# SECTIUNEA 1

# RAPORTUL STIINŢIFIC ŞI TEHNIC (RST)

## FAZA DE EXECUŢIE NR. 3

## CU TITLUL: „Implementarea, testarea soluţiei şi înregistrarea de date curente în sistemul pilot”

- **RST – raport stiintific si tehnic in extenso***
- **PVAI – proces verbal de avizare interna**
- **PVRLP – procese verbale de receptie a lucrarilor de la parteneri**
- **PF – protocol de fnalizare (numai pentru faza finala)**

* pentru Programul 4 “Parteneriate in domeniile prioritare” se va utiliza modelul din Anexa 1

**Anexa 1 - RST**

# Raportul Stiintific si Tehnic (RST) in extenso

**Proiect:** CARDIONET, nr. 2459 - PN2 Parteneriate
**Contract**: Nr. 11-001/14.sept.2007
**Titlul proiectului:** Sistem integrat pentru supraveghere continua in retea inteligenta e-Health a pacientilor cu afectiuni cardiologice - CardioNet.

**Etapa III** **Implementarea, testarea solutiei si culegerea de date curente in sistemul pilot**

**Termen predare**: 30.06.2009

# I. Obiective

## I.1 Obiectivele proiectului CARDIONET

**Obiective generale**

- implementarea infrastructurii unui sistem regional de telecardiologie care sa interconecteze medici de familie, policlinici si spitale, cu posibilitati de extindere la nivel national
- asigurarea colaborarii intre unitatile de cardiologie cu reteaua de asistenta primara pentru realizarea "dosarului electronic individual" al pacientilor
- realizarea schimbului eficient de informatii intre unitati spitalicesti din teritoriu, inclusiv serviciile de urgenta, pentru asigurarea unei interventii rapide in caz de urgente

**Obiective specifice:**

- identificarea celor mai eficiente echipamente si metodologii de monitorizare a pacientilor cu cardiopatie ischemica, tulburari de ritm, tratament anticoagulant
- analiza fluxurilor de informatii si a interactiunilor specifice dintre principali participanti ai sistemului de sanatate, cu aplicare in cardiologie (pacient, medici de familie, cardiologi, directiile sanitare, Ministerul sanatatii, Casa nationala de asigurari de sanatate si Ministerul muncii)
- realizarea unui sistem integrat si distribuit de achizitie, stocare, procesare si acces la informatii medicale cardiologice
- sistemul realizat va permite: monitorizare ECG continua a pacientilor aflati in evidenta retelei de asistenta primara cu patologie de tip aritmie (paroxistica, recurenta), precum si patologie coronariana; monitorizarea tensiunii arteriale a pacientilor incadrati in clase inalte de risc cardiovascular dupa initierea terapiei antihipertensive; urmarirea prin dozare ambulatorie a timpilor de coagulare (teste rapide) a eficientei tratamentelor anticoagulante cronice; evaluarea in spitalele teritorale prin ecocardiografie a patologiei pediatrice si/sau evaluarea periodica a valvularilor protezati
- elaborarea pe baza analizelor statistice a unor masuri profilactice si optimizarea tratamentelor pentru acest grup populational;
- asigurarea de mijloace pentru educatia continua a specialistilor in aceste domenii si formarea de noi specialisti prin implicarea in activitatile de cercetare a tinerilor masteranzi, doctoranzi si studenti .

## I.2 Obiectivele fazei de executie nr. 3

**Obiectivul general:**

Implementarea soluției de telemedicina pentru pacienti cu afectiuni cardio-vasculare, testarea sistemului si initierea activitatii de culegere a datelor medicale. Etapa a 3-a include: implementarea ontologiei domeniului si transpunerea acesteia intr-o baza de date, implementarea politicilor de securitate si de control a accesului la datele medicale, realizarea sistemului de culegere de date, realizarea unui portal de acces la informatii si servicii medicale si culegerea de date medicale.

**Obiective specifice:**

- poiectarea si implementarea ontologiei domeniului cardio-vascular, definirea conceptelor de baza si a relatiilor existente intre concepte
- proiectarea si implementarea modelelor de date si implicit a bazelor de date aferente; implementarea unei baze de date care sa pastreze atat informatiile ce decurg din ontologie cat si datele medicale curente ale pacientilor
- proiectarea si implementarea unei aplicatii destinate medicilor de familie (denumita in continuare „Aplicatia Medic de familie”), aplicatie ce va permite achizitia, stocarea, vizualizarea si transmiterea datelor medicale ale pacientilor
- Aplicatia Medic de familie va oferi suportul pentru interactiunea dintre medic si pacient atat de la distanta cat si la cabinetul medical
- stabilirea unei interfete de comunicatie intre diferite aplicatii medicale ce deservesc entitati medicale distincte
- proiectarea si implementarea unui portal medical ce va oferi acces la informatii si servicii medicale specializate pe domeniul cardio-vascular
- initierea procesului de culegere a datelor si analiza statistica a acestora

# II. Rezumat

Sistemul CardioNet, implementat în cadrul etapei curente, este un sistem distribuit de urmărire, monitorizare şi inregistrare a pacienţilor cu afectiuni cardio-vasculare. Sistemul a fost conceput cu scopul de a facilita interacţiunea dintre pacienţi şi medici, utilizând în acest scop cele mai avansate modele şi tehnologii informatice şi de comunicaţie.

La elaborarea proiectului s-au luat în considerare rezultatele analizelor efectuate în etapele anteriare privind:

- soluţii şi sisteme similare de telemedicină şi telecardiologie
- cadrul legislativ şi practica medicală curentă din ţară
- standardele existente în domeniu.

Având în vedere comlexitatea sistemului medical ce trebuia să se modeleze, s-a decis adoptarea unei soluţii distribuite, în care mai multe aplicaţii autonome, implementate la nivelul diferitelor entităti medicale, cooperează în vederea îndeplinirii funcţiilor de inregistrare şi monitorizare a pacienţilor. Ca urmare a acestei abordări s-au implementat mai multe componente autonome, care împreună formează sistemul CardioNet.

Astfel, s-a implementat o aplicaţie de tip Medic de familie, ce oferă suport pentru interacţiunea dintre pacienţi şi medici de familie, atât în mod direct (la cabinetele medicale) cât şi indirect prin intermediul Internetului. Aplicaţia permite planificarea consulturilor, efectuarea de consulturi on-line, off-line şi la cabinetele medicale. Datele culese în cadrul episoadelor medicale sunt înregistrate într-o bază de date construită conceptual pe baza unei ontologii a domeniului. Baza de date este flexibilă şi modificabilă în mod dinamic; noi concepte şi instante medicale pot fi adăugate fără monificarea structurii bazei de date sau a programului. Aplicaţia implementează o politică strictă de acces la datele medicale, ce garantează intimitatea pacienţilor dar în acelaşi timp oferă specialistilor acces la date statistice utile în analizele şi evaluările medicale.

Cu ajutorul aplicaţiei pacientul poate să dialogheze cu medicul de familie prin intermediul Internetului; astfel consultaţiile de rutină pot fi efectuate de la domiciliul pacientului fără să fie necesară deplasarea acestuia la cabinet. Pacientul va descrie starea sa medicală prin bifarea unui set de semne şi simptome predefinite precum si prin comentarii transmise sub formă de text. Medicul, pe baza celor declarate de pacient si a unor observatii de specialitate va genera un diagnostic şi un tratament adecvat. Alegerea diagnosticului şi a medicaţiei se face dintr-o lista predefinită în ontologie.

In vederea realizării schimubului de informaţii cu alte aplicaţii medicale similare, aplicaţia medic de familie expune un set de servicii web prin care baza de date poate fi interogată. In raport sunt prezentate în detaliu toate componentele aplicaţiei şi modul de implementare a acestora.

A doua componentă importantă a sistemului este aplicaţia de asistenţă pentru diagnoză şi tratament. Această componentă, accesibilă opţional de către medic pe parcursul unui consult, ajută medicul în investigarea şi diagnosticarea corectă a pacienţilor. Pe baza unui set iniţial de semne şi simptome aplicaţia selectează un set posibil de planuri de diagnoză. Aceste planuri conţin indicatii privind întrebările suplimentare care trebuie puse pacientului în vederea catalogării afecţiunii; apoi, pe baza răspunsurilor date sugerează un anumit diagnostic şi o anumită schemă de tratament.

A treia componentă a sistemului este un portal medical. Acest portal are rolul de a coagula componentele distribuite ale sistemului, oferind o cale unică de acces la informaţii şi servicii medicale. Portalul are in principal rol de informare atât a pacienţilor cu afecţiuni cardio-

vasculare cât şi a medicilor. Sunt postate informaţii privind realizări recente în domeniul cardio-vascular, informaţii despre entităti medicale şi despre servicii medicale oferite de medici înregistraţi în sistem.

Cu scopul monitorizării de la distanţă a pacienţilor s-a conceput şi implementat un sistem de achiziţie a parametrilor medicali bazat pe o reţea de senzori de tip wireless. Cu ajutorul retelei se pot culege atât valori de parametrii ce caracterizează starea pacientului (ex. temperatură, ECG) cât şi parametri ai mediului în care pacientul îşi desfăşoară activitatea (temperatură ambientală, umiditate, etc.).

Toate aceste componente concură la îndeplinirea funcţiilor de bază ale sistemului: facilitatea dialogului pacient-medic, monitorizarea de la distanţă a pacienţilor şi informarea pacienţilor şi a personalului medical.

Un aspect important ce s-a avut în vedere a fost achiziţia şi procesarea statistică a datelor medicale. Un capitol al raportului prezintă tehnicile de prelucrare şi analiză a datelor utilizate în proiect şi câteva rezultate preliminare obţinute.

In această fază a priectului s-a realizat implementarea sistemului şi testarea pe componente. In faza următoare se are în vedere integrarea acestor componente şi efectuarea de teste funcţionale. De asemenea, se are în vedere încărcarea bazei de date cu informaţii medicale reale culese de la cele 3 entităti medicale cuprinse în consorţiul proiectului.

Rezultatele ştiinţifice obţinute în etapa curentă au fost sintetizate în 4 articole ştiinţifice, prezentate în cadrul unor conferinţe internaţionale din domeniul informaticii aplicate şi din domeniul cardiologiei.

In ceea ce priveste achiziţiile de echipamente prevăzute iniţial în proiect, datorită diminuării fondurilor alocate pe anul 2009, acestea au fost amânate până în anul următor. Activităţile de cercetare-dezvoltare au fost reorientate astfel încât aceste întârzieri să nu afecteze obiectivele iniţiale ale proiectului.

# 1 Introducere

## 1.1 Consideratii generale

Prin proiectul de faţă se urmăreşte realizarea unui sistem integrat de înregistrare şi urmărire a pacienţilor cu afecţiuni cardiace prin utilizarea celor mai noi tehnologii informatice şi de comunicaţie. Se doreşte ca sistemul să permită culegerea şi vizualizarea de la distanţă a informaţiilor despre pacienţi, informaţii stocate într-o bază de date distribuită. Sistemul trebuie să asigure schimbul de informaţii medicale în timp real între medicul de familie, policlinică şi spital.

In cadrul prezentei etape s-a realizat un sistem informatic distribuit, bazat pe tehnologii inovative, ce solutionează principalele probleme legate de asistenţa medicală a bolnavilor cu afectiuni cardio-vasculare, de la aspectele de informare şi dialog pacient-medic până la cele legate de urmărirea de la distanţă a pacenţilor. Soluţia propusa si implementata conduce la îmbunătaţirea metodelor de prevenţie şi intervenţie rapidă în bolile cardiovasculare, asigura metodologii noi de supraveghere medicală şi contribuie la eficientizarea sistemului de sănătate publică în domeniul patologiei cardiologice (cardiopatie ischemica). S-a avut în vedere realizarea de structuri de date medicale compatibile cu standardele internaţionale în vederea integrării cu alte proiecte europene.

Proiectul contribuie la dezvoltarea de metodologii info-comunicationale, în domeniul medical în general şi cardiologic în special, prin utilizarea tehnologiilor informatice de ultimă oră în practica medicală din ţara noastră.

Aplicaţiile de telemedicină au ca obiectiv utilizarea tehnologiilor informatice şi de comunicaţie avansate în scopul asigurării de servicii medicale la distanţă. Telecardiologia furnizează servicii medicale utilizând dispozitive de telemetrie portabile folosite pentru monitorizare ECG, urmărire semne vitale (tensiune arterială, frecvenţă cardiacă, frecvenţă respiratorie), probe biologice de tipul testelor rapide (troponină, pro-BNP, D-dimeri). Progresele în telecomunicaţii fac ca la acest moment să se poată transmite inclusiv informaţii complexe precum examinari ecocardiografice, evaluari invazive de tipul angiografiei sau cateterismului cardiac, precum şi proceduri terapeutice de tipul angioplastiei percutane.

Utilizarea acestui sistem pe scară largă va contribui la îmbunătaţirea stării de sănătate a populaţiei prin eficientizarea sistemului de sănătate publică, îmbunătaţirea asistenţei de sănătate primare, secundare şi terţiare şi o adresabilitate şi complianţă mai ridicată la tratament din partea pacienţilor.

Rezultatele acestui proiect vor fi utile atât medicului specialist cardiolog (printr-o evidenţa mai bună a pacienţilor după instituirea tratamentului de specialitate), asistentei medicale primare şi secundare (prin consultarea bazelor de date şi prin schimbul de informaţii între specialişti) cât şi pacientului (prin creşterea complianţei la tratament, scăderea disconfortului determinat de deplasările la medicul de familie şi specialistul cardiolog, reducerea duratei spitalizării, reducerea numărului de zile de incapacitate de muncă).

## 1.2 Structura generală a sistemului CardioNet

Pe baza studiilor şi a analizelor efectuate în etapele anterioare, în cadrul etapei curente s-a trecut la implementarea soluţiei de telemedicină, care cuprinde mai multe componente autonome:

- o aplicație destinată medicilor de familie
- o aplicație destinată pentru urmărirea şi tratarea pacienților în unități spitaliceşti,
- o aplicație de asistare a medicilor în diagnosticarea şi tratamentul pacienților cu afectiuni cardio-vasculare
- un sistem de culegere de date bazat pe senzori
- un portal medical

**Aplicația „medic de familie**" oferă suportul informatic pentru:

- culegerea de date medicale şi inregistrarea acestora în format electronic, sub forma de fişe medicale de pacienți
- asigurarea dialogului de la distanță între pacienți şi doctorul de familie
- realizarea schimbului de informații medicale referitoare la pacienți cu alte aplicații medicale (ex: spital sau laborator)

**Aplicația destinată spitalelor** modelează mecanisme de culegere a datelor medicale şi de urmărire a pacientilor, specifice acestui tip de entitate medicală. Sistemul permite înregistrarea informațiilor de internare, urmărire curentă şi tratament, precum şi generarea documentului de externare. Această aplicație permite transferul electronic al datelor către şi dinspre aplicația „medic de familie". Acest schimb de informații este posibil datorită faptului că pentru ambele aplicații s-a adoptat aceeasi ontologie şi structură a bazei de date.

**Aplicația de asistență în diagnosticare şi tratament** ajută medicii în procesul de diagnosticare a bolilor cardiace pe baza semnelor şi semnelor culese şi a unor planuri de diagnoză prestabilite. In mod similar sistemul ghidează medicul în alagerea şi ajustarea medicației pentru pacienți. Aplicația este utilă mai ales pentru medicii începători şi pentru cei nespecializați în boli cardio-vasculare (ex: medici de familie). Acest instrument de asistare obligă personalul medical la formalizarea diferitelor proceduri de diagnoză şi tratament, permite reutilizarea experienței acumulate şi stocate în planuri de tratament şi evită cazurile de mal-praxis.

**Sistemul de culegere de date medicale** se compune dintr-un set de senzori conectați într-o retea wireless, având rolul de a culege în mod continuu valori ale parametrilor medicali şi de mediu. Aceşti parametri caracterizează starea de sănătate a pacientului (ex: puls, tensiune arterială, ECG, concentrație de oxigen, etc.), dar şi condițiile de mediu în care îşi desfăşoară activitatea (temperatură, umiditate, etc.) Software-ul dezvoltat pentru acest sistem asigură comunicația între noduri şi accesul transparent (pe baza de nume) la parametri măsurați.

**Portalul medical** cumulează informatii şi servicii cu caracter medical, accesibile atât madicilor cât şi pacienților. Scopul portalului este acela de a oferi un punct unic de acces către diferite servicii medicale implementate în cadrul proiuectului CardioNet. Ajută la informarea pacienților privind diferitele boli cardio-vasculare şi modul de tratare a acestora. Nu este menit să înlocuiască consultul unui medic, ci furnizează datele necesare pentru o mai bună prevenție şi profilaxie a bolilor de inimă. Ofera un forum de dialog între pacienți şi cadre medicale. Într-o etapă următoare se prevede includerea în structura portalului a unor servicii de stocare centralizată a datelor medicale ale pacienților şi servicii de conversie a datelor în vederea schimbului de date între aplicații medicale diferite.

# 2 Cardionet- Modele, arhitecturi, standarde, protocoale si structuri de date adaptate solutiei proiectului Cardionet

## 2.1 Cardionet – implementare modele, arhitecturi, standarde si structuri de date

Arhitectura sistemului distribuit Cardionet este definita ca fiind nivelul cel mai inalt de abstractizare, care prezinta strategia de rezolvare a problemelor domeniului ales[3-RM-ODP]. Ea a fost definita in general in etapele anterioare dar pentru generalizarea solutiilor de interoperabilitate a componentelor si pentru ca si depozitele de date accesate prin sistem devin distribuite, mai revenim cu unele precizari asupra designului in ansamblu. Unele din precizarile pe care le vom face in continuare sunt caracteristice portalului Cardionet si structurilor sale de date.

### 2.1.1 Cardionet- adaptare modele, metode, concepte si arhitecturi la domeniul proiectului

Pentru rigoarea si coerenta modelarii, s-a plecat de la urmatoarele idei:

-Sistemul va face distintie intre aplicatiile operationale si cele de diseminare, analiza sau suport de decizie. Aplicatiile operationale, implica urmarirea fluxului curent de operatii ale proceselor de ingrijire a sanatatii pacientilor si, in consecinta trebuie sa proceseze datele conform cerintelor acestor fluxuri culegere si de prelucrare. Activitatile operationale sunt in general simple, accesand putine inregstrari si trebuie sa aiba ca timp de raspuns un “timp real” in acceptiunea actorilor din domeniu, daca tehnologic este posibil. Deoarece activitatile de la nivelul operational sunt fundamentale pentru procesul economic in sine, baza de date trebuie sa aiba integritatea asigurata, sa ramana intacta, sa fie sigura si actuala. In general, multe baze de date operationale stocheaza numai datele cele mai recente, prin suprascriere sau prin metode de arhivare a datelor vechi, in domeniul medical prima metoda nu este potrivita cu cerintele de prelucrare.

-Prin constrast, aplicatiile analitice contin cereri complexe care prelucreaza uzual cantitati mai mari de date si permit organizatiei sa ia decizii strategice. In aceste situatii datele vor fi stocate sub forma unor depozite de date (datawarehousing) (figura II.1), care pot combina date provenind de la mai multe surse. Un depozit de date isi preia informatiile prin alimentare periodica de la bazele de date operationale si le completeaza eventual cu date provenite de la alte surse, proces reprezentat in figura II.1. Cele mai multe depozite de date stocheaza datele parametrizat, cu momentul la care acestea s-au produs, astfel incat utilizatorii sa poata solicita, ulterior, date dupa criterii temporale si sa poata decela tendintele unor parametri in anumite perioade de timp. Aplicatiile analitice tind sa aibe putine actualizari, altele decat alimentarile periodice cu noi inregistrari.

-Descompunerea sistemelor mari in subsisteme distribuite pe mai multe nivele. Un nivel este practic un subsisetm care se construieste peste subsistemele de la nivelul inferior de abstractizare. Pot exista si subsisteme care activeaza simultan, in paralel cu alte subsisteme. Descompunerea se va produce iterativ, dupa cerintele din structurile organizationale implicate, conform cu cerintele fluxurilor de prelucrare din domeniu.

-Separarea proceselor de prelucrare logica de interfetele utilizator. In mod normal acesta decuplare este vizibila in structura logica si functionalita a sistemului. Interfetele utilizator au nevoie de un alt tip de codificare si se adreseaza cu precadere altor actori implicati (browsere sau persoane fizice). Deseori este de preferat ca aplicatiile sa fie organizate ca biblioteci de componente(API)), pentru ca apoi sa se invoce componentele adecvate conform cerintelor de prelucrare. Pot exista interfete utilizator multiple care acceseaza in comun codul aplicatiei, dar pot diferi prin formatul de prezentare si prin informatiile pe care le prezinta spre vizualizare.

-Orientarea prelucrarii spre OOP(Object Oriented Programming). In anumite cazuri informatii cum ar fi : controlul, regulile sau restrictiile se vor transfera spre obiecte, conducand la un nivel mai ridicat de complexitate a datelor.

-Transferarea spre serverul de baze de date a unor sarcini de prelucrare. Trebuie pastrat un echilibru intre rolul "motorului" bazei de date si componentele de procesare ale aplicatiilor. Un server de baze de date poate face mult mai mult decat simpla stocare si regasire a datelor ; acesta este o "masina de calcul" (MS SQL 2008 in cazul Cardionet) puternica spre care se pot transfera o parte din sarcinile de prelucrare(proceduri stocate, trighere, view-uri, reporting services, etc), reducand timpul general de realizare a interogarilor, respectiv de obtinere a unor seturi de date solicitate.

-Sa se proiecteze interfete specializate pentru actori (roluri) precum si pentru interactiunea automata cu alte sisteme. Arhitectura generala a sistemului poate fi afectata de constrangerile solicitate de aceasta interactiune, automata sau "manuala", cu alte sisteme. Este de preferat ca datele introduse manual (de operatori umani) sa fie cat mai reduse, pentru diminuarea pe cat posibil a surselor de eroare.

Se recomanda, daca este posibil, utilizarea unui proces formal pentru gandirea unei arhitecturi tinand cont totodata de costuri, fezabilitate, riscuri si analizand desigur si arhitecturi si solutii alternative [2].

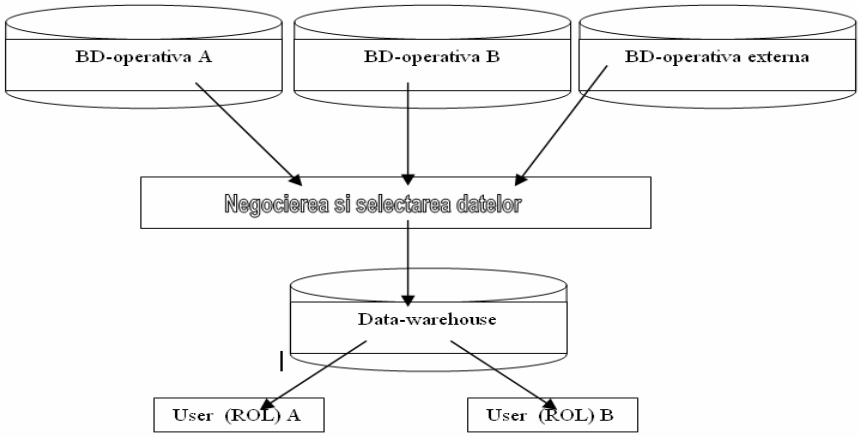



Figura II.1. Schema de creare a depozitelor specializate din bazele de date operationale

### 2.1.1.1 Conceptualizarea datelor

Cerintele de standardizare si interoperabilitate impun structurilor de date din domeniu proiectului sa adapteze si sa coreleze cerintele de prelucrare in conformitate cu:

- ariile geografice de operare: globale, regionale, nationale sau locale;
- sub-domeniile de interes- resurse umane, marketing, asigurari sociale, cercetare, analize statistice sau de specialitate si medical (cardiovascular).
- nevoile de internationalizare- acelasi concept este reprezentat vizual diferit

- novoile cultural/legale- colectarea si stocarea anumitor informatii ,traditional este permisa sau nu de reglementarile uzuale;
- nevoile financiare- se pot referi la legislatia locala, regionala sau globala.

Componentele standardizarii datelor cuprind:

- partile semantice ale datelor si structurilor de date
- modelul complet al metadatelor pentru semantica lor
- clasele si obiectele asociate acestora
- conventii de nume (reprezentari multilingve)
- seturi de valori permise, acceptate standardizat
- tratarea unitara a tuturor aspectelor de prelucrare, inclusiv tratarea erorilor.

Un Element de Date privit ca o entitate de domeniu, poate fi privit ca fiind compus din trei parti :

- Ierahia semanticilor,
- Elemente de date derivate si compuse, incluzand si “Concepte”
- Elementul de Date propriuzis.

Un concept important este acela ca un element de date este utilizat ca un sablon semantic, reprezentand o marcare a semanticilor faptelor domeniului. Ierahiile semanticilor domeniului sunt asemanatorae taxonomiilor conceptelor datelor, organizate pe criterii diverse, cum ar fi :

- temporale (primul, ultimul, urmatorul, anual, trimestrial etc)
- geografic ( local, mondial, european, regional, national, etc)
- sau precizia de aparitie( estimat, exact, final, etc)

Atunci cand aceste ierarhii sunt utilizate pentru construirea de “concepte” aceastea includ un nume al elementului de date, putandu-se genera nume diferite (sinonime) si abrevieri in mod automat. Modelul metadatelor unui Element de date a fost reglementat prin standardul ISO 11179. In consecinta exista formalizari propuse ale metadatelor care cuprind elemente de date, domeniile lor de valori, maparea valorilor domeniului, asignarea tipurilor de date, etc. Aceste constructii prestabilite ajuta la gestionarea semnificatiei datelor, iar abordarea inca de la inceput a proiectarii in acesta maniera a condus la un model de date mult mai apropiat de modelul real al domeniului (cardiovascular pentru setul de date pilot).

Motivatiile acestei modalitati de proiectare a bazelor de date sunt:

- Domeniile proceselor de prelucrare sunt strans legate conceptual de structurile de date si de fluxurile de prelucrare in care procesul economic rezida;
- Compunerea elementelor de date se face prin agregarea din secvente specifice mai multor elemente. Compunerea elementelor de date se poate face si imbricat, conducand la premise pentru o arhtectura “document oriented”;
- Conceptele reprezinta seturi de idei, abstractiuni, cu reprezentare si semnificatii in lumea reala, identificabile si supuse acelorasi reguli de comportament. Conceptele sunt utilizate ca si baza pentru specificarea cunostintelor dintr-un domeniu de interes.
- Conceptul poate reprezinta o colectie de concepte care sunt relationate si care care sunt reprezentate printr-un concept colectiv;
- Elementele de date sunt sabloane ale faptelor domeniului si sunt independente de contextul tehnologic;
- Clasificarea elementelor de date reprezinta scheme sau ontologii necesare pentru intelegerea categoriilor de cunostinte din domeniu;
- Elementele de date derivate reprezinta rezultatul unor transformari sau calcule efectuate fie asupra unei colectii de elemente de date, fie asupra unei compuneri de elemente de date;
- Valorile permise pot fi controlate prin semantici individuale care sunt asignate elementelor de date, pentru sustinerea unei specificatii complete a elementuluii de date. Instantele acestor

semantici pot fi organizate ierarhic. Numele asignate lor pot deveni parte explicita a numelui elementuluii de date;
- Domeniile de valori sunt seturi de valori care pot fi discrete sau interval si care sunt permise in contextul general al semnificatiilor specificate din domeniul de interes. Domeniile de valoare sunt utilizate atat de elementele de date cat si de SGBD-uri, ca subseturi de atribute ale acestor entitati si sunt atribuite coloanelor tabelelor SGBD.

#### *2.1.1.1.1 Modelul de date specificat*

Modelele de date specificate, numite si modele conceptuale, permit expertilor in domeniu sa defineasca, independent de contextul tehnologic, structurile de date standard necesare pentru :

- executarea politicilor functionale ale domeniului
- stabilirea conceptelor, structurilor de date si valorilor admise
- rezolvarea cerintelor de prelucrare
- definirea de proprietati abstracte si reale
- realizarea de tranzactii specifice domeniului
- stabilirea nevoilor de resurse umane implicate
- definirea de evenimente semnificative si reactiile asociate necesare.

Odata ce aceste concepte au fost modelate, ele pot fi utilizate ca punct de start pentru activitatea de modelare a bazelor de date. Aceste modele conceptuale de date, asa cum vom vedea in paragrafele urmatoare, se compun din : subiecte, entitati si atribute. Fiecare atribut trebuie mapat peste un element de date, devenind astfel un derivat al semanticilor acestuia.
Pot fi utilizate, de asemenea si ierarhiile semantice pentru formarea de cuvinte care compun numele atributului. In cazul in care cuvintele semantice sunt asignate unui element de date, acestea sunt automat mostenite de un atribut. In cazul in care este necesara aplicarea unei reguli semantice mai restrictiva unui atribut, acesta poate fi asignata. Semanticile atributului sunt, in consecinta, toate semanticile asociate acestuia, propriile-i entitati si semnaticile pe care le mosteneste de la elementele de date din modelul de date. Aceasta reprezinta de fapt managementul semantic real al semnificatiei datelor. Prin aceasta strategie, definirea atributelor, a sinonimelor si acronimelor poate fi realozata in mod automat.
Aceste atribute vor fi mapate apoi, peste domeniul de valori, astfel incat seturile de valori sunt automat restrictionate prin domeniul de valori mostenit de la elementul de date al atributului, sau prin asignarea unui domeniu de valori si mai restrictiv.
Pentru implementarea unui astfel de model sunt necesare tabele pentru urmatoarele informatii:

- atribute
- atribute si valori
- entitati
- cheia candidata a entitatii
- cheia candidata a entitatii si atributului
- cheia straina a entitatii
- cheia straina a entitatii si atributului
- cheia primara a entitatii
- cheia primara a entitatii si atributului
- Subiect

Nevoile de reprezentare a informatiilor mentionate pentru componentele modelului de date sunt impuse de urmatoarele caracteristici:

- atributele sunt manifestari ale semanticilor elementului de date in interiorul unei entitati apartinand subiectului asociat;

- Domeniile de valori ale atributelor sunt seturi de colectii de valori, rafinand astfel domeniul de valori, pentru un atribut particular;
- Atributul si valorile sunt relatii care exista intre un atribut si seturile de semantici asignate. Aceste semantici asignate sunt totdeauna un subset ale celor asignate atributului elementului de date parinte;
- O entitate este o expresie bine definita a unei politici intr-un doemniu de interes. Colectia tuturor entitatilor din domeniu poate defini complet setul de politici ale acestuia.
- Cheia candidata a entitatii reprezinta o colectie de atribute dintr-o entitate care, la momentul primirii valorilor, sunt utilizate pentru regasirea sau actualizarea unei linii de date din entitate(in cazul in care entitatea este o tabela). Nu este permis ca atributele cheii candidate sa se suprapuna unele cu altele (ortogonalitate) sau cu cheia primara a entitatii;
- Cheia straina a entitatii reprezinta o legatura directa cu cheia primara a entitatii. Numele cheii straine trebuie sa defineasca cat mai bine relatia pe care o reprezinta cheia. Atributelor cheii straine nu li se permite sa se suprapuna cu cheia primara. Suplimentar pentru cheile straine exista si reguli aditionale care gestioneaza inserarile, actualizarile si stergerile;
- Cheia primara a entitatii reprezinta o colectie de atribute apartinand entitatii. Intr-o entitate nu poate fi decat o singura cheie primara. Atributele cheii principale trebuie sa fie ortogonale;
-Subiectele reprezinta concepte din domeniu. Subiectele pot fi organizate ierahic si abordeaza resursele esentiale ale domeniului.

Modelele specificate de date sunt considerate distribuite in domeniul de interes, sunt independente de contextul tehnologic, si sunt specifice domeniului.

#### *2.1.1.1.2 Modelul de date implementat*

Modelul datelor implementate, cunoscut si ca modelul logic, este de fapt un model al schemelor bazelor de date. Schema devine astfel un container pentru tabele, iar tabelele devin containere pentru coloane.
Aceasta este in opozitie cu de modelul datelor specificate unde temele sunt colectii abstracte de entitati. De aceea este posibil ca in modelul datelor specificate, relatiilor sa li se permita sa fie trans-tematice, dar in modelul datelor implementate relatiile nu pot depasi schemele. Aceste modele de date implementate constau din scheme, care la randul lor constau din tabele, care contin coloane. O schema si colectiile de tabele aferente reprezinta de fapt utilizari ale entitatilor si atributelor acestora. O tabela poate contine o intreaga entitate, o parte a unei entitati, entitati multiple sau o entitate de mai multe ori (instantieri diferite).
Ierarhiile semantice, definite in modelul elementului de date pot fi utilizate, de asemena pentru a forma cuvinte care, in substanta lor contin numele coloanei. In cazul in care cuvintele semantice respective sunt asignate unui element de date, atunci coloana le va mosteni in mod automat.
In final, este posibila si asignarea “manuala”, la o coloana a unei semnatici mai restrictive, proprie acesteia, caz in care numele coloanei este trebuie ajustat. Semanticile unei coloane sunt, in consecinta, toate lucrurile asociate acesteia, si tot ceea ce mosteneste de la elementul de date. Mecanismul prezentat devine una dintre cele mai importante resurse de gestiune a semnificatiei dalelor.
Aceasta strategie permite proiectantilor de baze de date sa utilizeze colectii standardizate de specificatii de fapte pentru a construi schemele bazelor lor de date. Fiecare coloana, din fiecare tabela este mapata peste un atribut, care, la randul sau este mapat peste elemente de date si care, in continuare sunt mapate pe nivele superioare ale semanticii despre concepte, care la randul lor contin restrictiile de doemniu. La proiectarea bazei de date pentru modelul de date implementat sunt necesare tabele pentru a reprezenta urmatoarele informatii:

- coloane
- coloane si valori
- scheme
- tipul de date SQL
- tabele
- chei candidate
- chei straine
- cheia primare.

Nevoile de reprezentare de mai sus pentru acest model survin deoarece:

- coloanele sunt manifestari ale semanticilor elementelor de date dintr-o tabela a schemei. Nu este obligatoriu ca o coloana sa mapeze atribute ale unei singure entitati;
- Domeniile de valori ale coloanelor sunt subseturi ale domeniilor de valori ale atributelor;
- O tabela este o expresie bine definita a unei politici in cadrul unei scheme. Ideal, colectia tuturor tabelelor dintr-o schema trebuie sa defineasca o politica completa;
- Cheile candidate ale tabelei reprezinta o colectie de coloane din respectiva tabela care la momentul instantierii sunt utilizate pentru regasirea sau actualiazrea unei linii unice;
- Cheia straina a tabelei reprezinta o asociere cu cheia primara. Coloanele cheii straine trebuie sa poata fi sterse din tabela fara pierderea politicii pe care tabela o reprezinta. Coloanele cheii straine nu se suprapun peste coloanele cheii primare.
- Cheia primara a tabelei reprezinta o colectie de coloane apartinand tabelei care permit, la momentul interogariilor, regasirea sau actualizarea unei singure linii a tabelei. Coloanele cheii primare trebuie sa fie ortogonale;
- Schema reprezinta o structura de baza de date pentru tabele si relatii in cadrul domeniului abordat;
- Tipurile de date SQL sunt o clasificare a valorilor reprezentate de coloanele tabelelor unei baze. Fiecare tip de date SQL impune un set de reguli relative la valorile permise si la operatiile acceptate a se realiza asupra lor.

#### *2.1.1.1.3 Modelul de date operational*

Modelele de date operationale, numite si modele fizice sunt anolage modelelelor implementate (modelele logice ale datelor) cu exceptia fatului ca sunt legate direct de un SGBD si o paltforma hardware specificata. Bazele de date fizice sunt acele structuri care contin datele reale si care alimenteaza cu seturi de date procesele de prelucrare. Aceste modele operationale constau din scheme, tabele si coloane SGBD.

Deoarece modelele de date operatioanle sunt, de asemenea, containere, relatiile intre modelul implementat si cel operational nu sunt simple relatii de forma unu la unu. Un exemplu edificator pentru aceasta observatie este dat de insasi proiectarea si structurarea depozitelor de date. Obiectivul principal al unui astfel de depozit este de a mixa date din surse diferite si eterogene, intr-o baza de date unica, selectand datele dupa criteria prestabilite.

Asfel pot exista diferentieri ale proiectiilor datelor fizice (avand in vedere ca in general un astfel de depozit este o stocare distribuita in spatii fizice si adeseori chiar cultural, eterogene).

Dat fiind ca modelul datelor operationale este direct implementat de un SGBD, conceptele anterioare se vor particulariza in limbajul specific al acestui SGBD. Astfel acest model de date are nevoie de tabele pentru a cuprinde urmatoarele informatii:

- Baza de date
- Starea bazei de date
- Coloanele SGBD
- Schema SGBD

- Tabelele SGBD
- Chei candidate
- Chei straine
- Chei primare
- Chei secundare
- Tipuri de date SGBD

Necesitatea acestor tabele se justifica dupa cum urmeaza:

- O baza de date este o colectie de tabele, cu o structura de tip schema, care poate cuprinde: colectia de date originara, spatiul tranzactiilor, baza de date a domeniilor de interes, depozitul de date sau datele referite.
- Natura bazei de date este data de tipul proceselor economice si de utilizare a bazei de date.
- Starea bazei de date va codifica starile posibile ale unei baze de date. De obicei valorile tipice ale acestei clasificari sunt: dezvoltare, test si productie.
- Coloanele SGBD sunt expresii ale semanticilor, dintr-o schema SGBD.
- Un tip de date SGBD este o clasificare a valorilor reprezentate printr-o coloana de date. Sunt specifice producatorului SGBD. Fiecare tip de date SGBD, impune reguli asupra valorilor datelor, precum si asupra operatiilor permise.
- O tabela este o expresie bine definita a unei politici din schema SGBD.
- Cheile candidate, primare, straine si secundare se definesc similar, dar in termeni reglementati de SGBD.

#### *2.1.1.1.4 Modele de vizualizare a datelor*

Modele de vizualizare formalizeaza interfetele (relatiile) dintre bazele de date, componentele asociate proceselor de prelucrare si cele de afisare a lor pentru operatorii umani. Vizualizarile pot fi mapate peste scheme diferite, permitand astfel aplicatiilor sa acceseze baze de date multiple, care lucreaza sub diferite SGBD-uri.

O particularitate importanta este faptul ca numele unei coloane din vizualizare poate fi diferit de numele coloanei din baza de date din care este derivata. Astfel, numele coloanelor vizualizarilor poate fi setat la ultima mostenire din elementul de date pe care-l refera. Adica definirea coloanei vizualizarii, poate fi mostenita din elementul de date. Acest lucru va permite o procesare automatizata.

Concluzionand, gestionarea semnificatiei datelor este de un interes major si duce mai departe pe un alt plan conectivitatea asigurata prin interoperabilitate.

Tehnologiile si resursele necesare pentru interpretarea datelor, bazate pe descoperirea acestora sunt destul de variate si fac obiectul unor dese schimbari tehnologice. In schimb abordarea prin crearea de elemente de date standardizate, bazate pe concepte standardizate, au adus stabilitatea necesara si au condus la aparitia de pachete de aplicatii specializate pe domenii economice (ex: palnificarii resurselor intreprinderii (ERP), SAP, BAAN, ABAS,…).

Abordarea proiectarii prin semantici standardizate si bazata pe politici si metodologii de interactiune, va permite operatorilor din domeniu sa devine capabili sa gestioneze si semnificatia datelor si sa execute totodata, unitar procesele de prelucrare, focalizate pe business nu pe tehnologie.

### 2.1.1.2 Metode de integrare: ingineria directa si inversa

Este foarte important ca platformele utilizate in nivelele dependente de tehnologie sa fie capabile sa reconstituie, printr-un proces de refacere, toate nivelele de date si de metadata cuprinse in

modelele de date initiale. Ingineria inversa, ca proces in sine, este extrem de valoroasa pentru cazurile in care sunt necesare, pentru integrare si interoperabilitate, operatii cum ar fi: "intersectii" si "reuniuni" de modele de date, care provin din sisteme diferite, deja existente.

In cazul intersectiei fiecare model de date al sistemului, deja existent, este importat in metabaza, dupa care, unul cate unul, cu structura de date din tabele, este "translatat" spre nivelul modelului implementat.

Odata completat, modelul de date implementat, reprezinta intersectia tuturor modelelor de date deja existente. Aceasta intersectie poate fi importata in modelul de date operational, creindu-se astfel un nou model, care poate fi exportat (ca schemei DDL a SQL) pentru a realiza proiectarea noii baze de date, care reprezinta astfel intersectia bazelor de date deja existente.

In cazul reuniunii, modelul nou de date implementat inseamna proiecatrea unei baze de date noi, pentru domeniul de interes. Aceasta este mai ampla decat toate bazele de date care o alimenteaza. Mecanismul tehnic de reprezentare a bazei de date este similar cu cel descries in cazul intersectiei. Exista trei scopuri pentru utilizarea ale acestei tehnici si anume:

- crearea de semantici pentru gestiunea datelor la nivel de intreprindere (organizatie);
- crearea de modele de date de tip intersectie pentru comunitatile sistemelor deja existente si care au deja incetatenite schimburi de date;
-crearea de modele de reuniune, care reprezinta baze de date construite pentru domenii de interes, pentru comunitati care vor folosi baze de date mai cuprinzatoare.

Similar cu acest mecanism este folosit si mecanismul ingineriei directe, care urmeaza secventa relativ fireasca de constituire a modelelor de date implementate pornind de la entitati ca subiecte ale modelelor de date specificate, cu urmarea finala a realizarii modelelor de date operationale si obtinerea intregului set de coloane implicate.

Dintr-o perspectiva istorica, in directia interconectivitatii sistemelor se remarca urmatoarele tendinte :

• *Integrare* – au la baza crearea unor modele comune, standardizate;
• *Unificare* – sisteme eterogene unificate pe baza unui metamodel comun;
• *Federalizare* – intercomunicare fara nici un sablon comun.

In primul caz, am putea exemplifica prin incercarile de integrare software, promovata de sistemele ERP. S-a dovedit a fi un obiectiv dificil, chiar putin probabil de atins in mediul inter-organizational, adeseori din motive de concurenta.

Unificarea presupune definirea unor concepte si metamodele, cat mai generale si daca este posibil universal valabile. Partenerii isi pastreaza, in mare masura, specificul intern (organizarea activitatilor, reguli de prelucrare, etc.) dar in relatiile de colaborare, adera la respectivele standarde. Este tendinta actuala si promite mult din perspectiva obtinerii unui nivel ridicat de interoperabilitate (planetara).

Federalizarea presupune adaptarea dinamica a sistemului unei organizatiei, in functie de relatiile de cooperare pe care le stabileste la un moment dat. Tendinta este, de regula, asociata cu stiinta reprezentarii cunostintelor (inteligenta artificiala) si se afla , in plina dezvoltare prin noi orientari, date de tehnologiile Semantic Web.

Federalizarea reprezinta modelul in care sistemele existente nu vor face nici o modificare in propriile modele, eventual vor coopera la alimentarea unitara a unor nivele superioare arhitectural, prin care se va realiza interoperabilitatea.

La o privire generala, eforturile de unificare semantica in construirea modelelor de intreprindere sunt directionate de:

-perspectiva reglementarilor impuse pentru implementare:
-standarde reglementate ISO sau CEN;
-standarde „deschise" (generate de consortii nonprofit continand: specificatii si unelte software suport cu scopul de a impune standarde „de facto" noi sau diferite fata de cele reglementate ISO);
-perspectivele domeniului abordat: metodologii; concepte si metamodele; notatii si semantici pentru modelare conceptuala; structuri sintactice pentru reprezentare si executie; structuri pentru obtinerea interoperabilitatii (comunicatii, mesagerie); dictionare de termeni.

#### 2.1.1.3 Concepte si metamodele arhitecturale de sisteme informatice, e-Health

Literatura ultimilor ani abunda in articole orientate spre identificarea unui metamodel si unificarea conceptelor pentru o modelare unitara a unei intreprinderii sau al unui domeniu de interes. Iata doar cateva initiative:
- propuneri de unificare a modelelor ERP (ABAS, BAAN, SAP)– construirea unei baze unice si necesitatea introducerii elementelor de QS (Quality of Service), din perspective multiple, inclusiv cea economica.
- Object Process Metodology - obiectele si procesele sunt primitive folosite la modelare cu importanta egala, desi un proces nu poate exista fara o interactiune cu un anumit obiect. Metodologia propune si introduce si un nou formalism grafic de design.
- definirea unei metodologii de armonizare a arhitecturii afacerilor cu cea software, pornind de la concepte organizationale: entitati de intreprindere, obiective economice, procese, reguli economice.
- modelarea intreprinderii pornind de la arhitecturi SOA (Service Oriented Architecture) si un metamodel, ce extinde UML, cu entitati cum ar fi: resurse, produs, proces, operatii, reguli (mecanism de prioritati), roluri (actori), context (obiective si set de reguli).
Figura urmatoare arata un mod de a descrie legaturile dintre entitati si relatii, in domeniul de interes "healthcare", pe care il propune modelul HL7-RIM.

Modelul HL7 RIM consta dintr-un mare numar de clase grupate intr-un numar de 6 blocuri care alcatuiesc asa numitul schelet ("backbone") al modelului, figura II.2. Aceste blocuri sunt:

*Entity:* reprezinta entitati fizice (lucruri si fiinte) care sunt implicate in procesele de "healthcare";

*Role:* stabilesc rolurile destinate acestor entitati dupa cum participa ele la procesele de "healtcare";

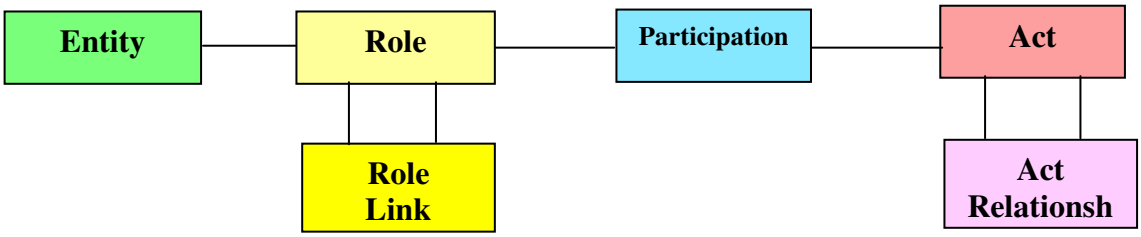
*RoleLink:* reprezinta relatiile dintre rolurile individuale;

*Participation:* descrie contextul unui "act medical" in termeni: cine l-a executat, pentru cine a fost executat, unde, cand, etc.;

*Act:* reprezinta actiuni ("acte medicale") executate in scop "healtcare", care pot fi documentate si gestionate;

*ActRelationship:* reprezinta legaturi intre aceste acte (actiuni) medicale.

Acest "backbone" reprezentat in figura II.2, poate fi citit de la stanga la dreapta astfel: O Entitate (ex. Persoana) joaca un Rol (ex. Pacient) si participa in niste "Acte medicale" (ex. Consultatie). La fel poate fi citit si de la dreata spre stanga: La un "Act medical" (ex. Operatie de apendicita) a participat o Entitate (ex. Persoana) avand un Rol (ex. Nurse)




HL7 Reference Information Backbone @-[HL7-RIM]

Figura II.2. Modelul HL7-RIM -grupurile principale de clase HL7-RIM

## 2.2Cerinte de conformitate. Standarde si reglementari in domeniul sanatatii si eHealth

Sistemele informatice medicale in general precum si cele specifice inregistrarii electronice a datelor medicale, EHR (Electronic Health Record) sunt in centrul preocuparilor unor importante organisme internationale, regionale si locale(nationale) pentru optimizarea lor si pentru asigurarea interoperabilitatii intre sisteme.
Acestea preocupari acopera unele aspecte mai generale, cum sunt definirea si scopul sistemelor EHR, tipurile de date si sistemele de codificare a informatiilor, sau specificarea in detaliu a unor modele informatice pentru stocarea si regasirea datelor, a documentelor si a modalitatilor permise pentru schimbul de informatii medicale.

### 2.2.1 Cadrul international, organisme de reglementare. Caracteristici e-Health

Pe plan mondial, principala organizatie implicata in standardizare este ISO (International Standards Organization), domeniul medical fiind in sarcina comitetului tehnic TC 215. In Europa activitatea de standardizare este organizata in cadrul CEN (Comité Européen de Normalisation), in anul 1991 fiind infiintat un comitet tehnic pentru informatica medicala - TC251.
In 1995 s-a aprobat: "Directory of the European Standardisation Requirements for Healthcare Informatics and Telematics. Programme for the Development of Standard" - ce constituie programul de lucru al CEN/TC 251.
Principalele standarde pentru sistemele de evidenta electronica a datelor medicale sunt:

-ISO DTR 20514 - EHR definition and scope
-ISO TS 18308 - EHR Requirements
-CEN TS 14796 - Data Types
-CEN/TC 251 EN 13606 - EHR Communications
-HL7 - EHR Functional Specification
-HL7 - Templates specification
-HL7 - Clinical Document Architecture
-DICOM – Digital Imaging and Communications in Medicine
-EDIFACT , XML – Messaging standards
-Health informatics - Standardized computer protocols for ECG - PrENV 1064:2000 version 1.3 2000-10
- VITAL-FILE -Health informatics - File exchange format for Vital Signs- New WI Proposal - File Exchange Format for Vital Signs

- EPAS -Health informatics - Evaluation of Physological Analysis Systems - Testing Physiological Measurement Software -- Part 1: 2000-10-10, Addendum on ECG Measurement Systems 2002-02-11
- OpenEHR – archetypes and tempaltes

Dintre organismele internationale mai importante amintim:

#### 2.2.1.1 Organisme globale - ISO/TC 215

Comitetul tehnic pentru informatica medicala al ISO a luat fiinta in anul 1998 prin participarea a 25 de state membre si a 14 observatori. Pana in anul 2004 sub directa responsabilitate a grupului de lucru TC 215 au fost elaborate un numar de 14 standarde.
Amintim:

-**TR 20514:2005** Raport tehnic privind Dosarul electronic de sanatate – definitie, scop si context. Cuprinde "o clasificare pragmatica a dosarului electronic de sanatate (EHR)" , furnizand definitii ale principalelor categorii de sisteme EHR si descrieri ale caracteristicilor acestora. Raportul face o distinctie clara intre continutul EHR si structura acestuia, asa-numitul "generic EHR" definind o structura generica in masura sa asigure o larga aplicabilitate pentru toti utilizatorii si sistemele EHR prezente si viitoare.

-**TS 18308:2004** - Specificatie tehnica privind cerintele medicale si tehnice pentru arhitectura EHR. Scopul acestui standard este de a "realiza o colectie de cerinte clinice si tehnice pentru arhitectura EHR care sa sustina utilizarea, partajarea si schimbul inregistrarilor electronice medicale intre diferite sectoare de ingrijire a sanatatii, diferite tari si diferite modele de furnizare a ingrijirilor pentru sanatate" .

-**TR 18307:2001** Raport tehnic privind Interoperabilitatea si Compatibilitatea in standardele pentru comunicatie si schimb de mesaje. Descrie un set de caracteristici cheie pentru obtinerea interoperabilitatii si compatibilitatii in schimbul de informatii medicale intre sisteme informatice.

-**ISO 21731:2006** Standard privind modelul informatic de referinta (RIM) – preluat din standardul HL7 v.3

-**ISO 17432:2004** – Mesaje si comunicatie, acces web la obiectele DICOM

-**ISO 20301:2006** – Carduri de sanatate. Caracteristici generale

-**ISO 20302:2006** – Carduri de sanatate. Sisteme de numerotare si proceduri de inregistrare a identificatorilor personali

#### 2.2.1.2 Organisme europene - CEN/TC 251

Organismul recunoscut in Europa in domeniul standardelor pentru informatica medicala este CEN/TC 251 (European Committee for Standardization - Technical Committee on Health Informatics). Acesta are competenta si responsabilitatea organizarii, coordonarii si monitorizarii activitatii de dezvoltare a standardelor in domeniul informaticii medicale, precum si a promulgarii acestor standarde. CEN publica si pune la dispozitie diverse materiale numite ***standarde europene*** (European Standards - **EN**), ***pre-standarde europene*** (European Pre-standards - **ENV**) si ***rapoarte CEN*** (CEN Reports - **CR**) .

Din 1990 cand a fost creat si pana in prezent, CEN/TC251 a elaborat peste 50 de documente (standarde, prestandarde si rapoarte). In cadrul proiectelor de cercetare europene, au fost elaborate o serie de prestandarde, vizand vocabularul utilizat in informatica medicala, arhitectura fisei electronice de sanatate, arhitectura sistemelor informatice de sanatate precum si cadrul informatiei in domeniul ingrijirilor de sanatate. Incepand cu anul 1999, au inceput sa fie

publicate o serie de standarde pe baza revizuirii prestandardelor din etapa anterioara. Un aport deosebit in acest proces l-a avut echipa de experti EHRcom Task Force ce lucreaza la revizuirea prestandardului ENV 13606:2000 in vederea elaborarii unui standard (EN) care sa asigure interoperabilitatea cu noile specificatii HL7(Health Level 7).
Amintim standardele CEN/TC 251 relevante pentru domeniul EHR:

-**EN 13606** – Electronic healthcare communication - se bazeaza pe specificatiile proiectului openEHR, fiind structurat pe cinci capitole:
  -EN 13606 - 1 Model de referinta – adoptat in 2007
  -prEN 13606 - 2 Arhetipuri
  -prEN 13606 - 3 Arhetipuri de referinta si Terminologie
  -prEN 13606 - 4 Cerinte de securitate si reguli de distributie a datelor
  -prEN 13606 - 5 Modele pentru schimbul de date
-**EN 1068:2005** - Inventarierea sistemelor de codificare
-**CEN/TS 14796:2005 –** Specificatie tehnica privind tipurile de date
-**prEN 12967:2006 -** Arhitectura Sistemelor Informatice Medicale (HISA)
-**CEN/TS 14463:2003** - Sintaxa pentru reprezentarea continutului sistemelor medicale de clasificare (CIaML).
-**EN 12052:2004-** Health informatics-Digital imaging- Communication, workflow and data management
-**EN 12251:2004** - Secure User Identification for Health Care - Management and Security of Authentication by Passwords
-**EN 12264:2005** - Categorial structures for systems of concepts
-**EN 12381:2005 -** Time standards for healthcare specific problems
-**EN 14822-1:2005** - General purpose information components. Overview
-**EN 14822-2:2005** - General purpose information components. Non-clinical
-**EN 14822-3:2005 -** General purpose information components. Clinical
-**EN 1064:2005 -** Standard communication protocol - Computer-assisted electrocardiography

Unele dintre aceste standarde au fost deja adoptate ca standarde romanesti, altele vor deveni treptat obligatorii, urmare a integrarii Romaniei in UE.
In Romania, ASRO - Asociatia de Standardizare din Romania – constituita ca organism national de standardizare in baza prevederilor OG 39/98 si a Legii nr.355/2002, este recunoscuta ca organism national de standardizare prin HG 985/2004". ASRO este membru cu drepturi depline CEN si CENELEC - Comitetul European pentru Standardizare in domeniul Electrotehnicii; este membru observator al ETSI - Institutul European de Standardizare in domeniul Telecomunicatiilor si membru al ISO - Organizatia Internationala de Standardizare si CEI - Comisia Electrotehnica Internationala. In cadrul ASRO a fost constituit comitetul tehnic **TC 319** pentru standardizarea in domeniul informaticii medicale.

### 2.2.2 Modele conceptuale de sisteme medicale

#### 2.2.2.1 Modelul European - HCIS-Healthcare Information System Architecture.

**Arhitectura Sistemului Informatic pentru Sanatate.**

Acest model arhitectural a fost preluat in Romania prin translatarea standardelor Europene, cu plecare de la versiunea in limba engleza a prestandardului european ENV

00251003 / 1996 **HCIS-Arch - Healthcare Information System Architecture**, elaborat de echipa de lucru PT1-013 a CEN/TC 251 sub mandat BC-IT 205.

“Structura organizationala a sanatatii consta in toate tarile europene, dintr-o retea de centre distribuite in teritoriu, caracterizate printr-un inalt grad de eterogenitate si diversitate, din perspective organizationale, logistice, clinice si chiar culturale. Structura centrelor, luate individual, se desfasoara de la o organizare agregata, pe verticala, spre integrarea unui set de zone functionale specializate, cu nevoi si caracteristici specifice. O astfel de situatie determina doua necesitati principale, intr-un anume fel conflictuale. Pe de o parte, este necesar sa se sustina efectiv cerintele specifice ale fiecarei unitati sau utilizator in cea mai adecvata si eficienta modalitate, iar pe de alta parte este vital sa se asigure consistenta si integrarea intregii organizatii, atat la nivel local cat si teritorial.” [HISA]

“Arhitectura este proiectata ca baza atat pentru compararea, evolutia si integrarea sistemelor existente cat si pentru planificarea si proiectarea la nivelul superior a unor noi sisteme deschise si modulare, capabile sa ofere un suport consistent si integrat pentru cerintele clinice, organizationale si manageriale ale organizatiilor de sanatate.“[ HISA]

“Acest Standard European stabileste principiile generale pentru Arhitectura Sistemului Informatic pentru Sanatate si caracteristicile unui set de baza de servicii utilizate in cadrul sistemelor informatice pentru sanatate, care sunt necesare pentru a sustine majoritatea cerintelor legate de tratamentul subiectului ingrijirii sanatatii.

Acest Standard European este independent de orice mediu tehnologic specific si nu implica adoptarea directa sau indirecta a unor solutii specifice organizationale, de proiectare sau implementare.

Standardul European specifica caracteristicile unui numar de Componente Comune legate de Ingrijirea Sanatatii prin identificarea serviciilor asigurate de componentele selectate din setul de baza de servicii. Nu este abordata structura interna a fiecarei componente. Specificarea componentelor se afla la nivelul conceptual care descrie clesele principale de servicii asigurate si care specifica structura datelor intretinute si regasite de componenta prin serviciile identificate.

Standardul European specifica, in cadrul contextului Componente Comune legate de Ingrijirea Sanatatii, comportarea externa a fiecarui serviciu in raport cu functia sa si informatiile pe care le pune la dispozitia sistemului.

Standardul European este aplicabil sistemelor informatice pentru orice tip de organizatie de sanatate.

Standardul European este aplicabil numai pentru acele aspecte legate direct de subiectii individuali ai ingrijirii. Sunt excluse aspectele de management si administrare ale infrastructurii de sanatate.” [HISA]

Modelarea sistemului Cardionet in ansamblu precum si a portalului au tinut cont de recomandarile si tendintele prezentate si de acest act normativ. Sunt prezentate in continuare cateva elemente din reglementarile ISO conexe domeniului “healthcare”:

| | | | |
|---|---|---|---|
| ISO | 1087 | 1990 | Vocabularul terminologiei |
| ISO | 10241 | 1992 | Standarde ale terminologiei internationale - pregatire si macheta |
| ENV | 12017 | 1995 | Vocabularul informaticii medicale |

-ISO/IEC 11179 Metadata Registries, Parts 1-6:

-Framework

-Classification

-Registry metamodel and basic attributes

-Formulation of data definitions
-Naming & identification principles
-Registration

-ISO/IEC TR 20943-1, Procedures for achieving MDR content consistency-Data elements published July, 2003. Available free at: www.jtc1.org

-ISO/IEC TR 20943-3, Procedures for achieving MDR content consistency- Value Domains published in March, 2004.

-ISO/IEC 20944 series – API's and other interfaces

Iata cateva definitii propuse prin aceste reglementari:

**" aplicatie:**

set de componente ale unui sistem informatic general, capabil sa asigure suport complet si consistent cerintelor si activitatilor unei zone functionale sau unitati individuale din organizatia ce foloseste sistemul informatic.

**componenta:**

principala structura a arhitecturii reprezentand orice tip de element al Sistemului Informatic, de sine statator si identificabil in mod evident, capabil de autonomie in interactiunea cu celelalte componente.

**serviciu:**

functie asigurata de o componenta celorlalte componente din Sistemul Informatic pentru Sanatate.

Fiecare serviciu este invocat de o componenta printr-un mecanism formal, documentat conform unei sinteze formale neambigue, depinzand de limbajele de programare suportate.

**artefact software:**

set self-consistent de programe executabile pe calculator, identificabil printr-un criteriu neambiguu definit de creatorul sau utilizatorul sau.

Nu include echipamentul pe care rezida.

**simboluri si abrevieri:**

Structura datelor intretinute si regasite de fiecare componenta comuna legata de ingrijirea sanatatii este descrisa la nivel conceptual, prin utilizarea urmatoarelor obiecte:

-entitate;
-relatie;
-ierarhie;
-subset;
-atribut.

Aceste meta-tipuri de obiecte informationale sunt definite mai jos, impreuna cu notatiile lor grafice acolo unde se aplica.

Pentru acest Standard European referintele la entitati sunt in majuscule. Pentru a imbunatati lizibilitatea in text numele de entitati sunt adesea folosite la plural.

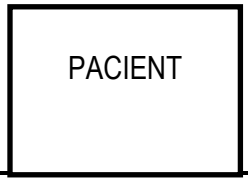
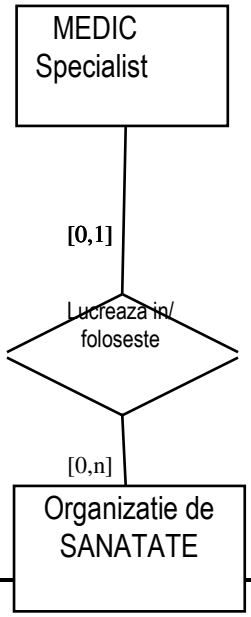
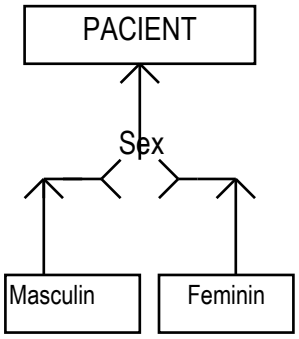
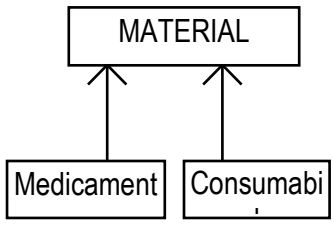
| | |
|---|---|
| ENTITATE:<br>o entitate este orice obiect autonom al domeniului de interes cu un set de proprietati comune numite atribute.<br><br>O entitate se reprezinta grafic printr-un dreptunghi. |  |
| RELATIE:<br>o relatie este o activitate care conecteaza doua entitati. O relatie este reprezentata grafic printr-un romb, conectat la entitatile implicate.<br>O relatie este intotdeauna bidirectionala, ea definind o legatura reciproca intre conceptele conectate. Din acest motiv rombul contine doua nume, specificand intelesul relatiei in raport cu fiecare entitate.<br><br>Privita din oricare parte relatia are o cardinalitate, care specifica numarul minim si maxim de aparitii ale relatiei. In diagrama cardinalitatile sunt scrise chiar langa linia ce conecteaza entitatea cu relatia (alaturi de relatie).<br>Relatiile pot avea proprietati specifice, ce se leaga de faptul insusi si nu depind de entitatile conectate prin relatie. Aceste proprietati se numesc *atribute*. |  |
| IERARHIE:<br>o ierarhie clasifica entitatlile introducand o separare a lor pe baza unui/unor atribut/e.<br>Fiecare instanta a entitatii apartine unei singure sub-entitati. Figura alaturata prezinta o clasificare a pacientilor dupa atributul *sex*: un pacient este fie din sub-entitatea barbatesc, fie din femeiesc.<br>Clasificarea ierarhie este descrisa grafic printr-un set de sageti ce leaga sub-entitatile de entitatea de origine.<br>Atributul de diferentiere este scris pe linia ce conecteaza toate sub-entitatile. |  |
| SUBSET:<br>un subset creeaza o clasificare intr-o entitate, identificand un grup de instante cu proprietati comune, dar fara a determina o separare in entitate. Figura alaturata prezinta doua subseturi: Medicament si Consumabil, ce reprezinta doua grupuri din entitatea: Materiale. Evident, un material consumabil poate fi si un medicament.<br>Din punct de vedere grafic, un subset este reprezentat printr-o<br>sageata conectand sub-entitatea cu entitatea. |  |

Diagrama grafica pentru fiecare componenta ilustreaza, conceptual, informatiile gestionate, regasite si referite de componenta. Tipurile de obiecte informatice prezentate in aceste diagrame prin linii punctate  sunt doar regasite sau referite de componenta. ” [HISA]
Prezentam de asemenea in fig II.3, o reprezentare pentru: modele de obiecte si clase HISA:

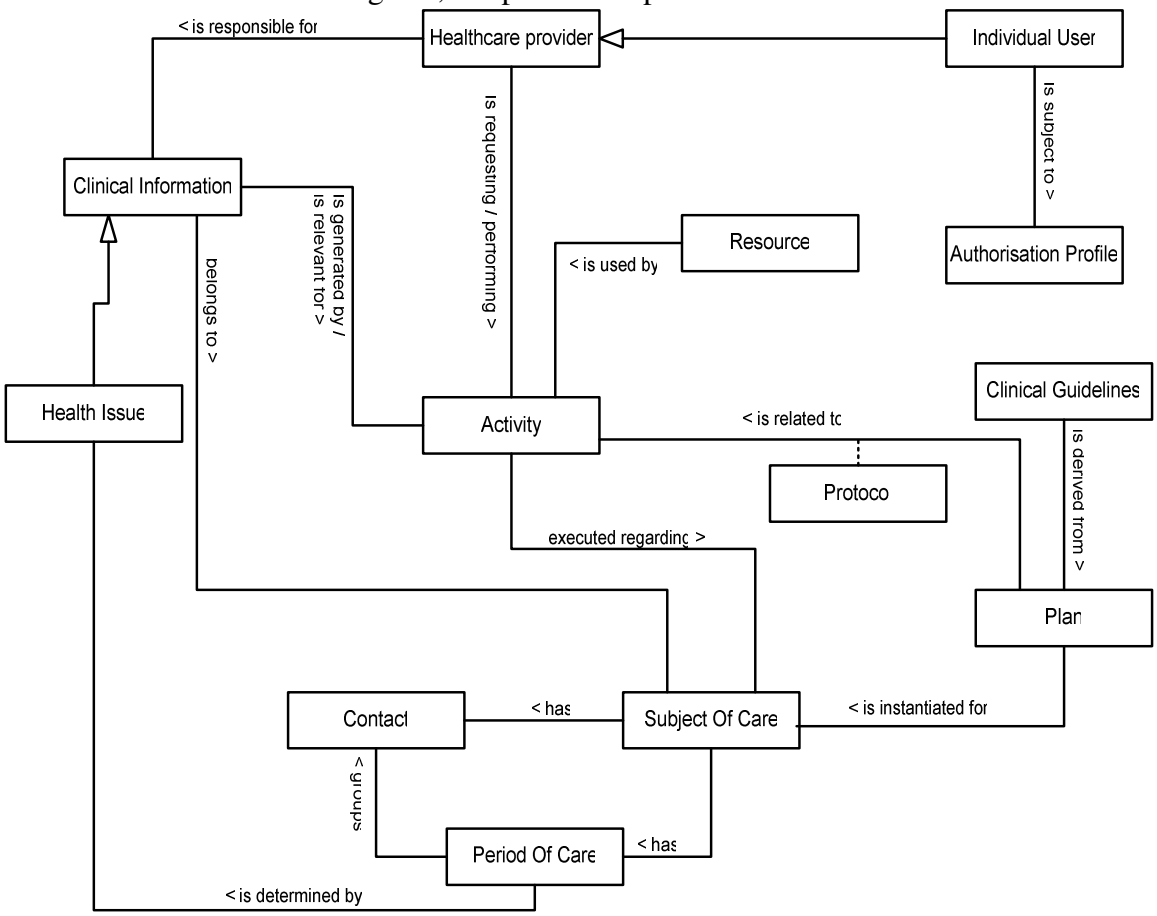



Figura II.3. High level model of the information objects @-[HISA]
[Health informatics — Service architecture - Part 2: Information viewpoint; **ISO/TC 215** Date:  2007-02 **12967-2**  - Secretariat:  ANSI]

HISA- Diagrama UML pentru clasele semnificative ale modelului HISA este prezentata in figura II.4:

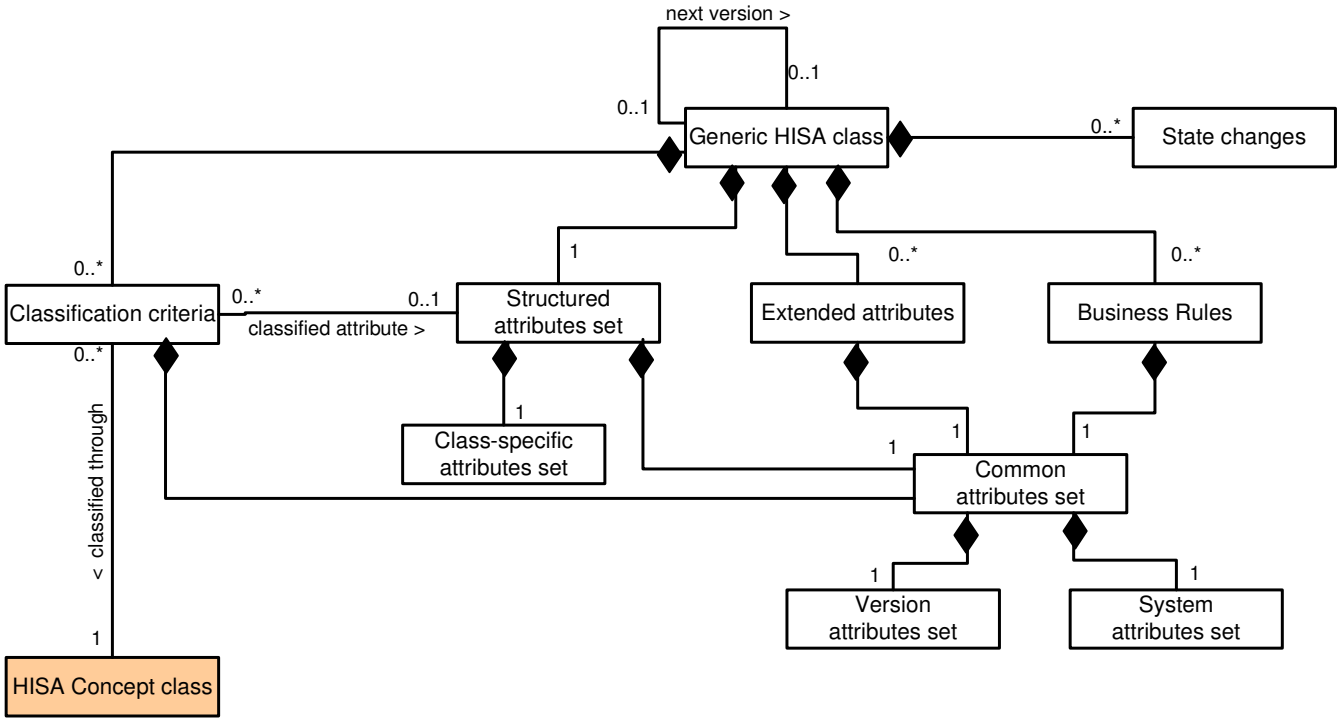



Figura II.4. Clasele generice HISA @-[HISA]

[Health informatics — Service architecture - Part 2: Information viewpoint; **ISO/TC 215** Date: 2007-02 **12967-2** - Secretariat: ANSI]

Modelul HISA are multe aspecte de formalizae valoroase care trebuie analizate si incluse in orice model modern de sistem distribuit de sanatate cu reale “aptitudini” de interoperare cu alte sisteme.

### 2.2.2.2 Modelul interoperabil HL7 – RIM

Elaborarea unor astfel de modele are rolul de a abstractiza cat mai mult posibil solutia propusa pentru un domeniu, aplicat in cazul nostru la “healthcare”. Modelul HL7-RIM, figura II.5, asa cum a mai fost prezentat si in etapele anterioare ale proiectului Cardionet, face parte din clasa “Model-Driven Development”, are un nivel ridicat de generalitate si impreuna cu modelul HISA au stat la baza modelarii solutiei de colaborare intre actorii care vor aplica modelul Cardionet.

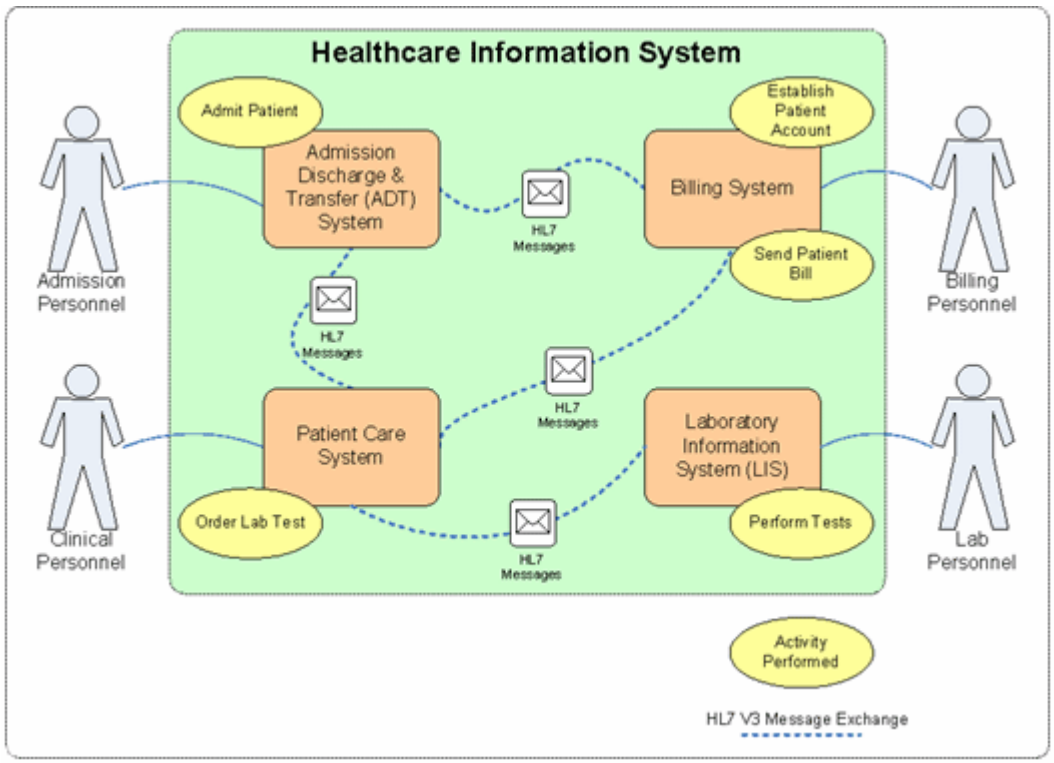



Figura II.5. Model cadru de colaborare intre parteneri, pe baza de mesaje HL7 @-[HL7-SF]
**[HL7-SF]- Microsoft & Blueprint Technologies [BPT] - *Health Level 7 (HL7) Software Factory*: a vision for a software factory in the context of Health Level Seven Version 3 (V3).**

O reprezentare grafica, la un alt nivel de formalizare a grupurilor principale de clase ale modelului HL7-RIM (compus din circa 65 clase) este data in figura II.6.:

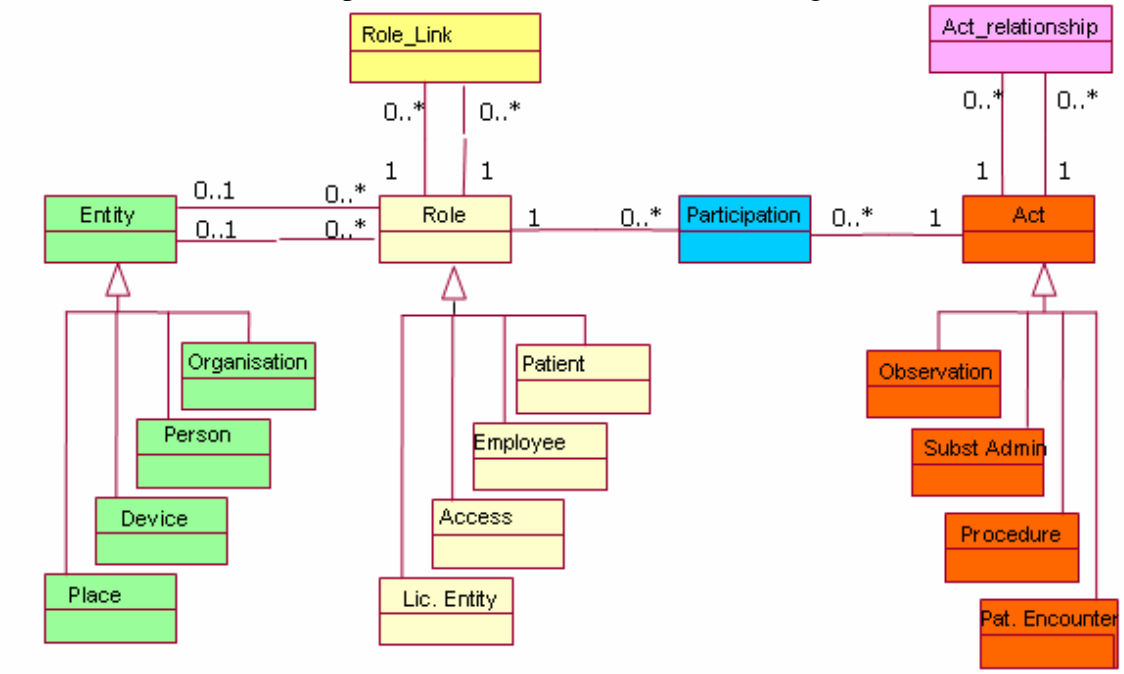



Figura II.6.: Modelul HL7 – RIM : clasele principale @-[HL7-RIM]
Descrierea tuturor acestor clase si a modelului in sine depaseste cadrul acestei lucrari, dar au stat la baza elaborarii structurilor de date Cardionet.

Pe langa aceste modele adoptate pentru a asigura un nivel ridicat de interoperabilitate trebuie tinut cont si de oportunitatile tot mai mari oferite de posibititatile tehnice sporite de comunicatie si tehnologiile de tip e-Health.

In data de 12 Februarie 2009, la Brussels, la Centrul de Politici Europene, au fost trasate principalele tendinte in Politica Europeana de Sanatate Publica pentru 2009.
Cerinte de importanta majora in sanatate constitue printre altele si urmatoarele preocupari:
1. Cresterea calitatii vietii si crestere sperantei de viata

-vizeaza prevenirea si controlul bolilor in faza incipienta. UE va aloca un buget majorat in directia prevenirii chiar si in contextul crizei economice , deoarece costurile de prevenire sunt in general mult mai mici fata de cele de tratament.

2. Managementul pandemiilor

- vizeaza detectarea si prevenirea epidemiilor ( HIV, flu,...)

3. Suportul dinamic in sistemele de sanatate prin noile tehnologii

- vizeaza in general imbunatatirea calitatii vietii.

Si in Romania exista preocupari in aceste domenii. Astfel, in luna mai 26-27, a avut loc la Bucuresti, la Hotel Hilton, editia a IV- a E-HEALTH CONGRESS 2009, cu tematica: *"Solutii pentru imbunatatirea calitatii, sigurantei si eficientei in practica medicala"*

Evenimentul din acest an a reunit speakeri de top din Europa reprezentand provideri care pot oferi solutii concrete pentru imbunatirea sistemelor de sanatate conform celor mai actuale tendinte eHealth. Congresul a fost focalizat pe E-Health si telemedicina, prezentand o gama larga de produse si componente IT&C corelate cu aplicatii medicale care conduc la imbunatatirea actului medical, reducerea costurilor, eficientizarea ingrijirii pacientilor.

## 2.3Cadrul local, reglementari MSP- Ministerul Sanatatii Publice si legislatie nationala

La proiectarea structurilor de date ale portalului Cardionet am tinut cont de: cerintele sistemului de sanatate din Romania, reglementate printr-o serie de acte normative ale Ministerului Sanatatii Publice, de standardele internationale din domeniu sanatatii si eHealth, de modelele arhitecturale propuse spre standardizare si in special de cerintele nationale si internationale de interoperabilitate.

### 2.3.1 Reglementarile nationale specifice domeniului sanatatii

La nivelul anului 2008 in Romania, aceste reglementari au fost stabilite prin:

„-ORDIN NR. 99/12.02.2008 privind aprobarea Regulilor de validare a cazurilor spitalizate in regim de spitalizare continua si a Metodologiei de evaluare a cazurilor invalidate pentru care se solicita revalidarea.

-ORDIN NR. 949 privind modificarea Ordinului presedintelui Casei Nationale de Asigurari de Sanatate nr. 26/2007 pentru aprobarea utilizarii formularelor unice pe tara, fara regim special, necesare raportarii activitatii furnizorilor de servicii medicale

-ORDIN nr. 2144 privind modificarea si completarea Ordinului ministrului sanatatii publice si al presedintelui Casei Nationale de Asigurari de Sanatate nr. 1.781/CV 558/2006 pentru aprobarea Normelor metodologice de aplicare a Contractului-cadru privind conditiile acordarii asistentei medicale in cadrul sistemului de asigurari sociale de sanatate pentru anul 2007

-HOTARARE nr. 1534 privind modificarea si completarea Hotararii Guvernului nr. 1.842/2006 pentru aprobarea Contractului-cadru privind conditiile acordarii asistentei medicale in cadrul sistemului de asigurari sociale de sanatate pentru anul 2007

-ORDIN nr. 2017/918 din 28.11.2007, privind modificarea si completarea Ordinului ministrului sanatatii publice si al presedintelui Casei Nationale de Asigurari de Sanatate nr. 1781/CV 558 558/2006 pentru aprobarea Normelor

metodologice de aplicare a Contractului-cadru privind conditiile acordarii asistentei medicale in cadrul sistemului de asigurari sociale de sanatate pentru anul 2007
-ORDIN nr. 717 din 7 septembrie 2007 privind modificarea si completarea Ordinului ministrului sanatatii publice si al presedintelui Casei Nationale de Asigurari de Sanatate nr. 1.781/CV 558/2006 pentru aprobarea Normelor metodologice de aplicare a Contractuluicadru privind conditiile acordarii asistentei medicale in cadrul sistemului de asigurari sociale de sanatate pentru anul 2007
-ORDIN nr. 1166 din 2 iulie 2007 privind modificarea si completarea Ordinului ministrului sanatatii publice si ai presedintelui Casei Nationale de Asigurari de Sanatate nr. 1.781/CV558/2006 pentru aprobarea Normelor metodologice de aplicare a Contractului-cadru privind conditiile acordarii asistentei medicale in cadrul sistemului de asigurari sociale de sanatate pentru anul 2007
-Ordinul MSP 1081 / CNAS 639 pentru modificarea si completarea Ordinului MSP si al presedintelui CNAS nr.1782/576/2006 privind inregistrarea si raportarea statistica a paci- entilor care primesc servicii medicale in regim de spitalizare continua si spitalizare de zi
-Ordinul Ministerului Sanatatii Publice 1080 - 20.06.2007 privind introducerea si utilizarea Clasificarii internationale si a bolilor de sanatate inrudite, Revizia 10, Modificarea Australiana
-Ordinul CNAS 208/2007 pentru completarea Ordinului nr. 37/2007 privind aprobarea Regulilor de validare a cazurilor spitalizate in regim de spitalizare continua si a Metodologiei de evaluare a cazurilor invalidate pentru care se solicita revalidarea
-Ordinul CNAS 106/2007 pentru aprobarea utilzarii aplicatiei de colectare a setului minim de date la nivel de pacient in regim de spitalizare de zi - SpitalizareZi
-Ordinul CNAS 26/2007 pentru aprobarea utilizarii formularelor unice speciale, fara regim special, necesare raportarii activitatii furnizorilor de servicii medicale
-Ordinul CNAS 37/2007 privind aprobarea Regulilor de validare a cazurilor spitalizate in regim de spitalizare continua si a Metodologiei de evaluare a cazurilor invalidate pentru care se solicita revalidarea
-Ordinul MSF 1782/2006 privind inregistrarea si raportarea statistica a pacientilor care primesc servicii medicale in regim de spitalizare continua si spitalizare de zi (Anexa 1: Nomenclatorul investigatiilor de laborator; Anexa 2: FOCG; Anexa 3: Instructiuni FOCG; Anexa 4: FSZ; Anexa 5: Instructiuni FSZ; Anexa 6: SMDPC; Anexa 7: SMDPZ)
-Ordinul MSF 1781/2006 pentru aprobarea normelor metodologice de aplicare a contractului cadru privind conditiile acordarii asistentei medicale in cadrul sistemului de asigurari sociale de sanatate pentru anul 2007
-HG 1842 pentru aprobarea Contractului-cadru privind conditiile acordarii asistentei medicale in cadrul sistemului de asigurari sociale de sanatate pentru anul 2007
-HG 1841 pentru aprobarea Listei cuprinzand denumirile comune internationale corespunzatoare medicamentelor de care beneficiaza asiguratii in tratamentul ambulatoriu, cu sau fara contributie personala, pe baza de prescriptie medicala, in sistemul de asigurari sociale de sanatate."
Surse de date: [7],[8].

### 2.3.2 Seturi minime de date obligatoriu de inregistrat in format electronic

Colectarea electronica a datelor despre pacienti a fost reglementata de MSP prin: "ORDIN nr. 29 din 20 ianuarie 2003 privind introducerea colectarii electronice a Setului minim de date la nivel de pacient (SMDP) în spitalele din Romania.
Avand in vedere:
*- prevederile Ordinului ministrului sanatatii si familiei si al presedintelui Casei Nationale de Asigurari de Sanatate nr.102/34/2002 privind aprobarea strategiei pentru implementarea finantarii bazate pe caz în spitalele din Romania;*
*- prevederile Ordinului ministrului sanatatii si familiei si al presedintelui Casei Nationale de Asigurari de Sanatate nr.1.021/113/2002 privind aprobarea Planului actiunilor majore pentru implementarea finantarii bazate pe caz în spitalele din Romania, în perioada 2002-2003;*
*- prevederile Ordinului ministrului sanatatii si familiei nr. 798/2002 privind introducerea în spitale a formularului "Foaia de observatie clinica generala",*
*vazand Referatul de aprobare al Centrului de Calcul, Statistica Sanitara si Documentare Medicala Bucuresti si al Institutului National de Cercetare-Dezvoltare în Sanatate Bucuresti nr. DB 396 din 20 ianuarie 2003,*

*în temeiul Hotararii Guvernului nr.22/2001 privind organizarea si functionarea Ministerului Sanatatii si Familiei, cu modificarile si completarile ulterioare,* ***Ministrul Sanatatii siFamiliei (MSF)*** *emite urmatorul ordin:*
*Art. 1 Începand cu data de 1 ianuarie 2003 se introduce colectarea electronica a datelor clinice la nivel de pacient în toate spitalele din Romania.*
*Art. 2 Datele clinice la nivel de pacient se culeg din foaia de observatie clinica generala, introdusa în sistemul informational al spitalului prin Ordinul ministrului sanatatii si familiei nr. 798/2002 si formeaza Setul minim de date la nivel de pacient, ...„*

Seturile minime de date ce trebuie colectate, in Romania, in format electronic, conform MSP sunt:

### 2.3.2.1 Setul minim de date la nivel de pacient

"Ordin MSF 798/2002: Anexa nr. 1:
Datele din foaia de observatie clinica generala care formeaza Setul minim de date la nivel de pacient (SMDP)
1. Numarul de identificare a cazului externat
2. Numarul de identificare a spitalului
3. Numarul de identificare a sectiei
4. Numarul foii de observatie clinica generala
5. Numele pacientului
6. Prenumele pacientului
7. Localitatea
8. Judetul
9. Data nasterii
10. Sexul
11. Codul numeric personal
12. Tipul asigurarii de sanatate
13. Casa de asigurari de sanatate
14. Data internarii
15. Ora internarii
16. Tipul internarii
17. Diagnostic la internare
18. Data externarii
19. Ora externarii
20. Tipul externarii
21. Starea la externare
22. Diagnosticul principal si diagnosticele secundare, la externare
23. Interventiile chirurgicale efectuate
24. Data interventiei chirurgicale principale
25. Explorarile functionale efectuate
26. Investigatiile radiologice efectuate
27. Greutatea la nastere (doar pentru nou-nascuti)
28. Transferurile intraspitalicesti (sectia; data)
29. Medicul curant
30. Medicul operator."

### 2.3.2.2 Setul minim de date la nivel de pacient pentru spitalizarea continua

” Ordin MSF 1782/2006: Anexa nr. 6:
Datele din foaia de observatie clinica generala care formeaza Setul minim de date la nivel de pacient pentru spitalizarea continua (SMDPC):
1. Codul de identificare a cazului externat
2. Codul de identificare a spitalului
3. Codul de identificare a sectiei
4. Numarul foii de observatie clinica generala
5. Codul numeric personal al pacientului
6. Numele pacientului
7. Prenumele pacientului
8. Sexul
9. Data nasterii
10. Judetul de domiciliu al pacientului
11. Localitatea de domiciliu a pacientului
12. Cetatenia
13. Greutatea la nastere (doar pentru nou-nascuti)
14. Ocupatia
15. Nivelul de instruire
16. Statut asigurat
17. Tipul asigurarii de sanatate
18. Casa de asigurari de sanatate
19. Categoria de asigurat
20. Tipul internarii
21. Criteriul de internare
22. Data internarii
23. Ora internarii
24. Diagnosticul la internare
25. Data externarii
26. Ora externarii
27. Tipul externarii
28. Starea la externare
29. Diagnosticul principal si diagnosticele secundare la externare
30. Interventia chirurgicala principala efectuata
31. Data interventiei chirurgicale principale
32. Alte proceduri: interventii chirurgicale, explorari functionale, investigatii radiologice etc., precum si numarul acestora
33. Transferurile intraspitalicesti (sectia; data)
34. Codul de parafa al medicului curant
35. Codul de parafa al medicului operator
36. Situatiile speciale (suspiciune de accident de munca, suspiciune de boala profesionala, accident rutier, vatamare corporala)”

### 2.3.2.3 Setul minim de date la nivel de pacient pentru spitalizarea de zi

„Ordin MSF 1782/2006: Anexa nr. 7:

Datele din fisa de spitalizare de zi care formeaza Setul minim de date la nivel de pacient pentru spitalizarea de zi (SMDPZ):
1. Codul de identificare a cazului externat
2. Codul de identificare a spitalului
3. Codul de identificare a sectiei
4. Numarul fisei de spitalizare de zi
5. Codul numeric personal al pacientului
6. Numele pacientului
7. Prenumele pacientului
8. Sexul
9. Data nasterii
10. Judetul de domiciliu al pacientului
11. Localitatea de domiciliu a pacientului
12. Cetatenia
13. Ocupatia
14. Nivelul de instruire
15. Statut asigurat
16. Tipul asigurarii de sanatate
17. Casa de asigurari de sanatate
18. Categoria de asigurat
19. Data deschiderii fisei
20. Data închiderii fisei
21. Tipul de serviciu de spitalizare de zi
22. Data vizitei
23. Diagnosticul principal si diagnosticele secundare la închiderea fisei
24. Procedurile efectuate
25. Investigatiile de laborator efectuate
26. Codul de parafa al medicului curant"

#### 2.3.2.4 Seturi de date solicitate conform HL7

Standardul HL7 a fost partial prezentat si in documentatia etapelor anterioare. Din motive de spatiu exemplificam totusi si aici doar doua astfel de colectii de date, din totalul colectiilor Hl7:

Coloana "Element Name" din urmatoarele doua tabele ne arata cerintele HL7 pentru codificarea informatiilor pentru "identificare pacient", respectiv pentru reprezentarea datelor despre "Alergii".

Exemplu de Set de date pentru "Identificare Pacient" (HL7 - PID - PATIENT IDENTIFICATION) [www.hl7.org] :

| Seq | HL7 Data Type | Used at DUMC | DUMC Max Len | Tbl# | Element Name |
|---|---|---|---|---|---|
| 1 | SI | N | 0 | | Set ID - Patient ID |
| 2 | CK | Y | 40* | | Patient ID (External ID) |
| 3 | CM | Y | 10* | | Patient ID (Internal ID) |
| 4 | ST | N | 0 | | Alternate Patient ID |
| 5 | PN* | Y | 25* | | Patient Name |
| 6 | ST | N | 0 | | Mother's Maiden Name |
| 7 | TS | Y | 8* | | Date of birth |

| | | | | | |
|---|---|---|---|---|---|
| 8 | ID | Y | 1 | 0001 | Sex |
| 9 | ST* | Y | 16* | | Patient Alias |
| 10 | ID | Y | 1 | 0005 | Race |
| 11 | AD | Y | 105* | CTRY | Patient Address |
| 12 | ID | Y | 3* | CNTY | County Code |
| 13 | TN | Y | 13* | | Phone Number - Home |
| 14 | TN | Y | 19* | | Phone Number - Business |
| 15 | ST | N | 0 | | Language - Patient |
| 16 | ID | Y | 1 | 0002 | Marital Status |
| 17 | ID | Y | 3 | 0006 | Religion |
| 18 | CK | Y | 8* | | Patient Account Number |
| 19 | ST | Y | 9* | | SSN - Patient |
| 20 | CM | N | 0 | | Driver's License Number |
| 21 | CK | Y | 10* | | Mother's Indicator |
| 22 | ID | N | 0 | | Ethnic Group |
| 23 | ST | N | 0 | | Birth Place |
| 24 | ID | N | 0 | | Multiple Birth Indicator |
| 25 | NM | N | 0 | | Birth Order |
| 26 | ID | N | 0 | | Citizenship |
| 27 | CE | N | 0 | | Veterans Military Status |

Exemplu de Set de date pentru reprezentarea datelor despre alergii(HL7-AL1 - ALLERGY INFORMATION) [www.hl7.org]

| Seq | HL7 Data Type | Used at DUMC | DUMC Max Len | Tbl# | Element Name |
|---|---|---|---|---|---|
| 1 | SI | N | 0 | | Set ID |
| 2 | ID | *F* | 2 | 0127 | Allergy Type |
| 3 | CE | Y | 60 | | Allergy Code/Mnemonic/Descrip |
| 4 | ID | *F* | 2 | 0128 | Allergy Severity |
| 5 | ST | *F* | 0 | | Allergy Reaction |
| 6 | DT | *F* | 0 | | Identification Date |

***** indicates variance from HL7 Standard.***

***'F' in the "Used at DUMC" column indicates that the field is not currently in use, but is planned in the future.***

La proiectarea modelului de date Cardionet am tinut cont: de cerintele nationale reglementate de MSP, de cerintele privind seturile de date necesare in EHR din standarde internationale (HL7, OpenEHR), precum si de modelul de sistem propus de ESC (Societatea Europeana de Cardiologie) prezentat in figura II.7., pentru a avea in final un model de sistem modern bazat pe o colectie de date capabila sa asigure o larga interoperabilitate cu alte sisteme existente sau viitoare.

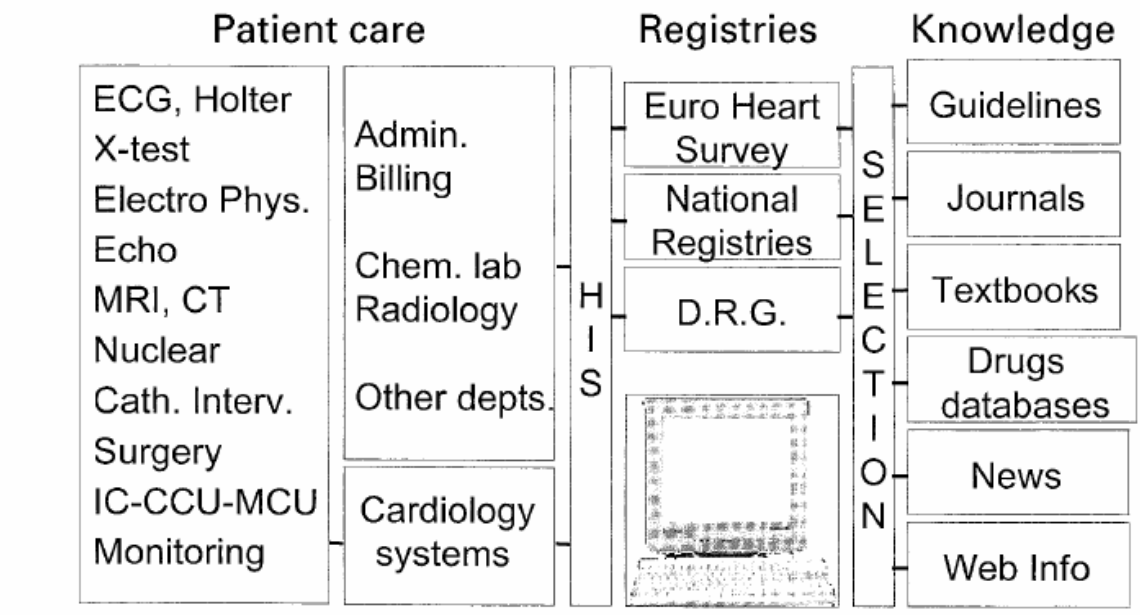



Figura II.7.: Model de sistem informatic pentru cardiologie @[ESC-EHR]
[ *European Heart Journal* (2002) **23,** 1148–1152 doi:10.1053/euhj.2002.3269, available online at http://www.idealibrary.com on : „The Cardiology Information System: the need for data standards for integration of systems for patient care, registries and guidelines for clinical practice"]

## 2.4Cardionet- Implementare Foaie de Observatii Clinice Generale "FOCG"

Reuniunea seturilor de date studiate, coroborata cu reglementarile internationale si cu practicile IT existente in domeniu au stat la baza structurilor de date ale bazelor operationale si ale bazelor si depozitelor portalului Cardionet.

Asa cum s-a aratat si in paragrafele anterioare, obligativitatea introducerii FOCG (Foaie de Observatii Clinice Generale) in format electronic a fost reglementata prin ordinul MSP 798/2002 si a completarilor aduse ulterior. (ex. OMSF Nr. 88/2004).

Asa cum am mai aratat si in paragrafele anterioare, MSP [8], reglementeaza in continuare si prin odinul (figura II.8):

**MINISTERUL SĂNĂTĂŢII PUBLICE**
Nr. 1782 /28.12.2006

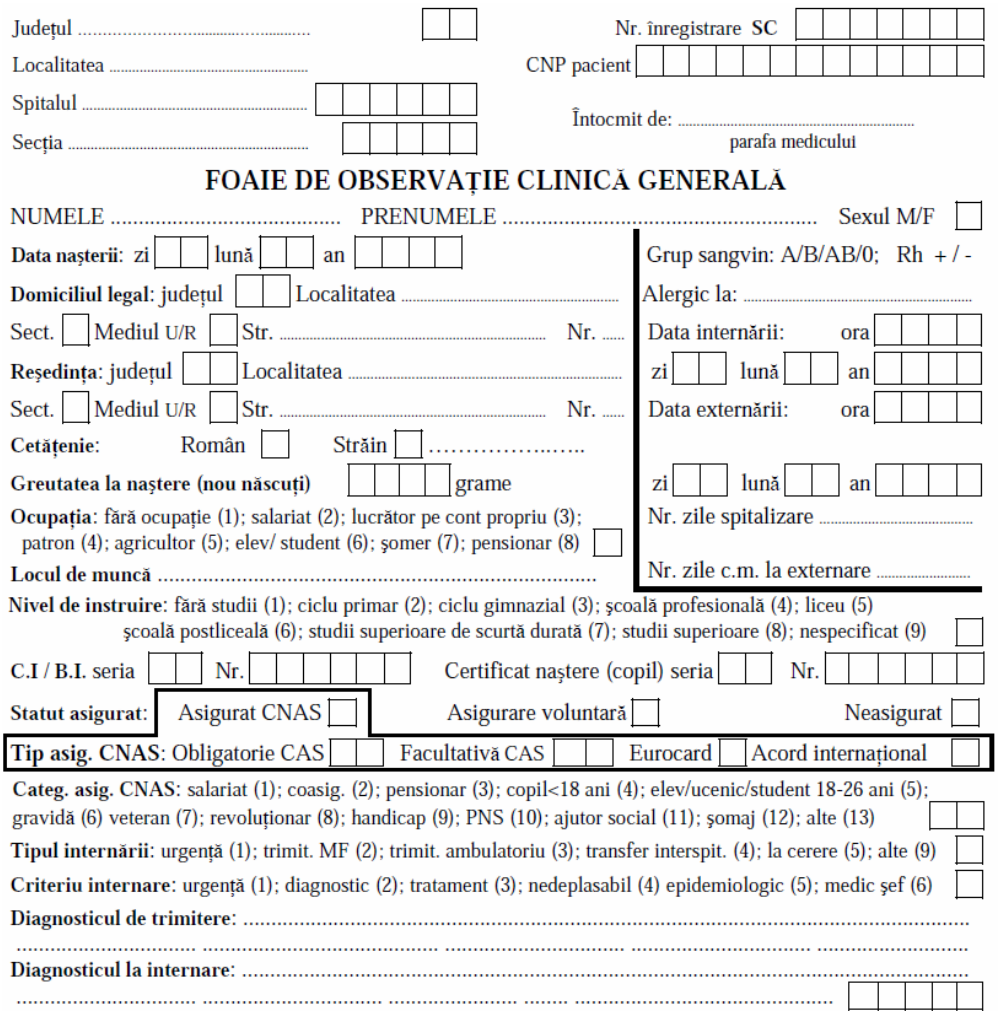
**CASA NAŢIONALĂ DE ASIGURĂRI DE SĂNĂTATE**
Nr. 576 /28.12.2006

**ORDIN**

**privind înregistrarea şi raportarea statistică a pacienţilor care primesc servicii medicale în regim de spitalizare continuă şi spitalizare de zi**

Figura II.8. Ordin MSP, inregistrari electronice de date despre pacienti

seturile minime de date si formatul general pentru documentul "Foaie de Observatie Clinica Generala", dupa care se vor face inregistrarile si raportarile de date medicale, in unitatile din sistemul sanitar romanesc. O parte din prima pagina (din 8 pagini) se poate vedea in figura II.9. reprezentata mai jos:



Județul ..... Nr. înregistrare SC

Localitatea ..... CNP pacient

Spitalul .....

Secția ..... Întocmit de: .....
parafa medicului

**FOAIE DE OBSERVAȚIE CLINICĂ GENERALĂ**

NUMELE ..... PRENUMELE ..... Sexul M/F

**Data nașterii**: zi lună an

Grup sangvin: A/B/AB/0; Rh + / -

**Domiciliul legal**: județul Localitatea .....

Alergic la: .....

Sect. Mediul U/R Str. ..... Nr. .....

Data internării: ora

**Reședința**: județul Localitatea .....

zi lună an

Sect. Mediul U/R Str. ..... Nr. .....

Data externării: ora

**Cetățenie**: Român Străin .....

**Greutatea la naștere (nou născuți)** grame

zi lună an

**Ocupația**: fără ocupație (1); salariat (2); lucrător pe cont propriu (3); patron (4); agricultor (5); elev/ student (6); șomer (7); pensionar (8)

Nr. zile spitalizare .....

**Locul de muncă** .....

Nr. zile c.m. la externare .....

**Nivel de instruire**: fără studii (1); ciclu primar (2); ciclu gimnazial (3); școală profesională (4); liceu (5) școală postliceală (6); studii superioare de scurtă durată (7); studii superioare (8); nespecificat (9)

**C.I / B.I.** seria Nr. Certificat naștere (copil) seria Nr.

**Statut asigurat**: Asigurat CNAS Asigurare voluntară Neasigurat

**Tip asig. CNAS**: Obligatorie CAS Facultativă CAS Eurocard Acord internațional

**Categ. asig. CNAS**: salariat (1); coasig. (2); pensionar (3); copil<18 ani (4); elev/ucenic/student 18-26 ani (5); gravidă (6) veteran (7); revoluționar (8); handicap (9); PNS (10); ajutor social (11); șomaj (12); alte (13)

**Tipul internării**: urgență (1); trimit. MF (2); trimit. ambulatoriu (3); transfer interspit. (4); la cerere (5); alte (9)

**Criteriu internare**: urgență (1); diagnostic (2); tratament (3); nedeplasabil (4) epidemiologic (5); medic șef (6)

**Diagnosticul de trimitere**: .....

**Diagnosticul la internare**: .....

Figura II.9. FOCG- pagina 1

In sistemul Cardionet exista structuri de date care vor putea inregistra toate informatiile solicitate prin aceste reglementari ale MSP [8], grupate in date generale ale FOCG si date specifice sub-domeniului cardiovascular abordat de proiect. (Introducand ulterior structuri specifice pentru alte sub-domenii medicale sistemul va putea fi extins si pentru alte specialitati medicale.)

In cardiologie realizarea FOCG presupune completarea unui numar destul de mare de date specifice acestui domeniu (modelul de foaie pentru cardiologie este prezentata in capitolul despre portal), cu consecinte atat in gestionarea cat si in analizele ulterioare efectuate asupra datelor. Pe langa datele minimale solicitate in general, colectarea informatiilor în cardiologie trebuie sa cuprinda investigatii caracteristice specialitatatilor cardiologice: cardiologie interventionala, electrofiziologie, coronariana, chirurgie, preventie si recuperare cardiovasculara, cu includerea unei grile de evaluare a riscului cardiovascular.

Transpunerea foii de observatie clinice generale în formatul actual impus de MSP este un obiectiv al acestui proiect, colectarea datelor urmand sa fie realizata in mod unitar in toate componentele sistemului.

S-a realizat proiectarea de specificatii minime pentru realizarea unor subsisteme departamentale pe specialitatile cardiologice selectate pentru proiect. Solutia modulara aleasa va permite extinderea sistemului spre alte specialitati cardiologice(electrofiziologie, cardiologie

interventionala, ecocardiografie, preventie si recuperare cardiovasculara) sau din alte domenii medicale. Preluarea datelor din diverse surse (laborator, investigatii imagistice, sectii clinice, explorari functionale) si necesitatatile de interoperabilitate au impus folosirea de standarde industriale larg acceptate (prezentate si in documentele etapelor anterioare Cardionet):
- DICOM pentru transferul imaginilor medicale si pentru intercomunicatie între diferite echipamente medicale;
- HL7 bazat pe limbajul XML;
- standarde de tip EHR (Asociatia Romana de Evidenta Electronica a datelor medicale).

Alte aspecte de care am tinut cont au fost legate de standardele SIUI [9] impuse de CNAS, Casa Nationala de Sanatate, la sfarsitul anului 2007.
Sistemul Cardionet contine într-o prima varianta componente functionale pentru actorii principali: spitale, medici de familie, laboratoare, unitati administrative, personal medical, utilizatori), cu urmatoarele sarcini functionale:
- gestiune FOCG si documente de internare-externare;
- gestiune si transfer informatii între actorii principali ai proiectului si ulterior intre acestia si alte institutii (ASP, SNSPMS, MSP) ;
- Codificarari specifice in foile de observatie si rapoarte de analiza pe baza datelor colectate;
- componente de codificare si prevalidare DRG [7] care va crea premise pentru schimb de date cu autoritatile de specialitate; (pentru analiza indicatorilor clinici, analiza de finantare si buget pe grupe DRG, calculul costurilor clinice, etc.)

Toate reglementarile si normativele (atat nationale cat si internationale) cat si tendintele actuale, au stat la baza proiectarii modulelor sistemului distribuit Cardionet, a structurilor sale de date precum si a componentelor si depozitelor de date ale portalului asociat, care va fi prezentat in unul din capitolele urmatoare. Structurile de date generale si cele specializate pentru “actorii” participanti la proiect si componentele de prelucrare vor fi prezentate atat in documentele etapei curente cat si ale etapei urmatoare (decembrie 2009) impreuna cu testele de implementare efectuate.

**Referinte bibliografice:**

**Capitolul II:**

[1 -HISA] European Prestandard ENV 12967-1:1998 Medical Informatics – Healthcare Information System Architecture (HISA) – Part 1: Healthcare Middleware Layer. CEN TC/251, 1998.

[2] -Reynolds M., Wejerfeld I. Short Strategic Study – Health Information Infrastructure–Final report. CEN/TC 251/N00-074. CEN Health Informatics TC 251. CEN/TC 251. Sep 2000. http://www.centc251.org

[3 -RM-ODP]: The Reference Model for Open Distributed Processing. http://www.rm-odp.net/.

[4 -HL7-SF]- Microsoft & Blueprint Technologies [BPT] - *Health Level 7 (HL7) Software Factory*: a vision for a software factory in the context of Health Level Seven Version 3 (V3)

[5 –HL7-RIM]- Salford Health Informatics Research Environment (SHIRE) - Report to the Information Standards Advisory Board, 2004

[6 -ESC-EHR] *European Heart Journal* (2002) **23,** 1148–1152: 10.1053/euhj.2002.3269, http://www.idealibrary.com: „The Cardiology Information System: the need for data standards for integration of systems for patient care, registries and guidelines for clinical practice"

[7]http://www.drg.ro/legislatie/

[8]http://www.ms.ro/legislatia-in-vigoare/

[9]. http://www.cnas.ro/in_SIUI/

# 3 Proiectarea şi implementarea ontologiei domeniului

## 3.1 Consideratii generale

Realizarea unui sistem de telemedicina in domeniul cardio-vascular comporta doua aspecte majore: partea de medicina propriu-zisa si cea informatica. Pe partea de medicina trebuie sa se formalizeze toate tipurile de acte medicale in vederea transpunerii lor in forma informatica. Multe proceduri si scheme de tratament care acum au la baza intuitia, experienta sau priceperea medicului trebuie sa capete o forma concreta, riguroasa. Din punct de vedere informatic problemele care se punt sunt cele privind modelarea datelor medicale, stabilirea fluxurilor de date, a schemelor de interactiune dintre utilizatori si sistem si a mijloacelor de comunicatie intre aplicatiile medicale.

Un sistem de telemedicina presupune cooperarea si schimbul de informatii intre componente/aplicatii medicale distribuite geografic, care pot sa difere ca arhitectura, functionalitate si implementare. Diferentele provin din cerintele specifice pentru fiecare entitate medicala (medic de familie, spital, laborator de analize, autoritate sanitara, etc.) pentru care s-a proiectat si implementat cate o aplicatie medicala. Pentru a asigura compatibilitatea si interoperabilitatea dintre aceste aplicatii datele vehiculate trebuie sa aiba acceasi semnificatie si structura. Dar, domeniul medical este complex, informatiile cu care se opereaza pot sa fie foarte diverse si pot fi interpretate diferit. De exemplu o anumita tensiune arteriala masurata si stocata de o aplicatie poate fi considerata o valoare normala pentru o anumita categorie de varsta, sex sau grupa de boala, dar se considera o valoare patologica in alte cazuri. Chiar si in cazul bolilor sau a tratamentelor exista diferente de interpretare.

Construirea unui sistem coerent si complet de telemedicina, orientat catre bolile cardiovasculare presupune in primul rand definirea unui model conceptual bazat pe ontologia domeniului. Ontologia trebuie sa acopere toate conceptele abstracte utilizate intr-un sistem de telemedicina si relatiile dintre acestea.

Principalele categorii de concepte avute in vedere sunt:

- actorii sistemului (pacient, medic de familie, spital, laborator de analiza, autoritati de control sau de finantare, etc.),
- documentele medicale (fisa pacient, fisa de consult, reteta, fisa de internare/externare, foaie de trimitere, et.),
- procedurile de tratament (consult simplu, internare, tratament ambulatoriu, etc.),
- resurse fizice utilizate (echipamente medicale mobile si fixe, echipamente de calcul si de comunicatie),
- componentele program (servere de baze de date, interfete/aplicatii client, componente de comunicatie).

Pe baza practicii medicale curente se stabilesc relatiile si interconditionarile dintre concepte, precum si scenariile de utilizare a sistemului. Un scenariu de utilizare este o formalizare, o descriere riguroasa a interactiunilor dintre componentele sistemului. De exemplu un consult poate sa implice un pacient, un medic de familie si eventual un medic specialist, presupune un anumit flux de documente medicale (fisa pacient, foaie de trimitere, buletin de analize, reteta, etc.), anumite relatii de responsabilitate privind generarea documentelor si drepturi de acces la informatiile medicale.

Trecerea de la un sistem clasic de evidenta pe hartie, la un sistem informatizat impune specificarea si formalizarea tuturor tipurilor de interactiune care pot sa apara intre actorii

sistemului si sistemul informatic propriu-zis. Trebuie de asemenea sa se precizeze documentele ce trebuie generate in fiecare etapa a unei secvente de tratament, continutul acestora, locul si modul de pastrare a informatiilor medicale, si drepturile de acces la acestea.

In urma consultarii cu cadre medicale implicate in sistemul medical romanesc actual s-a tras concluzia ca in multe situatii sistemul actual de evidenta si gestiune din sanatate nu are reguli clare si precise de culegere, stocare si acces la informatii medicale. Regulile utilizate (acolo unde exista) se bazeaza pe niste practici locale specifice fiecarei unitati medicale. De aceea un obiectiv important al proiectului de fata este tocmai acela de a formaliza actul medical, de a introduce reguli clare si unitare in conformitate cu legislatia nationala, cu prevederile unor acorduri internationale si in concordanta cu standardele existente (ex: HL7, CDA, RIM, etc.).

## 3.2 Definirea ontologiei pentru domeniul cardio-vascular

Exista o serie de cercetari recente care au permis conturarea unei ontologii medicale cu aplicabilitate pentru domeniul cardio-vascular [Prcela,2006, 2007], [Davis, 2003]. Rezultatele acestor cercetari au demonstrat viabilitatea abordarii bazata pe ontologie. In cadrul proiectului de fata ne-am inspirat din cateva propuneri de ontologii deschise. Adoptarea unor scheme ontologice deja existente ne va permite sa dezvoltam o platforma medicala deschisa compatibila cu alte realizari similare din domeniu. In mod concret am considerat ca punct de plecare ontologia propusa in cadrul proiectului european „Hart failure Ontology" (http://lis.irb.hr/heartfaid/). Aceasta ontologie solutioneaza in speta aspectele medicale privind patologia cardio-vasculara, dar rezolva in mai mica masura problemele legate de gestiunea actului medical. De aceea am extins ontologia propusa in cadrul acestui proiect cu o serie de noi concepte si relatii.

In ontologia propusa conceptele sunt organizate ierarhic, cu scopul de a indica relatiile de mostenire dintre conceptele mai generale si cele particulare. De exemplu „Anomalie atriala" este un concept ce mosteneste atributele conceptului de „Boala cardiaca", care la randul sau deriva din conceptul „Boala a sistemului cardio-vascular" si care mai departe decurge din conceptul de „Diagnostic". Relatiile dintre concepte se stabilesc prin atribute de tip poantor ce indica alte concepte. In capitolul 3.3 vor fi prezentate cele mai importante relatii existente intre concepte.

In continurare se vor prezenta principalele concepte ale ontologiei propuse.

### 3.2.1 Caracteristici de pacient

In categoria generica de „caracteristici pacient" sunt incluse toate acele concepte si instante ce caracterizeaza un pacient din punct de vedere clinic si demografic. Subclasele acestui concept sunt:

- caracteristici demografice
- semne si simptome
- diagnostice
- prognoze
- alte caracteristici de pacient

Caracteristicile demografice cuprind atat conceptele de identificare a tipului de pacient (grupa de varsta, sex, rasa, categoria de asigurare, etc.) cat si datele de identificare (nume, adresa, CNP, etc.)

Din punct de vedere medical mult mai importante sunt conceptele de „semne si simptome” si respectiv „diagnostice”. In Anexa 1 se prezinta ierarhia de concepte care deriva din conceptul de Diagnostic, iar in Anexa 2 diagrama de concepte pentru semne si simptome.

#### 3.2.1.1 Diagnostice

Este un concept important ce grupeaza patologia cardio-vasculara. In subclasele generate din acest concept sunt grupate si clasificate diferitele afectiuni ale sistemului circulator ( Afectiuni arteriale si de circulatie) si ale inimii (Boli de inima si Insuficienta cardiaca). Principalele atribute ale acestui concept sunt:

- Nume: text ce indica numele bolii
- Definitie: text ce descrie un anumit diagnostic
- Cod_diagnostic: cod unic de indexare a bolilor cardiovasculare
- indicat_de_semn_simptom: legatura la un subconcept sau instanta a conceptului de semne si simptome
- factori_de_risc: legatura la subconceptele sau instantele clasei „factori de rist”
- tratat_prin: legatura la conceptul de tratament (instanta sau subclasa)
- cauzeaza_semne_simptome: legatura la semne sau simptome cauzate de o anumita boala
- sinonime: legatura la clasa de corelare a termenilor sinonimi

Instante directe ale acestui concept sunt: Lipsa_diagnostic si Diagnostic_necunoscut. Un diagnostic identificat este indicat printr-un subconcept sau o instanta a acestuia. Exista diagnostice din categoria bolilor cardio-vasculare si respectiv ale unor alte tipuri de boli. Avand in vedere obiectivul aplicatiei, bolile legate de alte organe sunt mai putin detaliate. In schimb cele ce vizeaza sistemul cardio-vascular sunt ierarhizate si clasificate in detaliu, dupa cum urmeaza:

- anomalii arteriale si de circulatie
- insuficienta cardiaca (directa)
- boli ale inimii

In prima categorie „anomalii arteriale si de circulatie” sunt incluse urmatoarele tipuri de boli:

- arteroscleroza
- anomalii de tensiune
- congestii
- anomalii minerale: calciu, magneziu, glucoza, potasiu si sodiu
- anomalii ale celulelor rosii
- anomalii ale lipidelor
- tromboze

In a doua categorie sunt incluse acele boli care sunt direct legate de o insuficienta cardiaca:

- disfunctii ale inimii: diastolic si sistolic
- forme de insuficienta cardiaca

In categoria bolilor de inima au fost incluse:

- anomalii atriale
- aritmii cardiace
- anomalii atrio-ventriculare de conductie
- hipertrofia inimii
- cardiopatii
- anevrism cardiac
- anomalii de ritm cardiac

- anomalii de valve
- perturbatii de conductie intraventriculare
- dispunctii ale nodului sinusal

#### 3.2.1.2 Semne si simptome

In categoria de semne si simptome sunt incluse toate acele elemente pe baza carora se stabileste un diagnostic. Exista elemente masurabile si detectabile prin diferite metode de masurare si investigare ( ele genereaza semne) si respectiv elemente ce decurg fie din starea generala a pacientului fie dintr-o descriere a acestuia. De exemplu tensiunea arteriala sau o diagrama ECG reprezinta semne iar durerea de cap declarata de pacient se incadreaza in categoria de simptome. Cele mai relevante semne si simptome pentru bolile cardio-vasculare au fost grupate dupa cum urmeaza:

- semne:
    - semne cardio-vasculare:
        - presiune sangvina
        - ritm cardiac
        - semne privind circulatia sangvina
        - murmure si sunete cardiace
    - semne pulmonare (de respiratie
    - endeme ale extremitatilor
    - semne privind greutatea
    - schimbari ale tenului
- simptome:
    - simptome legate de insuficienta cardiaca (oboseala, palpitatii, crestere/descrestere in greutate)

### 3.2.2 Pacienti

Este un concept ce grupeaza toate atributele cel descriu starea de sanatate a unui pacient si informatiile demografice de identificare. Cele mai importante atribute ale acestui concept sunt: nume

- cod de identificare pacient
- varsta, sex, inaltime, greutate
- conditia de asigurat
- are semne si simptome: legatura la o clasa se semne sau simptome
- medicatia urmata: legatura la un concept sau instanta de medicatie
- medicatia prescrisa: legatura la un concept sau instanta de medicatie
- diagnostic: legatura la un subconcept sau instanta de diagnostic
- factori de risc: legatura la instante de risc

### 3.2.3 Tratament

Conceptul de tratament este un arhetip pentru clasele de medicatie, proceduri medicale, recomandari, interventii chirurgicale si tratament cu aparate medicale. Principalele atribute ale conceptului sunt:

- nume tratament

- cod unic de identificare
- periodicitatea tratamentului
- durata pana la urmatoarea investigatie

#### 3.2.3.1 Medicatia

Conceptul de medicatie este cel mai complex dintre cele legate de tratament; ierarhia de medicatie grupeaza diverse clase de medicamente recomandate4 pentru diverse tipuri de diagnostice. In ontologie au fost cuprinse in speta acele medicamente si grupe de medicamente care au importanta in medicatia bolilor cardio-vasculare. Principalele subclase sunt:

- medicatia cardiopatiilor:
    - agenti anti-aritmii
    - diuretice
    - agenti de fibrilatie
    - vaso-dilatatori
    - agenti inotropici
    - inhibitori ACE
    - alti agenti
- alte medicatii:
    - antihistaminice
    - steroizi
    - toxine
    - anticolinergice
    - medicatii specifice

Principalele atribute ale conceptului de medicatie sunt:

- denumire
- cod unic de identificare
- timpul pana la urmatoarea investigatie

In cazul medicatiei cardiopatiilor se adauga urmatoarele atribute:

- eficient impotriva: legatura la diagnostic
- cantitatea administrata (mg)
- indicata in simptome si semne: legatura la semne si simptome
- doza zilnica
- doza maxima zilnica recomandata
- doza de mentenanta
- doza initiala
- trateaza: legatura la diagnostic
- efecte secundare: legatura la diagnostic si la semne si simptome
- a nu se folosi cu: legatura la medicatie
- folosit in caz de intoleranta la: legatura la medicatie

#### 3.2.3.2 Proceduri medicale

Grupeaza diferitele tipuri de proceduri medicale aplicate in cazul bolnavilor cardiaci (ex: socuri electrice, infuzii, intubare endotraheana, etc.) . Atributele conceptului sunt:

- denumire
- cod unic de identificare
- timpul pana la urmatorul test

- perioada de testare

#### 3.2.3.3 Dispozitive

In aceasta categorie intra dispozitivele de stimulare cardiaca (pacemaker). Atributele conceptului sunt:

- denumire
- cod unic de identificare
- eficient in caz de : legatura la diagnostic
- imbunatateste: legatura la diagnostic si la tratament
- efecte secundare: legatura la semne, simptome si diagnostice
- modifica: legatura la semne si simptome

#### 3.2.3.4 Recomandari

Recomandarile vin in completarea tratamentului si a medicatiei prescrise. Acestea au rolul de a reduce riscul incidentei unor accidente cardio-vasculare. Dintre instantele posibile ale acestui concept se pot aminti:

- reducerea consumului de alcool
- evitarea expunerii la soare
- verificarea periodica a tensiunii sangvine
- efectuarea de exercitii fizice
- reducerea gradului de obezitate
- evitarea fumatului

#### 3.2.3.5 Interventii chirurgicale

Acest concept modeleaza diferitele tipuri de interventii chirurgicale specifice pentru bolole cardio-vasculare. Exemple de instante ale acestui concept: angioplastie coronariana, Bypass arterial-coronarian, transplant, revascularizare, operatia valvei mitrale, etc. Principalele atribute ce descriu aceasta clasa sunt:

- denumire
- cod unic de identificare
- imbunatateste: legatura la caracteristici pacient, tratament
- indicat pentru: legatura la diagnostic, si la semne si simptome
- nu se recomanda pentru: legatura la Diagnostic

### 3.2.4 Testare

In acest concept sunt grupate toate aspectele legate de diversele forme de investigatii medicale. Cuprinde diverse tipuri de masuratori si teste, examinare fizica, precum si limite normale de masurare. Cele mai importante grupe de masuratori sunt:

- masurari eco-cardiografice
- masurari electrocardiografice:
    - masurari ECG obisnuite
    - masurari cu holter
    - masurari ECG cu 12 derivatii
- teste si masuratori de sange si biochimice
    - lipide
    - stare tiroida

    - urina
- examinari fizice:
    - tensiune sangvina
    - ritm cardiac
    - masuratori fizice legate de inima
    - masuratori legate de plamani
- teste la efort
- Radiografii ale pieptului
- Masurari ale inimii cu rezonanta magnetica

### 3.2.5 Planuri de tratament

Planurile de tratament modeleaza un set de actiuni si contraindicatii, preconditii si conditii pentru tratamentul unei anumite afectiuni. Principalele tipuri de actiuni sunt:

- prescriere de medicamente
- trimitere pacient (la analize sau cu scop de internare)
- efectuarea unui test
- interventie chirurgicala

Aceste actiuni pot aparea si sub forma de contraindicatii (ex: a nu se face interventie chirurgicala, a nu se efectua un anumit test, etc.). Clasa „prescriere medicamente" cuprinde mai multe subclase pentru principalele grupe de medicamente administrate in afectiunile cardio-vasculare (ex: anticoagulante, beta-blocante, etc.).
Atributele mai importante ale actiunii „prescriere de medicamente" sunt:

- grupa de medicamente
- prescris pentru diagnostic: legatura la un diagnostic
- prescriere pentru semne si simptome: legatura la semne si simptome
- perioada de medicatie
- plan diagnostic sau semne si simptome: legatura la diagnostic sau la semne si simptome
- nivelul de severitate
- cresterea lenta a dozei (da sau nu)
- necesita stabilizare (da sau nu)

### 3.2.6 Concepte legate de insuficienta cardiaca

#### 3.2.6.1 Factori de risc

In aceasta categorie sunt inclusi acei factori care au influenta asupra incidentei accidentelor cardio-vasculare si in general asupra bolilor de inima. Sunt factori care se leaga de sange, de starea fiziologica generala, factori demografici si istorici, factori clinici, etc. In corelatie cu factorii de risc se stabilesc perioadele de verificare si de efectuare a unor teste.

## 3.3 Relații intre concepte

O ontologie este o specificație formala a obiectelor, conceptelor si a entitatilor dintr-un anumit domeniu de interes si a relațiilor care se stabilesc intre acestea. In cazul ontologiei medicale a căror concepte au fost prezentate anterior, relațiile care se stabilesc variază in

complexitate. Mai jos va fi prezentat un subset al mulțimii relațiilor definite in ontologia medicala din subdomeniul afecțiunilor cardiace. In figurile de mai jos sunt reprezentate conceptele si relațiile dintre acestea sub forma de diagrama statica de clase.

In Fig. 1 sunt reprezentate relatiile intre conceptul Clasificare clinica simptomatica si conceptele Diagnostic, Semne si simptome si Asistenta medicala. Prin relațiile stabilite se definește semantic acest concept. Clasificarea clinica simptomatica a insuficientei cardiace se realizează in funcție de diagnostic si simptome, iar fiecărei clase ii sunt asociate mai multe tipuri de asistenta medicala.

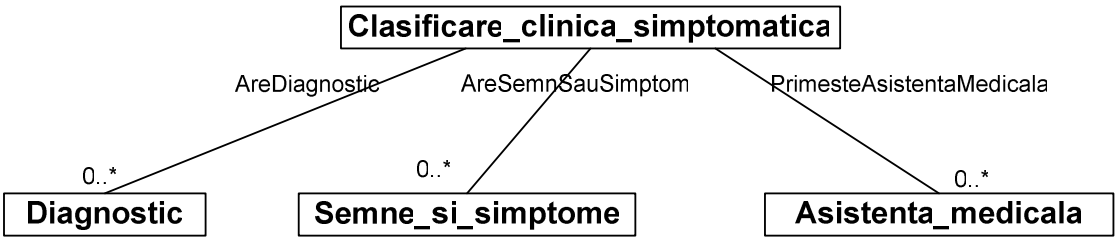



Fig. 1 Relațiile definite intre conceptul Clasificare_clinica_simptomatica si alte concepte

Intre conceptul Diagnostic si alte concepte se stabilesc relații complexe. In Fig. 2 este reprezentat un subset al acestor relații. Un diagnostic este indicat de anumite simptome si de anumite rezultate ale testelor realizate, sau poate fi cauzat de un anumit tip de medicație. Un diagnostic poate indica un anumit factor de risc cardiologic, o clasa de boli cardiace, sau un alt posibil diagnostic. De asemenea, pe baza unui diagnostic se poate indica un tratament si anumite teste de laborator care trebuie efectuate. Conceptul Diagnostic este o subclasa a conceptului Caracteristici_pacient.

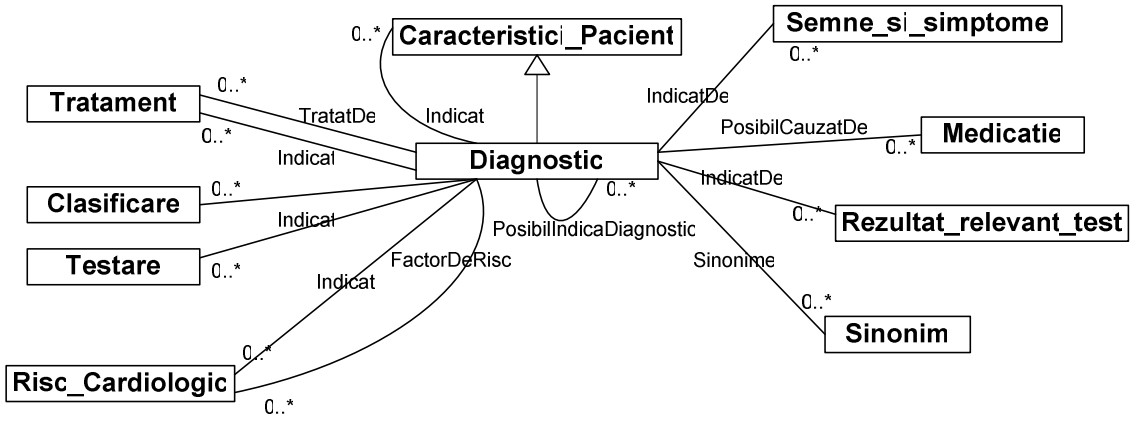



Fig. 2 Relațiile definite intre conceptul Diagnostic si alte concepte

In Fig. 3 sunt reprezentate o parte din relațiile stabilite intre conceptul Pacienti si alte concepte din ontologie. Un pacient poate avea unul sau mai multe diagnostice, poate prezenta anumite simptome, nu poate tolera un anumite tratamente sau a suferit anumite intervenții chirurgicale. De asemenea pacientului i se poate sugera o lista de teste de laborator care pot fi prescrise sau poate primi asistenta medicala.

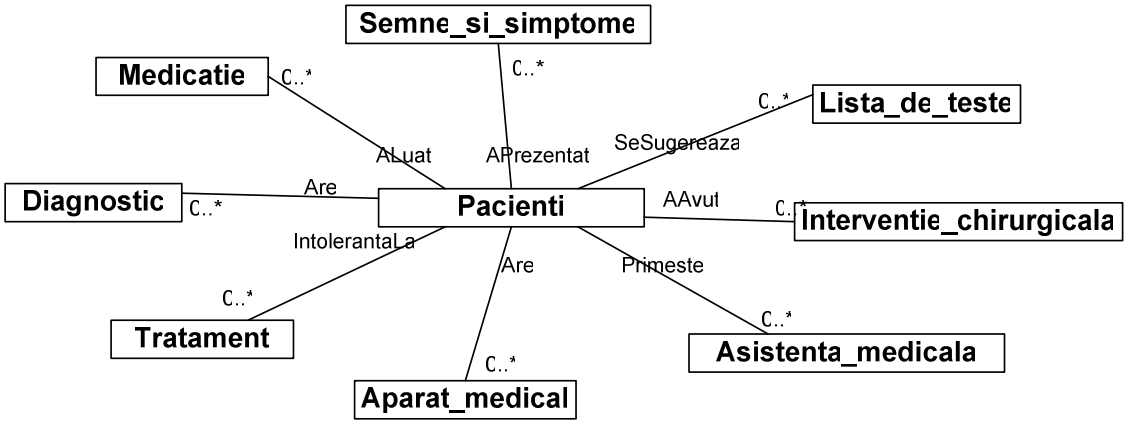



Fig. 3 Relațiile definite intre conceptul Pacienti si alte concepte

Conceptul Prescriere_medicatie este in relație cu conceptele Diagnostic, Semne_si_simptome si Medicatie, aşa cum se poate vedea in Fig. 4. Acest concept este o subclasa a conceptului de baza Plan. Medicația se poate prescrie daca este pus un anumit diagnostic, sau daca exista anumite simptome. In cadrul unui plan de prescriere, exista medicamente specifice care pot fi prescrise.

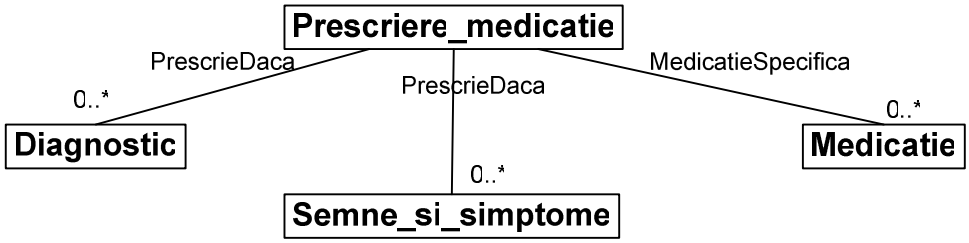



Fig. 4 Relațiile definite intre conceptul Prescriere_medicatie si alte concepte

Conceptul Examinare_fizica este o subclasa a conceptului Testare. Sunt stabilite relații intre conceptul Examinare_fizica si conceptele Diagnostic, Semne_si_simptome si Masuratori_test. Relațiile sunt reprezentate in Fig. 5. In timpul examinării fizice se pot detecta diagnostice, simptome si se pot realiza anumite masuratori.

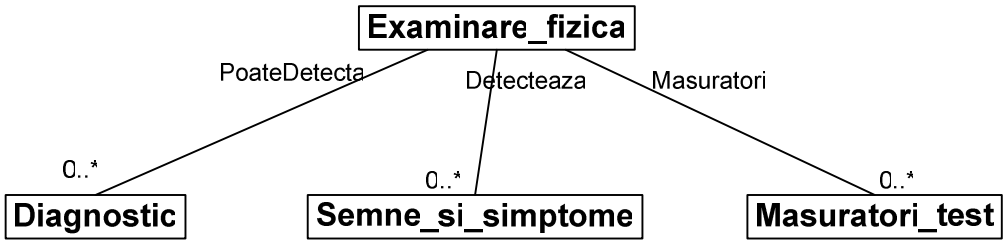



Fig. 5 Relațiile definite intre conceptul Examinare_fizica si alte concepte

In Fig. 6 sunt reprezentate relațiile intre conceptul Grup_medicatie_insuficienta_cardiaca si conceptele Caracteristici_pacient, Diagnostic, Tratament si Semne_si_simptome. Conceptul Grup_medicatie_insuficienta_cardiaca este o subclasa a conceptului Tratament. Un grup de medicamente pentru insuficienta cardiaca imbunatateste caracteristicile unui pacient, este eficient pentru anumite diagnostice, poate fi indicat intr-un anumit tratament si poate cauza efecte secundare vizibile prin anumite simptome.

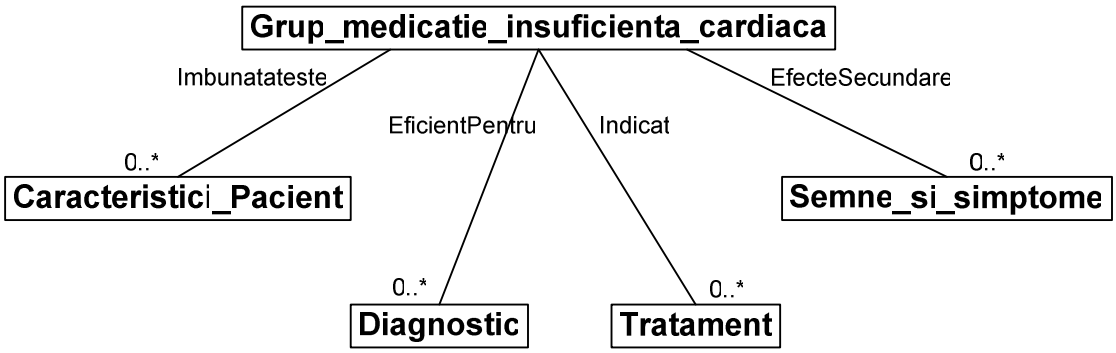


Fig. 6 Relațiile definite intre conceptul Grup_medicatie_insuficienta_cardiaca si alte concepte

## 3.4 Transpunerea ontologiei în structura bazei de date

### 3.4.1 Structura ontologiei

Elementul central al ontologiei este ***instanta***; o instanta este un element reprezentativ care are un corespondent concret in lumea reala; acest corespondent poate fi un obiect, o fiinta, notiune, categorie etc...

Pentru structurarea instantelor ele se grupeaza in ***concepte***. Un concept este o descriere generala dupa care se grupeaza instante, un nume colectiv dat unui set de parametrii specifici ce descriu o lista de instante similare. De ex. daca avem instantele "paracetamol", "ampicilina", "penicilina" conceptul care le grupeaza poate fi "medicamente". La conceptul "medicamente" se asociaza instante care au parametrii care satisfac conditia de a fi medicament (adica sa aiba forma de pastila, capsula, sau lichid injectabil, sa fie folosit la tratarea unor conditii medicale etc...).

Pentru un nivel suplimentar de structurare conceptele sunt organizate in mod ierarhic (***ierarhie de concepte***) pe baza a cat sunt de generale sau concrete. Un concept va fi superconceptul altui concept, daca defineste niste parametrii care alcatuiesc un subset al parametrilor ce definesc subconceptul. De exemplu conceptul "medicatie" va fi un superconcept al conceptului "antibiotice" fiindca toate instantele care satisfac conditiile de a fi antibiotice satisfac si conditiile de a fi medicamente, dar nu si invers, deci setul de conditii pentru medicamente este un subset al setului de conditii pentru antibiotice. Conceptul cel mai general care are un set gol, de conditii se numeste "THING" si sta la radacina ierarhiei arborescente de concepte.

Deoarece se folosesc instante pentru reprezentarea tuturor lucrurilor din lumea reala, intuitiv putem deduce ca trasaturile specifice unor lucruri vor fi si ele reprezentate prin instante. In ontologie relatia **Obiect – caracteristica – valoarea** se va reprezenta ca si **Instanta i1 – caracteristica – instanta i2** unde i1 este instanta despre care se discuta, iar i2 este valoarea unei caracteristici asignate instantei i1. De ex. daca vrem sa exprimam ca pentru ampicilina, paracetamol este contraindicat atunci i1 va fi ampicilina, caracteristica va fi "contraindicat" iar i2 va fi paracetamol.

Pentru a exprima relatiile de genul **instanta – caracteristica – instanta**, si elementul de tip caracteristica trebuie formalizat. Aceste caracteristici se vor numi ***slot***-uri; relatia se transforma in Instanta i1 – slot s – instanta i2 care se citeste: Instanta i1 are in slotul s valoarea i2. (ex. "ampicilina are in slotul contraindicat valoarea paracetamol" este echivalent cu expresia "paracetamolul este contraindicat pentru ampicilina")

In acest stadiu, teoretic un slot poate primi valori de orice gen, adica orice instanta poate intra in orice slot. Se va impune o constrangere de valori posibile care pot intra intr-un slot. De ex. in slotul contraindicat se doreste sa intre doar medicamente (intuitiv nu putem spune ca ampicilina are ca si contraindicatie instante ce nu apartin conceptului "medicament", de exemplu "durere de cap" – care este instanta conceptului "simptoma"). Aceste constrangeri se formalizeaza prin semne de slot – ***signslots*** **–** un semn de slot defineste o asociere de gen **slot s - concept c** ce reprezinta faptul ca in slotul s pot intra ca si valori instante ale conceptului c (sau instante ale subconceptelor lui c - intuitiv instantele care satisfac conditiile conceptelor din subarborele lui c satisfac si conditiile lui c deci pot intra ca si valori)

In adaptarea ontologiei pentru aplicatia curenta s-a mai introdus o arbitrare:

- Se porneste de la problema ca pe langa lista de instante predefinite din ontologie (instantele avand corespondente din lumea reala) aplicatia in sine mai creeaza dinamic elemente in lumea reala care trebuie si ele reprezentate sub forma unor instante speciale.
- La instantele generate de aplicatie se vor lega slot-uri in mod similar cu cazul instantelor predefinite.
- Instantele generate dinamic de sistem se impart in 5 categorii:
  - Persoane (reprezinta rolul unei persoane reale transpuse in sistem)
  - Pacienti
  - Doctori
  - Episoade medicale
  - Pasi de tratament
- Este nevoie sa precizam pentru fiecare slot, la ce tip de entitate poate fi legat – una din cele 5 tipuri generate sau tipul standard de entitati predefinite. Relatia **slot s – tipul entitatii t** este reprezentat prin relatii numite ***entityslot.***

### 3.4.2 Transpunerea ontologiei in baza de date:

Lista de instante predefinite este stocata in tabela ***instances*** – elementele principale stocate sunt:

- denumirile in mai multe limbi – Romana si Engleza.
- un id de tip string pentru identificarea instantei in elementele de nivel innal ale sistemului sau in comunicarea inter-sisteme.
- un id de tip intreg pentru identificarea instantei in interiorul bazei de date (join-urile si ordonarile in BD se fac mai usor pe baza de intreg).

Lista de concepte este stocata in tablea ***concepts*** – elementele principale stocate sunt:

- denumirile in mai multe limbi – Romana si Engleza.
- un id de tip string pentru identificarea conceptului in elementele de nivel innal ale sistemului sau in comunicarea inter-sisteme.
- un id de tip intreg pentru identificarea slotului in interiorul bazei de date (join-urile si ordonarile in BD se fac mai usor pe baza de intreg).

Relatiile ierarhice dintre doua concepte stocate in tabela concepts se exprima prin intrari tabela ***concepthierarchies*** – o intrare contine referinte la doua concepte (referinta bazata pe id – ul de tip intreg) una semnificand superconceptul, iar celalalt subconceptul direct al superconceptului.

Pentru a exprima relatiile de gen instanta – caracteristica – valoare dintre 2 instante baza de date contine tabela ***valuesslot*** care face legatura dintre o instanta de referinta, un slot

(caracteristica) si o instanta de valoare. Pentru a stoca pe langa valori de tip instanta si valori simple ca si string si intreg asocierea la instanta valoare s-a mutat din tabela principala valuesslot intr-o tabela secundara ***valuessinst*** iar pe modelul acestei tabele s-au mai construit 2 tabele una pentru valori de tip string, si una pentru valori de tip numeric, aceste tabele sunt ***valuesstr*** si ***valuesint***

Pentru a stoca caracteristicile in baza de date s-a definit o tabela ***slots*** care contine lista de caracteristici posibile. Elementele principale stocate sunt:

- denumirile in mai multe limbi – Romana si Engleza.
- un id de tip string pentru identificarea slotului in elementele de nivel innal ale sistemului sau in comunicarea inter-sisteme.
- un id de tip intreg pentru identificarea slotului in interiorul bazei de date (join-urile si ordonarile in BD se fac mai usor pe baza de intreg).

Pentru a definii constrangerile de slot – concepte ale caror instante pot intra in sloturi, baza de date contine un tabel numit ***signslot*** – o intrare din tabela face legatura dintre un slot si un concept.

Pentru stocarea instantelor generate dinamic s-au constuit tabele proprii persons, patients, mds, episodes, steps – din considerari de performanta deoarece aceste instante sunt tratate in permanenta de sistem si este mai fezabil sa se stocheze separat. Pentru reprezentarea constragerii de tip slot – tip de entitate s-a introdus in baza de date tabelul ***entityslots*** care contine tuple ce asociaza un anumit slot, cu un anumit tip de entitate. De ex sloturile originale din ontologia de baza vor avea toate intrari in aceasta tabela ca re ii asociaza cu entitati de tip M (de la Medical knowlege base) care sunt entitatile originale – instantele predefinite din ontologie.

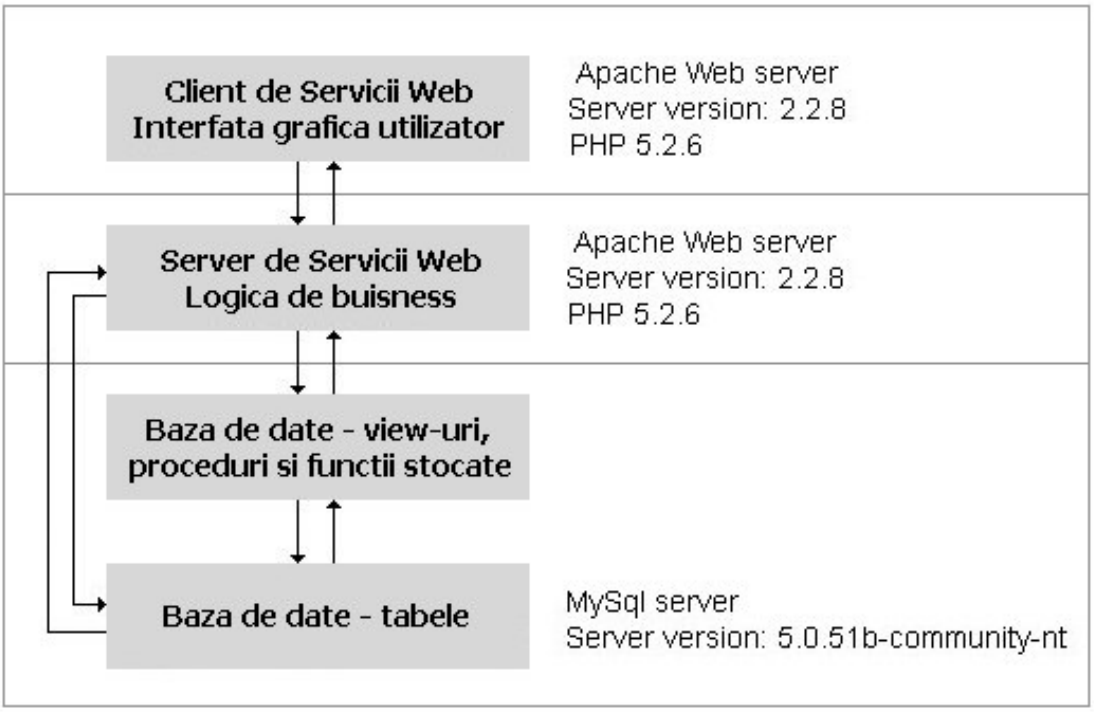
# 4 Aplicația „Medic de familie”

In cadrul proiectului CardioNET un modul important constituie aplicatia medic de familie, care permite interactia pacient – doctor, si permite accesul la servicii medicale la distanta prin intermediul internetului. In continuare se prezinta o descriere detaliata structurala si functionala a palicatiei medic de familie.

## 4.1 Descriere structurala a aplicatiei “medic de familie”

Sistemul consta din 3 parti majore: **Baza de Date** (implementat in mysql 5.0.51b), **Serverul de Servicii Web** (implementat ca si o web-aplicatie PHP ruland pe server apache) si **Clientul de Servicii Web** (implementat ca si o web-aplicatie PHP ruland pe server apache).

In functionare si interactiunea cu utilizatorul, componentele sunt legate in forma ierarhica:




- Modulul 1: Baza de date contine:
  - Un set de tabele pentru stocarea informatiilor referitoare la utilizatori/persoane/pacienti/doctori sessiuni de login si ontologie.
  - Un set de vederi pentru usurarea accesului la date
  - Un set de functii si proceduri stocate pentru introducerea si extragerea unor date complexe.
- Modulul 2: serverul de servicii web, contine partea majora a logicii de business, implementeaza:
  - Filtre de securitate (pentru prevenirea accesului neautorizat la date)
  - Functii de autentificare
  - Functii de extragere a datelor
  - Functii de prelucrare si salvare a datelor noi
  - Etc ...

- Modulul 3: clientul de servicii web dialogheaza cu utilizatorul prin intermediul interfetei grafice, si comunica cu serverul de servicii web prin protocolul SOAP. Modulul 3 nu contine logica de business.
    - obtine date si informatii existente in sistem de la Serverul de servicii web pe baza carora genereaza interfata grafica.
    - Primeste date si informatii de la utilizator prin interfata grafica pe care le trimite mai departe pentru prelucrare la serverul de WS.

## 4.2 Descriere functionala pe componente

### 4.2.1 Descrierea bazei de date a aplicatiei

Baza de date a aplicatiei stocheaza atat date cu caracter permanent legate de ontologia domeniului cat si date curente privind pacienti, medici, episoade medicale si detalii referitoare la etape de diagnoza si tratament. Baza de date contine tabele pentru stocarea informatiilor referitoare la:

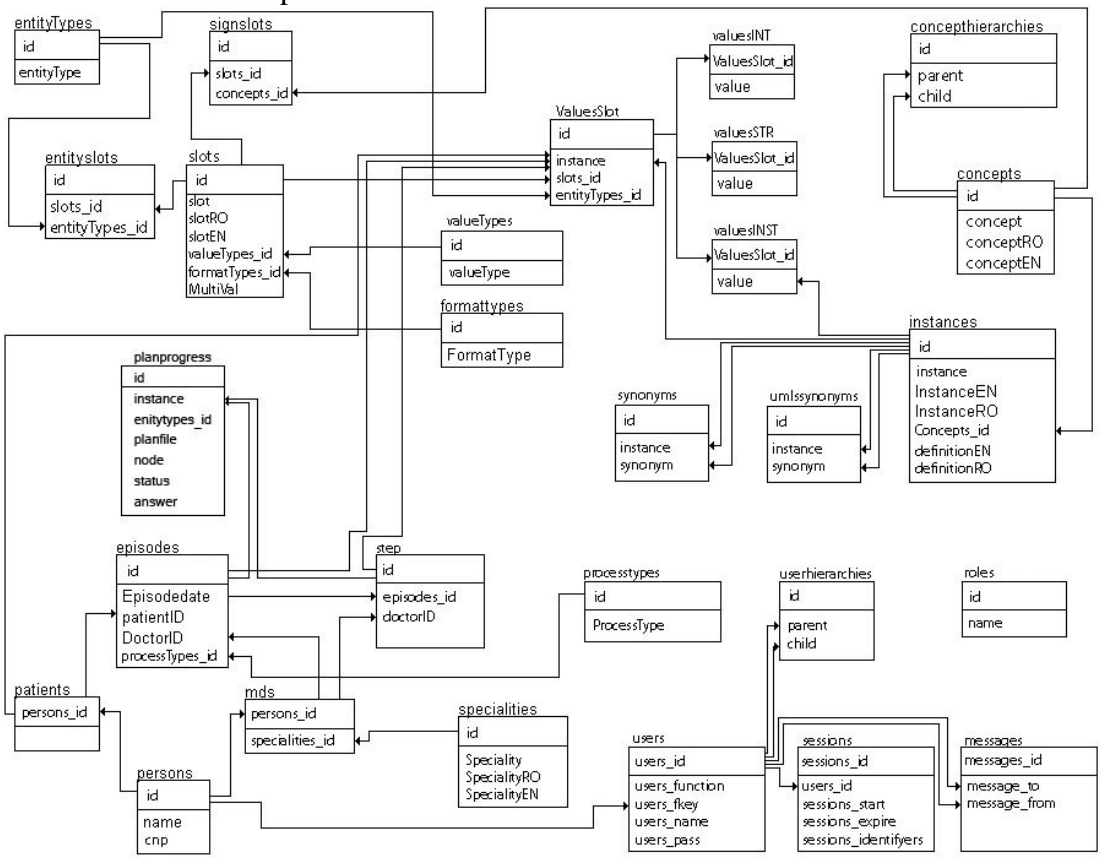
- Persoane:
    - **Persons** – pentru stocarea persoanelor
    - **Users** – pentru a lega un utilizator si o parola la persoana respectiva
    - **Patients** – pentru a marca persoanele care sunt pacienti in sistem
    - **Mds** – pentru a marca persoanele care sunt doctori in sistem
- Roluri, sessiuni si securitate: Policy-ul de securitate este descris in detaliu la 2.2.4.5:
    - **Roles** - pentru a stoca rolul utilizatorului (pacient, doctor sau administrator de sistem)
    - **Userhierarchies** – pentru a stoca relatiile ierarhice dintre persoane – necesar pentru determinarea drepturilor de acces la anumite informatii (doctorul are acces la informatiile pacientilor proprii, un doctor are acces la informatiile subordonatilor sai)
    - **Specialities** – pentru a stoca tipul de doctor (de familie, specialist etc ...)
    - **Sessions** – pentru a stoca informatiile referitoare la sesiuni active (data de start, data expirarii, key de encriptare etc...)
    - **Messages** – pentru a stoca mesajele trimise intre utilizatori prin intermediul serviciului de mesagerie tip “messenger” incorporat in sistem (numit CNmessenger)
    -
- Ontologie
    - **Concepts** – pentru a stoca conceptele din ontologie (un concept este un nume colectiv pentru un set de lucruri din lumea reala- ex. daca *paracetamol* este lucrul din lumea reala atunci conceptul parinte va fi “medicatie”)
    - **Instances** – pentru a stoca referinte la lucruri din lumea reala si pentru a le lega de concepte.
    - **Concepthierarchies** – pentru a stoca relatia hierarhica dintre concepte generale si concepte mai concrete
    - **Synonyms** – pentru compatibilitate cu alte sisteme care folosesc alte cuvinte pentru a reprezenta aceleasi lucruri din lumea reala

- **Umlssynonyms –** pentru compatibilitate sporita cu sisteme care folosesc expresiile din UMLS.
- **Slots** – pentru stocarea denumirilor proprietatilor referitoare la o instanta tinta definite in ontologie (o proprietate poate fi :
  - O caracteristica numerica de gen greutate, varsta etc...
  - O valoare de tip string de gen nume, descriere etc...
  - O alta instanta atribuita ca si caracteristica instantei prime
- **Valuetypes** – pentru a preciza ce tipuri de date intra la care proprietate (slot)
- **Valuesslot** – face legatura dintre instanta si slot, este folosit pentru stocarea valorii efective care intra la proprietatea (slotul) unui lucru (instanta) din lumea reala.
- **Valuesint** – daca valuesslot stocheaza valoarea unei caracteristici numerice, valoarea numerica se stocheaza in subtabela valuesint a tabelei principale de valori valuesslot
- **Valuesstr** – daca valuesslot stocheaza valoarea unei caracteristici de tip string, valoarea textuala se stocheaza in subtabela valuesstr a tabelei principale de valori valuesslot.
- **Valuesinst** – daca valuesslot stocheaza valoarea unei caracteristici de tip alta instanta atribuita, cheia straina (obtinuta duin tabela instances) ce poanteaza spre instanta atribuita ca si caracteristica, se stocheaza in subtabela valuesinst a tabelei principale de valori valuesslot.
- **Signslots** – in cazul sloturilor (proprietatilor) care iau ca valori instante din ontologie, se va marca care sunt acele concepte ale caror instante pot intra ca si valoare la slotul respectiv. (ex. la slotul “Medicatie Curenta” pot intra instante ale conceptului “Medicament” care sunt “Paracetamol”, “Aspirina” etc... )
- **Entitytypes** – pentru a extinde mecanismul de atribuire a proprietatilor in asa fel incat nu doar instantele de concepte din ontologie sa aiba proprietati ci si persoanele (proprietati de genul “Gender” “Marital status”) si episoadele medicale (proprietati de genul “Semne si simptome”, “Diagnostic” etc...). In entity types sunt marcate toate tipurile de entitati la care se pot lega proprietati (P-pacient, E-episode, M- medical knowledge base (acesta fiind originalul, default-ul ), etc...)
- **Entityslots** – pentru a lega o proprietate (un slot) de tipul de entitate (P,E,M etc...) care o poate avea.
- **Formattypes** – pentru ca datele stocate sub o anumita forma pot fi interpretate in mai multe feluri (un numar poate fi numar simplu sau timestamp, caz in care necesita un mod de afisare special) aceasta tabela stocheaza modurile de afisare de baza pentru proprietati (sloturi).

- Date medicale legate de ontologie
  - **Episodes** – pentru stocarea unui episod medical (doar datele generale de gen data, pacient, doctor se stocheaza in tabela, restul proprietatilor se stocheaza in valuesslot, valuesinst, valuesint si valuesstr)
  - **Processtypes** – pentru identificarea tipurilor de procese (actualmente avem “Consultatie”- care consta dintr-un singur pas, atomic dpdv a timpului si

”Tratament” – care consta din mai multi pasi si se desfasoara dealungul unei perioade de timp)

- **Steps** – pentru stocarea unui pas de tratament (doar datele generale de gen data, doctor se stocheaza in tabela, restul proprietatilor se stocheaza in valuesslot, valuesinst, valuesint si valuesstr)
- **PlanProgress** – pentru stocatrea informatiilor legate de planuri de tratament. Leaga un episod medical sau un pas de tratament de un plan medical stocat sub forma unui xml si precizeaza pozitia curenta in parcurgere pentru planul respectiv.




**Exemple de utilizare a tabelelor din baza de date**

**ex1: inregistrare pacient:** inregistram un pacient cu un cont de utilizator (user = pacient X, parola = pass), numele lui este John Doe, are CNP-ul 1234567890123 si este necasatorit:

1) se introduce in persons o inregistrare cu *id*-ul = 15, se seteaza *nume* si *cnp* la valorile John Doe si 1234567890123.
2) In tabela slots exista un slot numit MaritalStatus care are id-ul 49 si are valuetypes_id = “C” – adica slotul va avea ca si valori instante de concept
3) In tabela signslots exista o inregistrare care arata ca slot-ul cu id-ul 49 poate lua ca si valori, instante ale conceptului 127
4) In tabela concepts se regaseste conceptul cu id-ul 127 care este “MaritalStatus”
5) In tabela Instances gasim instantele conceptului 127, care sunt:
   a) 1999 – necasatorit
   b) 2000 – casatorit
   c) 2001 – divortat

   d) 2002 – vaduv(a)
6) Se introduce in valuesslot o inregistrare cu:
   a) Id-ul = 1239 – valoarea este generata (auto increment)
   b) entitytypes_id = "R" – ne spune ca ne referim la o persoana
   c) instance = 15 – ne spune ca ne referim la persoana cu id-ul 15
   d) slots_id = 49 – ne spune ca setam valoarea pentru slot-ul 49 (MaritalStatus)
7) Se introduc valori in cele 3 tabele valuesinst, valuesint si valuesstr
   a) De fapt introducem doar in valuesinst, pentru ca acest slot poate avea doar valori de tip instante de concept (asta stim de la valuetypes_id) , inregistrarea din valuesinst va avea:
      i) valuesslot_id = 1239 – specifica faptul ca inregistrarea specifica partea valorica (de tip instanta de concept) al valorii de slot 1239 din valuesslot
      ii) value = 1999 – indicand faptul ca instanta "necasatorit" este valoare introdusa.
8) Se introduce in pacienti o intrare cu persons_id = 15 marcand faptul ca persoana este pacient.
9) se introduce un users o intrare cu campurile
   a) name = "Pacient X"
   b) pass=hash-ul md5 al cuvantului "pass"
   c) persons_id = 15 marcand faptul ca utilizatorul este legat de persoana respectiva
   d) roles_id = "P" semnificand faptul ca acest utilizator are drepturi de pacient

**ex2: inregistrare episod simplu:** inregistram faptul ca pacientul John Doe din ex1. a fost consultat de doctorul Ixulescu (care are id-ul de persoana 5) la data de 17.6.2009 pentru ca avea durere severa de cap, iar doctorul Ixulescu a diagnostizat o migrena si a incheiat consultatia.

1) se introduce in episodes o inregistrare cu *id*-ul = 30, se seteaza *episodedatebegin* si *episodedateend* la valoarea 17.6.2009, *patientid* la valoarea 15 (id-ul lui John Doe), doctorid la 5 (Ixulescu), *processtypes_id* la C (indicand ca este un singur consult si nu un tratament)
2) In tabela slots exista un slot numit HasSignOrSymptom care are id-ul 96 si are valuetypes_id = "CI" – adica slotul va avea ca si valori instante de concept si valori numerice.
3) In tabela signslots exista o inregistrare care arata ca slot-ul cu id-ul 96 poate lua ca si valori, instante ale conceptului 191
4) In tabela concepts se regaseste conceptul cu id-ul 191 care este "Signs and symptoms"
5) In tabela concepthierarchies parcurgand ierarhia gasim ca 191 este superconcept al sub conceptului 202
6) In tabela concepts se regaseste conceptul cu id-ul 202 care este "Symptoms"
7) In tabela Instances gasim instantele conceptului 202, printre care si
   a) 750 – durere de cap
8) Se introduce in valuesslot o inregistrare cu:
   a) Id-ul = 1240 – valoarea este generata (auto increment)
   b) entitytypes_id = "E" – ne spune ca ne referim la un episod
   c) instance = 30 – ne spune ca ne referim la episodul cu id-ul 30
   d) slots_id = 96 – ne spune ca setam valoarea pentru slot-ul 96 (HasSignOrSymptom)
9) Se introduc valori in cele 3 tabele valuesinst, valuesint si valuesstr
   a) De fapt introducem doar in valuesinst si valueint, pentru ca acest slot poate avea doar valori de tip instante de concept si valori numerice (asta stim de la valuetypes_id)
   b) Inregistrarea din valuesinst va avea:
      i) valuesslot_id = 1240 – specifica faptul ca inregistrarea specifica partea valorica (de tip instanta de concept) al valorii de slot 1240 din valuesslot.

ii) value = 750 – indicand faptul ca instanta “durere de cap” este valoare introdusa.

c) Inregistrarea din valuesint va avea:
   i) valuesslot_id = 1240 – specifica faptul ca inregistrarea specifica partea valorica (de tip instanta de concept) al valorii de slot 1240 din valuesslot.
   ii) Value = 9 indicand faptul ca (pe o scala de la 0 la 10) este o durere severa de cap.

10) In tabela slots exista un slot numit HasDiagnosis care are id-ul 124 si are valuetypes_id = “C” – adica slotul va avea ca si valori instante de concept.
11) In tabela signslots exista o inregistrare care arata ca slot-ul cu id-ul 124 poate lua ca si valori, instante ale conceptului 62
12) In tabela concepts se regaseste conceptul cu id-ul 62 care este “Diagnosis”
13) Similar cazului anterioar in ierarhia de concepte si instante gasim instanta de migrena care este instanta a unui sub-concept al lui “Diagnostic” si care are id-ul 1151.
14) Se introduce in valuesslot o inregistrare cu:
   a) Id-ul = 1241 – valoarea este generata (auto increment)
   b) entitytypes_id = “E” – ne spune ca ne referim la un episod
   c) instance = 30 – ne spune ca ne referim la episodul cu id-ul 30
   d) slots_id = 124 – ne spune ca setam valoarea pentru slot-ul 124 (HasDiagnosis)
15) Se introduc valori in cele 3 tabele valuesinst, valuesint si valuesstr
   a) De fapt introducem doar in valuesinst pentru ca acest slot poate avea doar valori de tip instante de concept (asta stim de la valuetypes_id)
   b) Inregistrarea din valuesinst va avea:
      i) valuesslot_id = 1241 – specifica faptul ca inregistrarea specifica partea valorica (de tip instanta de concept) al valorii de slot 1241 din valuesslot.
      ii) value = 1151 – indicand faptul ca instanta “migrena” este valoare introdusa.

**ex3: inregistrare episod cu mai multi pasi:** Sa presupunem ca similar exemplului 2 dorim sa inregistram un consult medical, cu diferenta ca la data de 17.6.2009 doctorul da doar un diagnostic prezumtiv, iar in data de 18.6.2009 se continua consultatia, iar dupa masurarea pulsului (pulsul este de 90) ii da diagnostic final de migrena.

Notam doar diferentele:

La 1: *episodedateend* initial ramane NULL

La 10-15: in loc de slotul HasDiagnosis (cu id-ul 124) punem SuggestedDiagnosis (cu id-ul 125)

In continuare :

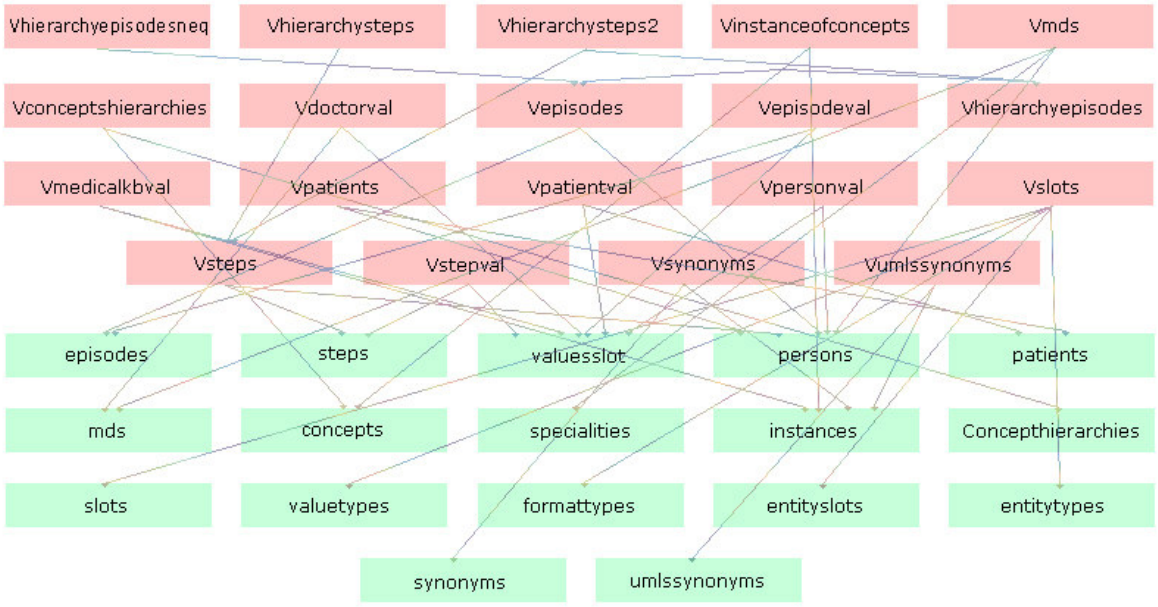
1) se introduce in steps o inregistrare cu *id*-ul = 11, se seteaza *stepdatebegin* si *stepdateend* la valoarea 18.6.2009, *patientid* la valoarea 15 (id-ul lui John Doe), doctorid la 5 (Ixulescu), *episodes_id* la 30 (indicand ca este un pas de tratament asociat cu episodul 30)
2) Se stocheaza pulsul similar ex2. (pasii 2-9) cu diferenta ca:
   a) (la 7 ) in loc de instanta de durere de cap “Headache” (id-ul 750), se foloseste instanta de puls “HeartRateValue” (id-ul 2004)
   b) (la 8) in loc de entitytypes_id = “E” se pune entitytypes_id=”S” (reprezinta faptul ca aceasta valoare se refera la pas si nu la episod) iar in loc de instance = 30 se pune instance = 11 (reprezinta id-ul pasului la care se refera valoarea)
   c) (la 9.) c.) ) in loc de valoarea 9 ce reprezinta severitatea durerii de cap, introducem valoara 90 ce reprezinta valoarea pulsului.

3) Se introduce diagnosticul final similar cazului de la ex2. cu diferenta ca diagnosticul se adauga la pasul 11 si nu la episodul 30
4) Se doreste incheierea episodului deci se introduce data 18.6.2009 in *episodedateend*- care initial a fost lasat NULL.

Baza de date contine urmatoarele vederi:

- **Vconceptshierarchies** – vedere realizata din combinarea tabelelor *concepts* si *concepthierarchies* , contine perechi de string-id-uri de concepte si are rolul de a indica relatiile ierarhice nemijlocite existente intre diferite concepte (relatii definite in ontologie)
- **Vdoctorval** – vedere realizata in principal din tabelele *mds* si *valuesslot* , contine tuple ce asociaza atribute si valoarea atributelor ce caracterizeaza un doctor
- **Vepisodes** – vedere realizata in principal din *episodes* si *persons,* contine tuple ce reprezinta datele generale ce se refera la un anumit episod realizat de un anumit doctor pentru un anumit pacient.
- **Vepisodeval** – vedere realizata in principal din *episodes* si *valuesslot* contine tuple ce asociaza atribute si valoarea atributelor cu un episod la care se refera.
- **Vhierarchyepisodes** – vedere realizata din *vepisodes*, contine perechi de id-uri de episoade, semnificand ordinea cronologica dintre doua episoade (data inceperii la precedent este <= data inceperii la successor).
- **Vhierarchyepisodesneq** – vedere realizata din *vepisodes*, contine perechi de id-uri de episoade, semnificand ordinea cronologica dintre doua episoade (data inceperii la precedent este < data inceperii la successor).
- **Vhierarchysteps** – vedere realizata din *vsteps*, contine perechi de id-uri de pasi, semnificand ordinea cronologica dintre doua pasi de tratament apartinand aceluiasi episod (data inceperii la precedent este <= data inceperii la successor).
- **Vhierarchysteps2** – vedere realizata din *vhierarchyepisodes* si *vsteps*, si contine perechi de id-uri de pasi, semnificand ordinea cronologica dintre doua pasi de tratament apartinand unor episoade diferite (data inceperii episodului la care apartine precedentul este <= data inceperii episodului la care apartine successorul).
- **Vinstancesofconcepts** – vedere realizata din *instances* si *concepts* contine perechi de string-id-uri instanta-concept, semnificand faptul ca o anumita instanta este instanta unui concept anume
- **Vmds** – vedere realizata din *mds persons* si *specialities*, contine tuple ce descriu un medic(numele, id-ul numeric de persoana, specialitatea).
- **Vmedicalkbval** – vedere realizata din *instances* si *valuesslot*, contine tuple ce asociaza atribute si valoarea atributelor ce caracterizeaza o instanta.
- **Vpatients** – vedere realizata din *patients* si *persons*, contine tuple ce descriu un pacient (numele, id-ul numerci de persoana).
- **Vpatientval** – vedere realizata din *patients* si *valuesslot*, contine tuple ce asociaza atribute si valoarea atributelor, ce caracterizeaza un pacient.
- **Vpersonval** – vedere realizata din *persons* si *valuesslot*, contine tuple ce asociaza atribute si valoarea atributelor, ce caracterizeaza o persoana.
- **Vslots** – vedere realizata din *slots*, *valuetypes*, *formattypes*, *entityslots* si *entitytypes*, contine tuple ce reprezinta slot-uri (atribute) cu proprietatile lor (ce fel de valoare ia, sub ce format se afiseaza, la ce entitati se poate referi etc...)

- **Vsteps** – vedere realizata in principal din *steps* si *persons*, contine tuple ce reprezinta datele generale ce se refera la un anumit step realizat de un anumit doctor pentru un anumit pacient.
- **Vstepval** – vedere realizata in principal din *steps* si *valuesslot,* contine tuple ce asociaza atribute si valoarea atributelor, ce caracterizeaza un pas.
- **Vsynonyms** – vedere realizata din *synonyms* si *instances* contine tuple cu date (numele in mai multe limbi string-id-ul etc...) referitoare la perechi de instante care pot fi sinonime intre ele.
- **Vumlssynonyms** – vedere realizata din *umssynonyms* si *instances* contine tuple cu date (numele in mai multe limbi string-id-ul etc...) referitoare la perechi de instante care pot fi sinonime intre ele pe baza standardului UMLS.




Baza de date contine urmatoarele functii si proceduri stocate:
FUNCTII:

- **Episodeidof -** folosit pentru obtinerea id-ului unui episod daca se cunoste data de start, pacientul si doctorul pentru care a fost creat
  - Parametrii:
    - pepisodedate date – data episodului medical
    - ppatientid int – id-ul patientului la care apartine episodul medical
    - pdoctorid int – id-ul doctorului care a creat episodul medical pentru pacient (cheie primara din *episodes*)
  - Returneaza: id-ul de tip int al episodului cautat
- **Stepidof** - folosit pentru obtinerea id-ului unui pas dintr-un episod daca se cunoste data de start, id-urile pacientului si al doctorului pentru care a fost creat
  - Parametrii:
    - pstepdate date – data de inceput a stepului cautat
    - pepisodeid int – Id-ul episodului la care apartine stepul cautat
    - pdoctorid int – Id-ul doctorului care a creat step-ul cautat
  - Returneaza:

        - Id-ul de tip int al step-ului cautat (cheie primara din *steps*)
    - 
- **Conceptidof -** folosit pentru obtinerea id-ului unui concept daca se cunoste numele
    - Parametrii:
        - pconcept varchar(256) – numele (string-id-ul) conceptului
    - Returneaza:
        - id-ul de tip int al conceptului cautat (cheie primara din *concepts*)
- **Instanceidof -** folosit pentru obtinerea id-ului unei instante daca se cunoaste numele
    - Parametrii:
        - pinstance varchar(256) – numele (string id-ul) instantei
    - Returneaza:
        - Id-ul de tip int al instantei cautate (cheie primara din *instances*)
- **Slotidof -** folosit pentru obtinerea id-ului unui slot daca se cunoaste numele
    - Parametrii:
        - pname varchar(256) – numele (string id-ul) slotului
    - Returneaza:
        - Id-ul de tip int al slotului cautat (cheie primara din *slots*)
- **Useridof -** folosit pentru obtinerea id-ului unui utilizator daca se cunoaste numele persoanei pentru care a fost creat
    - Parametrii:
        - pname varchar(50) – numele persoanei
    - Returneaza:
        - Id-ul de tip int al primului (de obicei unicul) utilizator legat de persoana cu numele pasat ca si parametru (cheie primara din *users*)
- **Personidof -** folosit pentru obtinerea id-ului unei persoane daca se cunoaste numele
    - Parametrii:
        - pname varchar(50) – numele persoanei
    - Returneaza:
        - Id-ul de tip int al persoanei cautate (cheie primara din *persons*)
- **Specialityidof -** folosit pentru obtinerea id-ulei specializari de doctor daca se cunoaste numele
    - Parametrii:
        - pspeciality varchar(20) – numele specialitatii doctorului
    - Returneaza:
        - Id-ul de tip int al specialitatii cautate (cheie primara din *pecialities*)
- **Insval -** folosit pentru a insera o valoare a unei proprietati - legate la o entitate din baza de date - in tabela valuesslot si subtabelele sale valuesstr, valuesint si valuesinst
    - Parametrii:
        - pinstanceid int – id-ul entitatii (instanta, persoana, episod etc...) la care se leaga proprietatea
        - pentitytype char(1) – tipul entitatii (E – episod, R-persoana etc...)
        - pslotid int – id-ul slotului (care este proprietatea pe care o setam)
        - pvaluestr varchar (256) – valoarea de tip string care se stocheaza daca proprietatea este de tip string (valuetype este S) – proprietatile pot fi combinate SI, CS sau CSI

        - pvalueint int – valoarea de tip int care se stocheaza daca proprietatea este de tip int (valuetype este I) – proprietatile pot fi combinate SI, CI sau CSI)
        - pvalueinstanceid int – valoarea de tip pointer (cheie straina din *instances*) la instanta care se stocheaza ca si valoare pentru proprietate daca proprietatea este de tip instance (valuetype este C ) – proprietatile pot fi combinate CS, CI sau CSI
        - pvaluetype varchar(3) – tipul valorii simple sau combinate care se introduce – poate fi C, S, I, CS, CI, SI sau CSI.
    - Returneaza:
        - Codul de stare a operatiei:
            - 0 – daca inserarea a fost efectuata cu succes
            - 11 – daca slot-ul cu id-ul si valuetype-ul specificat nu exista
            - 12 – daca slot-ul cu id-ul dat nu este legat la tipul de entitate specificat (ex. entitatea (lucrul din lumea reala) episod nu poate avea slot-ul (caracteristica) height)
            - 13 – daca instanta pe care o inseram nu este descendent macar al unui concept dintre conceptele valide pentru acest slot
            - 14 – daca entitatea de tip pentitytype si cu id-ul
            - 15 – daca apare o eroare neprecizata
            - pinstanceid nu exista in baza de date.
            - 21 – daca valuetype-ul este invalid (nu se afla in tabela *valuetypes* deci cu este C,S,I,CS,CI,SI sau CSI)
            - 22 – daca valorile de intrare ce ar corespunde la valuetype nu sunt completate (de ex daca valuetype e S si pvaluestr este NULL)
- **Issuperconcept -** folosit pentru a determina daca un concept este descendentul unui concept de baza sau nu
    - Parametrii:
        - pparent int – id –ul conceptului de radacina
        - pchild int – id-ul conceptului care ne intereseaza
    - Returneaza:
        - O valoare intreaga care este
            - 1 daca pchild este insasi pparent sau un concept descendent a lui pparent (pchild este un concept mai restrictiv al conceptului general pparent ex. antibiotice este un concept copil al conceptului medicamente).
            - 0 in caz contrar
- **Accessgetslotval -** folosit pentru a determina daca un utilizator are sau nu dreptul de a *citi* o informatie de gen proprietate din baza de date, IMPLEMENTEAZA REGULA DE BAZA PE CARE SE BAZEAZA RUTINELE DE IMPLEMENTARE A SECURITATII
    - Parametrii:
        - puid int – id-ul utilizatorului (cheie primara in tabela *users*)
        - pinstanceid int – id-ul entitatii la care este legat proprietatea care urmeaza a fi citita.
        - pentitytype char(1) – tipul entitatii (E-episode, R-person etc...)
    - Returneaza:
        - O valoare intreaga care este

                - 1 daca utilizatorul are drept de citire asupra proprietatii
                - 0 in caz contrar
- **Accessputslotval -** folosit pentru a determina daca un utilizator are sau nu dreptul de a *scrie* o informatie de gen proprietate din baza de date, IMPLEMENTEAZA REGULA DE BAZA PE CARE SE BAZEAZA RUTINELE DE IMPLEMENTARE A SECURITATII
    - Parametrii:
        - puid int – id-ul utilizatorului (cheie primara in tabela *users*)
        - pinstanceid int int – id-ul entitatii la care este legat proprietatea care urmeaza a fi scrisa.
        - pentitytype char(1) ) – tipul entitatii (E-episode, R-person etc...)
    - Returneaza:
        - O valoare intreaga care este
            - 1 daca utilizatorul are drept de scriere asupra proprietatii
            - 0 in caz contrar

PROCEDURI:

- **InsEntitySlot -** folosit pentru a inregistra in baza de date faptul ca un anumit slot (proprietate) poate sa apara la o anumita entitate (instanta, episod, persoana etc...)
    - Parametrii:
        - pSlot VARCHAR(256) – id-ul slotului de interes
        - vEntityType CHAR(1) – tipul entitatii la care legam slotul
    - Returneaza:
        - nimic
- **InsSlotSign -** folosit pentru a inregistra in baza de date faptul ca un anumit slot (proprietate) - daca este de tip C,CI,CS sau CSI – poate avea in campul C (instanta) doar instante care sunt in arborele conceptelor transmise ca si parametru.
    - Parametrii:
        - pSlot VARCHAR(256) – id-ul slotului de interes
        - listConcepts VARCHAR(2560) – un string compus din string-id-urlile conceptelor valide pentru slot, despartite cu virgula (ex. daca la slotul “Alergie” vrem sa intre doar medicamente care apartin la conceptele de “Antibiotic” si “Analgezic” stringul va fi de forma “Antibiotic, Analgezic”)
    - Returneaza:
        - nimic
- **InsSynonym -** folosit pentru a introduce o inregistrare in tabela *synonyms* – inregistrare care leaga doua instante din tabela *instances* si marcheaza faptul ca cele doua instante sunt sinonime.
    - Parametrii:
        - pInstance VARCHAR(256) – id-ul instantei principale
        - listSynonyms VARCHAR(256) – string compus din string-id-urile instantelor – care pot fi sinonime instantei principale – despartite cu virgula.
    - Returneaza:
        - nimic

- **InsUMLSSynonym -** folosit pentru a introduce o inregistrare in tabela *umlssynonyms* – inregistrare care leaga doua instante din tabela *instances* si marcheaza faptul ca cele doua instante sunt sinonime conform standardului impus de UMLS
    - Parametrii:
        - pInstance VARCHAR(256) – id-ul instantei principale
        - listSynonyms VARCHAR(2560) – string compus din string-id-urile instantelor – care pot fi sinonime instantei principale – despartite cu virgula.
    - Returneaza:
        - nimic

### 4.2.2 Elemente funcționale comune pentru aplicatia Client si aplicatia Server:

#### 4.2.2.1 Sistemul de securitate implementat prin standardul RSA

##### *4.2.2.1.1 Modulul RSA - Javascript*

Pentru transferul securizat de mesaje intre doctro si pacient in cadrul serviciului de mesagerie "CNmessenger" s-a dezvoltat o pereche de module ce implementeaza protocolul RSA. Pentru ca clientul la serviciile de mesagerie este implementata in Javascript s-a implementata modulul RSA in javascript iar pentru ca serverul de mesagerie este implementat in PHP s-a implementat si modulul RSA in PHP.

Modulul RSA – Javascript contine 5 scripturi JS care se afla in folderul CLIENT/RSA_JS respectiv SERVER/RSA_JS:

- rsab64.js – este script-ul principal contine functii pentru generare a cheilor, importare a unor chei externe si functii pentru encriptare respectiv decriptare a unor string-uri sau array-uri de string.
- rsa.js – operatii de conversie array < - > mpi (multi precision integer)
- base64.js – operatii de conversie intre string normal < - > base64 string
- hex.js – operatii de conversie string < - > binar < - > hexadecimal
- keygen.js – functii ce implementeaza algoritmul RSA de generare a cheilor.

##### *4.2.2.1.2 Modulul RSA - PHP*

Pentru transferul securizat de mesaje intre doctro si pacient in cadrul serviciului de mesagerie "CNmessenger" s-a dezvoltat o pereche de module ce implementeaza protocolul RSA. Pentru ca clientul la serviciile de mesagerie este implementata in Javascript s-a implementata modulul RSA in javascript iar pentru ca serverul de mesagerie este implementat in PHP s-a implementat si modulul RSA in PHP.

Modulul RSA – PHP contine 8 scripturi PHP care se afla in folderul CLIENT/RSA_PHP respective SERVER/RSA_PHP:

- RSA_B64.php – este script-ul principal contine functii wrapper pentru generare a cheilor, importare a unor chei externe si functii pentru encriptare respectiv decriptare a unor string-uri sau array-uri de string.
- RSA.php – contine clasa Crypt_RSA care este folosita ca si baza a wrapperilor (implementeaza functionalitatea de baza)

- Key.php – contine o clasa wrapper pentru stocare unei chei – are metode de auto validare, conversie in stringm, si importare
- KeyPare.php – contine o clasa wrapper pentru o pereche de chei – functii de generare a cheilor, conversii din si in string - uri ASN.1,
- MathLoader.php – pentru alegerea pachetului matematic folosit in generarea si efectuarea operatiilor asupra numerelor foarte mari folosite la generarea cheilor.
- Math/BCMath.php – pachet matematic BCMath
- Math/BigInt.php – pachet matematic BIG INT
- Math/GMP.php – pachet matematic GMP

#### 4.2.2.2 Modulul de inregistrare a mesajelor interne - LogWrite

Fisierul CLIENT/include/logWrite.php respectiv SERVER/include/logWrite.php contine functii pentru scrierea unor mesaje de gen LOG in fisiere cu nume predefinite din radacina aplicatiei “survey_log.txt” “err_log.txt” si “index.html”.

- Functia *logWriteLn* este folosita pentru a scrie mesaje generale legate de starea sistemului (pentru a inregistra faptul ca algoritmul a ajuns la un anumit punct, si pentru a afisa valorile unor variabile interne in timpul rularii algoritmului) si este folosita in general pentru debugging.
  - Primeste ca si parametru de intrare un string, un numar sau un array, in cazul in care primeste array afiseaza structura interna a array-ului.
  - Ca si iesire scrie continutul parametrilor de intrare in fisierul “survey_log.txt” urmat de un caracter de new line “\n”.
- Functia *errWriteLn* este folosita pentru a scrie mesaje de eroare in cazul unor tentative de abuz sau in situatii care nu ar trebui sa apara in regimul de functionare normala a sistemului.
  - Primeste ca si parametru de intrare un string, un numar sau un array, in cazul in care primeste array afiseaza structura interna a array-ului.
  - Ca si iesire scrie continutul parametrilor de intrare in fisierul “err_log.txt” urmat de un caracter de new line “\n”.
- Functia *logWriteLnHtml* este folosita pentru a scrie mesaje generale legate de starea sistemului atunci cand aceste mesaje au un continut informativ vast si au nevoie de o anumita formatare vizuala (output-ul este salvat intr-un fisier html si poate fi vizionat cu usurinta, fiind numit index.html este fisierul default care se afiseaza daca se navigheaza cu browserul in directorul radacina al aplicatiei). Functia este folosita in general pentru debugging.
  - Primeste ca si parametru de intrare un string, un numar sau un array, in cazul in care primeste array afiseaza structura interna a array-ului. In general va primii string formatat cu tag-uri html.
  - Ca si iesire scrie continutul parametrilor de intrare in fisierul “index.html” urmat de un caracter de new line “\n”.

#### 4.2.2.3 Modul de conversie/parsare XML-PHP – xmlParser

Fisierul CLIENT/include/xmlParser.php respectiv SERVER/include/xmlParser.php contine 4 functii si o clasa, pentru parsarea fisierelor xml conversia intre modurile de reprezentare, si gestionarea informatiilor obtinute din xml.

Exista doua moduri de reprezentare a datelor obtinute din fisierele xml:

- Un array plan (nu are array-uri in elemente)
- O structura arborescenta, alcatuita din noduri

Functiile:

- **xml_parse_into_struct_from_file** – primeste ca si parametru path-ul la fisierul xml care trebuie parsat, foloseste functia nativa PHP *xml_parse_into_struct* pentru a parsa fisierul intrun array plan, dupa modificari minore (cu functia *flatten*) (transformarea modului de reprezentare a atributelor – sa fie compatibila cu Serviciile Web) returneaza array-ul.
- **xml_parse_into_tree_from_struct** – primeste ca si parametru array-ul plan returnat de *xml_parse_into_struct_from_file* si il transforma intr-o reprezentare arborescenta, alcatuita din noduri care sunt instante ale clasei *NODE*.
- **xml_parse_into_tree_from_file** – combina functionalitatile celor doua functii primeste ca si parametru path-ul la fisierul xml, apeleaza cele 2 functii anterioare si returneaza structura arborescenta.
- **Flatten** – primeste ca si parametru de intrare un array asociativ si il transforma intr-un nested array, mutand valoarea de identificare a elementelor, in valori efective (ex. din elementul cu id-ul “ceva” care are valoarea “altceva” – “ceva” => “altceva” – va genera un element cu id numeric si valoare array de genul: 0 => array(“var” => “ceva”,”val”=>”altceva”))

Clasa:

- **NODE** – este folosit pentru reprezentarea unui nod din structura arborescenta.
  - Are Campii:
    - Name – numele nodului din xml
    - Value – valoarea text din interiorul nodului xml
    - Attributes – array asociativ cu numele atributelor => valorile atributelor nodului xml
    - childNodes – array de referinte la nodurile care sunt fii ai nodului curent
    - parentNode – referinta la nodul parinte al nodului curent
  - Are metodele:
    - __construct($n,$a,$v) – constructor-ul clasei
    - __destruct() – destructor-ul clasei
    - setParams($n,$a,$v) – pentru setarea numelui, valorii si a atributelor
    - appendChild($node) – pentru adaugarea unui nod copil
    - addParent($node) – pentru a adauga o referinta la nodul parinte
    - getFirstChild($tagName = null) – pentru a obtine primul copil (daca tagName este string atunci primul copil cu numele = tagName, daca tagName este array atunci primul copil al carui nume se afla in tagName)
    - getChildNodes($tagName = null) – pentru a obtine o lista cu toti copii (daca tagName este string atunci lista va contine doar copii cu numele = tagName, daca tagName este array lista va contine copii care )

- toString() – pentru debug, returneaza o reprezentare –generata recursiv – formatata html (cu <UL>) a structurii arborescente incepand cu nodul curent.

#### 4.2.2.4 Conventii generale

##### *4.2.2.4.1 Structuri de date in comunicare*

In comunicarea dintre client si server sunt niste grupari de date foarte frecvent intalnite, de exemplu datele referitoare la un slot – valoarea/valorile stocate in slot, modul de afisare, valori implicite etc... – sau episoadele medicale – data inceperii si incheierii, doctorul si pacientul intre care s-a desfasurat, si toate sloturile cu toate valorile stocate in ele etc... – De aceea aceste grupari frecvente de date se vor transmite intodeauna prin intermediul unor structuri de date complet definite, care vor avea (in anumite cazuri) campuri necompletate (cand nu este necesar, cand omiterea se cere explicit sau cand datele nu exista). Continutul structurilor va fi deci schimbatoare dar, arhitectura generala a structurii ramane la fel pentru a facilita o tratare uniforma in toate cazurile. Structurile de date impuse ca si standard sunt urmatoarele:

- **CSI** – un triplu ce poate contine orice combinatie a 3 valori (C-string-id-ul unei instante, S-string simplu, I-intreg simplu) se foloseste pentru reprezentarea unei valori (combinate sau simple) stocate intr-un slot sau intr-un field al unei tabele
  - **C – string** – este string-id-ul instantei ce intra ca si valoare in slot (tipul slotului trebuie sa fie C,CI,CS sau CSI ca aceasta valoare sa fie completata)
  - **S – string** – este un string simplu care intra ca si valoare in slot (tipul slotului trebuie sa fie S,CS,SI sau CSI ca aceasta valoare sa fie completata)
  - **I – string** – este un intreg simplu care intra ca si valoare in slot (tipul slotului trebuie sa fie I,CI,SI sau CSI ca aceasta valoare sa fie completata) *ATENTIE !!! ca si definitie este de de tip string si nu int pentru ca s-a extins functionalitatea sa poata contine valori numerice de toate felurile (chiar si float) nu doar numere intregi*
- **CSI_SET** – este un array de tip CSI
- **LANG_ARRAY** – este o structura ce poate contine un set de texte in mai multe limbi. Se foloseste la transmiterea unor nume/denumiri in mai multe limbi.
  - **RO – string** – este valoarea string-ului in limba romana
  - **EN – string** – este valoarea string-ului in limba engleza
  - Altele pot fi introduse in viitor
- **instance_overrides** – in cazul transmiterii unor instante (o instanta este un lucru din lumea reala semn simptom, medicament etc...) exista un set de parametrii default care se folosesc la toate instantele transmise, atunci cand se genereaza elementele de interfata grafica pe baza instantelor (de ex tipul formatarii, valoarea implicita, numele de grup etc...). Uneori insa este nevoie ca la anumite instante acesti parametrii sa se supraincarce (ex. daca la toate semnele si simptomele transmise avem doar checkbox, dar la cefalee avem si checkbox si slider) atunci aceasta structura *instance_overrides* va contine parametrii supraincarcati:
  - **formattype – string** – este tipul de formatare pentru elementul de interfata generat din instanta la care se refera supraincarcarea (din instanta se pot genra checkbox, radiobutton, checkbox + slider etc ...)

    - **defaultvalue – CSI –** este valoarea implicita luata de inputurile generate pentru interfata daca nu exista valoare setata inca
    - **name – LANG_ARRAY** – este eticheta pusa pentru instanta (ex. eticheta checkboxului) care in unele cazuri se poate supraincarca pentru a produce o interfata mai inteligibila.
    - **display – string** – este o optiune care precizeaza faptul ca elementul de interfata generat din instanta este vizibil din start sau este ascuns si se aduce la suprafata ulterior prin actiunea unor script-uri javascript (ex. cand navigam in concept – tree si inseram un checkbox care era ascuns pana la momentul respectiv)
    - **interactive – boolean** – este o optiune care spune daca controlul generat din instanta este read only sau interactiv, permite sau nu modificarea si inserarea datelor.
    - **group – string** – este numele grupului la care apartine elementul de interfata produs din instanta (folosit la gruparea butoanelor radio mutual exclusive)
- **instance –** folosit pentru gruparea datelor legate de o singura instanta, precum numele id-ul, numarul de ordine si valorile supraincarcate
    - **instance_name – LANG_ARRAY** – numele instantei (lucru din lumea reala "headache" – "cefalee") in mai multe limbi
    - **instance – string –** string-id-ul instantei (pentru identificare unica in tot sistemul)
    - **instance_order – int** – numarul de ordine a instantei. Se foloseste in cazul in care in template-urile din care se alcatuieste interfata grafica utilizator nu exista un loc definit pentru fiecare instanta, dar vrem totusi sa avem o anumite ordonare, sortarea se va face conform acestui numar de ordine.
    - **instance_overrides – instance_overrides** – valorile supraincarcate ale unei instante (explicat mai sus)
- **instance_SET** – este un array de tip *instance*
- **concepttree** – daca avem un slot (o proprietate) care poate avea ca si valoare instantele (lucrurile din lumea reala) care apartin unui concept sau sunt descendentii unui concept (adica apartin unui concept care este subconcept al unui concept) aceasta structura va contine o reprezentare a sub-arborelui unui anumit concept. (ex. conceptul de baza este *medicamente* care are ca si subconcept - printre altele – si conceptul *antibiotice*, iar conceptul *antibiotice* are niste instante precum *penicilina*, *ampicilina* etc...)
    - **concept – string** – este string-id-ul conceptului curent
    - **concept_name - LANG_ARRAY** – este numele in mai multe limbi ai conceptului.
    - **concept_order – int** – numarul de ordine a conceptului. Se foloseste in cazul in care in template-urile din care se alcatuieste interfata grafica utilizator nu exista un loc definit pentru fiecare concept, dar vrem totusi sa avem o anumite ordonare, sortarea se va face conform acestui numar de ordine.
    - **concept_instances - instance_SET** – array-ul cu instantele care sunt descendenti directi ai conceptului curent
    - **concept_children - concepttree_SET** – array-ul cu conceptele care sunt descendenti directi (subconcepte) ai conceptului curent

- **concepttree_SET** – este un array de tip *concepttree*
- **PROPERTY** – este folosit la transmiterea unor date, in general legate de o proprietate de tip slot. Cand avem de transmis o data legata de o anumita entitate (de ex. un episod medical) un singur fragment de data este stocat ori in slot ori intr-un camp static ale unei tabele. Desi cele mai multe informatii sunt stocate in sloturi, avem unele care sunt in campuri. Pentru a le trata uniform s-a definit o structura de date care sa le acomodeze pe ambele.
    - **Name – LANG_ARRAY** – numele in mai multe limbi
    - **Value – CSI_SET** – array-ul de valori stocate. In caz de slot putem avea un array cu 0, 1 sau mai multe elemente, fiecare element poate avea campurile C, S, I completate. In caz de field (camp din tabela) avem un array de 0 sau maxim 1 elemente, iar doar una din cele trei campuri C,S sau I va fi completata.
    - **Defaultvalue – CSI –** valoarea implicita
    - **Formattype – string** – tipul de formatare (tipul elementului de interfata generat)
    - **Valuetype – string –** tipul valorii (C,S,I,CS,CI,SI sau CSI)
    - **Isslot – booleanb –** true daca este slot, false daca este camp
    - **Table – string** – daca este data stocata in campul unei tabele, *Table* contine numele tabelei.
    - **Field – string** – daca este data stocata in campul unei tabele, *Field* contine numele campului.
    - **Concepttree – concepttree_SET** – pentru a stoca structura arborescenta care contine instantele ce pot intra ca si valori in sloturi (daca slotul este de tip C,CS,CI sau CSI) si pozitia lor in ierarhia de *superconcepte-subconcepte-instante ale conceptelor* definita in ontologie.
    - **Pkeyfield – string** – daca este data stocata in campul unei tabele, care este numele campului de cheie primara in tabela respectiva.
    - **Slot – string** – daca este data stocata in slot, care este string-id-ul slotului in care este stocata data.
    - **Entity – string** – daca este data stocata in slot, la care tip de entitate se refera (persoana, episod, etc...) (observatie: daca era data din tabel pe baza tabelului ar fi fost evident la ce entitate se refera)
    - **Entity_id – int** – daca este data stocata in slot, care este id-ul entitatii la care se refera (cheia primara din tabela entiotatii - daca *Entity* este “E” atunci *Entity_id* va fi cheia primara din tabela *episodes* care marcheaza episodul la care se refera data).
    - **ID – int**
        - – daca este data stocata in slot, id-ul (sau lista de id-uri) din tabela de valori (in caz ca modificam o valoare veche trebuie sa stim care este valoarea veche care trebuie modificata – cheie primara din valuesslot)
        - - daca este data stocata in camp, atunci va contina valoarea lui *Pkeyfield* pentru care intrarea in tabela *Table* denota entitatea la care se refera data.
    - **Addable – boolean** – daca este data stocata in slot, acest camp precizeaza daca slotul respectiv poate contine mai multe valori pentru aceeasi entitate sau

numai o singura valoare (ex. slotul *simptome* poate contine mai multe valori dar slotul *varsta* poate contine o singura valoare).
    - **Interactive – boolean** – precizeaza daca elementele de interfata generate pe baza informatiei continute in aceasta structura sunt interactive sau nu (sunt read-only sau pot si introduce si modifica valori)
    - **Filterfunction** – **string** – este o informatie suplimentara folosita la stocarea valorilor introduse de utilizator. Daca anumite valori mai necesita niste prelucrari inainte de stocare (ex. generarea unor informatii noi pe baza informatiilor introduse – din valoarea numerica a pulsului aplicatia introduce automat si *puls ridicati* sau *puls scazut* daca este cazul) acest camp contine numele functiei de filtrare care se va aplica asupra valorilor introduse, inainte de stocarea datelor.
    - **Display – string** – precizeaza daca elementele de interfata generate pe baza informatiei continute in aceasta structura, se vor afisa initial, sau vor ramane invizibile pana cand vor fi introduse in interfata de catre un script javascript (ex. alegerea unor elemente pe baza navigarii in arborele generat din ontologie)
- **PROPERTY_SET** – este un array de tip *PROPERTY*
- **PROPERTY_SET_LIST** – este un array de tip *PROPERTY_SET* deci un array de array-uri de tip *PROPERTY*
- **CONSULT_DESCRIPTOR** – este folosit pentru a transmite date legate de un consult medical sau un tratament medical (date legate de un episod sau step)
    - **Id – int** – id-ul de tip intreg al episode-ului (cheie primara din tabela *episodes*) daca descriem un step (id-ul va fi id-ul episodului la care apartine step-ul)
    - **processtypes_id – string** – daca este episod, poate fi C (Consult – fara pasi de tratament ) sau T (Treatment – cu pasi de tratament)
    - **entity – string** – tipul consultulu medical E (Episode) sau S(Step)
    - **entity_id – int** – id-ul de tip intreg al episode-ului sau al stepului (cheie primara din tabela *episodes* sau *steps*)
    - **patientname – string** – numele pacientului
    - **doctorname – string** – numele doctorului
    - **patientid – int** – id-ul de tip intreg al pacientului (cheie primara din *patients*)
    - **doctorid – int**– id-ul de tip intreg al doctorului (cheie primara din *mds*)
    - **datebegin – string** – data de start al episodeului sau stepului
    - **dateend – string** – data de incheiere a episodeului sau stepului
    - **pingpong – int** – marcheaza orientarea atentionarii, adica este:
        - 0 daca datele medicale trebuie aduse in atentia doctorului
        - 1 daca datele medicale trebuie aduse in atentia pacientului
        - NULL daca datele medicale sunt expirate (servesc doar ca si EHR dar nu trebuie aduse in atentia nimanui)
    - **PROPS – PROPERTY_SET** – datele (din sloturi si campuri) care se refera la episodeul sau stepul curent
    - **STEPS – CONSULT_DESCRIPTOR_SET** – daca este episod, acest camp va contina un array de *CONSULT_DESCRIPTOR*-uri completate cu datele stepurilor care apartin episodului curent.
- **CONSULT_DESCRIPTOR_SET** – este un array de tip *CONSULT_DESCRIPTOR*

- **SPECIALITY –** este folosit pentru a transmite date legate de specialitatile pe care doctorii le pot avea
    - **id – int** – id-ul de tip intreg al specialitatii
    - **speciality – string** – id-ul de tip string al specialitatii
    - **name – LANG_ARRAY** – numele specialitatii in mai multe limbi
- **SPECIALITY_SET** – este un array de tip *SPECIALITY*
- **DOCTOR_DESCRIPTOR** – este folosit pentru a transmite date legate de un doctor (date de identificare)
    - **id – int** – id-ul de tip intreg a doctorului
    - **name – string** – numele persoanei care este si doctor
    - **cnp – string** – codul numeric personoal al doctorului
    - **speciality – string –** denumirea specialitatii doctorului
- **DOCTOR_DESCRIPTOR_SET** – este un array de tip *DOCTOR_DESCRIPTOR*
- **PATIENT_DESCRIPTOR** – este folosit pentru a transmite date legate de un doctor (date de identificare)
    - **id – int** – id-ul de tip intreg a pacientului
    - **name – string** – numele persoanei care este si pacient
    - **cnp – string** – codul numeric personoal al pacientului
    - **doctorid – int –** id-ul de tip intreg al doctorului care este medicul de familie al pacientului
- **PATIENT_DESCRIPTOR_SET** – este un array de tip *PATIENT_DESCRIPTOR*

#### *4.2.2.4.2 Conventii de nume in baza de date*

- Fiecare tabela (care au cheie primara proprie – cheia primara nu este cheie straina din alt tabel) va avea cheia primara numita *id*
- Fiecare tabela care are cheie straina din alta tabela (aceeasi cheie straina apare doar o singura data) va avea campul cheii straine cu o denumire de genul *numetabela_id* unde "numetabela" es numele tabelei in care cheia straina este cheie primara.

#### *4.2.2.4.3 Conventii de stocare a datelor*

Exista posibilitatea de a stoca valori combinate in baza de date, cum s-a mentionat si in 2.1 o proprietate poate avea valori de tip C(instanta de concept) S(string) I (intreg) sau orice combinatie a acestora (CS CI SI CSI). Exista cazuri insa in care o constrangere suplimentara trebuie impusa asupra acestor valori, de exemplu daca in campul I se doreste stocarea intensitatii unui siptom este nevoie de o conventie care sa spuna care valoare a lui I la ce nivel de intensitate corespunde. Se foloseste aceeasi conventie atat la introducerea datelor cat si la afisarea lor. Atat timp cat la introducere si la extragere datele crude se filtreaza prin aceeasi conventii nu este nevoie ca sistemul de stocare a datelor (baza de date) sa aiba cunostinte asupra acestor conventii.

##### 4.2.2.4.3.1 Conventia semnelor si simptomelor cu intensitate:

Putem avea o proprietate numita "are semn sau simptom" (in general se referea la un episod medical sau la un pas de tratament medical) . Proprietatea de semn si simptom este de tip CI, in C intra instanta de semn sau simptom iar I in cele mai multe cazuri ramane gol, insa la putine semne si simptome speciale, la care este important sa se mentioneze si intensitatea

acestora I va contine prin conventie un numar intreg intre 0 si 10 care va semnala intensitatea (0 fiind intensitate scazuta, iar 10 intensitate maxima)

#### 4.2.2.4.3.2 Conventia dosajului la medicatie:

Putem avea o proprietate numita “medicatie” (in general se referea la un episod medical sau la un pas de tratament medical – specifica medicamentele prescrise de doctor pentru pacient) . Proprietatea de medicatie este de tip CS, in C intra instanta de medicament iar S va contine informatii referitoare la dosajul medicamentului. S va contine prin conventie un text formatat xml in care va apare un singur tag xml numit medication de forma : “<medication DURATION="durata" ISINFINITE="infinit" DOSE="doza" UNIT="unit" NUMBER="numar" MEALS="mese_si_conditii" OBS="observatie" />” unde:

- **durata** - este un numar intreg ce desemneaza numarul de zile cat tine tratamentul cu medicatia respectiva.
- **infinit** – este un string care poate lua valorile “true” sau “false”, “true”- daca medicatia se va lua in continuu, deci durata este nespecificata, “false” daca durata este specificata si se va respecta.
- **doza** – numar intreg ce denota numarul de unitati / o singura doza (ex. daca avem 1 doza de 500 mg, atunci *doza* va fi = 500 iar *unit* va fi = mg)
- **unit** – string care precizeaza unitatea de masura in care este precizat o singura doza. Poate fi:
  - “mg” – pentru miligrame
  - “g” – pentru grame
  - “ml” – pentru mililitrii
  - “mm2” – pentru milimetri cub
  - “cm2” – pentru centimetri cub
  - “comp” – pentru comprimate sau capsule
  - “plic” – pentru plicuri
- **numar** – numarul de doze care se iau intr-o zi.
- **mese_si_conditiib** – un string care precizeaza daca medicamentul se ia inaintea in timpul sau dupa mancat. Poate lua valorile:
  - “false” – nu are legatura cu mancarea
  - “before” – se ia medicamentul pe stomacul gol, adica inainte de mancare
  - “during” – se ia medicamentul in timpul mesei.
  - “after” – se ia medicamentul dupa mancare.
- **observatie –** daca doctorul vrea sa adauge informatii suplimentare de orice gen, care nu puteau fi inregistrate in campurile anterioare, atunci are posibilitatea sa scrie un text cu orice continut in care aduce aceste precizari suplimentare.

### *4.2.2.4.4 Conventii de formatare a datelor in raspunsul interfetei grafice (variabile POST si GET standard HTML)*

Elementele de interfata generate din aceeasi lista de PROPERTY – uri vor avea un nume de identificare comun pe care il notam de acum in colo cu ***“inputsname”*** si un nume de identificare propriu – notam cu ***“cheie”*** - care va fi un numar de la 0 la M unde M este numarul elementelor de interfata adica lungimea listei de PROPERTY-uri. In unele cazuri elementele de interfata vor fi compuse – precum am mentionat si in cazul anterior 2.3.4.3.1 – in acest caz sub-

elementele de interfata vor primi si ele un nume de identificare propriu – notam cu ***"cheia 2"*** – care va fi un numar de la 0 la N unde N este numarul sub-elementelor de interfata.

Pentru a reprezenta datele incapsulate intr-o lista de elemente PROPERTY s-au folosit input-uri care corespund campurilor din structura PROPERTY asa cum acesta este definita la 2.2.4.1:

- **Name** – nu are corespondent pentru ca nu are valoare informationala, este folosit doar ca si informatie vizuala pentru utilizator (lable-ul proprietatii).
- **Defaultvalue** – nu are corespondent pentru ca nu are valoare informationala, este folosit doar ca si informatie vizuala pentru utilizator (valoare implicita completata daca nu exista informatie introdusa explicit de utilizator).
- **Formattype** – nu are corespondent pentru ca nu are valoare informationala, este folosit doar ca si informatie vizuala pentru utilizator (tipul de element de interfata sau sub-element de interfata generat).
- **Display** – nu are corespondent pentru ca nu are valoare informationala, este folosit doar ca si informatie vizuala pentru utilizator (daca se afiseaza sau se ascund sub-elementele de interfata generate).
- **Concepttree** – nu are corespondent pentru ca nu are valoare informationala, este folosit doar ca si informatie vizuala pentru utilizator (lista cu valori posibile, si ierarhia lor – nu se intorc toate valorile ci doar cele care s-au introdus iar ele intra in *value*).
- **Addable** – nu are corespondent pentru ca nu are valoare informationala, este folosit doar ca si informatie vizuala pentru utilizator (daca exista deja o valoare asociata cu aceasta proprietate, atunci se permite introducerea unei valori suplimentare doar daca addable-este adevarat – elementul de interfata care permite introducerea unei valori noi (si nu modificarea unei valori existente) se genreaza sau nu).
- **Interactive** – nu are corespondent pentru ca nu are valoare informationala, este folosit doar ca si informatie vizuala pentru utilizator (Daca este fals atunci se va genera un element de interfata pasiv, care permite doar vizualizarea datelor, iar in caz contrar se afiseaza un element de interfata intercativ care permite modificarea/stergerea elementului de interfata).
- **Valuetype** – tipul valorii returnate C,S,I,CS,SI,CI sau CSI- etse important pentru stocarea valorii in baza de date, deci se introduce intr-un input de tip hidden cu numele: ***inputsname***_*isslot*_***cheie***
- **Isslot** – flag-ul care specifica daca informatia provine din (si trebuie reintrodusa in) slot sau camp static din tabela. Valoarea se introduce intr-un input numit ***inputsname***_*isslot*_***cheie.***
- **Table** – Numele tabelei din care provin (si in care trebuie reintroduse) datele (daca isslot este fals). Valoarea se introduce intr-un input numit ***inputsname***_ table***_cheie.***
- **Field** – Numele campului din care provin (si in care trebuie reintroduse) datele (daca isslot este fals). Valoarea se introduce intr-un input numit ***inputsname***_ field***_cheie.***
- **Slot** – Numele slotului din care provin (si in care trebuie reintroduse) datele (daca isslot este adevarat). Valoarea se introduce intr-un input numit ***inputsname***_ slot ***_cheie.***

- **Pkeyfield** – Numele campului care este cheie primara din ***table*** este important pentru identificarea inregistrarii din tabela in care se va modifica campul (daca isslot este fals). Valoarea se introduce intr-un camp numit ***inputsname***_ pkeyfield ***_cheie.***
- **Entity** – Tipul entitatii la care se refera informatia (cand isslot este adevarat, - important la stocare). Valoarea se introduce intr-un camp numit ***inputsname***_ entity***_cheie.***
- **Entity_id** – Id-ul de tip intreg al entitatii la care se refera informatia (cand isslot este adevarat, - important la stocare). Valoarea se introduce intr-un camp numit ***inputsname***_entity_id***_cheie***
- **Filterfunction** – este numele functiei folosit drept filtru prin care se va trece valoarea inainte de stocare. Valoarea se introduce intr-un camp numit ***inputsname****_filterfunction****_cheie.***
- **Value** – In PROPERTY-ul original acesta era un sir de elemente C, S si I, se transpune in mai multe input-uri cu numele de forma:
  - ***inputsname_ cheie _cheia 2***_C – pentru valoare de instanta de concept.
  - ***inputsname_ cheie _cheia 2***_S – pentru valoarea de string
  - ***inputsname_ cheie _cheia 2***_I – pentru valoarea numerica

  Pentru a putea aduna valorile inapoi intr-un sir de valori ***cheia 2*** este un intreg care merge de la 0 la N unde N este numarul de valori. La reconstructie mai este nevoie si de numarul N, care este introdusa intr-un input cu numele ***inputsname _ cheie***_cnt.
- **ID** – In PROPERTY-ul original acesta era un sir de ID-uri cu lungimea X (unde X <= N). Daca o valoare este introdusa in prealabil, si se modifica sau se sterge, trebuie sa se cunoasca id-ul inregistrarii din tabelul ***valuesslot*** care ii corespunde (in cazul in care avem slot-uri cu valori multiple, nu ne ajunge sa stim care este entitatea si slotul ci trebuie sa cunoastem si id-ul valorii din slot pentru a identifica in mod unic inregistrarea – doar asa se poate modifica/sterge). Pentru fiecare valoare veche va exista un input cu numele ***inputsname_ cheie _cheia 2***_id care va contine ID-ul de valuesslot pentru valoarea respectiva. Cand se reintroduc valorile si ID-urile in PROPERTY, acestea se organizeaza in asa fel incat primele X valori si cele X id-uri se introduc in paralel, astfel incat sa existe corespondenta de numar de ordine dintre o valoare si ID-ul sau. Ultimele N-X valori care sunt valorile introduse noi, nu vor avea corespondent in sirul ID.

### *4.2.2.4.5 Politici de securitate*

Deoarece sistemul gestioneaza date de natura personala si / sau medicala definirea si respectarea unor policy-uri de securitate este obligatorie. Termenii si conditiile de acces la date implementate de sistem se definesc astfel:

- Orice utilizator are dreptul sa citeasca si sa scrie datele care tind de persoana lui (slot-uri legate de entitati person mds si patient)
- Orice doctor are dreptul sa citeasca datele personale ale pacientilor sai, dar nu si sa le modifice.

- Orice doctor are dreptul sa citeasca datele personale ale pacientilor care apartin unui doctor subordonat (ierarchia intre doctori se exprima prin ierarhia utilizatorilor lor, in tabela *userhierarchies*)
- Fiecare pacient are dreptul sa-si vizualizeze datele medicale (date legate de episod medical sau pas de tratament) dar nu sa si le modifice.
- Un doctor are acces de citire la datele medicale create intre datele X si Y (episodul medical de care se leaga datele a fost inceput la data X si incheiat la data Y) ale unui pacient daca:
    - Pacientul este tratat de el (daca este pacientul lui)
        - Daca pacientul este inscris la el (este doctor de familie)
        - Dac exista un episod medical cu data de inceput P si data de sfarsit Q cu conditia ca:
            - P>X
            - sau X>P si Q>X adica episodul a inceput inainte de X dar s-a terminat dupa X (pe parcursul episodului de la X sau poate chiar dupa)
            - sau X>P si Q=null, adica episodul a inceput la P inaintea lui X dar nu s-a terminat inca
    - Orice doctor are acces la acele informatii medicale la care subordonatii lui au acces
- Un doctor are acces de scriere si modificare asupra datelor medicale, doar daca datele au fost create de el, si daca episodul medical nu este depasit (nu se permit alterarea datelor din trecut).

#### 4.2.2.5 Fisierul de configurare a sistemului

Pentru un deployment cat se poate de usor a sistemului, setarile de configurare atat pentru parte CLIENT cat si pentru partea SERVER sunt salvate intr-un singur fisier de configurare din radacina aplicatiei, numit *config.inc.php*. Acest fisier contine datele de acces la baza de date folosit de modulul SERVER, si datele de conectare la serviciile web pentru CLIENT. De regula datele de conectare la serviciile web vor poanta la modulul SERVER.

### 4.2.3 Aplicatia "Client"

Este componenta aplicatiei care este responsabila pentru managementul interfetei utilizator, nu contine elemente de logica de buisness, toate operatiile de buisness logic sunt efectuate de Server (server-ul de servicii web), clientul face doar apel la functionalitatile oferite de server.

#### 4.2.3.1 Structura aplicatiei Client

Scriptul principal al aplicatiei client este CLIENT/index.php in acesta sunt incluse toate celelalte parti care alcatuiesc interfata:

PHP SCRIPTURI:

- **include/logWrite.php** - contine functii pentru scrierea unor mesaje de gen LOG in fisiere cu nume predefinite din radacina aplicatiei. Functiile sunt descrise in cap. 2.2.2.
- **include/utility_functions.php** - contine functii de uz general de manipulare a unor array-uri, conversie din array-uri imbricate in array plan, etc... Functiile sunt descrise la 2.3.3
- **include/xmlParser.php** – contine 4 functii si o clasa, pentru parsarea fisierelor xml conversia intre modurile de reprezentare, si gestionarea informatiilor obtinute din xml. Descriere detaliata la 2.2.3.
- **SOAP_Wrapper.php** – contine o functie wrapper numita callWebService care apeleaza un serviciu web de pe serverul de WS default sau un alt WS server a carui adresa se paseaza ca si parametru. Descriere detaliata in 2.3.2
- **SessionManagement.php** – contine logica de manipulare a sesiunii de login, alegerea si stocarea limbii alese, logare, delogare. Descriere detaliata la 2.3.6
- **language/RO.php** – contine varianta romana a cuvintelor si expresiilor folosite in pagina. Textele sunt stocate intr-un array asociativ numit $lang vare contine ca si identificator un string simplificat dar reprezentativ in limba engleza si ca si valoare string-ul folosit in pagina (in limba romana).
- **language/EN.php** – contine varianta engleza a cuvintelor si expresiilor folosite in pagina. Textele sunt stocate intr-un array asociativ numit $lang vare contine ca si identificator un string simplificat dar reprezentativ in limba engleza si ca si valoare string-ul folosit in pagina(in limba engleza).
- **include/bTemplate.php** – contine clasa bTemplate [bTemplate] originala cu imbunatatirile aduse pentru a o adapta mai bine functionalitatii siteului. Descriere detaliata la 2.3.4.1
- **include/inteligentTemplateFill.php** – contine clasa bTemplateInteligentFill care este o clasa wrapper pentru bTemplate, constructorul primeste ca si parametru o instanta a clasei bTemplate cu care va lucra. Clasa ofera functionalitati de completarea a template-urilor in mod inteligent. Descriere detaliata la 2.3.4.2
- **menuFunctions.php** – contine 3 structuri predefinite si 3 functii pentru alcatuirea meniurilor pentru doctor,pacient si vizitator. Structurile definesc continutul textual si referinta linkurilor din meniuri, iar functiile construiesc meniul din informatiile incapsulate in structuri. Descriere detaliata la 2.3.5.
- **include/CNmessengerResponse.php** – este scriptul php responsabil pentru comunucarea prin ajax, incapsuleaza functionalitatile de comunicare utilizate de serviciul de mesagerie CNmessenger. Desi este inclus in index.php ca toate celelalte scripturi, functionalitatea sa este complet separata de modul normal de functionare. Cand pagina este accesata in mod standard de catre utilizator, prezenta scriptului nu are nici un efect asupra output-stream-ului. Daca insa pagina este accesata prin ajax cu un parametru special, output stream – ul este generat in exclusivitate de acest script. Dupa ce genereaza tot output-ul face apel la functia die(), astfel impiedicand restul scripturilor php sa-si produca efectul pe output stream.
- **include/outputCreateFunctions.php** – este scriptul care contine functiile de generare a unor elemente de interfata care au ca si sursa de date un slot sau un camp din tabela, si ***nu sunt*** interactive. Detalii la 2.3.4.3.1

- **include/inputCreateFunctions.php** – este scriptul care contine functiile de generare a unor elemente de interfata care au ca si sursa de date un slot sau un camp din tabela, si ***sunt*** interactive. Detalii la 2.3.4.3.2.
- **SPECIFIC_PAGES/navigator.php** – continutul paginii 0 din interfata utilizator specifica vizitatorului. Cand un utilizator acceseaza sistemul pentru prima data, si nu este logat inca , sau nici nu are cont de utilizator, prima pagina va contine un text introductiv/descriptiv si niste instructiuni simple pentru utilizarea corecta a sistemului. De asemenea contine si un navigator grafic care ii permite utilizatorului sa navigheze pe pagina unde isi poate crea un cont pe sistem, in cazul in care inca nu are.
- **SPECIFIC_PAGES/subscribe.php** – continutul paginii 1 din interfata utilizator specifica vizitatorului. Aceasta pagina contine un formular pe care un utilizator nou il completeaza la inceput, pe baza datelor completate va obtine un cont de utilizator.
- **SPECIFIC_PAGES/patient/page.0.php** – continutul paginii 0 din interfata utilizator specifica pacientului. Contine un text introductiv/descriptiv si niste instructiuni simple pentru utilizarea corecta a sistemului. De asemenea contine si un navigator grafic care ii permite utilizatorului de tip pacient sa navigheze la paginile care prezinta functionalitatile (serviciile medicale) accesibile pacientului.
- **SPECIFIC_PAGES/patient/page.1.php** – continutul paginii 1 din interfata utilizator specifica pacientului. Contine un formular in care pacientul isi poate completa/modifica datele personale.
- **SPECIFIC_PAGES/patient/page.2.php** – continutul paginii 2 din interfata utilizator specifica pacientului. Contine controale pentru navigarea intre/cautarea/vizualizarea episodelor medicale din trecut (care pot sau nu fi in continua derulare pana in prezent), deci prin aceasta pagina se acceseaza istoricul medical.
- **SPECIFIC_PAGES/ patient/page.4.php** – continutul paginii 4 din interfata utilizator specifica pacientului. Contine un formular in care pacientul poate efectua un consult offline (isi completa semnele si simptomele, si cere o rezolutie de la doctor).
- **SPECIFIC_PAGES/doctor/page.0.php** – continutul paginii 0 din interfata utilizator specifica doctorului. Contine un text introductiv/descriptiv si niste instructiuni simple pentru utilizarea corecta a sistemului. De asemenea contine si un navigator grafic care ii permite utilizatorului de tip doctor sa navigheze la paginile care prezinta functionalitatile (serviciile medicale) accesibile doctorului.
- **SPECIFIC_PAGES/doctor /page.1.php** – continutul paginii 1 din interfata utilizator specifica doctorului. Contine un formular in care doctorul isi poate completa/modifica datele personale.
- **SPECIFIC_PAGES/doctor /page.4.php** – continutul paginii 4 din interfata utilizator specifica doctorului. Contine lista tuturor consultatiilor offline noi si continuate, efectuate de pacienti si un formular in care doctorul poate modifica/adauga date referitoare la episodul medical (consultatia offline) pe care il solutioneaza.

CSS-URI:

- **CSS/general.css** – clase css generale, pentru structura generala a paginii, elemente comune dintre interfata doctor si interfata pacient, elemente de pozitionare.

- **CSS/menu.css** – clase css pentru meniul principal al paginii (bara de meniuri de sub header)
- **CSS/box.css** – clase css pentru meniuri secundare de tip box, care apar la nevoie in mai multe pagini, sub forma de cutii de liste ce contin intrari de meniu, in partea laterala a paginii.
- **CSS/big_icons.css** – clase css pentru icoanele mari grafice din paginile de navigare, atat la doctor cat si la pacient
- **CSS/patient.css** – clase css specifice elementelor din paginile pacientului
- **CSS/doctor.css** – clase css specifice elementelor din paginile doctorului
- **CSS/forms.css** – clase css pentru elemente din formulare.
- **CSS/slider.css** – clase css pentru controalele de tip slider din form-uri
- **CSS/CNmessenger.css** – clase css pentru elemente legate de serviciul de mesagerie Cnmessenger.
- **CSS/datetime.css** – clase css pentru controalele de tip date, time, si datetime
- **CSS/calendar.css** – clase css pentru managger-ul de calendar

JAVASCRIPTURI:

- **JS/lib/prototype.js** – script din colectia *script.aculo.us* pentru generarea unor efecte dinamice de uz general [Script.aculo.us]
- **JS/src/scriptaculous.js** – script din colectia *script.aculo.us* pentru generarea unor efecte dinamice de uz general [Script.aculo.us]
- **JS/src/unittest.js** – script din colectia *script.aculo.us* pentru generarea unor efecte dinamice de uz general [Script.aculo.us]
- **JS/src/builder.js** – script din colectia *script.aculo.us* pentru generarea unor efecte dinamice de uz general [Script.aculo.us]
- **JS/src/controls.js** – script din colectia *script.aculo.us* pentru generarea unor efecte dinamice de uz general [Script.aculo.us]
- **JS/src/effects.js** – script din colectia *script.aculo.us* pentru generarea unor efecte dinamice de uz general [Script.aculo.us]
- **JS/src/slider.js** – script din colectia *script.aculo.us* pentru generarea unor efecte dinamice de uz general [Script.aculo.us]
- **JS/src/sound.js** – script din colectia *script.aculo.us* pentru generarea unor efecte dinamice de uz general [Script.aculo.us]
- **JS/src/dragdrop.js** – script din colectia *script.aculo.us* pentru generarea unor efecte dinamice de uz general [Script.aculo.us]
- **JS/GENERAL.js** – contine functii generale de manipualre a DOM-ului
- **JS/CNmessengerAjax.js** – contine functiile care gestioneaza comunicarea prin ajax (folosit la implementarea serviciului de mesagerie)
- **JS/CNmessengerWindow.js** – contine functii care gestioneaza elementele de interfata folosite de serviciile de mesagerie (ex. gestionarea updatarea window-ului in care pacientul vorbeste in direct cu doctorul)
- **JS/dateTimeChoosers.js** – contine functii ce genereaza un element de interfata prin care se pot introduce date si timpi intr-un form. Functia principala primeste ca si parametru intervalul de timi intre care se va incadra timpul sau data generata, la incheiere introduce un datetime in format string intr-un input prestabilit.

- **JS/GENERAL_TIME.js** – contine functii generale folosite la calcule de timpi (transformare din timestamp in string si invers, calculul zilelor – corespondenta dintre data si zi a saptamanii etc ... )
- **JS/calendar.js** – contine functii ce genereaza un element de interfata prin care se poate efectua manipularea unui calendar de evenimente. Functia principala primeste ca si parametru un set de evenimente existente (cu data durata si denumire) un set de intervale de timp si date invalide. La incheiere returneaza data si durata unui eveniment nou introdus.
- **JS/treeControl.js** – contine functii ce genereaza un element de interfata prin care se poate parcurge un arbore si se pot alege frunce ale arborelui (se foloseste la alegerea instantei care se doreste sa se introduca ca si valoare a unui slot de tip C, CI, CS sau CSI – de ex. pentru alegerea medicamentului concret din arborele "medicamente" atunci cand in slotul "Medicatie" se doreste introducerea acestuia)
- **JS/sliderControl.js** – contine functii ce manipuleaza elementele de interfata de tip *slider* (sunt folosite la precizarea intensitatii unor simptome – ex. slotul "SemneSiSimptome" este de tip CI deci intra in slot o valoare C de tip instanta care specifica tipul simptomului – de ex. durere de cap - iar daca e cazul in I intra un integer – prin conventie intre 1 si 10 – care specifica intensitatea simptomului - de ex. daca I=2 inseamana ca exista o durere de cap de intensitate slaba - )
- **JS/printer.js** – contine functii cu care se poate imprima o parte a unei pagini HTML, functiile creaza un Window nou (window sau tab nou in browserul din care este accesat pagina) in care copiaza o sub-portiune a paginii originale si apeleaza functia window.print(). Apare dialogul de imprimare standard dupa care se trimite jobul la imprimanta si window-ul creat se inchide automat.
- JS/openclose.js – contine o functie care deschide sau inchide un element de tip container (care contine alte elemente) din interfata (ex. un DIV DOM) – este un wrapper pentru efectele de "TOGGLE" din colectia de scripturi *script.aculo.us*
- RSA_JS/rsa.js – script din colectia RSA_JS – explicat mai sus.
- RSA_JS/base64.js – script din colectia RSA_JS – explicat mai sus.
- RSA_JS/hex.js – script din colectia RSA_JS – explicat mai sus.
- RSA_JS/keygen.js – script din colectia RSA_JS – explicat mai sus.
- RSA_JS/RSA_B64.js – script din colectia RSA_JS – explicat mai sus.
- JS/swfobject.js – script folosit pentru inserarea dinamica a unor obiecte de tip swf (flash) in pagina (este folosit la inserarea player-ului flash folosit la mesaje de sistem audibile)
- tiny_mce/tiny_mce.js – script ce genereaza un element de interfata de tip text editor cu care se pot introduce texte cu formatare (font, size, bold, italic etc... ) [tiny_mce]

#### 4.2.3.2 Modulul de cuplare (wrapper) SOAP pentru clientul WS

Fisierul *CLIENT/SOAP_Wrapper.php* contine o functie wrapper numita callWebService care apeleaza un serviciu web de pe serverul de WS default sau un alt WS server a carui adresa se paseaza ca si parametru.

- **callWebService** – functie care primeste numele unei functii remote (serviciu web), un set de parametrii, si optional o adresa de wsdl si un host. Functia creaza un *SoapClient* prin

care apeleaza functia remote cu numele dat, pasand parametrii primiti si returneaza valoarea primita de la functia remote. Daca parametrii de wsdl si host sunt setati, se vor folosi in loc de cele dafault pentru a accesa serverul de servicii web.

#### 4.2.3.3 Functii utilitare de uz general

In fisierul *include/utility_functions.php* sunt definite niste functii de uz general de manipulare a unor array-uri, conversie din array-uri imbricate in array plan, etc...

- **concat** – concateneaza 2 sau 3 array-uri, returneaza un array compus prin concatenare.
- **sortinsertInstances –** insereaza un element (in cazul uzual o instanta) intr-un array de elemente conform unui numar de ordine. Elementele sunt array-uri asociative care au un camp numit "instance_order". Functia efectueaza un sort-insert al elementului nou, pe baza numerelor intregi din "instance_order".
- **convertConceptTreeToOrderedInstanceList** – scoate elemente (instante) dintr-o structura arborescenta (elementele sunt array-uri asociative si au sus mentionat-ul camp "instance_order") si le insereaza intr-o lista cu ajutorul functiei *sortinsertInstances* astfel lista creata din arbore va fi ordonata conform "instance_order" al elementelor.
- **getConceptAncestorsOfInstance** – primeste ca si parametru un nod de tip frunza (o instanta) dintr-un tree (tree-ul de instante si concepte) extrage din tree si returneaza o lista, in care pe prima pozitie va fi numele (string-id-ul) frunzei (instantei) iar pe elemente succesive vor fi numele (string-id-urile) nodurilor stramosi (conceptele) din care provine, pana la nodul radacina (conceptul de baza).
- **Contains** – determina daca un array contine un element anume sau nu.
- **fillNewPropsFromOldProps** – daca avem doua liste cu elemente de tip PROPERTY si vrem sa construim o singura lista din cele doua, aceasta functie supraincarca(daca corespund)/appendeaza(daca nu corespund) elemetele dintr-o lista cu/la elementele din cealalta lista. Functia se utilizeaza de exemplu in cazul in care avem un set mic dar "up to date" de date obtinute de la utilizator si un set mare dar "depricated" de date obtinute din baza de date si vrem sa afisam toate datele intr-un formular astfel incat datele mai recente sa se supraincarce peste corespondentele lor din datele invechite, dar daca exista date invechite care nu au corespondent intre datele noi, atunci acelea sa ramana neschimbate si sa se afiseze.

#### 4.2.3.4 Motorul de generare a vederilor pe baza de modele (template-uri)

##### *4.2.3.4.1 Modulul de baza a motorului - bTemplate*

Fisierul CLIENT/include/bTemplate.php contine definitia clasei bTemplate [bTemplate] care contine anumite parti originale (preluate din originalul bTemplate) si anumite parti adaugate (adaugate de programatorii CardioNET). Metodele principale folosite sunt urmatoarele:

- **bTemplate** – constructor original.
- **Set** – functie originala. Folosita pentru setarea valorii asociate cu un tag de template
- **Set_cloop** – functie originala. Folosita pentru setarea valorilor asociate cu un loop conditional de template
- **Reset_vars** – functie originala. Folosita pentru stergerea tuturor valorilor setate.

- **Fetch** – functie originala. Folosita pentru aplicarea valorilor asupra unui fisier de template si obtinerea string-ului rezultant.
- **Get** – functie adaugata. Folosita pentru a obtine o valoare setata anterior (folosit la valori care se appendeaza)

O descriere detaliata a structurii si a sintaxei fisierelor template folosite de bTemplate se gaseste pe http://www.massassi.com/bTemplate/index.php?page=docs&section=intro

#### *4.2.3.4.2 Modulul de generare inteligenta a vederilor - intelligentTemplateFill*

Fisierul CLIENT/include/inteligentTemplateFill.php contine definitia clasei bTemplateInteligentFill care este o clasa wrapper pentru bTemplate, constructorul primeste ca si parametru o instanta a clasei bTemplate cu care va lucra. Clasa ofera functionalitati de completarea a template-urilor in mod inteligent si optiunea de a completa mai multe fisiere template in acelas timp. Metode definite in calasa bTemplateInteligentFill:

- **__construct** – constructorul clasei, primeste ca si parametru o instanta a clasei bTemplate pe care o va folosi ca si miez, si un array de fisiere template pe care le va folosi.
- **set_unset_tags** – metoda pentru a seta toate tag-urile de un anumit tip (ex. toate if-urile) care nu sunt inca setate la o valoare default (ex. false)
- **find_tagnames** – metoda de a gasi numele tuturor tag-urilor template de un anumit tip (ex. sa aflam numele tuturor if-urilor tin toate fisierele template folosite, ca dupa aceea sa le putem parcurge si sa le setam pe acelea care nu sunt inca setate – folosit in *set_unset_tags*)
- **set_inputs_fromPROPS –** metoda de a seta in mod inteligent elementele de interfata ce alcatuiesc un form parcurge toate elementele si face apel la functiile set_field sau set_slot daca elementul de interfata s-a generat din camp respectiv din slot.

- **set_field** – metoda care insereaza elementul de interfata generat din continutul unui camp din baza de date.
  - In prima faza se cauta un tag template de forma <tag:FIELD_*numeleCampului*_TABLE_*numeleTabelei*_value />
  - Daca tag dedicat campului nu exista, elementul de interfata se introduce la tag-ul de tip *generic bin* care contine toate elementele care nu au tag template dedicat. Tag-ul *generic bin*-ului are forma <tag: genericBin />
- **set_slot field** – metoda care insereaza elementul de interfata generat din continutul unui slot.
  - Prima data se incearca gasirea unui tag de template dedicat instantei respective de forma <tag:SLOT_*numeleSlotului*_INSTANCE_*numeleInstantei*_value /> unde *numeleSlotului* este string-id-ul slot-ului in care intra instanta curenta, iar *numeleInstantei* este string-id-ul instantei curente
  - Daca tag de template dedicat instantei nu exista se incearca gasirea unui tag care corespunde unui concept predecesor al instantei. Se parcurge un array de predecesori obtinut prin metoda descrisa la 2.3.3 – functia *getConceptAncestorsOfInstance.* Tag-ul conceptului este de forma <tag:SLOT_*numeleSlotului*_CONCEPT_*numeleConceptului*_value /> unde *numeleSlotului* este string-id-ul slot-ului in care intra instanta curenta, iar *numeleConceptului* este string-id-ul concepului curent.

- Daca tag dedicat campului nu exista, elementul de interfata se introduce la tag-ul de tip *generic bin* care contine toate elementele care nu au tag template dedicat. Tag-ul *generic bin*-ului are forma <tag: genericBin />

- **templateFilesContainTag** – metoda care scaneaza toate fisierele template pentru a gasi un anumit tag de template. Returneaza true daca tag-ul cautat exista in una sau mai multe fisiere template, returneaza false in caz contrar.
- **getSlotKey** – metoda care returneaza cheia de tip integer cu care s-a generat elementul de interfata pentru un slot anume. Pentru a avea o lista continua de intrari numarate si ordonate total, numele acestora se genereaza nu din numele slotului sau campului din care se genereaza, ci i se aloca un numar de ordine care este un intreg intre 0 si numarul total al elementelor de interfata. Dar la unele scripturi ce controleaza interfata este nevoie sa se cunoasca numele elementelor de interfata ce corespund unui anumit slot, de aceea asocierea dintre slot si numarul de ordine se stocheaza in momentul generarii elementelor de interfata si se poate extrage cu functia *getSlotKey.*

### *4.2.3.4.3 Modul de generare a elementelor de interfata – inputCreateFunctions, outputCreateFunctions*

#### 4.2.3.4.3.1 Output create functions:

Cand se doreste doar vizualizarea, fara posibilitati de modificare, a unor proprietati (slot-uri/campuri + continutul lor) apartinand unei entitati (episod medical / persoana etc...) datele se extrag din baza de date, se toarna intr-un array de elemente de tip PROPERTY (descris la 2.2.4.1). Pentru a obtine un element de interfata static generat dintr-un astfel de PROPERTY, PROPERTY-ul se transmite ca si parametru functiei *createOutput* care pe baza informatiilor de formatare din PROPERTY face apel mai departe la functia coraspunzatoare de generare a unui element specific de interfata. (exista generatoare separate pentru date de tip, text scurt, text lung, numar intreg, data calendaristica etc...) Functiile asociate cu generarea/compunerea elementelor pasive sunt:

- **createOutput** – este o functie wrapper, care se apeleaza mereu, pe baza tipului de element de interfata care trebuie generat, face apel mai departe la functiile generatoare specifice fiecarui tip de element de interfata.
- **createBooleanOutput** – se apeleaza cand tipul elementului este boolean. Afiseaza o singura proprietate cu o singura valoare de tip boolean (true/false).
- **createNumericOutput** – se apeleaza cand tipul elementului este intreg. Afiseaza o singura proprietate cu o singura valoare de tip intreg (de fapt numeric – chiar si float).
- **createStringOutput** – se apeleaza cand tipul elementului este string scurt. Afiseaza o singura proprietate cu o singura valoare de tip string scurt.
- **createLongstringOutput** – se apeleaza cand tipul elementului este string lung. Afiseaza o singura proprietate cu o singura valoare de tip string lung.
- **createDateOutput** – se apeleaza cand tipul elementului este data calendaristica. Afiseaza o singura proprietate cu o singura valoare de tip data calendaristica (luna/data/an).
- **createTimeOutput** – se apeleaza cand tipul elementului este de tip timp. Afiseaza o singura proprietate cu o singura valoare de tip timp (ora/minut/secunda).

- **createDateTimeOutput** – se apeleaza cand tipul elementului este de tip moment in timp. Afiseaza o singura proprietate cu o singura valoare de tip moment absolut de timp (luna/data/an ora/minut/secunda).
- **getOutputElementsFromConcepttree** – cand pentru o anumita proprietate putem avea valori multiple (in cazul in care avem un slot si se stie ca in slotul respectiv poate intra ca si valoare oricare dintr-o lista de instante posibile, eventual impreuna cu valori S - string sau I - int ), si valorile sunt cunoscute, avem cazuri in care pentru o singura proprietate trebuie sa construim mai multe elemente de interfata (de ex. daca avem proprietatea numita semn si simptom in care intra N semne si simptome – toate cunoscute din ontologie – atunci trebuie sa construim N sub-elemente de interfata – fiecare cu propriul checkbox si lable – una pentru fiecare instanta.) Aceasta functie face o parcurgere a structurii arborescente in care sunt stocate valorile posibile (instantele posibile) si apeleaza functia *getDisplayFromInstance* de generare a unui singur sub-element de interfata, combinand sub-elementele se genereaza un element de interfata compus.
- **getDisplayFromInstance** – aceasta functie decide conform tipului de sub-element de interfata cerut, care va fi functia apelata pentru generarea sub-elementului de interfata, apeleaza functia generatoare si returneaza elementul de interfata generat.
- **getInstanceWithCheck** – genereaza un sub-element de interfata compus din numele instantei si un icon care specifica prezenta sau absenta instantei respective ca si valoare a slotului la care apartine elementul de interfata compus printre altele si din sub-elementul acesta.
- **getInstanceWithNoCheck** – genereaza un sub-element de interfata compus doar din numele instantei.
- **getInstanceSlider** – genereaza un sub-element de interfata compus din numele instantei si un slider care afiseaza numarul intreg asociat cu instanta (ex. intensitatea unui simptom) – trebuie sa avem tip de valori (valuetype) CI sau CSI . Mentionam faptul ca numarul trebuie sa fie intre 0 si 10 sa produca un slider valid. Conventia este descrisa la 2.2.4.3.1.
- **getInstanceString** – genereaza un sub-element de interfata compus din numele instantei si un camp de tip text. In campul de text se afiseaza continutul stringului S asociat cu instanta C – trebuie sa avem tip de valori (valuetype) CS sau CSI.
- **getInstanceMedication** – genereaza un sub-element de interfata compus din numele instantei (acest generator se foloseste doar la medicamente, deci instanta va fi un medicament) si un icon dotat cu un popup, in popup se afiseaza dosajul medicamentului, informatiile fiind luate din - si stocate in - campul string asociat cu instanta – trebuie sa avem tip de valori (valuetype) CS sau CSI. Conventia este descrisa la 2.2.4.3.2.
- **getString**– genereaza un sub-element de interfata compus din un camp de tip text. In campul de text se afiseaza continutul stringului S– trebuie sa avem tip de valori (valuetype) S, SI, CS sau CSI.

- **getSlider –** genereaza un sub-element de interfata compus dintr-un singur slider. Sliderul afiseaza valoarea I – numerica conform conventiei de la 2.2.4.3.1.– trebuie sa avem tip de valori (valuetype) I, SI, CI sau CSI.

#### 4.2.3.4.3.2 Output create functions:

Cand se doreste vizualizarea si/sau modificarea unor proprietati (slot-uri/campuri + continutul lor) apartinand unei entitati (episod medical / persoana etc...) datele se extrag din baza de date, se toarna intr-un array de elemente de tip PROPERTY (descris la 2.2.4.1). Pentru a obtine un element de interfata interactiv generat dintr-un astfel de PROPERTY, PROPERTY-ul se transmite ca si parametru functiei *createInput* care pe baza informatiilor de formatare din PROPERTY face apel mai departe la functia coraspunzatoare de generare a unui element specific de interfata. (exista generatoare separate pentru date de tip, text scurt, text lung, numar intreg, data calendaristica etc...)

Comparativ la functiile de generare de output-uri, in cazul inputurilor pe langa functiile care genereaza elementele de interfata, mai este nevoie si de functii care sunt menite sa adune datele turnate in interfata utilizator si modificate prin interactiunea cu utilizatorul, si sa le toarne inapoi in structuri de tip PROPERTY – pentru salvarea in BD. Elementele de interfata generate din aceeasi lista de PROPERTY – uri vor avea un nume de identificare comun pe care il notam de acum in colo cu ***"inputsname"*** si un nume de identificare propriu – notam cu ***"cheie"*** - care va fi un numar de la 0 la M unde M este numarul elementelor de interfata adica lungimea listei de PROPERTY-uri. In unele cazuri elementele de interfata vor fi compuse – precum am mentionat si in cazul anterior 2.3.4.3.1 – in acest caz sub-elementele de interfata vor primi si ele un nume de identificare propriu – notam cu ***"cheia 2"*** – care va fi un numar de la 0 la N unde N este numarul sub-elementelor de interfata.

Functiile asociate cu generarea/compunerea/dezcompunerea elementelor active (adica conversia PROPERTY->element de interfata si element de interfata->PROPERTY) sunt:

- **createSummary** – este o functie care creaza un sumar de informatii referitor la elementele de interfata generate. De fapt genereaza un input de tip hidden in care stocheaza numarul elementelor de interfata generate dintr-un set de elemente de tip PROPERTY.
- **createHiddenInput** – este o functie care creaza un sumar de informatii referitor la un singur element de interfata generat. Creaza un set de input-uri de tip hidden in care stocheaza informatii referitoare la un element de interfata generat dintr-un singur PROPERTY. Un PROPERTY poate descrie ori un grup de informatii referitoare la un camp dintr-o tabela sau un grup de informatii referitoare la un slot, exista deci astfel de inputuri care sunt comune, dar exista si unele care sunt specifice grupului de informatie din PROPERTY:
    - Elemente comune:
        - **Isslot** (inputul are numele = ***inputsname***_*isslot*_***cheie*** ) specifica daca informatia provine din slot sau camp.
        - **Valuetype** (inputul are numele = ***inputsname***_*isslot*_***cheie*** ) specifica daca informatia provine din slot sau camp.

        - **Filterfunction** (inputul are numele = ***inputsname_filterfunction_cheie*** ) specifica numele functiei de tip filtru prin care trebuie trecut PROPERTY-ul dupa ce s-a reconstruit din elementele de interfata, si inainte de a fi stocat in baza de date.
        - **Cnt** (inputul are numele = ***inputsname_cheie***_cnt ) contine numarul de sub-elemente de interfata care compun elementul de interfata curent.
    - Daca PROPERTY este camp din tabela:
        - **Table** (inputul are numele = ***inputsname***_ table***_cheie*** ) constine numele tabelei din care provine proprietatea.
        - **Field** (inputul are numele = ***inputsname***_ field***_cheie*** ) contine numele campului din tabela *Table* din care provine proprietatea.
        - **Pkeyfield** (inputul are numele = ***inputsname***_ pkeyfield ***_cheie*** ) contine numele campului care este cheie primara in tabela *Table*.
    - Daca PROPERTY este slot:
        - **entity** (inputul are numele = ***inputsname***_ entity***_cheie*** ) contine un caracter care denota tipul entitatii la care se refera informatia din slot. ('E' daca e episdo medical, 'P' daca e pacient, 'R' daca e persoana, etc...)
        - **entity_id** (inputul are numele = ***inputsname***_entity_id***_cheie*** ) contine numele campului din tabela *Table* din care provine proprietatea.
        - **slot** (inputul are numele = ***inputsname***_ slot ***_cheie*** ) contine string-id-ul slotului in care sunt stocate informatiil curente.
- **createInput** – este o functie wrapper, care se apeleaza mereu, pe baza tipului de element de interfata care trebuie generat, face apel mai departe la functiile generatoare specifice fiecarui tip de element de interfata.
- **createBooleanInput** – se apeleaza cand tipul elementului este boolean. Afiseaza si permite modificarea unei singure proprietati care are o singura valoare de tip boolean (true/false).
- **createNumericInput** – se apeleaza cand tipul elementului este numeric. Afiseaza si permite modificarea unei singure proprietati care are o singura valoare de tip numeric.
- **createStringInput** – se apeleaza cand tipul elementului este string. Afiseaza si permite modificarea unei singure proprietati care are o singura valoare de tip string.
- **createLongstringInput** – se apeleaza cand tipul elementului este string lung. Afiseaza si permite modificarea unei singure proprietati care are o singura valoare de tip string lung.
- **createDateInput** – se apeleaza cand tipul elementului este data calendaristica. Formateaza data, afiseaza si permite modificarea datei. (data/luna/an)
- **createTimeInput** – se apeleaza cand tipul elementului este timp. Afiseaza si permite modificarea unei singure proprietati care are o singura valoare de tip timp (ora/minut/secunda).
- **createDateTimeInput** – se apeleaza cand tipul elementului este un moment in timp (adica si data si si timp). Afiseaza si permite modificarea unei singure proprietati care are o singura valoare de tip moment in timp (data/luna/an ora/minut/secunda).
- **getInputElementsFromConcepttree** – cand pentru o anumita proprietate putem avea valori multiple (in cazul in care avem un slot si se stie ca in slotul respectiv

poate intra ca si valoare oricare dintr-o lista de instante posibile, eventual impreuna cu valori S - string sau I - int ), si valorile sunt cunoscute, avem cazuri in care pentru o singura proprietate trebuie sa construim mai multe elemente de interfata (de ex. daca avem proprietatea numita semn si simptom in care intra N semne si simptome – toate cunoscute din ontologie – atunci trebuie sa construim N sub-elemente de interfata – fiecare cu propriul checkbox si lable – una pentru fiecare instanta.) Aceasta functie face o parcurgere a structurii arborescente in care sunt stocate valorile posibile (instantele posibile) si apeleaza functia *getControlFromInstance* de generare a unui singur sub-element de interfata, combinand sub-elementele se genereaza un element de interfata compus.

- **createListInput** – se apeleaza atunci cand desi avem multiple valori cunoscute pentru o anumita proprietate, proprietatea poate lua la un moment dat doar una dintre valorile posibile (de exemplu daca proprietatea este “Stare civila” si avem lista cunoscuta de instante posibile “Casatorit” “Necasatorit”, “Divortat”, “Vaduv(a)”) atunci nu se va construi un element de interfata compus ci unul simplu care va contine un dropdown (combo box) din care se va alege doar o singura varianta din cele posibile. Optiunile din dropdown sew obtin cu functia *getOptionList*
- **getOptionList** – se apeleaza pentru a construi un sir de optiuni care va fi folosita ca si continutul unui dropdown.
- **getControlFromInstance** – aceasta functie decide conform tipului de sub-element de interfata cerut, care va fi functia apelata pentru generarea sub-elementului de interfata, apeleaza functia generatoare si returneaza elementul de interfata generat.
- **getCheckSliderFromInstance** – genereaza un sub-element de interfata compus din numele instantei, un checkbox ce indica prezenta sau absenta instantei repective si un control de tip slider care specifica o intensitate (se foloseste in general la semne si simptome).
- **getMedicationControlFromInstance** – genereaza un sub-element de interfata compus din numele instantei, un checkbox ce indica prezenta sau absenta instantei repective si un control de tip dosaj medicatie prin care se pot specifica detalii in legatura cu dosajul medicatiei.
- **getCheckInputFromInstance** – genereaza un sub-element de interfata compus din numele instantei, un checkbox ce indica prezenta sau absenta instantei repective si un input in care se afiseaza continutul I – numeric asociat cu instanta.
- **getInputFromInstance** – genereaza un sub-element de interfata compus din numele instantei si un input in care se afiseaza continutul I – numeric asociat cu instanta.
- **getRadioSliderFromInstance** – genereaza un sub-element de interfata compus din numele instantei, un buton de tip radio ce indica prezenta sau absenta instantei repective si un control de tip slider care specifica o intensitate (se foloseste in general la semne si simptome).
- **getCheckFromInstance** – genereaza un sub-element de interfata compus din numele instantei si un checkbox ce indica prezenta sau absenta instantei repective.
- **getRadioFromInstance** – genereaza un sub-element de interfata compus din numele instantei si un buton radio ce indica prezenta sau absenta instantei respective.
- **getTreeSearchControl** – cand pentru o anumita proprietate avem valori multiple, numarul valorilor este mare si pentru fiecare valoare potentiala trebuie generat un sub-element de interfata, continutul paginii s-ar aglomera foarte mult. Exista

posibilitatea de a ascunde majoritatea sub-elementelor de interfata – acolo unde valoarea din care este generata nu este foarte relevanta, sau este rar introdusa. (de ex. sub-elementul de interfata ce coresopunde unui semn sau simptom des intalnit precum *durerea de cap* va fi in permanenta afisat, dar unul mai rara intalnit, de exemplu *edem al bratului stang* va fi ascuns) Pentru a avea posibilitatea de a folosi si acele sub-elemente de interfata insa care sunt ascunse, acestea pot fi aduse la vedere cu ajutorul unui control numit treeSearchControl. treeSearchControl-ul este de fapt un element de interfata compus din doua parti:

- in partea stanga apare un navigator, similar navigatorului de directoare din windows explorer, cu care se poate parcurge ierarhia de concepte asociata cu slotul la care este legat treeSearchControl-ul, adica daca proprietatea (slotul) poate primii ca si valori concepte generale (adica orice instanta a conceptului general si orice instanta a unor concepte mai restrictive care sunt descendentii conceptului general)
- in partea dreapta este o lista a instantelor care apartin conceptului selectat din structura arborescenta in partea stanga. La selectarea instantei din aceasta lista, elementul de interfata asociat cu instanta respectiva este adusa la vedere (daca era ascuns pana atunci) iar daca contine un ckeckbox, sau un buton radio, acesta devine bifat (intuitiv daca utilizatorul a ales aceasta instanta putem deduce ca vrea sa o si adauge ca si valoare la slotul respectiv deci pentru reducerea work-loadului se bifeaza automat si nu se asteapta ca utilizatorul dupa ce a facut elemetul vizibil sa il mai si bifeze).

Pentru a genera cele doua parti ale interfetei treeSearchControl sunt folosite doua functii

*recursiveInstanceDisplay* – pentru fiecare concept si subconcept genereaza o lista cu instantele conceptului.

*recursiveBranchDisplay* – din ierarhia de concepte din ontologie, genereaza un control de tip arbore care permite navigarea la un concept anume.

- **recursiveInstanceDisplay** – genereaza o singura lista de instante pentru un singur concept. Toate listele pentru toate conceptele generate vor fi suprapuse pentru a genera partea dreapta a treeSearchControl-ului.
- **recursiveBranchDisplay** – genereaza o structura arborescenta avand ca noduri conceptele din ontologie (doar cele care sunt asociate cu slot-ul curent), si legaturile dintre nodiri se fac conform ierarhiei de concepte din ontologie.
- **getPROPS** – functie ce aduna datele din variabilele POST si GET si reface lista de PROPERTY-uri. La generarea interfetei de tip afisare + modificare s-a extras din baza de date o lista de elemente PROPERTY din care s-a generat interfata utilizator. Dupa ce utilizatorul a efectuat niste operatii de adaugare modificare stergere de date in interfata, datele updatate trebuie culese, convertite inapoi in lista de elemente PROPERTY, pentru a fi salvate inapoi in baza de date. La generarea interfetei datele generale referitoare la slot sau camp de tabela sunt turnate in input-uri de tip hidden, de catre functia *createHiddenInput* iar datele specificce (efectiv continutul) sunt transformate in controale interactive complexe de catre functiile infasurate de wrapper-ul *createInput*. La apelarea functiei getPROPS datele se afla in variabilele POST si GET conform standardului prezentat la 2.2.4.4.

#### 4.2.3.5 Meniul aplicatiei

In cadrul aplicatiei exista 3 tipuri de meniu:

- Meniul pentru un utilizator in vizita (care nu este logat). Acest meniu este de dimensiune redusa deoarece functionalitatile (serviciile medicale) majore ale sistemului pot fi accesate doare de utilizatori autorizati. Acest meniu permite navigarea la pagina de inregistrare a unui utilizator nou.
- Meniul pentru utilizator pacient. Permite navigarea de catre pacient la paginile cu functionalitatile specifice pacientului.
- Meniul pentru utilizator doctor. Permite navigarea de catre doctor la paginile cu functionalitatile specifice doctorului.

In fisierul ***menuFunctions.php*** exista 3 structuri predefinite si 3 functii pentru alcatuirea meniurilor pentru doctor,pacient si vizitator. Structurile definesc continutul textual si referinta linkurilor din meniuri, iar functiile construiesc meniul din informatiile incapsulate in structuri. La fiecare iteratie (refresh de pagina) structurile de date predefinite se reactualizeaza cu cuvinte specifice limbii curente, cuvintele se obtin din fisierele de limbi. Fiecare functie primeste ca si parametrii un fisier template si un numar ce denota numarul de ordine a meniului care este selectat, Functiile completeaza template-ul primit ca si parametru cu informatii din structura de date corespunzatoare (se ia in considerare si care meniu va fi selectat - highlight). Elementul de interfata – meniul – astfel generat se returneaza.

#### 4.2.3.6 Managementul sessiunii de lucru

Mentinerea sesiunii in cadrul aplicatiei este efectuat de rutinele din fisierul **SessionManagement.php**. Informatiile majore care se mentin in sesiune sunt *starea de autentificare* (legat de starea utilizatorului - este sau nu logat) si *limbajul ales* (limba in care se prezinta interfata utilizator in mod permanent). Actiunile majore efectuate in manipularea variabilelor de sesiune sunt: *Logarea*, *delogarea* si *schimbarea limbii:*

- Logarea:
  - Se face apel la serviciul web de logare (cum am mentionat la 2.3 toate functionalitatile, inclusiv cel de autentificare sunt in partea de server de servicii web) cu parametrii introdusi de utilizator.
  - In caz de succes serviciul web returneaza datele de sesiune (niste date referitoare la utilizator) inclusiv in identificator de sesiune, care va fi un hash de 32 de caractere generat pentru identificarea sesiunii de login. Dupa logare orice serviciu web apelat de utilizator va primii ca si parametru acest hash pentru identificarea utilizatorului. Serverul se asigura ca utilizatorul are dreptul sa efectueze o operatie de date pe baza acestui hash.
  - In caz de eroare returneaza false.
  - Daca logarea a fost efectuata cu succes, informatiile de sesiune returnate se stocheaza in sesiunea nativa php, pentru aplicatie client.
- Delogarea:
  - Se face apel la serviciul web de delogare (cum am mentionat la 2.3 toate functionalitatile, inclusiv cel de autentificare sunt in partea de server de servicii web) cu identificatorul de sesiune.
  - Se reseteaza structura nativa de sesiune php, in afara de parametrul *language*.
- Schimbarea limbii:

- Efectul se manifesta doar in interfata, deci nu se apeleaza logica de buisness, se seteaza variabila de sesiune php *language* in conformitate cu optiunea utilizatorului.

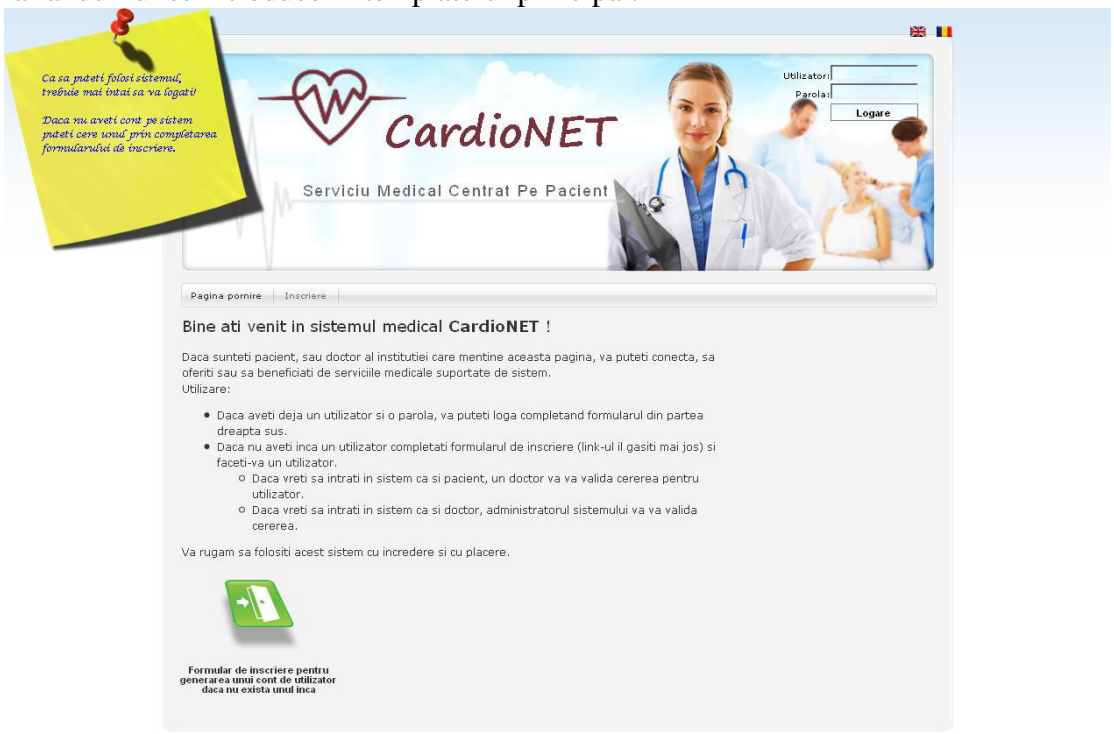
#### 4.2.3.7 Pagini utilizator

Interfata aplicatiei are acelasi cadru pentru cele 3 tipuri de utilizatori, doar continutul se schimba. Cadrul comun este definit in template-ul principal *templates/PageFrame.tpl.*

##### *4.2.3.7.1 Pagini pentru Vizitator*

###### 4.2.3.7.1.1 Pagina de navigare

Se completeaza template-ul *templates/navigator.tpl* cu navigatorul principal al interfetei vizitator (creat din icon-uri grafice mari), navigatorul se construieste din elemente de tip icon mare pe baza template-ului *templates/BIG_ICON.tpl* si se introduce in template-ul paginii 0 care la randul lui se introduce in template-ul principal.




###### 4.2.3.7.1.2 Pagina de inscriere

Se apeleaza serviciile web *GetNewUserForm*, *GetListOfAvailableDoctors* si *GetSpecialities* pentru a obtine continutul formului de inscriere (informatiile personale / demografice + nume utilizator si parola, tipul utilizatorului cerut si doctor de familie sau specializare). Daca o operatie de inscriere este efectuata, se aduna informatiile din variabilele POST si GET asa cum s-a descris la 2.3.4.3.2. si se apeleaza serviciul web *RegisterNewUser*, pentru a salva datele introduse.

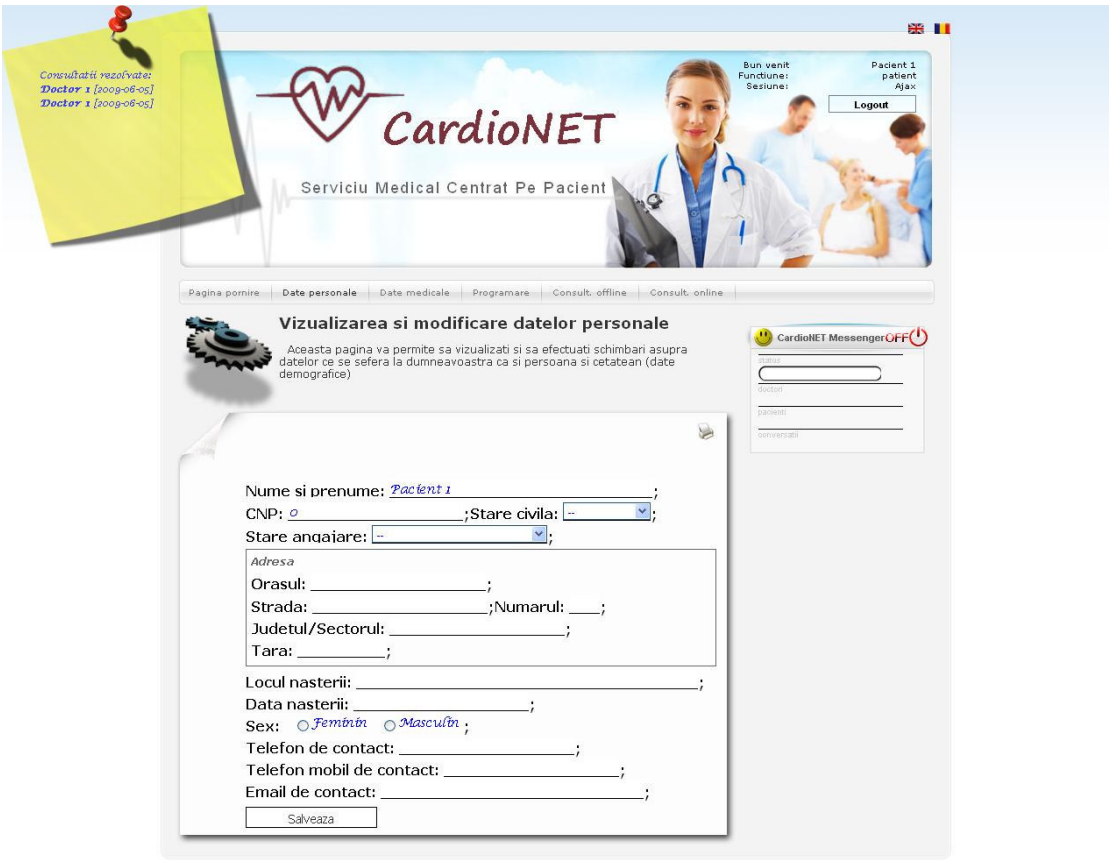

## *4.2.3.7.2 Pagini pentru Pacient*

### 4.2.3.7.2.1 Pagina de navigare

Se completeaza template-ul *templates/patient/page.0.tpl* cu navigatorul principal al interfetei pacient (creat din icon-uri grafice mari), navigatorul se construieste din elemente de tip icon mare pe baza template-ului *templates/BIG_ICON.tpl* si se introduce in template-ul paginii 0 care la randul lui se introduce in template-ul principal.


#### 4.2.3.7.2.2 Pagina cu date demografice

Se apeleaza serviciul web *GetFormContent* cu parametrul “Generalinfo“ care returneaza continutul formului de informatii generale (informatiile personale / demografice) legate de o persoana. Cu continutul returnat de *GetFormContent* se completeaza template-ul *templates/FORMS/generalinfo.tpl* iar continutul generat se introduce in template-ul principal al paginii 1 *templates/patient/page.1.tpl* care la randu lui se introduce in template-ul principal. Daca o operatie de salvare este efectuata de utilizator, se aduna informatiile din variabilele POST si GET asa cum s-a descris la 2.3.4.3.2. si se apeleaza serviciul web *SetData* pentru a salva datele introduse si modificate.



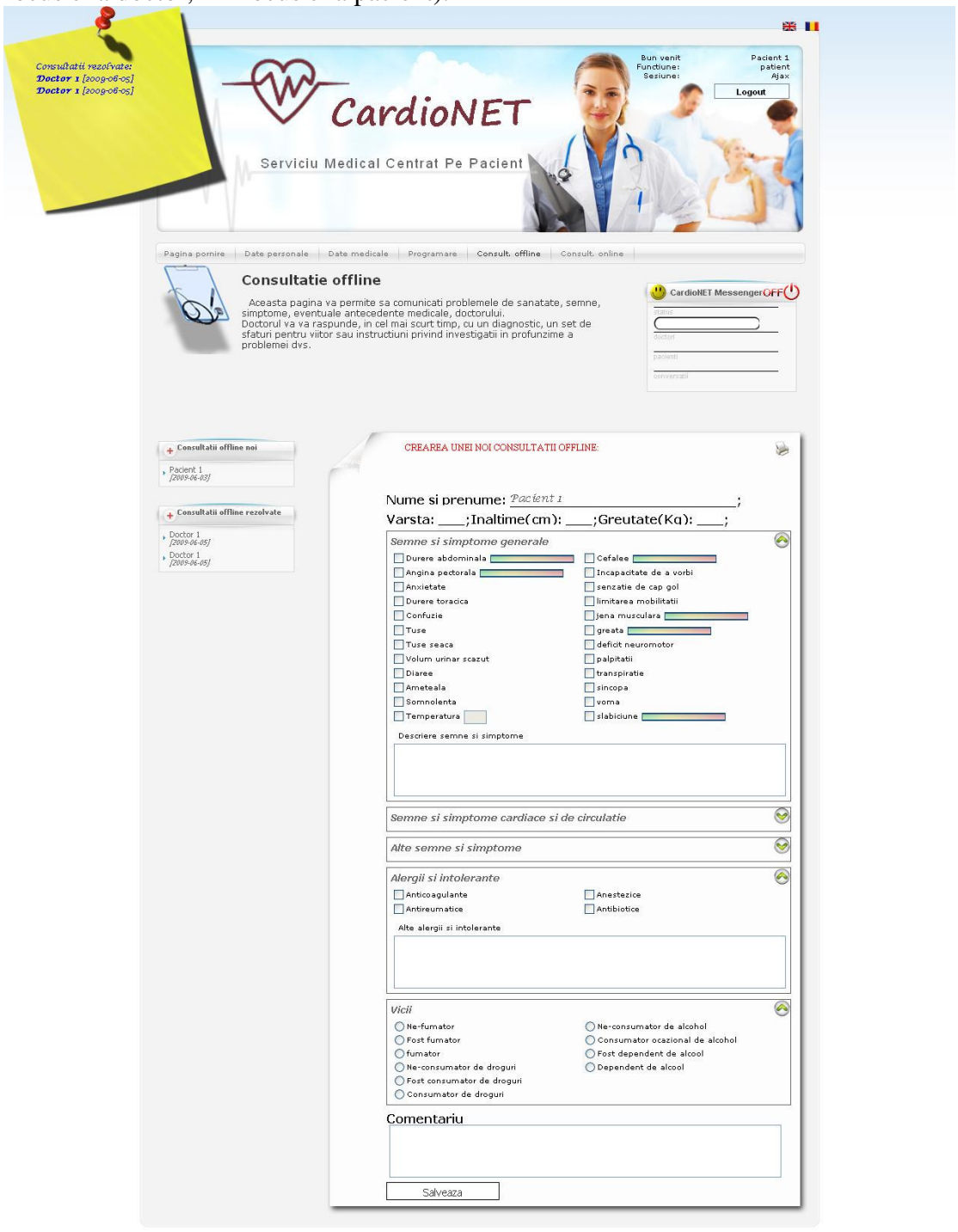

#### 4.2.3.7.2.3 Pagina cu date medicale

Pagina are 2 parti, prima parte este un set de liste unde apar episoadele medicale care apartin pacientului grupate dupa tipul lor (nou, vechi, in decurs, terminat, etc...) Pentru obtinerea listelor de episoade se apeleaza servicii web specifice fiecarui tip *GetNewConsultOfflineList*, *GetPendingConsultOfflineList* etc... Se construiesc listele de episoade pe baza template-ului *templates/BoxList.tpl* si se adauga la template-ul principal al paginii 2 *templates/patient/page.2.tpl.* Partea a doua este un form in care se afiseaza un episod medical sau un pas de tratament apartinand unui episod medical, care este vizualizat. Se obtin datele referitoare la episodul sau pasul vizualizat prin apelarea serviciilor web *GetEpisodeSingle* si *GetStepSingle* iar cu aceste date se completeaza templateurile corespunzatoare, dupa care formul se introduce in template-ul principal al paginii 2 care la randul lui se introduce in template-ul principal. Daca episodul vizualizat este de tip consult offline continuat, adica medicul necesita informatii suplimentare pentru a continua, se obtine continutul unui form secundar, prin intermediul caruia, pacientul va introduce datele suplimentare prin intermediul unui textbox. Se apeleaza serviciul *GetFormContent* cu parametrul “patient_response” pentru a obtine continutul formului secundar, se construieste form-ul secundar pe baza template-ului *templates/FORMS/patientResponse.tpl* si se adauga la template-ul principal al paginii 2 *templates/patient/page.2.tpl.* Daca o operatie de raspuns este efectuata de utilizator, se aduna informatiile din variabilele POST si GET asa cum s-a descris la 2.3.4.3.2. si se apeleaza serviciul web *PatientResponse* pentru a salva raspunsul pacientului si pentru a returna focusul episodului la doctor (focusul se seteaza cu un flag-ul – campul pingpong din baza de date: 0 – focus e la doctor, 1 – focus e la pacient).


#### 4.2.3.7.2.4 Pagina pentru efectuarea unui consult offline

Pagina are 2 parti, prima parte este un set de liste unde apar episoadele medicale care sunt consultatii offline noi sau continuate si apartin pacientului. Pentru obtinerea listelor de consultatii offline se apeleaza servicii web specifice fiecarui tip *GetNewConsultOfflineList*, *GetPendingConsultOfflineList* etc... Se construiesc listele de episoade pe baza template-ului *templates/BoxList.tpl* si se adauga la template-ul principal al paginii 4 *templates/patient/page.4.tpl.* Partea a doua este un form in care i se permite pacientului sa-si creeze un consult offline nou si sa completeze un set limitat de posibile semne simptome, sindroame si efecte, etc... (sunt selectate acele care pot fi intelese si completate corect de catre orice pacient, nu necesita cunostinte medicale). Se obtin datele referitoare la continutul form-ului de consult ofline nou prin apelarea serviciului *GetFormContent* cu parametrul “Consultation”, iar cu aceste date se completeaza templateul *templates/FORMS/consultation.tpl*, dupa care formul se introduce in template-ul principal al paginii 4 care la randul lui se introduce in template-ul principal. Daca o operatie de salvare este efectuata de utilizator, se aduna informatiile din

variabilele POST si GET asa cum s-a descris la 2.3.4.3.2. si se apeleaza serviciul web *ConsultOffline* pentru a salva datele introduse intr-un nou consult offline si pentru a seta focusul episodului la doctor (focusul se seteaza cu un flag-ul – campul pingpong din baza de date: 0 – focus e la doctor, 1 – focus e la pacient).



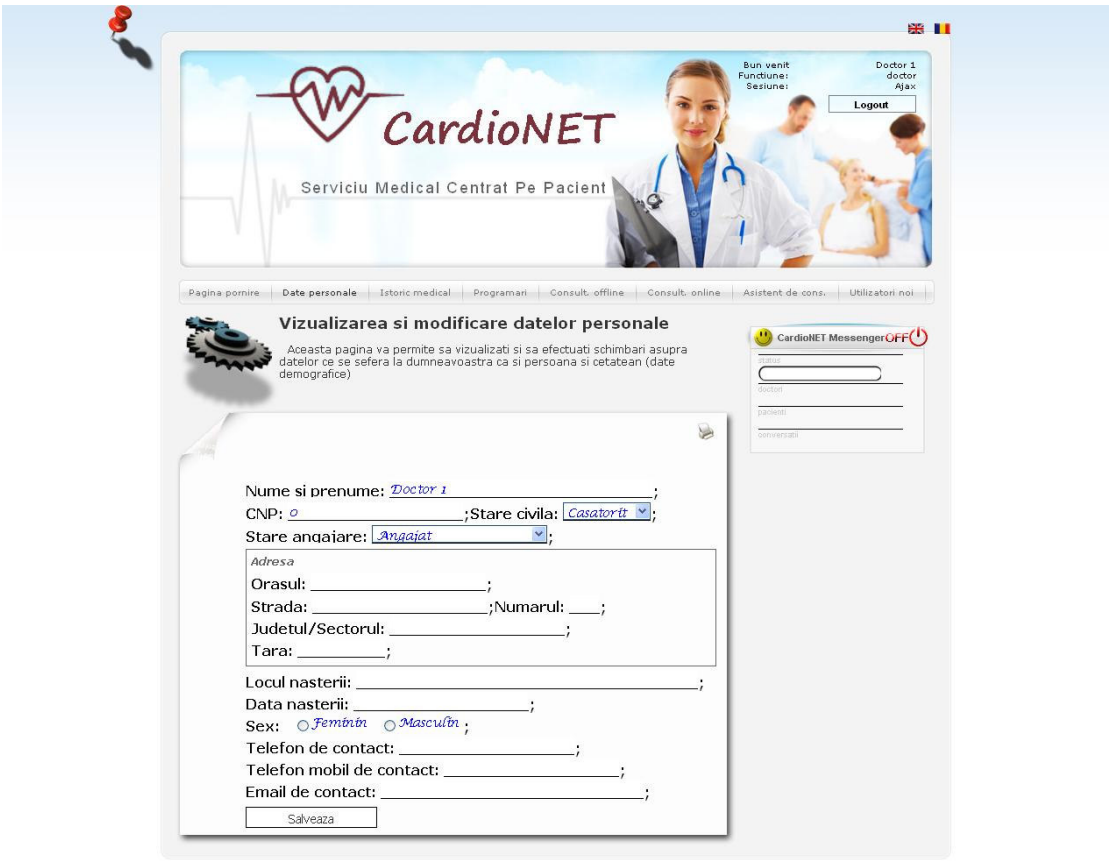
### *4.2.3.7.3 Doctor*

#### 4.2.3.7.3.1 Pagina de navigare

Se completeaza template-ul *templates/doctor/page.0.tpl* cu navigatorul principal al interfetei doctorului (creat din icon-uri grafice mari), navigatorul se construieste din elemente de tip icon mare pe baza template-ului *templates/BIG_ICON.tpl* si se introduce in template-ul paginii 0 care la randul lui se introduce in template-ul principal.


#### 4.2.3.7.3.2 Pagina cu date demografice

Se apeleaza serviciul web *GetFormContent* cu parametrul “Generalinfo“ care returneaza continutul formului de informatii generale (informatiile personale / demografice) legate de o persoana. Cu continutul returnat de *GetFormContent* se completeaza template-ul *templates/FORMS/generalinfo.tpl* iar continutul generat se introduce in template-ul principal al paginii 1 *templates/doctor/page.1.tpl* care la randu lui se introduce in template-ul principal. Daca o operatie de salvare este efectuata de utilizator, se aduna informatiile din variabilele POST si GET asa cum s-a descris la 2.3.4.3.2. si se apeleaza serviciul web *SetData* pentru a salva datele introduse si modificate.




#### 4.2.3.7.3.3 Pagina pentru solutionarea unui consult offline

Pagina are 2 parti, prima parte este un set de liste unde apar episoadele medicale care sunt consultatii offline noi sau continuate (adica au flag-ul pingpong setat la 0 – focus-ul este la doctor). Pentru obtinerea listelor de consultatii offline se apeleaza servicii web specifice fiecarui tip *doc_GetNewConsultOfflineList*, *doc_GetPendingConsultOfflineList*. Se construiesc listele de episoade pe baza template-ului *templates/BoxList.tpl* si se adauga la template-ul principal al paginii 4 *templates/doctor/page.4.tpl.* Partea a doua este un form in care i se permite doctorului sa solutioneze/continue/respinga un consult offline nou sau continuat si sa completeze toate datele posibile ce pot fi asociate cu un episod medical (semne simptome, sindroame si efecte, medicamentatie, interventii chirurgicale etc...). Se obtin datele referitoare la continutul form-ului de solutionare consult ofline prin apelarea serviciilor *doc_GetEpisodeSingle* si *doc_GetStepSingle*. Cu aceste date se completeaza template-urile *templates/FORMS/episodeEdit.tpl*, *templates/FORMS/episodeView.tpl*, *templates/FORMS/stepEdit.tpl, templates/FORMS/stepView.tpl,* formul obtinut astfel se introduce in template-ul colector *templates/doctor/medical_data_viewedit.tpl* care la randul lui este integrat in template-ul principal al paginii 4 *templates/doctor/page.4.tpl*. Daca o operatie de salvare/rezolvare/continuare/respingere este efectuata de utilizator, se aduna informatiile din variabilele POST si GET asa cum s-a descris la 2.3.4.3.2. si se apeleaza serviciul web *TmpsaveConsultOfflineSingle / ResolveConsultOfflineSingle / ContinueConsultOfflineSingle / RejectConsultOfflineSingle*.


#### 4.2.3.7.3.4 Pagina de acceptare/respingere utilizatori noi

Pagina are 2 parti, prima parte este o lista unde apar pacientii care s-au inscris pe pagina si a caror utilizator nu a fost validat inca. Pentru obtinerea listei de pacienti noi se apeleaza serviciul web *doc_GetNewPatientUserList*. Se construieste lista de pacienti pe baza template-ului *templates/BoxList.tpl* si se adauga la template-ul principal al paginii 7 *templates/doctor/page.7.tpl.* Partea a doua este un form in care se pot vizualiza datele completate de pacient la inscriere, doctorul va accepta sau respinge cererea de utilizator pe baza acestor date. Continutul formului se obtine prin apelarea serviciului *doc_GetNewPatientUserSingle* si prin completarea template-ului *templates/FORMS/generalinfoView.tpl.* Form-ul astfel obtinut se adauga la template-ul principal al paginii 7 *templates/doctor/page.7.tpl*, care la randul lui va fi adaugat la template-ul principal al paginii. Daca o operatie de acceptare/respingere este efectuata de utilizator se apeleaza serviciile web *acceptNewPatientUser* respectiv *rejectNewPatientUser*.


### 4.2.4 Aplicatia “Server”

Este componenta aplicatiei care contine logica de buisness, nu are interfata grafica, are doar o interfata de servicii web exportate pentru interactiunea machine-to-machine.

#### 4.2.4.1 Structura aplicatiei Server

Scriptul principal al Serverului este *SERVER/index.php* in acesta sunt incluse toate celelalte parti care alcatuiesc logica de buisness:

SCRIPTURI PHP:

- **include/ connect_db_inc.php** – contine o clasa wrapper pentru functiile mysql. Se creeaza o instanta a clasei care va fi variabila globala si se va folosi pentru a accesa baza de date. Descriere detaliata la 2.3.2**.**
- **include/lastAction.php** – contine o functie care se apeleaza dupa ce toate celelalte scripturi php s-au rulat, efectueaza deconectarea de la serverul BD.
- **include/logWrite.php** - contine functii pentru scrierea unor mesaje de gen LOG in fisiere cu nume predefinite din radacina aplicatiei. Functiile sunt descrise in cap. 2.2.2.

- **include/xmlParser.php** – contine 4 functii si o clasa, pentru parsarea fisierelor xml conversia intre modurile de reprezentare, si gestionarea informatiilor obtinute din xml. Descriere detaliata la 2.2.3.
- **RPDEF/AUX_SecurityModule.php** – acest fisier reprezinta modulul de securitate, contine functii exportate ca si servicii web, dar accesate si in direct de alte functii din SERVER, care acceseaza baza de date si implementeaza policy-urile de securitate definite la 2.2.4.5 . Toate accesurile la baza de date, citiri si scrieri de date sunt efectuate prin functiile definite in acest modul. Descris in detaliu la 2.4.4.1
- **RPDEF/RPDEF_Session.php** – contine functii exportate ca si servicii web, care tind de login si de managementul sessiunii: logare, delogare, cerere utilizator etc...
    - **RPDEF/AUX_Session.php** – contine functii ajutatoare pentru serviciile web din *RPDEF_Session.php*: functii de generare a cheii de sesiune, obtinerea id-ului, numelui ... din identificatorul de sesiune etc...
    - Descris in detaliu la 2.4.4.2
- **RPDEF/RPDEF_CNmess.php** – contine functii exportate ca se servicii web, care tind de serviciul de mesagerie a aplicatiei – CNmessenger. Descris in detaliu la 2.4.4.3.
- **RPDEF/RPDEF_Forms.php** – contine functii exportate ca se servicii web, care tind de obtinerea continutului formurilor si salvarea datelor din formurile generate inapoi in baza de date.
    - **RPDEF/AUX_ Forms.php**– contine functii ajutatoare pentru serviciile web din *RPDEF_Forms.php.*
    - Descris in detaliu la 2.4.4.4.
- **RPDEF/RPDEF_Patient.php** – contine functii exportate ca si servicii web, care tind de utilizatorii de tip pacient si serviciile medicale accesate de acestea.
    - **RPDEF/AUX_ Patient.php** – contine functii ajutatoare pentru serviciile web din *RPDEF_Patient.php.*
    - Descris in detaliu la 2.4.4.5.
- **RPDEF/RPDEF_Doctor.php** – contine functii exportate ca si servicii web, care tind de utilizatorii de tip doctor si serviciile medicale accesate de acestea.
    - **RPDEF/AUX_ Doctor.php** – contine functii ajutatoare pentru serviciile web din *RPDEF_Doctor.php.*
    - Descris in detaliu la 2.4.4.6.
- **RPDEF/RPDEF_GenTypes.php** – contine definitiile tipurilor de date folosite in comunicarea prin serviciile web. Descrierea detaliata a tipurilor de date definite aici se gaseste la 2.2.4.1.
- **RPDEF/AUX_HelpfulFunctions.php** – contine functii php de uz general ex. *ConcatArrays* – functie ce concateneaza doua array-uri.

#### 4.2.4.2 Modulul de conectare la baza de date

Datele de conectare la baza de date (adresa server DB, utilizator, parola etc...) sunt stocate in fisierul de configurare *config.inc.php* – descris la punctul 2.2.5. Fisierul *include/connect_db_inc.php* contine o clasa wrapper – MySqlDbConnection – pentru functiile mysql. Se creeaza o instanta a clasei care va fi variabila globala si se va folosi pentru a accesa baza de date Daca se schimba baza de date de la mysql la mssql de exemplu, adaptarea aplicatiei

se face astfel foarte usor, doar prin schimbarea acestui wrapper. Clasa wrapper contine urmatoarele metode:

- **MySqlDbConnection** – contructorul clasei *MySqlDbConnection*, primeste ca si parametrii datele de conectare si face apel la metoda connect pentru stabilirea unei conexiuni.
- **connect** – wrapper pentru functia *mysqli_connect* realizeaza o conexiune la baza de date si seteaza charsetul care va fi folosit pe conexiune (de regula utf8.bin).
- **disconnect** – wrapper pentru functia *mysqli_close* inchide conexiunea spre baza de date.
- **query** – wrapper pentru functia *mysqli_query,* ruleaza un query pe baza de date, returneaza un set de rezultat. Daca query-ul esueaza efectueaza o reconectare si inceatrca query-ul din nou. Daca query-ul esueaza si dupa reconectare, se trece in error log.
- **query_first_row** – functie ce ruleaza un query pe baza de date si returneaza doar prima linie din setul de rezultat, daca setul nu este gol.
- **insert_id** – wrapper pentru functia *mysqli_insert_id*, returneaza id-ul ultimului element inserat in baza de date.
- **cleanConnection** – efectueaza o deconectare si o reconectare la serverul de BD, pentru eliminarea unor posibile erori ce se pot produce din cauza starii conexiunii.
- **num_rows** – wrapper pentru functia *mysqli_num_rows*, primeste ca si parametru un set de rezultate, returneaza numarul liniilor din setul de rezultate.
- **fetch_array** – wrapper pentru functia *mysqli_fetch_array*, primeste ca si parametru un set de rezultate, returneaza urmatoarea linie din setul de rezultate (conform poantorului intern al setului) sub forma unui array asociativ.
- **data_seek** – wrapper pentru functia *mysqli_data_seek*, primeste ca si parametru un set de rezultate si un numar inreg, seteaza poantorul intern al setului de rezultate la numarul primit ca si parametru.

#### 4.2.4.3 Modulul de cuplare (wrapper-ul) SOAP pentru server WS

Fisierul *SERVER/ SOAP_Wrapper_S.php* contine o functie ce genereaza XML – ul care va deservi ca si continut al WSDL-ului folosit la publicarea functiilor WS. In acest fisier se instantiaza si se initiaza serverul SOAP prin clasa nativa PHP *SoapServer*.

#### 4.2.4.4 Descrierea detaliata a Serviciilor web (proceduri remote)

##### *4.2.4.4.1 Modulul de securitate*

Modulul de securitate este definit in script-ul *SERVER/RPDEF/AUX_SecurityModule.php* care contine urmatoarele functii:

- **getFilledSlotInfo** – returneaza o structura de tip PROPERTY completat dintr-un slot si cu valori reale ale slotului in referinta la o entitate.
    - parametrii:
        - slot – string-id-ul slot-ului din care se genereaza PROPERY-ul
        - OVERRIDES – valorile care se vor supraincarca in structura PROPERTY (formattype, nume etc...).

        - Fromentity – tipul entitatii la care se refera slotul, in referinta la ce tip de entitate extragem valori reale pentru slot (R-persoana, P-pacient,E-episode etc...)
        - sessions_identifyers – identificatorul de sesiune, pe baza acestui id se identifica userul in numele caruia se face acest apel la serviciu. Daca utilizatorul nu este logat si nu are un identificator de sesiune valid, operatiile nu se efectueaza.
        - person_id – daca entitatea este de tip persoana, doctor sau pacient (Fromentity este R,P sau D) acest id identifica entitatea la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - episode_id – daca entitatea este de tip episod (Fromentity este E) acest id identifica episodul la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - step_id – daca entitatea este de tip step (Fromentity este S) acest id identifica pasul de tratament la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - instance_id – daca entitatea este de tip instantance (Fromentity este M) acest id identifica instanta de concept (din ontologie) la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - override_security – daca in caz special se face un apel la aceasta functie care in asa fel incat policy-ul de securitate nu ar permite realizarea operatiei, acest parametru setat la "true" marcheaza faptul ca policy-ul de securitate nu se va lua in considerare si se va efectua operatia in orice caz.
    - functionalitate:
        - Creeaza o structura de tip PROPERTY
        - Completeaza structura cu informatii de natura generala despre un anumit slot (apeland *getEmptySlotInfo*)
        - Completeaza structura cu informatii specifice legate de valoarea stocata in slot pentru o entitate identificata prin tip si id.
        - Returneaza PROPERTY-ul.
- **getEmptySlotInfo** – returneaza o structura de tip PROPERTY completat dintr-un slot.
    - parametrii:
        - slot – string-id-ul slot-ului din care se genereaza PROPERY-ul
        - OVERRIDES – valorile care se vor supraincarca in structura PROPERTY (formattype, nume etc...).
    - functionalitate:
        - Creeaza o structura de tip PROPERTY
        - Ccompleteaza structura cu informatii de natura generala despre un anumit slot
        - Returneaza PROPERTY-ul.
- **getFilledFieldInfo** – returneaza o structura de tip PROPERTY completat dintr-un camp a unei tabele si cu valori reale ale campului pentru o intrare a tebelei.
    - parametrii:
        - table – numele tabelei in care se gaseste campul de interes

        - field – numele campului de interes din tabela specificata.
        - pk_value – valoare cheii primare (id-ului) la intrarea din tabel de unde se extrage informatia.
        - OVERRIDES – valorile care se vor supraincarca in structura PROPERTY (formattype, nume etc...).
        - sessions_identifyers– identificatorul de sesiune, pe baza acestui id se identifica userul in numele caruia se face acest apel la serviciu. Daca utilizatorul nu este logat si nu are un identificator de sesiune valid, operatiile nu se efectueaza.
        - override_security – daca in caz special se face un apel la aceasta functie care in asa fel incat policy-ul de securitate nu ar permite realizarea operatiei, acest parametru setat la "true" marcheaza faptul ca policy-ul de securitate nu se va lua in considerare si se va efectua operatia in orice caz.
    - functionalitate:
        - Creeaza o structura de tip PROPERTY
        - Completeaza structura cu informatii de natura generala despre un anumit camp de tabela (apeland *getEmptyFieldInfo*)
        - Completeaza structura cu informatii specifice legate de valoarea stocata in campul respectiv la o intrare din tabela identificata de pk_value.
        - Returneaza PROPERTY-ul.
- **getEmptyFieldInfo** – returneaza o structura de tip PROPERTY completat dintr-un camp a unei tabele.
    - parametrii:
        - table – numele tabelei in care se gaseste campul de interes
        - field – numele campului de interes din tabela specificata.
        - OVERRIDES – valorile care se vor supraincarca in structura PROPERTY (formattype, nume etc...).
    - functionalitate:
        - Creeaza o structura de tip PROPERTY
        - Completeaza structura cu informatii de natura generala despre un anumit camp de tabela.
        - Returneaza PROPERTY-ul.
- **getValueTypeOfTableField** – returneaza tipul valorii care este stocat intr-un camp al unei tabele
    - parametrii:
        - table – numele tabelei in care se gaseste campul de interes
        - field – numele campului de interes din tabela specificata.
    - functionalitate:
        - returneaza una dintre valorile string "C","S","I","CS","CI","SI" sau "CSI" (in cazul practic returneaza doar "C","S" sau "I" pentru ca fizic nu este posibil ca un camp de tabela sa aiba valori multiple de gen "CS", "CI"etc...) in functie de ce tip de valoare este stocat in campul din tabela.
- **getPrimaryKeyOfTable** – returneaza numele campului ce serveste ca si cheie primara intr-o tabela.
    - parametrii:
        - table – numele tabelei de interes

- functionalitate:
  - returneaza numele campului de cheie primara al tabelei specificate, in general acesta va fi “id”

- **getPossibleSlotValueArray –** returneaza o lista de structuri de tip concepttree in care se specifica valorile posibile de instante care pot intra ca si valoare intr-un slot si pozitia acestor instante in ierarhia ontologiei. Aceasta functie este apelata de *getEmptySlotInfo* si structura returnata se introduce intr-o structura PROPERTY.
  - parametrii:
    - slot – numele slotului pentru care ne intereseaza instantele care pot intra ca si valoare.
    - conceptlist – o lista (structura) de concepte folosit la limitarea conceptelor si/sau la supraincarcarea unor valori la instantele conceptelor.
    - instancelist – o lista (structura) de instante folosit la limitarea instantelor si/sau la supraincarcarea unor valori la instante.
  - functionalitate:
    - Parcurge toate superconceptele ale caror instante (instante directe sau instante ale descendentilor) pot intra ca si valori in slot (de regula la un slot exista doar un singur super concept, dar exista si exceptii)
    - Apeleaza functia *getConceptStruct* care contruieste o singura structura de tip *concepttree* dintr-un singur super concept.
    - Concateneaza *concepttree*-urile obtinute intr-un array de tip *concepttree_SET*
    - Returneaza *concepttree_SET*-ul care la randul lui se introduce intr-o structuta PROPERTY in cadrul functiei *getEmptySlotInfo.*
- **getConceptStruct –** returneaza o singura structura de tip concepttree care are la baza un superconcept primit ca si parametru, se autoapeleaza recursiv, parcurge toata ierarhia ontologica pornind de la conceptul de baza, si culege toate subconceptele si instantele acestora.
  - parametrii:
    - concept – conceptul de la care porneste recursia.
    - conceptlist – o lista (structura) de concepte folosit la limitarea conceptelor si/sau la supraincarcarea unor valori la instantele conceptelor.
    - Instancelist – o lista (structura) de instante folosit la limitarea instantelor si/sau la supraincarcarea unor valori la instante.
    - concept_order – numarul de ordine al conceptului (este o valoare stocata pentru fiecare instanta, pentru a avea o ordonare totala a instantelor, fiecare instanta primeste numarul de ordine al conceptului parinte, iar un concept care nu este trecut in conceptlist va mosteni numarul de ordine de la ultimul stramos la care a fost specificat) sau al ultimului superconcept specificat in conceptlist.
    - concept_override – valorile de supraincarcare a ultimului concept stramos care a fost specificat.
  - functionalitate:
    - parcurge recursiv ierarhia de concepte pornind de la un super concept.

        - Pentru fiecare concept construieste o structura concepttree cu lista de instante (apeland *getInstanceStruct*) si cu lista de subarbori pentru descendenti (conceptree-uri descententi prin apel recursiv)
        - Numarul de ordine si valorile supraincarcate ale instantelor se pot lua:
            - Din valorile explicite din instancelist
            - Daca nu, din valorile explicite din conceptlist
            - Daca nu, valorile se pot mosteni de la ultimul stramos care a avul valori explicite din conceptlist.
        - Conditii de returnare (conceptree-ul construit dintr-un concept se returneaza doar daca contine informatii semnificative – daca contine undeva si instante):
            - Daca lista de concepte este non exclusiva:
                - se iau in considerare doar valorile supraincarcate
                - se returneaza conceptree-ul daca
                    - concept-ul curent are instante
                    - daca exista un conceptree pentru sub-concepti (adica undeva un subarbore este macar o instanta)
            - Daca lista de concepte este exclusiva:
                - Se iau in considerare si valorile de supraincarcare
                - Daca conceptul curent apare in lista de concepte:
                    - se returneaza conceptree-ul daca
                        - concept-ul curent are instante
                        - daca exista un conceptree pentru sub-concepti (adica undeva in subarbore este macar o instanta)
                - daca conceptul curent nu apare in lista de concepte dar exista un concepttree pentru sub-concepti (adica undeva in subarbore este macar un concept care are macar o instanta si care apare in lista)
                    - returnam concepttree-ul dar fara instantele conceptului curent
- **getInstanceStruct –** returneaza o lista de elemente de tip *instance* in care vor intra instantele unui singur concept primit ca si paramteru.
    - parametrii:
        - concept – conceptul pentru care se cauta instantele
        - instancelist – o lista (structura) de instante folosit la limitarea instantelor si/sau la supraincarcarea unor valori la instante.
        - concept_order – numarul de ordine mostenit de la conceptul parinte sau de la ultimul stramos al conceptului parinte care a avut setat numarul de ordine.
        - concept_override – valorile de supraincarcare mostenite de la conceptul parinte sau de la ultimul stramos al conceptului parinte care a avut setat valori de supraincarcare.
        - conceptlist – o lista (structura) de concepte folosit la limitarea conceptelor si/sau la supraincarcarea unor valori la instantele conceptelor.
    - functionalitate:

        - creeaza o list de elemente *instane* in care introduce instantele care apartin unu concept anume.
        - Daca lista de instante este exclusiva
            - Se introduc doar acele instante care sunt specificate in lista de instante
            - Instantele introduse vor avea valori supraincarcate ori din lista de instante, ori mostenite de la concept parinte sau concept predecesor al parintelui (pot fi si fara valori supraincarcate).
        - Daca lista de instante nu este exclusiva
            - Se introduc toate instantele care sunt instante ale conceptului
            - Instantele introduse vor avea valori supraincarcate ori din lista de instante, ori mostenite de la concept parinte sau concept predecesor al parintelui (pot fi si fara valori supraincarcate).
- **setSlotInfo** – functie ce seteaza o valoare sau o serie de valoari dintr-un singur slot, asociate cu o entitate anume
    - parametrii:
        - sessions_identifyers – identificatorul de sesiune, pe baza acestui id se identifica userul in numele caruia se face acest apel la serviciu. Daca utilizatorul nu este logat si nu are un identificator de sesiune valid, operatiile nu se efectueaza.
        - PROP – lista de structuri de tip PROPERTY care contine datele care se vor stoca in slot-uri.
        - person_id – daca entitatea este de tip persoana, doctor sau pacient (Fromentity este R,P sau D) acest id identifica entitatea la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - episode_id – daca entitatea este de tip episod (Fromentity este E) acest id identifica episodul la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - step_id – daca entitatea este de tip step (Fromentity este S) acest id identifica pasul de tratament la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - instance_id – daca entitatea este de tip instantance (Fromentity este M) acest id identifica instanta de concept (din ontologie) la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - override_security – daca in caz special se face un apel la aceasta functie care in asa fel incat policy-ul de securitate nu ar permite realizarea operatiei, acest parametru setat la “true” marcheaza faptul ca policy-ul de securitate nu se va lua in considerare si se va efectua operatia in orice caz.
    - functionalitate:
        - Functia verifica daca utilizatorul are permisiunea sa modifice valoarea din slotul specificat de PROPERTY pentru entitatea specificata de PROPERTY (in caz ca override_security nu este “true”)
        - Daca policy-ul de securitate este satisfacut, se parcurge lista de valori din PROP (PROP[‘value’])

        - Se efectueaza o updatare sau o inserare pentru fiecare valoare din PROPERTY (updatare daca ID-ul care corespunde valorii este setat => valoarea era existenta si trebuie modificata).
- **setFieldInfo** – functie ce seteaza o valoare dintr-un camp dintr-o tabela
    - parametrii:
        - sessions_identifyers – identificatorul de sesiune, pe baza acestui id se identifica userul in numele caruia se face acest apel la serviciu. Daca utilizatorul nu este logat si nu are un identificator de sesiune valid, operatiile nu se efectueaza.
        - PROP – lista de structuri de tip PROPERTY care contine datele care se vor stoca in slot-uri.
        - person_id – daca entitatea este de tip persoana, doctor sau pacient (Fromentity este R,P sau D) acest id identifica entitatea la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - episode_id – daca entitatea este de tip episod (Fromentity este E) acest id identifica episodul la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - step_id – daca entitatea este de tip step (Fromentity este S) acest id identifica pasul de tratament la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - instance_id – daca entitatea este de tip instantance (Fromentity este M) acest id identifica instanta de concept (din ontologie) la care se refera datele reale din slot, care se vor extrage si se vor introduce in PROPERTY.
        - override_security – daca in caz special se face un apel la aceasta functie care in asa fel incat policy-ul de securitate nu ar permite realizarea operatiei, acest parametru setat la “true” marcheaza faptul ca policy-ul de securitate nu se va lua in considerare si se va efectua operatia in orice caz.
    - functionalitate:
        - Functia verifica daca utilizatorul are permisiunea sa modifice valoarea din campul/tabela specificate de PROPERTY (in caz ca override_security nu este “true”)
        - Daca policy-ul de securitate este satisfacut, se parcurge lista de valori din PROP (PROP[‘value’]) – in acest caz va fi de lungime 1.
        - Se efectueaza o updatare campului specificat de *field* din tabela specificata de *table* in linia unde cheia primara (se obtine prin apel la *getPrimaryKeyFieldOfTable*) este egala cu *pk_value*.
- **deleteEntity** – functie care sterge o entitate (poate fi persoana, pacient, doctor, episod, pas medical sau instanta de concept)
    - parametrii:
        - sessions_identifyers – identificatorul de sesiune, pe baza acestui id se identifica userul in numele caruia se face acest apel la serviciu. Daca utilizatorul nu este logat si nu are un identificator de sesiune valid, operatiile nu se efectueaza.
        - entity – tipul entitatii la care se va sterge

        - entity_id – id-ul entitatii care se sterge
        - override_security – daca in caz special se face un apel la aceasta functie care in asa fel incat policy-ul de securitate nu ar permite realizarea operatiei, acest parametru setat la "true" marcheaza faptul ca policy-ul de securitate nu se va lua in considerare si se va efectua operatia in orice caz.
    - functionalitate:
        - Se sterg toate valorile din sloturi asociate cu entitatea care urmeaza sa fie stearsa
        - Se auto apeleaza recursiv si se sterg toate entitatile care sunt legate de aceasta entitate (ex. daca se sterge un pacient, se vor sterge toate episoadele medicale si toti pasii de tratament, asociate cu pacientul respectiv)
        - Se sterge entitatea propriuzisa.

#### *4.2.4.4.2 Modulul pentru controlul sessiunii de lucru*

Functiile exportate ca si servicii web din modulul pentru sesiune sunt definite in script-ul *SERVER/RPDEF/RPDEF_Session.php*, iar functiile ajutatoare (auxiliare) in *SERVER/RPDEF/AUX_Session.php.* Functiile sunt urmatoarele:

Functii WS:

- **GetNewUserForm** – returneaza continutul formului de inscriere pentru utilizatori noi.
    - parametrii: nu are
    - functionalitate: apeleaza functia *GetFormContent* cu parametrul "newUserInfo" pentru a obtine continutul formului de inscriere pentru utilizatopri noi.
- **GetListOfAvailableDoctors** – returneaza o lista de doctori (la inscriere un pacient isi va alege doctorul la care apartine)
    - parametrii:nu are
    - functionalitate:
        - construieste o lista de structuri de tip *DOCTOR_DESCRIPTOR*
        - umple lista cu doctorii din sistem
        - returneaza lista de doctori
- **GetSpecialities** – returneaza o lista de specialitati (la inscriere un doctor isi va alege specialitatea)
    - parametrii:nu are
    - functionalitate:
        - construieste o lista de structuri de tip *SPECIALITY*
        - umple lista cu specialitatile posibile
        - returneaza lista de specialitati
- **RegisterNewUser** – salveaza datele introduse de catre utilizator la crearea unui cont de utilizator nou.
    - parametrii:
        - uname – numele pentru contul utilizator cerut (acest nume se va folosi la login)
        - pass – parola pentru contul utilizator cerut (aceasta parola se va folosi la login)

        - roles_id – specifica daca contul de utilizator cerut este pentru doctor sau pacient.
        - specialities_id – daca rolul este de tip doctor, aici se specifica specialitatea doctorului.
        - Doctorid – daca rolul este de tip pacient, aici se specifica id-ul doctorului de familie la care se inscrie pacientul
        - PROPS – lista de elemente de tip PROPERTY datele personale completate de inscriere (nume, cnp, etc ... ). Aceste date se vor lega de persoana utilizatorului fie el doctor sau pacient.
    - functionalitate:
        - Se verifica CNP-ul
            - Daca CNP-ul nu este completat se returneaza un cod de eroare 2.
            - Daca lungimea nu este de 13 caractere se returneaza un cod de eroare 1.
            - Daca CNP-ul exista deja in baza de date se returneaza un cod de eroare 3.
        - Daca datele introduse sunt in regula
            - Se creaza o personana noua
            - Se creaza un doctor/pacient nou legat la persoana
            - Se creaza un utilizator nou care se leaga de persoana
            - Se introduc datele (din PROPS) asociate cu persoana in baza de date
            - In caz de succes returneaza 0
- **Login** – efectueaza o operatie de login, returnand date legate de sesiunea de login creata
    - parametrii:
        - user – numele de utilizator
        - pass – parola utilizatorului
    - functionalitate:
        - Verifica corectitudinea utilizatorului si a parolei
        - In caz de eroare (utilizator sau parola incorecta) returneaza 0
        - In caz de succes
            - Creeaza o structura de tip SESSION
            - Completeaza structura cu date
            - Apeleaza *generate_session* pentru a crea o sessiune de login si pentru a obtine identificatorul de sessiune asociat cu sessiunea noua
            - Returneaza structura SESSION
- **Logout** – efectueaza o operatie de logout, distruge sessiunea de login curenta
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - verifica daca exista o sessiune de login valida cu identificatorul precizat (apeland functia auxiliara *isValidUserSession)*
        - Sterge sessiunea cu identificatorul precizat din baza de date
        - Returneaza 1 la succes 0 la eroare.

- **ValidateSession** – efectueaza validare unei sessiuni de login, actualizeaza datele de sessiune
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - verifica daca exista o sessiune de login valida cu identificatorul precizat (apeland functia auxiliara *isValidUserSession)*
        - Daca exista sessiunea valida
            - Updateaza timpul de expirare a sessiunii
            - Completeaza structura SESSION cu datele curente
            - Returneaza structura SESSION
        - In caz de eroare returneaza false

Functii Aux:

- **getUserIdFromSession** – pe baza identificatorului de sessiune, returneaza id-ul de tip intreg al utilizatorului.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - Extrage din baza de date id-ul de tip intreg utilizatorului care este legat la sessiunea identificata de *sessions_identifyers*
- **getUserPersonIdFromSession** –
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - Extrage din baza de date id-ul de tip intreg al persoanei asociate cu utilizatorului care este legat la sessiunea identificata de *sessions_identifyers*
- **getUserFunctionFromSession** –
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - Extrage din baza de date functia (rolul – doctor / pacient) utilizatorului care este legat la sessiunea identificata de *sessions_identifyers*
- **getPersonNameFromSession** –
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - Extrage din baza de date numele persoanei asociate cu utilizatorului care este legat la sessiunea identificata de *sessions_identifyers*
- **isValidUserSession** –
    - parametrii:

        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - verifica daca exista o sessiune de login valida, identificata de *sessions_identifyers*, sessiunea este valida daca:
            - exista in baza de date o sessiune cu identificatorul *sessions_identifyers*
            - daca are IP-ul egal cu IP-ul de pe care utilizatorul curent se conecteaza
            - daca date expirarii sessiuni este un moment din viitor
        - Returneaza true sau false, true daca exista o sessiune valida, false in caz contrar.
- **generate_session** – genereaza o sessiune de login noua
    - parametrii:
        - USER – date legate de utilizator (ex. id-ul utilizatorului)
    - functionalitate:
        - Are ca si functionalitate ascunsa stergerea din BD a sessiunilor care au expirat de mai mult de o ora (pentru a evita umplerea bazei de date cu sessiuni expirate)
        - Se genereaza un identificator de sessiune ca si hash-ul md5 a unei variabile aleatoare cu valori intre 0 si 100000.
        - Daca exista o sessiune expirata (de mai putin de o ora), sau activa pentru utilizatorul curent, se updateaza sessiunea respectiva, pastrandu data de start si schimband data expirarii (se poate monitoriza cat de lung este o sessiune de login continua – cu intreruperi mai scurte de o ora) si identificatorul.
        - Daca nu exista sessiune expirata sau curenta pentru utilizatorul curent, se creeaza una cu identificatorul generat.
        - Se returneaza identificatorul.

#### *4.2.4.4.3 Modulul pentru serviciul de mesagerie - CNmessenger*

Functiile exportate ca si servicii web din modulul pentru CNmessenger sunt definite in script-ul *SERVER/RPDEF/RPDEF_CNmess.php*. Functiile sunt urmatoarele:

- **getOnlineDoctors** – returneaza lista doctorilor care sunt online pe serviciul de mesagerie
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - Selecteaza doctorii care sunt logati in sistem si sunt activi pe serviciul de mesagerie
        - Returneaza o lista cu doctori
- **getOnlinePatients** – returneaza lista pacientilor care sunt online pe serviciul de mesagerie
    - parametrii:

        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - Selecteaza pacientii care sunt logati in sistem si sunt activi pe serviciul de mesagerie
        - Returneaza o lista cu pacienti
- **getUndeliveredMessages** – returneaza lista mesaje care au ca si destinatar utilizatorul curent
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - extrage din baza de date toate mesajele care au ca si destinatar utilizatorul caruia ii apartine sessiunea identificata de *sessions_identifyers*, si care nu au fost livrati inca.
        - seteaza flagurile de mesaj livrat la acele mesaje care se trimt.
        - returneaza o lista de mesajele destinate utilizatorului
- **postMessage** – permite trimiterea unui mesaj prin serviciul de mesagerie
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - toid – id-ul de tip intreg al utilizatorului care este destinatarul mesajului.
        - msg – textul mesajului trimis (criptat cu algoritm RSA)
    - functionalitate:
        - decripteaza mesajul cu cheile de criptare RSA stocate in baza de date la sessiune.
        - introduce mesajul in baza de date si seteaza flag-ul de mesaj nelivrat.
- **mess_login** – efectueaza o operatie de conectare la serviciul de mesagerie
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - key_mod – parte a cheii RSA folosite pentru criptarea mesajelor - modulul din cheia RSA.
        - key_exp – parte a cheii RSA folosite pentru criptarea mesajelor - exponentul din cheia RSA.
    - functionalitate:
        - Seteaza flag-ul de conectat la serviciul de mesagerie (din baza de date / session)
        - Stocheaza perechea de chei de criptare RSA care se folosesc la criptarea/decriptarea mesajelor.
- **mess_logout** – efectueaza o operatie de deconectare de la serviciul de mesagerie
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:

        - Sterge flag-ul de conectat la serviciul de mesagerie (din baza de date / session)
        - Sterge perechea de chei de criptare RSA care se folosesc la criptarea/decriptarea mesajelor.
- **mess_status** –
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - status – textul care se seteaza ca si status pentru utilizatorul serviciului de mesagerie.
    - functionalitate:
        - Seteaza statusul utilizatorului (salveaza textul status in baza de date in sessions)

#### *4.2.4.4.4 Modulul general pentru form-uri*

Functiile exportate ca si servicii web din modulul pentru sesiune sunt definite in script-ul *SERVER/RPDEF/RPDEF_Forms.php*, iar functiile ajutatoare (auxiliare) in *SERVER/RPDEF/AUX_Forms.php.* Functiile sunt urmatoarele:

Functii WS:

- GetFormContent – functie care culege date care vor intra in continutul unui form pe baza unor xml-uri predefinite care specifica continutul formurilor
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - form – numele formului care se doreste completat (in cele mai multe cazuri numele formului este aceeasi cu numele fisierului xml care defineste continutul)
        - person_id – id-ul de tip intreg al persoanei la care se refera datele (o parte din date) din form.
        - episode_id – id-ul de tip intreg al episodului la care se refera datele (o parte din date) din form.
        - step_id – id-ul de tip intreg al pasului de tratament la care se refera datele (o parte din date) din form.
        - instance_id – id-ul de tip intreg al instantei de concept din ontologie la care se refera datele (o parte din date) din form.
        - override_security – daca in caz special se face un apel la aceasta functie care in asa fel incat policy-ul de securitate nu ar permite realizarea operatiei, acest parametru setat la “true” marcheaza faptul ca policy-ul de securitate nu se va lua in considerare si se va efectua operatia in orice caz.
    - functionalitate:
        - Apeleaza functia – de o singura data sau de mai multe ori – de extragere a datelor pe baza xml *getPropertySetBasedOnXml*. (De mai multe ori daca este un form compus din continuturi descrise in mai multe fisiere xml)

        - Returneaza lista de elemente de tip PROPERTY obtinuta prin concatenarea listelor de elemente de tip PROPERTY generate de apelurile la *getPropertySetBasedOnXml*.
- SetData – functie ce introduce date obtinute sub forma unui sir de elemente de tip PROPERTY in baza de date (face o sincronizare dintre sirul de PROPERTY-uri si baza de date)
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - PROPS – lista de elemente de tip PROPERTY din care trebuie extrase datele si introduse in BD.
        - person_id – id-ul de tip intreg al persoanei la care se refera datele (o parte din date) din lista.
        - episode_id – id-ul de tip intreg al episodului la care se refera datele (o parte din date) din lista.
        - step_id – id-ul de tip intreg al pasului de tratament la care se refera datele (o parte din date) din lista.
        - instance_id – id-ul de tip intreg al instantei de concept din ontologie la care se refera datele (o parte din date) din lista.
        - override_security – daca in caz special se face un apel la aceasta functie care in asa fel incat policy-ul de securitate nu ar permite realizarea operatiei, acest parametru setat la “true” marcheaza faptul ca policy-ul de securitate nu se va lua in considerare si se va efectua operatia in orice caz.
    - functionalitate:
        - parcurge lista PROPS
        - daca gaseste un element care are setat un *filterfunction* (numele functiei de filtru prin care se va trece valoarea inainte de stocarea ei in BD) apeleaza functia filtru care isi produce efectul asupra PROPS.
        - Dupa filtrare se mai parcurge o data PROPS si fiecare element se introduce in baza de date prin intermediul modulului de securitate.

Functii Aux:

- getPropertySetBasedOnXml – functie care culege date care vor intra in continutul unui form pe baza unui singur fisier xml predefinit
    - parametrii:
        - xmlfile – fisierul xml pe baza caruia se culeg datele care urmeaza sa intre in form.
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - person_id – id-ul de tip intreg al persoanei la care se refera datele (o parte din date) din lista.
        - episode_id – id-ul de tip intreg al episodului la care se refera datele (o parte din date) din lista.
        - step_id – id-ul de tip intreg al pasului de tratament la care se refera datele (o parte din date) din lista.
        - instance_id – id-ul de tip intreg al instantei de concept din ontologie la care se refera datele (o parte din date) din lista.

        - override_security – daca in caz special se face un apel la aceasta functie care in asa fel incat policy-ul de securitate nu ar permite realizarea operatiei, acest parametru setat la “true” marcheaza faptul ca policy-ul de securitate nu se va lua in considerare si se va efectua operatia in orice caz.
    - functionalitate:
        - Se parseaza fisierul XML
        - Pentru fiecare nod din care se genereaza PROPERTY (nod te tip input) se apeleaza o functie corespunzatoare care sa genereze PROPERTY-ul:
            - getEmptyFormContentFromSlotBasedOnXmlNode – daca se doreste generarea unui PROPERTY dintr-un slot, fara continut explicit, doar descriptor de slot.
            - getFilledFormContentFromSlotBasedOnXmlNode – daca se doreste generarea unui PROPERTY dintr-un slot, completat cu continut explicit – adcia valori din slot legate de o instanta anume getEmptyFormContentFromFieldBasedOnXmlNode – daca se doreste generarea unui PROPERTY dintr-un camp de tabela, fara continut explicit, doar descriptor de camp si tabela.
            - getFilledFormContentFromFieldBasedOnXmlNode – daca se doreste generarea unui PROPERTY dintr-un camp din tabela, completat cu continut explicit – adcia valoarea din camp.
        - Daca nodul are atasat o functie generatoare (similara cu functiile de filtrare doar ca acestea se ruleaza inainte de afisare si nu inainte de stocare) se apeleaza functia generatpare care-si produce efectul asupra setului de PROPERTY care s-a generat pana la acel moment.
- getEmptyFormContentFromSlotBasedOnXmlNode – creeaza un element de tip PROPERTY completat doar cu informatii generale referitoare la un slot, pe baza unui nod input din XML-ul care specifica continutul formului si din datele corespunzatoare din baza de date.
    - parametrii:
        - input – nod de tip NODE (definit in 2.2.3) creat pe baza unui nod din XML.
    - functionalitate:
        - Extrage valorile supraincarcate din nod-ul XML inclusiv listele valide de concepte si de instante (cu functiile *getConceptListFromTreeNodeList* si *getInstanceListFromTreeNodeList*)
        - Proceseaza valorile supraincarcate , si calculeaza numerele de ordine.
        - Obtine structura PROPERTY prin modulul de securitate, apeland functia *getEmptySlotInfo*
- getFilledFormContentFromSlotBasedOnXmlNode – creeaza un element de tip PROPERTY completat cu informatii generale referitoare la un slot si informatii specifice – date stocate in slot pentru o entitate anume – pe baza unui nod input din XML-ul care specifica continutul formului si din datele corespunzatoare din baza de date.
    - parametrii:
        - input – nod de tip NODE (definit in 2.2.3) creat pe baza unui nod din XML.

        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - person_id – id-ul de tip intreg al persoanei la care se refera datele (o parte din date) din lista.
        - episode_id – id-ul de tip intreg al episodului la care se refera datele (o parte din date) din lista.
        - step_id – id-ul de tip intreg al pasului de tratament la care se refera datele (o parte din date) din lista.
        - instance_id – id-ul de tip intreg al instantei de concept din ontologie la care se refera datele (o parte din date) din lista.
        - override_security – daca in caz special se face un apel la aceasta functie care in asa fel incat policy-ul de securitate nu ar permite realizarea operatiei, acest parametru setat la "true" marcheaza faptul ca policy-ul de securitate nu se va lua in considerare si se va efectua operatia in orice caz.
    - functionalitate:
        - Extrage valorile supraincarcate din nod-ul XML inclusiv listele valide de concepte si de instante (cu functiile *getConceptListFromTreeNodeList* si *getInstanceListFromTreeNodeList*)
        - Proceseaza valorile supraincarcate , si calculeaza numerele de ordine.
        - Obtine structura PROPERTY prin modulul de securitate, apeland functia *getFilledSlotInfo*
- getEmptyFormContentFromFieldBasedOnXmlNode – creeaza un element de tip PROPERTY completat doar cu informatii generale referitoare la un camp, pe baza unui nod input din XML-ul care specifica continutul formului si din datele corespunzatoare din baza de date.
    - parametrii:
        - input – nod de tip NODE (definit in 2.2.3) creat pe baza unui nod din XML.
    - functionalitate:
        - Extrage valorile supraincarcate din nod-ul XML (la campuri nu exista liste de concepte si instante)
        - Proceseaza valorile supraincarcate , si calculeaza numerele de ordine.
        - Obtine structura PROPERTY prin modulul de securitate, apeland functia *getEmptyFieldInfo*
- getFilledFormContentFromFieldBasedOnXmlNode – creeaza un element de tip PROPERTY completat cu informatii generale referitoare la un camp din baza de date si informatii specifice – date stocate in camp – pe baza unui nod input din XML-ul care specifica continutul formului si din datele corespunzatoare din baza de date.
    - parametrii:
        - input – nod de tip NODE (definit in 2.2.3) creat pe baza unui nod din XML.
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - person_id – id-ul de tip intreg al persoanei la care se refera datele (o parte din date) din lista.

        - episode_id – id-ul de tip intreg al episodului la care se refera datele (o parte din date) din lista.
        - step_id – id-ul de tip intreg al pasului de tratament la care se refera datele (o parte din date) din lista.
        - instance_id – id-ul de tip intreg al instantei de concept din ontologie la care se refera datele (o parte din date) din lista.
        - override_security – daca in caz special se face un apel la aceasta functie care in asa fel incat policy-ul de securitate nu ar permite realizarea operatiei, acest parametru setat la "true" marcheaza faptul ca policy-ul de securitate nu se va lua in considerare si se va efectua operatia in orice caz.
    - functionalitate:
        - Extrage valorile supraincarcate din nod-ul XML (la campuri nu exista liste de concepte si instante)
        - Proceseaza valorile supraincarcate , si calculeaza numerele de ordine.
        - Obtine structura PROPERTY prin modulul de securitate, apeland functia *getFilledFieldInfo*
- getInstanceListFromTreeNodeList – functie ce extrage o lista de instante cu valori supraincarcate si cu numere de ordine obtinute din nod-uri XML.
    - parametrii:
        - VALID_instances – lista de noduri de tip NODE (definit in 2.2.3) create pe baza unuor noduri din XML, alcatuiesc lista de instante.
    - functionalitate:
        - parcurge lista de instante, extrage valorile supraincarcate
        - creeaza un array de instante cu valori supraincarcate
        - returneaza array-ul
- getConceptListFromTreeNodeList – functie ce extrage o lista de concepte cu valori supraincarcate si cu numere de ordine obtinute din nod-uri XML.
    - parametrii:
        - VALID_concepts – lista de noduri de tip NODE (definit in 2.2.3) create pe baza unor noduri din XML, alcatuiesc lista de concepte.
    - functionalitate:
        - parcurge lista de concepte, extrage valorile supraincarcate
        - creeaza un array de concepte cu valori supraincarcate
        - returneaza array-ul.

#### *4.2.4.4.5 Modulul pentru utilizatori tip pacient*

Functiile exportate ca si servicii web din modulul pentru pacient sunt definite in script-ul *SERVER/RPDEF/RPDEF_Patient.php* si *SERVER/RPDEF/AUX_Patient.php*. Functiile sunt urmatoarele:

- ConsultOffline – efectuarea unui consult offline
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - PROPS – lista de elemete de tip PROPERTY care contine datele ce trebuie salvate in baza de date.

    - functionalitate:
        - creaza un episod nou de tip consult offline
        - salveaza datele din PROPS in episodul nou creat
- PatientResponse – continuarea unei consultatii offline de catre pacient
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - episode_id – id-ul de tip intreg al episodului continuat
        - PROPS – lista de elemete de tip PROPERTY care contine datele ce trebuie salvate in baza de date.
    - functionalitate:
        - Daca un consult offline a fost continuat de doctor pentru ca avea nevoie de informatii suplimentare, iar pacientul a furnizat acele informatii suplimentare
        - Se creaza un pas de tratament nou legat de episod
        - Se salveaza datele din PROPS in pas-ul nou creat
- GetEpisodeSingle – obtinerea detaliilor referitoare la un episod anume
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - episode_id – id-ul de tip intreg al episodului continuat
        - filldata – specifica daca descriptorul se completeaza sau nu cu date efective sau doar cu informatii generale referitoare la episod.
    - functionalitate:
        - Se construieste un descriptor de episod
        - Se completeaza cu date generale referitoare la episod(data, pacient, doctor etc...)
        - Daca filldata este true se completeaza date specifice legate de episod (date medicale)
- GetStepSingle – obtinerea detaliilor referitoare la un pas de tratament anume
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - step_id – id-ul de tip intreg al episodului continuat
        - filldata – specifica daca descriptorul se completeaza sau nu cu date efective sau doar cu informatii generale referitoare la step.
    - functionalitate:
        - Se construieste un descriptor de pas
        - Se completeaza cu date generale referitoare la pas (data, pacient, doctor etc...)
        - Daca filldata este true se completeaza date specifice legate de pas (date medicale)
- GetNewConsultOfflineList – obtinerea unei liste de episoade medicale care sunt de tip consultatie offline nou creata.
    - parametrii:

        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - Se construieste o lista de descriptori de episod
        - Se completeaza cu date generale (data, pacient, doctor etc...) referitoare la episoadele care satisfac conditia de a fi consultatii offline noi
        - Se returneaza lista de descriptori
- GetPendingConsultOfflineList – obtinerea unei liste de episoade medicale care sunt de tip consultatie offline continuate care asteapta evaluarea doctorului.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - Se construieste o lista de descriptori de episod
        - Se completeaza cu date generale (data, pacient, doctor etc...) referitoare la episoadele care satisfac conditia de a fi consultatii offline asteptand evaluare.
        - Se returneaza lista de descriptori
- GetRepliedConsultOfflineList – obtinerea unei liste de episoade medicale care sunt de tip consultatie offline care a fost continuat de doctor, doctorul necesitand informatii suplimentare.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - Se construieste o lista de descriptori de episod
        - Se completeaza cu date generale (data, pacient, doctor etc...) referitoare la episoadele care satisfac conditia de a fi consultatii offline continuate care necesita informatii suplimentare.
        - Se returneaza lista de descriptori
- GetResolvedConsultOfflineList – obtinerea unei liste de episoade medicale care sunt de tip consultatie offline rezolvata.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - Se construieste o lista de descriptori de episod
        - Se completeaza cu date generale (data, pacient, doctor etc...) referitoare la episoadele care satisfac conditia de a fi consultatii offline rezolvate.
        - Se returneaza lista de descriptori
- GetCurrentEpisodeList – obtinerea unei liste de episoade medicale care sunt de tip curent, adica nu sunt terminate inca.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:

        - Se construieste o lista de descriptori de episod
        - Se completeaza cu date generale (data, pacient, doctor etc...) referitoare la episoadele care  satisfac conditia de a fi consultatii in decurs de desfasurare.
        - Se returneaza lista de descriptori
- GetFinishedEpisodeList – obtinerea unei liste de episoade medicale care sunt incheiate.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - Se construieste o lista de descriptori de episod
        - Se completeaza cu date generale (data, pacient, doctor etc...) referitoare la episoadele care  satisfac conditia de a fi consultatii incheiate.
        - Se returneaza lista de descriptori

#### *4.2.4.4.6 Modulul pentru utilizatori tip doctor*

Functiile exportate ca si servicii web din modulul pentru doctor sunt definite in script-ul *SERVER/RPDEF/RPDEF_Doctor.php* si *SERVER/RPDEF/AUX_Doctor.php*. Functiile sunt urmatoarele:

- acceptNewPatientUser – functie pentru acceptarea unui utilizator nou creat pentru un pacient.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - user_id – id-ul de tip intreg al utilizatorului care se accepta ca si pacient.
    - functionalitate:
        - verifica drepturile utilizatorului (doar doctor sau administrator are dreptul sa accepte un pacient)
        - se verifica daca contul acceptat este chiar de tip pacient (sa nu fie doctor sau admin)
        - se activeaza contul.
- rejectNewPatientUser – functie pentru respingerea unui utilizator nou creat pentru pacient.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - user_id – id-ul de tip intreg al utilizatorului care se accepta ca si pacient.
    - functionalitate:
        - verifica drepturile utilizatorului (doar doctor sau administrator are dreptul sa respinga un pacient)
        - se verifica daca contul acceptat este chiar de tip pacient (sa nu fie doctor sau admin)
        - se respinge contul.
- doc_GetNewPatientUserList – folosit pentru obtinerea unei liste cu utilizatorii noi de tip pacient.

    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - creeaza o lista de structuri de tip USER_DESCRIPTOR
        - completeaza structura cu datele utilizatorilor noi
        - returneaza lista
- doc_GetNewPatientUserSingle – pentru obtinerea informatiilor (sub forma unei structuri USER_DESCRIPTOR ) cu care se completeaza form-ul in care se afiseaza detaliile utilizatorului nou care urmeaza sa fie acceptat sau respins.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - user_id – id-ul de tip intreg al utilizatorului pentru care se doreste obtinerea informatiilor.
        - filldata – flag care semnaleaza faptul ca se doreste sau nu completarea USER_DESCRIPTOR – ului cu date efective (datele personale din sloturi si campuri care se refera la persoana asociata cu utilizatorul)
    - functionalitate:
        - creeaza o structura de tip USER_DESCRIPTOR
        - completeaza structura cu date referitoare la utilizatorul nou identificate de user_id
        - returneaza structura
- doc_GetNewConsultOfflineList – functie pentru obtinerea unei liste de structuri de tip CONSULT_DESCRIPTOR completate pe baza unor episoade medicale ce satisfac conditia de a fi consultatii offline noi.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - construieste o lista de elemente de tip CONSULT_DESCRIPTOR
        - verifica drepturile utilizatorului (doar doctor sau administrator are dreptul sa vizualizeze lista de consultatii offline)
        - completeaza lista cu structuri CONSULT_DESCRIPTOR completate pe baza episoadelor care satisfac conditia de a fi consultatii offline noi.
        - Returneaza lista.
- doc_GetPendingConsultOfflineList – functie pentru obtinerea unei liste de structuri de tip CONSULT_DESCRIPTOR completate pe baza unor episoade medicale ce satisfac conditia de a fi consultatii offline in decurs de desfasurare.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
    - functionalitate:
        - construieste o lista de elemente de tip CONSULT_DESCRIPTOR
        - verifica drepturile utilizatorului (doar doctor sau administrator are dreptul sa vizualizeze lista de consultatii offline)

        - completeaza lista cu structuri CONSULT_DESCRIPTOR completate pe baza episoadelor care satisfac conditia de a fi consultatii offline in decurs de desfasurare.
        - Returneaza lista.
- doc_GetEpisodeSingle – functie pentru obtinerea unei structuri de tip CONSULT_DESCRIPTOR completate pe baza unui episod medical anume.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - episode_id – id-ul de tip intreg al episodului pentru care se doreste obtinerea informatiilor.
        - filldata – flag care semnaleaza faptul ca se doreste sau nu completarea CONSULT_DESCRIPTOR – ului cu date efective (datele medicale care se refera la episodul specificat)
    - functionalitate:
        - construieste o structura de tip CONSULT_DESCRIPTOR
        - completeaza structura cu date referitoare la episodul identificat de episode_id
        - returneaza structura
- doc_GetStepSingle – functie pentru obtinerea unei structuri de tip CONSULT_DESCRIPTOR completate pe baza unui pas de tratament anume.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - step_id – id-ul de tip intreg al pasului de tratament pentru care se doreste obtinerea informatiilor.
        - filldata – flag care semnaleaza faptul ca se doreste sau nu completarea CONSULT_DESCRIPTOR – ului cu date efective (datele medicale care se refera la pasul de tratament specificat)
    - functionalitate:
        - construieste o structura de tip CONSULT_DESCRIPTOR
        - completeaza structura cu date referitoare la pasul de tratament identificat de step_id
        - returneaza structura
- ResolveConsultOfflineSingle – functie care se apeleaza cand o consultatie offline devine rezolvata (se incheie cu o rezolutie de gen diagnostic, medicatie impusa etc...)
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - episode_id – id-ul de tip intreg al episodului care reprezinta consultatia offline care urmeaza a fi rezolvata
        - step_id – id-ul de tip intreg al pasului de tratament (daca exista) care este ultimul pas – la care se adauga datele medicale noi.
        - PROPS – lista de elemente de tip PROPERTY, contine informatiile de natura medicala care se salveaza la rezolvarea consultatiei.
    - functionalitate:

        - calculeaza data de terminare a episodului – care va fi data curenta
        - marcheaza episodul medical ca fiin incheiat
        - salveaza datele medicale noi din PROPS in episod sau in pas de tratament
            - daca consultatia offline era rezolvata din prima, datele medicale vor fi asociate cu episodul medical ce reprezinta consultatia
            - daca consultatia offline s-a continuat inainte de a fi rezolvata, datele medicale vor fi asociate atat cu episodul medical ce reprezinta consultatia cat si cu pasii de tratament asociati cu episodul.
- TmpsaveConsultOfflineSingle – functie care salveaza datele modificate pentru o consultatie dar lasa starea consultatiei nealterata.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - episode_id – id-ul de tip intreg al episodului care reprezinta consultatia offline care urmeaza a fi salvata temporar
        - step_id – id-ul de tip intreg al pasului de tratament (daca exista) care este ultimul pas – la care se adauga datele medicale noi.
        - PROPS – lista de elemente de tip PROPERTY, contine informatiile de natura medicala care se salveaza la rezolvarea consultatiei.
    - functionalitate:
        - nu altereaza starea episodului
        - salveaza datele medicale noi din PROPS in episod sau in pas de tratament.
- ContinueConsultOfflineSingle – functie care se apeleaza cand doctorul mai necesita informatii suplimentarepentru rezolvarea consultatiei
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - episode_id – id-ul de tip intreg al episodului care reprezinta consultatia offline care urmeaza a fi continuata
        - step_id – id-ul de tip intreg al pasului de tratament (daca exista) care este ultimul pas – la care se adauga datele medicale noi.
        - PROPS – lista de elemente de tip PROPERTY, contine informatiile de natura medicala care se salveaza la rezolvarea consultatiei.
    - functionalitate:
        - seteaza flag-ul de focus (pingpong) pentru a aduce episodul medical in atentia pacientului (doctorul necesita informatii suplimentare)
        - salveaza datele medicale noi din PROPS in episod sau in pas de tratament
- RejectConsultOfflineSingle – in cazul in care se constata un abuz al serviciului de consultatii offline aceasta functie respinge / sterge consultul offline.
    - parametrii:
        - sessions_identifyers – identificatorul de sessiune (hash de 32 caractere ce identifica sessiunea de login)
        - episode_id – id-ul de tip intreg al episodului care reprezinta consultatia offline care urmeaza a fi respinsa

        - step_id – id-ul de tip intreg al pasului de tratament (daca exista) care este ultimul pas – nu are rol aici.
        - PROPS – lista de elemente de tip PROPERTY, contine informatiile de natura medicala – nu are rol aici
    - functionalitate:
        - se verifica daca episodul identificat de episode_id este chiar o noua consultatie offline (nu se permite stergerea unu episod daca acesta contine date medicale completate de profesionisti – daca este o noua consultatie offline contine doar date completate de pacient)
        - se sterge episodul si toate datele medicale asociate de episod din baza de date.

# 5 Sistem de asistență pentru diagnoză şi tratament

## 5.1 Logica aplicației

Diagnosticarea progresivă a unui pacient este obiectivul principal al acestei implementări. Aplicația se adreseaza medicilor specialişti care descopera şi trateaza bolnavii cu afecțiuni cardio-vasculare. Pacienții descoperiți şi diagnosticați cu o astfel de afecțiune vor rămâne permanent în sistemul medical sub urmărirea atentă a doctorului specialist. Pentru a veni în ajutorul medicului în descoperirea unui bolnav care are semne de afecțiuni cardiace am implementat un sistem care să îl ajute şi să îl ghideze. Acest sistem se bazează pe informația existentă în planurile medicale create de cei de la eu-heartfaid ( http://lis.irb.hr/heartfaid/plans/ ). Aceste planuri sunt create de specialişti în domeniul medical şi sunt sub forma de arbori n-ari de decizii. De fapt reprezintă paşii care ar trebui urmați de doctorii specialişti atunci când consultă şi diagnostichează un pacient. Aceşti arbori decizionali arată sub forma următoare:

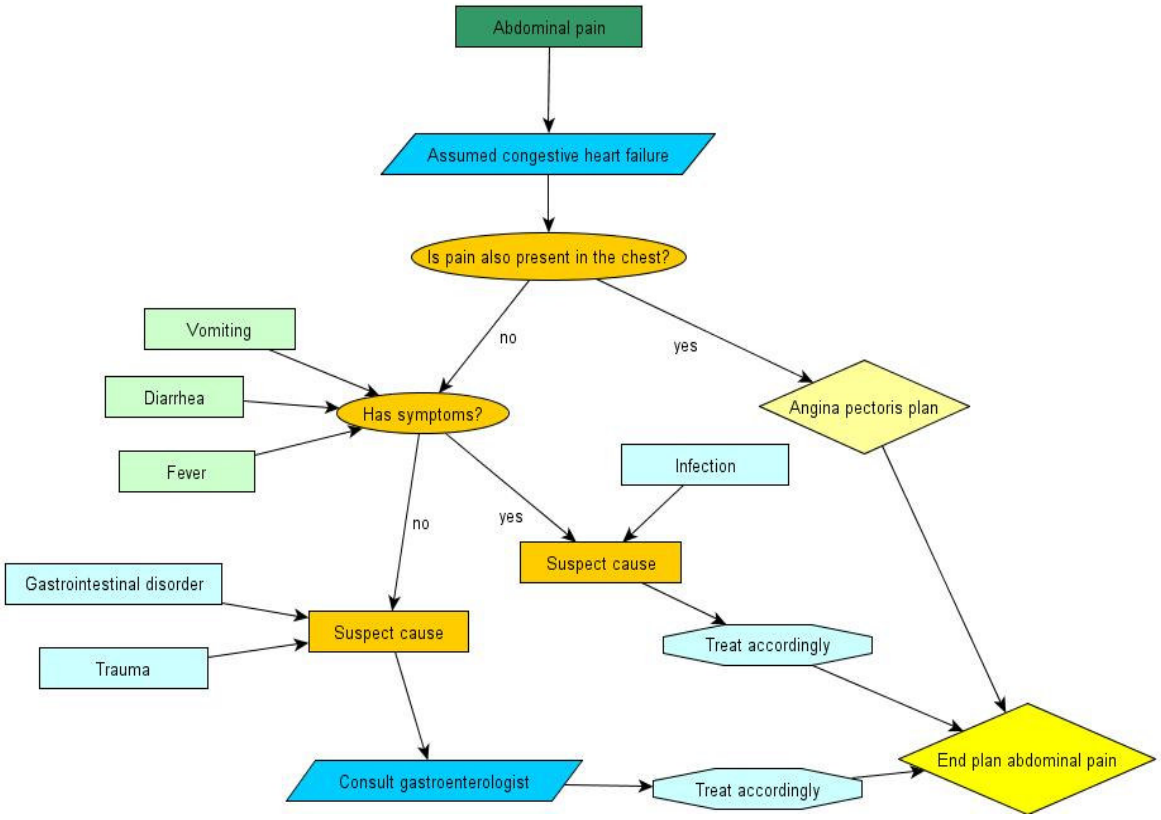



**Fig 5.1: Planul Abdominal pain**

În vederea transpunerii acestor cunoştințe într-o formă posibilă de înțeles şi de procesat de către un calculator am transformat informația existentă în fiecare plan în câte un fişier **.xml**. Ca atare întreaga asistență pe care o oferă calculatorul doctorului se bazează pe aceste fisiere xml create din arborii decizionali.

## 5.2 Descrierea planurilor de diagnoză şi tratament prin formatul XML

### 5.2.1 Elemente preliminare privind documentul XML

Pentru a întelege XML-urile utilizate trebuie să întelegem modul în care este formatat documentul. Tag-ul este folosit pentru a defini un element în XML. Este principala unealtă folosită şi trebuiesc respectate anumite reguli. Un tag va fi tot timpul cuprins de semnele '<' şi '>'. Ce este între cele două semne reprezintă numele tag-ului.
Tag-urile sunt de două feluri:

- tag de început - care indică începutul elementului şi este de forma '<tag>';
- tag de sfârşit - care indică terminarea definirii elementului şi arată ca tag-ul de început doar ca are imediat dupa semnul '<' un semn '/'.

Astfel că un tag arată în general în acest fel: <tag>continut</tag> . Seamănă cu un tag HTML. Aşa este dar cu o mica-mare deosebire: tag-urile HTML sunt predefinite, iar cele XML nu. Dacă în HTML avem: informatie este interpretat de HTML ca 'textul informatie dintre cele doua tag-uri trebuie să fie îngroşat'. În schimb în XML s-ar putea interpreta 'textul informatie este o dată de tip strong'.

Respectând condițiile sintactice impuse de specificațiile XML am folosit în gramatică utilizată 2 categorii de tag-uri :

- simple: prin aceste tag-uri ne referim doar la acelea de tipul

  <exemplu_tag>Informație relevantă </exemplu_tag>

- compuse: prin aceste tag-uri ne referim la acelea compuse care pot avea sub-tag-uri

  (respectiv tag-uri copil) de forma:

  <medicament> Captopril

  <initial_dosage>6.25 mg t.i.d. </initial_dosage>

  <maintenance_dosage> 25-50 mg t.i.d.</maintenance_dosage>

  <mean_daily_dose> 127 mg</mean_daily_dose>

  <target_dose> 50 mg t.i.d.</target_dose>

  </medicament>

Fişierele xml se bazează pe următorele concepte (concepte care sunt definite şi stocate în tabela ConceptType):

| Id | DescriptionRo | DescriptionEN |
|---|---|---|
| 1 | Diagnostic | Diagnosis |
| 2 | Test | Test |
| 3 | Cauza | Cause |
| 4 | Tratament | Treatment |

| | | |
|---|---|---|
| 5 | Indiciu | Tip |
| 6 | Simptom | Symptom |
| 7 | Sugestie | Suggestion |
| 9 | Medicament | Medicine |
| 10 | Doza initiala | InitialDosage |
| 11 | Doza de mentinere | Maintenance Dosage |
| 12 | Doza zilnica medie | Mean Daily Dosage |
| 13 | Doza optima | Target Dosage |
| 14 | Doza zilnica maxima | Maximum Daily Dosage |
| 15 | Risc | Risk |
| 17 | Plan | Plan |
| 18 | Intrebare | Question |
| 19 | Varianta | QuestionVariant |
| 20 | Presupunere | Assumption |
| 21 | Scop | Scope |
| 22 | Operatie | Surgery |
| 23 | Echipament | Device |
| 24 | Lista simptomuri | Simptom List |
| 26 | Lista cauze | Cause List |
| 27 | Schimbare plan | Change Plan |
| 28 | Inchidere plan | End Plan |
| 29 | Lista teste | Test List |
| 30 | Lista riscuri | Risk List |
| 31 | Risc slab | Weaker Risk |
| 32 | Risc mediu | Medium Risk |
| 33 | Risc mare | High Risk |
| 34 | Lista operatii | Surgery List |
| 35 | In afara scopului | Out of Scope |
| 36 | Lista procedee | Procedure List |
| 37 | Procedeu | Procedure |
| 38 | Doza prescrisa | Prescribed Dosage |

Aceste concepte reprezintă defapt tag-urile utilizate în fisierele xml create din arborii decizionali. Un exemplu uşor şi simplu de fişier în care avem atât tag-uri simple cât şi compuse este:

----------Abdominal pain plan.xml----------

```
<plan>
  <name> Abdominal pain plan </name>
  <assum> Assumed congestive heart failure </assum>
  <question>
    Is abdominal pain also present in the chest?
    <no>
```

```
    <simptom_list>
      <simptom> Vomiting </simptom>
      <simptom> Diarrhea </simptom>
      <simptom> Fever </simptom>
    </simptom_list>
    <question>
      Has simtoms vomiting, diarrhea or fever?
      <no>
        <possible_cause>
          <cause> Gastrointestinal disorder </cause>
          <cause> Trauma </cause>
        </possible_cause>
        <sugestion> Consult gastroenterologist </sugestion>
        <treatment> Treat acordingly </treatment>
        <endplan>abdominal pain</endplan>
      </no>
      <yes>
        <possible_cause>
          <cause>Infection</cause>
        </possible_cause>
        <treatment> Treat acordingly </treatment>
        <endplan>abdominal pain</endplan>
      </yes>
    </question>
  </no>
  <yes>
    <changeplan>Angina pectoris plan</changeplan>
    <endplan>abdominal pain</endplan>
  </yes>
 </question>
</plan>
```

### 5.2.2 Structura fişierului xml

Arborii decizionali descrişi sunt împarțiți în 2 categorii: arbori decizionali de investigare şi diagnosticare şi arbori decizionali de tratament.

1. Arborii decizionali de investigare sunt cei care conțin informațiile de care are nevoie doctorul pentru a diagnostica un pacient. Ele sunt planuri care încep cu o întrebare şi în funcție de raspunsul acordat la acea întrebare ghidează medicul în decoperirea şi diagnosticarea pacientului. Urmând paşii descrişi în aceste planuri se ajunge fie la o rezolutie, fie la o schimbare de plan (simpomele şi raspunsurile date de pacient ghidează doctorul într-o altă direcție), fie la un plan de tratament.

2. Arbori decizionali de tratament sunt cei care contin tratamentul medicamentos pe care îl va prescrie medicul odată ce a ajuns la o rezoluție şi a identificat boala pacientului.

Pentru ca aplicația să poată folosi în totalitate cunoştințele din arborii decizionali primul pas pe care l-am făcut a fost de a transforma fiecare arbore în parte în câte un fişier xml. Pasul doi a fost acela de a parsa fişierele şi de a le introduce într-o bază de date relațională în asa fel încât procesarea ulterioară din cadrul aplicației să fie mult mai uşoară.

În efectuarea pasului unu a fost nevoie să definesc o gramatică care să suporte întreg bagajul de cunoştințe existent în arborii decizionali. Am optat pentru crearea de fişiere xml uşoare, fara prea multe standarde, care să fie utile aplicației. Pentu ca aplicația să cunoască şi să poată utiliza fişierele este nevoie ca acestea să respecte un minim set de reguli, reguli pe care am să le indic în cele ce urmează.

Structura obligatorie a unui fişier xml de diagnosticare este:

```
<plan>
    <nume> Numele planului.xml (asa cum este el salvat în directorul care conține toate planurile) </nume>
    .....
    <question>
        <yes>
            .....
        </yes>
        <no>
            .....
        </no>
    </question>  .....
    --restul este optional--
</plan>
```

Structura obligatorie a unui fişier xml de medicație este:

```
<plan>
    <nume> Numele planului.xml (asa cum este el salvat în directorul care conține toate planurile) </nume>
    .....
    --restul este optional--
</plan>
```

Optional însemnand orice combinație de tag-uri definite şi utilizate în procesare.

Totodata este necesar să se folosească tag-urile definite. În cazul în care folosim un tag nedefinit aplicația pur şi simplu nu îl va lua în considerare şi nu va fi procesat (se va introduce în tabela Concept ca şi unknown dar vor fi complet ignorate ). Tag-urile sunt introduse în tabela ConceptType şi în BussinessConstants în cadrul aplicației.

Fiecare tag va defini o entitate în cadrul aplicației. Entitățile folosite sunt de 4 feluri:

1. Entități editabile: în aceasta categorie intră acele entități care pot fi editate de doctor în cazul în care ceea ce sugerează sistemul nu corespunde cu ceea ce doreşte medicul sau în cazul în care acesta trebuie sa completeze rezultate la anumite analize şi teste făcute de pacient: **test, surgery, treatment, prescribed_dosage**

2. Entități grupabile: acestea sunt singurele tag-uri care pot fi compuse şi în aceasta categorie intră:

- **simptom_list** (utilizată pentru a grupa într-o lista toate tag-urile **simptom**) grupare necesară pentru o procesare şi o afişare mai simplă a simptoamelor pe care le acuză pacienții,
- **possible_cause** (utilizată pentru a grupa într-o lista toate tag-urile **cause**) grupare necesară pentru o procesare şi o afişare mai simplă a cauzelor care stau la baza anumitor boli
- **check_risks** (utilizată pentru a grupa într-o lista toate tag-urile de tipul **risk** – medium,weaker, high) grupare necesară pentru o afişare mai simplă a ricurilor care trebuiesc luate în vedere de către medici în anumite situații
- **procedure_list** (utilizată pentru a grupa într-o lista toate tag-urile de tipul **procedure**) grupare necesară pentru o procesare şi o afişare mai simplă a procedurilor medicale pe care trebuie să le facă anumiți pacienți
- **test_list** (utilizată pentru a grupa într-o lista toate tag-urile de tipul **test**) grupare necesară pentru o procesare şi o afişare mai simplă a testelor pe care trebuie să le facă anumiți pacienți în anumite laboratoare de specialitate
- **surgery_list** (utilizată pentru a grupa într-o lista toate tag-urile de tipul **surgery**) grupare necesară pentru o procesare şi o afişare mai simplă a operațiilor medicale pe care trebuie să le facă anumiți pacienți
- **treatment** (utilizată pentru a grupa într-o lista toate tag-urile de tipul **medicament, suggestion, surgery_list, surgery** sau orice alt tag care reprezintă un sfat medical) grupare necesară pentru o procesare şi o afişare mai simplă a tratamentului medical pe care trebuie să îl facă anumiți pacienți
- **medicament (**utilizată pentru a grupa într-o lista toate tag-urile de tipul **medicament** şi numele medicamentelor prescrise **)** grupare necesară pentru o procesare şi o afişare mai simplă a medicamentelor prescrise de doctor pacienților
- **change_plan** grupare necesară pentru a putea încărca următorul plan, deci următorea entitate, deci avansare cu înca un nivel în încărcarea recursivă

3. Entități reutilizabile: în aceasta categorie intră acele entități care pot fi reutiliza acelaşi concept: **diagnosis, test, cause, treatment, symptom, suggestion, medicine, initial_dosage, maintenance_dosage, mean_daily_dosage, target_dosage, maximum_daily_dosage, prescribed_dosage, risk, question, surgery, device, out_of_scope, procedure, change_plan**

4. Entități unice: în aceasta categorie intră doar singura entitate unică existentă adica **plan**

### 5.2.3 Parsarea fişierelor xml în baza de date

După ce au fost create fişierele xml urmează pasul doi acela de a le introduce într-o bază de date relațională. Folosirea bazelor de date în loc de procesare directă pe xml se justifică prin faptul că: bazele de date sunt optimizate şi relaționate, sunt mult mai rapide, tratează erorile de scriere (care pot aparea în fisierele xml exemplu: suggestion / sugestion cu toate că în fişier ne referim la acelaşi tag) şi se pot obtine orice informații dorim fără sa fie nevoie să încarcăm fişierele de fiecare dată etc.

Pentru a putea utiliza toate informațiile medicale existente în fişierele xml folosesc 3 tabele:

1. ConceptType (în care am definit toate conceptele pe care le folosesc pe parcursul aplicației), am descris-o mai sus
2. Concept (este tabela care contine toată informația existentă în fisierele xml create din arborii decizionali) şi
3. ConceptHierarchy (folosită strict pentru a memora ordinea informațiilor, oferind ierarhia acțiunilor deci locul unde se fac relațiile).

Inițial concepusem baza de date ca având întrebări, planuri, iar toate conceptele erau legate de acestea în mod ierarhic. Pe parcurs mi-am dat seama că datorită acestei concepții posibilitatea de a naviga între diferitele planuri şi de a folosi diferitele concepte conținute în acestea limita serios posibilitatea de trecere de la un plan la altul (şi revenire) precum şi posibilitățile de a combina elementele. Ca şi exemplu, o întrebare aparținea întotdeauna unui plan, conceptele erau legate diferit de întrebări şi de planuri, căutarea după concepte era mai anevoioasă.

Varianta actuală presupune salvarea tuturor elementelor folosite în aplicație ca şi concepte şi diferențierea lor în cadrul aplicației. Prin urmare baza de date foloseşte exclusiv la stocarea informațiilor şi a legăturilor dintre ele. În cadrul variantei actuale există două structuri

importante, tabela de concepte unde sunt salvate absolut toate entitățile prezente în aplicație si tabela de ConceptHierarchy unde sunt făcute relațiile. Spre deosebire de multe alte aplicații, în tabela ConceptHierarchy legăturile nu sunt făcute între concepte ci între entități de tip ConceptHierarchy. Acest lucru oferă o flexibilitate mult mai mare în formarea legăturilor, practic oferă posibilități nelimitate de a lega conceptele, dar, la fel de important, oferă posibilitatea de a restrânge datele salvate ca şi concepte şi posibilitatea de reutilizare a unor concepte. De exemplu la o întrebare există două variante de răspuns: Da şi Nu, iar acestea reprezintă conceptul QuestionVariant. În loc să salvăm câte o pereche de astfel de concepte pentru fiecare concept de tip întrebare (Question) creăm o singură pereche de astfel de concepte după care le reutilizăm prin intermediul sistemului de ierarhizare. Practic putem reutiliza marea majoritate a conceptelor. Acest lucru ne oferă în primul rând posibilitatea de a restrânge volumul de date salvat (în cazul a mii de întrebări şi medicamente – inclusiv dozaje) şi, foarte important, ne permite să grupăm aceste concepte şi sa extragem date filtrate într-un mod foarte simplu (de exemplu: pacienții cărora li s-a administrat medicamentul X, sau pacienții care au avut dispozitivul Y instalat, sau toate medicamentele al căror dozaj zilnic este Z).

În momentul de față, datorită structurii bazei de date adăugarea de concepte noi este foarte simplă: baza de date rămâne aceeaşi (cu excepția tabelei ConceptType unde se vor adăuga noile concepte), iar singura care trebuie modificată este aplicația, în sensul că funcția de importare date să recunoască noile concepte, iar partea de consult care să fie actualizată pentru a putea utiliza noile concepte în cadrul unui consult.

Pentru a întelege mecanismul în care sunt introduse datele in cele 2 tabele (Concept şi ConceptHierarchy) luăm un exemplu concret.

- Alegem un fişier xml (mai simplu): Abdominal pain plan.xml

```
<plan>
 <name> Abdominal pain plan </name>
 <assum> Assumed congestive heart failure </assum>
 <question>
  Is abdominal pain also present in the chest?
  <no>
   <simptom_list>
    <simptom> Vomiting </simptom>
    <simptom> Diarrhea </simptom>
    <simptom> Fever </simptom>
   </simptom_list>
   <question>
    Has simtoms vomiting, diarrhea or fever?
    <no>
     <possible_cause>
      <cause> Gastrointestinal disorder </cause>
      <cause> Trauma </cause>
     </possible_cause>
     <sugestion> Consult gastroenterologist </sugestion>
     <treatment> Treat acordingly </treatment>
     <endplan>abdominal pain</endplan>
    </no>
```

```
    <yes>
      <possible_cause>
        <cause>Infection</cause>
      </possible_cause>
      <treatment> Treat acordingly </treatment>
      <endplan>abdominal pain</endplan>
    </yes>
  </question>
 </no>
 <yes>
  <changeplan>Angina pectoris plan</changeplan>
  <endplan>abdominal pain</endplan>
 </yes>
</question> </plan>
```

- Introducem fişierul în tabela Concept: la început se vor încarca toate conceptele. Cele care sunt reutilizabile se vor încărca doar o singură dată. Ne este necesară încarcarea aceloraşi concepte de câte ori se găsesc în planuri deorece tabela de ierarhie se foloseşte şi pentru acest lucru.

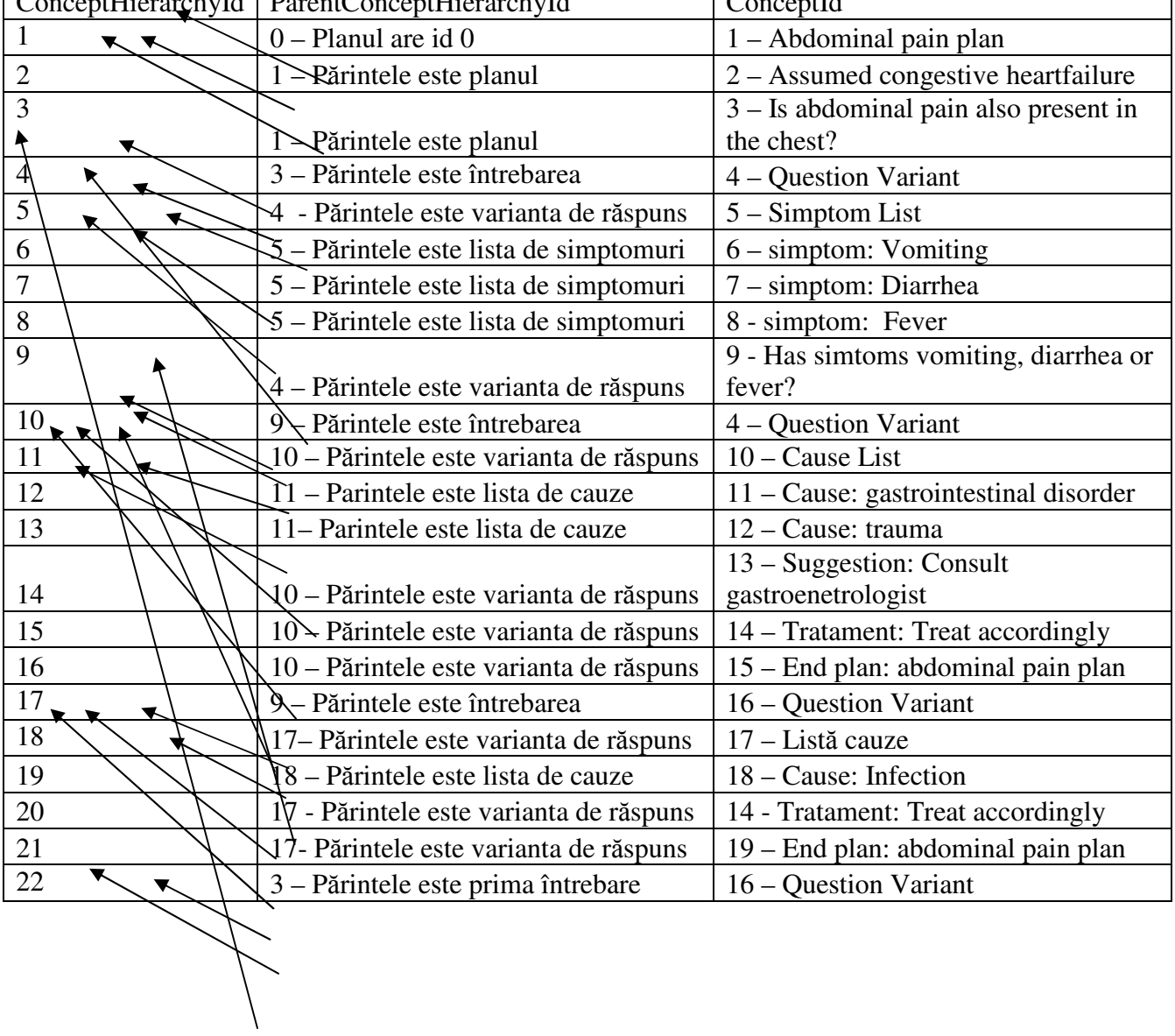
| ConceptId | ConceptTypeId | Description | IsAffirmative | NewConceptId |
|---|---|---|---|---|
| 1 | 17 | Abdominal pain plan | NULL | NULL |
| 2 | 20 | Assumed congestive heart failure | NULL | NULL |
| 3 | 18 | Is abdominal pain also present in the chest? | NULL | NULL |
| 4 | 19 | | FALSE | NULL |
| 5 | 24 | | NULL | NULL |
| 6 | 6 | Vomiting | NULL | NULL |
| 7 | 6 | Diarrhea | NULL | NULL |
| 8 | 6 | Fever | NULL | NULL |
| 9 | 18 | Has simtoms vomiting, diarrhea or fever? | NULL | NULL |
| 10 | 26 | | NULL | NULL |
| 11 | 3 | Gastrointestinal disorder | NULL | NULL |
| 12 | 3 | Trauma | NULL | NULL |
| 13 | 7 | Consult gastroenterologist | NULL | NULL |
| 14 | 4 | Treat acordingly | NULL | NULL |
| 15 | 28 | abdominal pain | NULL | NULL |
| 16 | 19 | | TRUE | NULL |

| | | | | |
|---|---|---|---|---|
| 17 | 26 | | NULL | NULL |
| 18 | 3 | Infection | NULL | NULL |
| 19 | 28 | abdominal pain | NULL | NULL |
| 20 | 27 | Angina pectoris plan | NULL | NULL |
| 21 | 28 | abdominal pain | NULL | NULL |

Oridecâteori în fişierul xml vom găsi un tag de schimbare plan (changeplan), corespunzător concepului 27 – Change Plan (din tabela ConceptType) se va căuta id-ul planului spre care se va face schimbarea şi va fi trecut în câmpul NewConceptId. Dacă planul înca nu există introdus în bd se trece null. Oricum la o schimbare de plan vom face căutarea acestuia şi după câmpul Description (câmp a cărui valoare trebuie să fie identică cu numele fişierului xml).

- Îl introducem şi în tabela ConceptHierarchy (am adăugat explicații ca să fie mai uşor de urmărit):

| ConceptHierarchyId | ParentConceptHierarchyId | ConceptId |
|---|---|---|
| 1 | 0 – Planul are id 0 | 1 – Abdominal pain plan |
| 2 | 1 – Părintele este planul | 2 – Assumed congestive heartfailure |
| 3 | 1 – Părintele este planul | 3 – Is abdominal pain also present in the chest? |
| 4 | 3 – Părintele este întrebarea | 4 – Question Variant |
| 5 | 4 - Părintele este varianta de răspuns | 5 – Simptom List |
| 6 | 5 – Părintele este lista de simptomuri | 6 – simptom: Vomiting |
| 7 | 5 – Părintele este lista de simptomuri | 7 – simptom: Diarrhea |
| 8 | 5 – Părintele este lista de simptomuri | 8 - simptom: Fever |
| 9 | 4 – Părintele este varianta de răspuns | 9 - Has simtoms vomiting, diarrhea or fever? |
| 10 | 9 – Părintele este întrebarea | 4 – Question Variant |
| 11 | 10 – Părintele este varianta de răspuns | 10 – Cause List |
| 12 | 11 – Parintele este lista de cauze | 11 – Cause: gastrointestinal disorder |
| 13 | 11– Parintele este lista de cauze | 12 – Cause: trauma |
| 14 | 10 – Părintele este varianta de răspuns | 13 – Suggestion: Consult gastroenetrologist |
| 15 | 10 – Părintele este varianta de răspuns | 14 – Tratament: Treat accordingly |
| 16 | 10 – Părintele este varianta de răspuns | 15 – End plan: abdominal pain plan |
| 17 | 9 – Părintele este întrebarea | 16 – Question Variant |
| 18 | 17– Părintele este varianta de răspuns | 17 – Listă cauze |
| 19 | 18 – Părintele este lista de cauze | 18 – Cause: Infection |
| 20 | 17 - Părintele este varianta de răspuns | 14 - Tratament: Treat accordingly |
| 21 | 17- Părintele este varianta de răspuns | 19 – End plan: abdominal pain plan |
| 22 | 3 – Părintele este prima întrebare | 16 – Question Variant |



| | | |
|---|---|---|
| 23 | 22 – Părintele este varianta de răspuns | 20 – Change plan: Angina pectoris plan |
| 24 | 22– Părintele este varianta de răspuns | 21 – End plan: abdominal pain plan |

Unul din motivele principale pentru care am optat pentru această variantă este acela că în anumite planuri este nevoie să se ştie ordinea în care se parcurge un fişier. Adică este posibil ca la un moment dat să ai o schimbare de plan (către un plan de medicaţie) iar mai apoi investigarea să continue. Un exemplu elocvent pentru această situaţie este Heart attack plan.xml unde pentru început trebuie verificată o listă de simptome, apoi există un tratament imediat, urmează nişte teste care trebuie să fie efectuate după care urmeză întrebarea decisivă. Alegând o variantă de răspuns vom găsi un tratament (format din sugestii şi operaţii) iar mai apoi o altă întrebare. Alegând o variantă de răspuns ajungem la un alt tratament (cu o clasă de medicamente prescrise iar mai apoi o schimbare de plan la un plan specific de medicaţie) după care urmeză o alta întrebare şi procesul continuă mai departe. Deci după schimbarea de plan este necesară revenirea în planul de unde am plecat pentru a continua investigarea. Mecanismul prin care se face revenirea la planul din care se pleacă este unul simplu. Pentru fiecare element se încarca doar lista de elemente care ţin strict de el. Deci o încarcare doar a unui nivel din planul respectiv. În cazul în care elementul este marcat ca şi grupabil, lucru care înseamnă că poate să aibă o lista dupa el (exemplu: Symptom_List), se mai merge un nivel. Prin urmare la încarcarea unui "change plan" se face legătura (dacă nu era făcută deja) către planul care punctează după care se încarcă primul nivel de copii ai planului conform schemei de mai sus. Toate elementele care sunt fraţi ai „change planu-lui" (prin fraţi întelegând elemente pe acelaşi nivel) dar după el devin inactive (adică invizibile) până cand planul se consideră a fi complet. Această verificare se face la fiecare randare (deci la fiecare postback).
Exemplu de fişier complex cu revenire extras din Heart attack plan.xml:

```
.....
<treatment> Prescribe medication
        <medicament>ARB</medicament>
        <changeplan>ARBs dosage plan</changeplan>
 </treatment>
 <question>
      Has left ventricular systolic dysfunction?
.....
</question>
```

### 5.2.4 Asistarea interactivă a medicului pe parcursul unui consult

Odată introdusă toata informaţia în baza de date, sub o formă uşor de înţeles şi într-o modalitate perfect optimizată, asistarea medicului în diagnosticarea şi tratarea pacienţilor este pasul imediat următor. Întreaga informaţie de diagnosticare se găseşte în baza de date. Aceste planuri sunt compuse în principal dintr-o listă de simpoame (lista care este opţională) şi o serie de întrebari cu ajutorul cărora se va stabili diagnisticul. În funcţie de simpoamele pe care le prezintă pacientul şi în funcţie de răspunurile acordate întrebarilor sistemul va conduce medicul, pe un drum ales de acesta, ajungând în cele din urma la un diagnostic sau la o schimbare de plan în cazul în care drumul ales nu este corect şi ceea ce prezintă pacientul sunt semnele clare a unei alte afecţiuni.

Acest proiect a fost gândit astfel încât să permită adăugarea unor consulturi automate cât mai complexe. Prin urmare design-ul bazei de date a luat în calcul posibilitatea de adăugare a noi concepte şi legături variate şi multiple între acestea.

Scopul meu a fost să permit consulturi cât mai complexe pentru a reuşi să redau cât mai aproape de realitate posibilitatea unui consult real în spital. Prin urmare planurile conţin o mulțime de concepte care sunt structurate ierarhic. Totodată un consult poate deveni foarte complex datorită posibilității de a „naviga" de la un plan la altul. În acest caz ceea ce reprezintă un consult devine ceva mult mai complex datorită posibilității de a combina informațiile cuprinse într-un plan strict de întrebări, cu legături către planuri de medicație, alte planuri cu întrebări, opțiuni (dintre care se poate alege una).

Inițial consultul între un medic şi un pacient începe prin o lista de simptoame pe care le acuză pacientul. Doctorul introduce aceste simpoame în sistem (căutându-le în drop down-ul aferent ). Pe măsura ce aceste simptoame sunt introduse, din toate planurile existente în baza de date, se selectează şi se afişează doar acelea care au simpomele respective. Ca atare se face o procesare simultană a tuturor planurilor şi în funcție de simptoamele introduse bază de cunoştințe se restrânge. Următorul pas este de a lua în considerare întrebările pe care ți le pun planurile selectate. În funcție de întrebarea selectată se parcurge planul ales (şi confirmat de doctor) şi diagnosticarea poate începe. În cazul în care doctorul a ales greşit întrebarea poate reveni şi poate modifica planul (prin alegerea altei întrebări). Consultul poate să înceapă şi prin alegerea directă a unui plan de diagnosticare, în cazul în care doctorul specialist ştie exact ce are pacientul. Un consult poate fi de 2 feluri:

1. Simplu : care are o rezoluție finală atunci când începe
2. Compus: care se desfăşoara în mai multe episode. Ne referim aici la un consult care necesită efectuare de teste de laborator, operații etc. Asta înseamnă că consultul nu are o rezoluție evidentă şi este necesar să se aştepte efectuarea anlizelor şi deci primirea rezultatelor. Ca atare ne vom referi la un episod ca şi un pas dintr-un consult compus. Odată terminat un episod (terminat însemnând ca se ajunge la nişte teste care trebuiesc efectuate şi e nevoie de rezultatul acestora ca să poți merge mai departe cu procesul de diagnosticare) acesta se salvează. Când pacientul revine cu rezultatele, procesul de diagnosticare continuă exact din punctul în care a fost salvat. Medicul introduce rezultatele analizelor şi noul episod începe. Aceşti paşi pot continua până când în cele din urmă se ajunge la un rezultat şi consultul poate fi încheiat. Rezultatul este diagnosticarea pacientului şi tratarea acestuia fie direct, fie prin intermdiul unui plan de medicație.

Planurile de medicație sunt tratate puțin diferit şi asta pentru că structura lor nu este ca şi a unui plan de diagnosticare. Flow-ul acestuia este putin diferit şi nu respectă aceeaşi structură. Ca atare acesta nu poate fi accesibil imediat doctorului pentru că nu are o întrebare de început. Structura unui astfel de xml este (ACE inhibitor dosage plan.xml):

<plan>

```
<name> ACE Inhibitor Dosage plan </name>
<medicament>
  Captopril
  <initial_dosage>6.25 mg t.i.d. </initial_dosage>
  <maintenance_dosage> 25-50 mg t.i.d.</maintenance_dosage>
  <mean_daily_dose> 127 mg</mean_daily_dose>
  <target_dose> 50 mg t.i.d.</target_dose>
</medicament>
<medicament>
  Enalapril
  <initial_dosage> 2.5 mg/day</initial_dosage>
  <maintenance_dosage> 10 mg b.i.d.</maintenance_dosage>
  <mean_daily_dose> 18.4 mg; 15.0 mg; 16.6 mg</mean_daily_dose>
  <target_dose> 20 mg b.i.d.; 10 mg b.i.d.</target_dose>
</medicament>
<medicament>
  Lisinopril
  <initial_dosage> 2.5 mg/day</initial_dosage>
  <maintenance_dosage> 5-20 mg/day</maintenance_dosage>
  <target_dose>High dose: 32.5-35 mg; Low dose: 2.5-5 mg </target_dose>
</medicament>
<medicament>
  Ramipril
  <initial_dosage> 1.25-2.5 mg/day</initial_dosage>
  <maintenance_dosage> 2.5-5 mg b.i.d.</maintenance_dosage>
  <target_dose> 5 mg b.i.d.</target_dose>
</medicament>
<medicament>
  Trandolapril
  <initial_dosage> 1 mg/day</initial_dosage>
  <maintenance_dosage> 4 mg/day </maintenance_dosage>
  <target_dose> 4 mg/day </target_dose>
</medicament>
</plan>
```

Însă cum scopul aplicației este acela de a ajuta doctorul şi nu a-l încurca, am făcut un xml generic, care să conțina toate planurile de medicație în asa fel încât, medicul, raspunzând la întrebari, să poată accesa direct planul de medicație dorit şi scrie rețeta. Fişierul este: Medications plan.xml care arată sub forma:

```
<plan>
        <name>Medications plan</name>
        <question>Do you want to see direcly a dosage plan?
                <yes>
                        <question>Do you want ACE inhibitor dosage?
                                <yes>
                                        <changeplan>ACE inhibitor dosage plan</changeplan>
                                        <endplan>Medications plan</endplan>
```

</yes>
<no>
<question>Do you want Aldosterone receptor antagonists dosage ?
<yes>
<changeplan>Aldosterone receptor antagonists dosage plan</changeplan>
<endplan>Medications plan</endplan>
</yes>
<no> .....
</plan>

În acest fel medicul specialist poate vedea direct un plan de medicație, cu dozele existente, astfel încât informația din ele îl poate ghida, fără a fi nevoit să se parcurgă anumite planuri de diagnosticare.

Un episod, odată ajuns la final, va avea un diagnostic automat, dat de sistem, dar există şi posibilitatea ca medicul să îşi introducă propriul diagnostic indiferent de ceea ce îi oferă sistemul.Totodată pe parcursul planurilor de diagnosticare doctorul are posibilitatea să introducă manual, de la tastatură, rezultatele analizelor, testelor efectuate de pacienți, iar pe parcursul planurilor de medicație are posibilitatea să introducă dozajul dorit la medicamente în cazul în care ceea ce îi oferă sistemul nu corespunde cu ceea ce doreşte.

Se păstrează un istoric la fiecare consult, respectiv la fiecare episod în parte. Istoricul va putea fi vizualizat în 2 forme. Una detaliată, cuprinzând pas cu pas episodul efectuat şi una sumară care cuprinde doar câteva informații esențiale pentru ca un medic specialist să vadă evoluția pacientului consultat cum ar fi: doctorul, pacientul, data episodului, diagnosticul automat dat de sistem, diagnosticul prescris de medic, simpoamele prezentate de pacient, testele pe care le-a efectuat cu valorile acestora, operațiile efectuate cu rezultatele avute, tratamentul / tratamentele recomandate, medicamentele administrate, device-uri instalate.

## 5.3 Structura aplicației de asistență

### 5.3.1 Definiția entităților

#### 5.3.1.1 Design-ul structurilor

Structurile de date reprezintă entitățile folosite pentru a transporta / reprezenta datele aplicației. Ele pot fi dezvoltate pentru orice nivel al aplicației: structuri pentru nivelul prezentare, nivelul aplicație sau nivelul date.

Pentru această aplicație structurile de date vor fi disponibile pentru toate nivelele iar fiecare entitate din sistem va reprezenta o tabelă din baza de date.

#### 5.3.1.2 Descrierea structurii de date utilizate pentru diagnosticare:

Toate elementele utilizate în stabilirea unui diagnostic aparțin unui plan. Prin urmare vor exista două tipuri de planuri: planuri de tratament (care vor conține în special informații despre medicamente şi dozajul acestora) şi planuri de diagnostic (care vor conține întrebări, opțiuni, teste, simptome, tratament, operații etc.)

#### *5.3.1.3 Concept*

Conceptul e o formă specială de entitate care poate reprezenta mai multe tipuri de valori (obiecte). În implementarea actuală conceptele sunt păstrate în tabela ConceptType.

Un caz special îl reprezintă dozele de medicament deoarece ele sunt de fapt sub-concepte în sensul că toate dozele de medicație au ca şi concept părinte un concept de tip medicament. Totodată în aplicație ele nu vor fi folosite fără a fi relaționate la un medicament.

#### *5.3.1.4 Plan de tratament*

Planurile de tratament sunt compuse dintr-o listă de medicamente. Pentru fiecare medicament sunt specificate una sau mai multe din dozele definite în cadrul tipurilor de concept.

**Plan de diagnosticare**

Planurile de diagnosticare sunt cele mai complexe. Ele sunt compuse în special din o listă inițială de simptoame (Concept – ConceptType=simptom) – care poate sa lipsească – şi o serie de întrebări cu ajutorul cărora se stabileşte diagnosticul (Question).
Întrebările sunt compuse din următoarele elemente:

- Textul întrebării
- Varianta de răspuns (QuestionVariant) – yes/no

### 1.2.1. Baza de date a aplicației

În imaginea următoare este prezentat design-ul cu entitățile implicate în formarea unui diagnostic.


**Fig 20: Baza de date pentru aplicație**

**Descrierea tabelelor utilizate pentru stocarea datelor:**

| Denumire tabelă | Descriere funcționalitate |
|---|---|
| **Users** | Această tabelă conține informațiile despre utilizatori: date personale + date pentru identificarea lor în aplicație |
| **UserType** | Aici sunt stocate tipurile de utilizatori. Există 3 tipuri de utilizatori: Pacienți, Doctori, Administratori.<br>Pacienții în momentul de față nu au acces la aplicație, ei sunt doar utilizați pentru consulturi.<br>Doctorii au anumite secțiuni restricționate, iar datele despre pacienți sunt filtrate pentru fiecare doctor.<br>Administratorii au acces în toate secțiunile aplicației. |
| **Concept** | Reprezintă entitatea care conține date de care aplicația se foloseşte. Aceste date pot reprezenta orice fel de informație relevantă pentru aplicație. În momentul de față sunt înregistrate mai bine de 30 de entități: plan, întrebare, simptom, test, operație etc. |
| **ConceptType** | Reprezintă tipurile de concepte care sunt înregistrate în aplicație. Totodată ele conțin un câmp special denumit SystemType care este un |

| | |
|---|---|
| | număr prin care aplicația identifică un anumit tip de concept. Tipurile de concepte sunt identificate prin intermediul acestui câmp şi nu pe baza denumirii (care poate varia în funcție de limbă) |
| **ConceptHierarchy** | În această tabelă sunt memorate toate relațiile posibile între conceptele memorate în baza de date. Singurele concepte pentru care există o înregistrare ConceptHierarchy care nu are părinte sunt conceptele de tip Plan. Pentru acestea câmpul ParentConceptHierarchyId are valoarea zero.<br>Relațiile între concepte sunt memorate cu ajutorul unei înregistrări de tip ConceptHierarchy care face legătura către conceptul memorat şi ParentConceptHierarchyId. Din această cauză acelaşi concept poate fi utilizat în planuri diferite sau pentru obținerea mai multor logici de diagnosticare. |
| **Consult** | Este utilizată pentru a memora un consult. Deoarece consultul este considerat ca fiind interacțiunea doctor-pacient de la primele simptome până la însănătoşirea pacientului, în această tabelă se memorează doar legături către doctor şi pacient. |
| **Episode** | Aici se memorează detaliile unui episod din cadrul unui consult. Deoarece un consult poate să se întindă de-a lungul mai multor întâlniri doctor-pacient, o astfel de întâlnire reprezintă un episod. Într-un episod se memoreaza atât date personale despre pacient cât şi elementele consultului propriu-zis.<br>În această tabelă sunt memorate date despre pacient la momentul întâlnirii precum şi un diagnostic automat rezultat în urma consultului împreună cu notele doctorului. |
| **EpisodeConcept** | Reprezintă un element al unui episod. Pentru fiecare element al consultului se salvează datele introduse de către medic (comentarii, dacă este selectat – în cazul unei operații sau a unui test – , dacă este vizibil sau dacă s-a introdus o valoare).<br>Aceste elemente reprezintă diferite concepte şi poziționarea lor în sistemul de diagnosticare cu ajutorul ierarhiei de concepte şi sunt memorate într-o ierarhie asemănatoare cu cea de la ConceptHierarchy. |

**Users:**

| Denumire câmp | Descriere |
|---|---|
| UserId | Id generat automat de către sistem |
| Username | Nume utilizator folosit pentru identificarea în sistem |
| Password | Parola utilizator folosită pentru identificarea în sistem |
| LastName | Numele de familie |
| FirstName | Prenumele |
| UserTypeId | Id către tipul de utilizator (vezi UserType) |
| CNP | CNP |
| BirthDate | Data naşterii |
| Male | Sexul utilizatorului |
| Bisexual | Pentru cazurile în care utilizatorii sunt bisexuali |

**UserType:**

| Denumire camp | Descriere |
|---|---|
| UserTypeId | Id generat automat de către sistem |
| SystemType | Id de sistem prin care aplicația identifică tipul de utilizator dorit |
| TypeNameRO | Denumirea în română |
| TypeNameEN | Denumirea în engleză |

**Concept:**

| Denumire camp | Descriere |
|---|---|
| ConceptId | Id generat automat de către sistem |
| ConceptTypeId | Id către tipul de concept (vezi ConceptType) |
| DescriptionRO | Descrierea în română.<br>Descrierea poate semnifica numele planului, textul întrebării sau noul plan (în cazul unui concept de tip ChangePlan) |
| DescriptionEN | Descrierea în engleză.<br>Descrierea poate semnifica numele planului, textul întrebării sau noul plan (în cazul unui concept de tip ChangePlan) |
| IsAffirmative | Folosit exclusiv pentru conceptele de tip QuestionVariant. Semnifică faptul dacă varianta este pozitivă (Yes) sau negativă (No) |
| NewConceptId | Id către noul plan către care se face trecerea. Folosit exclusiv în cazul conceptelor de tip ChangePlan.<br>În cazul în care planul dorit (salvat în câmpul DescriptionXX) nu este găsit acest id este lăsat fără date (NULL) şi respectiva acțiune nu se mai produce. |

**ConceptType:**

| Denumire camp | Descriere |
|---|---|
| ConceptTypeId | Id generat automat de către sistem |
| DescriptionRO | Descrierea în română |
| DescriptionEN | Descrierea în engleză |
| SystemType | Id de sistem prin care aplicația identifică tipul de concept |

**ConceptHierarchy:**

| Denumire camp | Descriere |
|---|---|
| ConceptHierarchyId | Id generat automat de către sistem |
| ParentConceptHierarchyId | Părintele unei înregistrări. Doar ierarhiile de concept care corespund conceptelor de tip plan nu au părinte (id = 0) |
| ConceptId | Conceptul reprezentat de înregistrarea respectivă.<br>Legăturile sunt realizate între ierarhiile de concepte în loc să fie realizate între concepte pentru a da mai mare flexibilitate legăturilor şi pentru a permite reutilizarea conceptelor de câte ori |

| | |
|---|---|
| | e nevoie reducând semnificativ numărul acestora. |

**Consult:**

| Denumire camp | Descriere |
|---|---|
| ConsultId | Id generat automat de către sistem |
| DoctorId | Id către utilizatorul de tip doctor care face consultul (vezi Users) |
| PatientId | Id către utilizatorul de tip pacient care participă în consult (vezi Users) |

**Episode:**

| Denumire camp | Descriere |
|---|---|
| EpisodeId | Id generat automat de către sistem |
| ConsultId | Id către consultul de care aparține episodul respectiv |
| EpisodeDate | Data la care s-a produs episodul |
| PatientHeight | Înălțimea pacientului |
| PatientWeight | Greutatea pacientului |
| IsMarried | Starea civilă a pacientului |
| Diagnosis | Diagnostic automat completat de către sistem în urma consultului |
| DiagnosisNotes | Diagnostic completat de către doctor |

**EpisodeConcept:**

| Denumire camp | Descriere |
|---|---|
| EpisodeConceptId | Id generat automat de către sistem |
| EpisodeId | Id către episodul de care aparține conceptul episodului |
| ParentEpisodeConceptId | Id către părintele conceptului episodului.<br>În cazul conceptelor episodului care nu au părinte (simptomele pacientului şi întrebările de început) acest id are valoarea 0. |
| ConceptHierarchyId | Id către ierarhia de concept. Este folosit pentru a identifica conceptul utilizat în cadrul sistemului de diagnosticare (folosirea doar a id-ului de concept nu este suficientă deoarece acelaşi concept poate apărea în mai multe planuri de diagnosticare) |
| ConceptId | Id către conceptul pentru care se memorează informațiile |
| IsVisible | Vizibilitatea conceptului episodului.<br>Un concept al episodului poate fi invizibil (nu este afişat) în cazul în care s-a făcut o schimbare de plan, conceptul de episod curent este o întrebare conținută în planul original situată după schimbarea de plan, iar planul la care s-a făcut schimbarea înca nu este considerat completat (are întrebări la care se aşteaptă alegerea unei variante) |
| IsSelected | Indică daca conceptul episodului este selectat.<br>De exemplu, în cazul unei liste de teste, unele teste pot să nu fie făcute de către doctor, caz în care nu selectează respectivele teste. |
| Value | Valoarea conceptului episodului. Folosit în special pentru |

| | |
|---|---|
| | dozajele recomandate de către doctor pentru un anumit medicament. |
| Comments | Comentarii făcute de doctor pentru respectivul concept al episodului (poate fi test, operație etc.) |

### 1.2.2. Descrierea interacțiunii între entitățile din baza de date

După cum se observă din diagrama bazei de date toate entitățile utilizate sunt salvate generic ca şi concepte. Tabela ConceptHierarchy permite crearea unui arbore n-ar în care toate tipurile de concepte pot fi salvate ca şi subconcepte ale unui alt concept. Există o singură excepție, care este reprezentată de conceptul de tip „Plan" care nu poate fi subconcept al nici unui alt concept. În continuare sunt prezentate tipurile de concepte şi modul cum interacționează între ele:

| Concept | Concept părinte | Descriere |
|---|---|---|
| Cauză | Listă cauze | **Rol în consult:** este afişată într-o listă de cauze posibile<br>**Afişare:** text simplu |
| Diagnostic | Î Variantă răspuns | **Rol în consult:** informare simplă sau o acțiune complexă<br>**Afişare:** ca şi text simplu sau ca şi o grupare de întrebări, teste, operații |
| Dispozitiv | Î Variantă răspuns | **Rol în consult:** utilizare în cadrul unui procedeu/operații<br>**Afişare:** text simplu |
| Doză de menținere | Medicament | **Rol în consult:** afişată pentru a delimita doza din medicamentul respectiv<br>**Afişare:** text simplu |
| Doză inițială | Medicament | **Rol în consult:** afişată pentru a delimita doza din medicamentul respectiv<br>**Afişare:** text simplu |
| Doză prescrisă | Medicament | **Rol în consult:** modificată de către medic pentru a se potrivi cazului curent<br>**Afişare:** casuță de text pentru introducerea valorii |
| Doză zilnică medie | Medicament | **Rol în consult:** afişată pentru a delimita doza din medicamentul respectiv<br>**Afişare:** text simplu |
| Indiciu (tip) | Plan | **Rol în consult:** rol de grupare a mai multor posibilități de tratament<br>**Afişare:** grupare de întrebări, medicație |
| În afara scopului | Î Variantă răspuns | **Rol în consult:** când nu se recunoaşte entitatea la import sau pentru a afişa opțiuni necunoscute<br>**Afişare:** text simplu |
| Întrebare | Diagnostic | **Rol în consult:** modalitate de a descoperi boala de care suferă pacientul<br>**Afişare:** text simplu + opțiuni de răspuns |
| | Î Variantă răspuns | |
| | Plan | |

| Concept | Concept părinte | Descriere |
|---|---|---|
| Î Variantă răspuns | Întrebare | **Rol în consult:** răspuns specific la o întrebare dată. Determină evoluția consultului şi a întrebărilor ulterioare<br>**Afişare:** text simplu + grupare de orice alte elemente |
| Listă cauze | Î Variantă răspuns | **Rol în consult:** listă posibilă de cauze care pot provoca simptomul (simptoamele) prezentat(e)<br>**Afişare:** grupare de cauze. Text simplu |
| | Plan | |
| Listă operații | Î Variantă răspuns | **Rol în consult:** listă de operații care pot fi sau trebuie efectuate<br>**Afişare:** grupare de operații. Pentru fiecare operație se va prezenta opțiune dacă s-a efectuat precum si note din partea doctorului. |
| Listă procedee | Î Variantă răspuns | **Rol în consult:** Procedeu care trebuie sa fie efectuat de către medic în cazul în care este nevoie<br>**Afişare:** text simplu |
| Listă riscuri | Plan | **Rol în consult:** Listă de riscuri care trebuie luate în considerare de către medic când va selecta continuarea tratamentului / consultului<br>**Afişare:** text simplu (grupare de riscuri) |
| Listă simptoame | Î Variantă răspuns | **Rol în consult:** listă posibilă de simptoame care, în caz că sunt prezente, trebuie adăugate în lista specială pentru simptoame<br>**Afişare:** grupare de simptoame. Text simplu |
| | Plan | |
| Listă teste | Î Variantă răspuns | **Rol în consult:** listă de teste care trebuie efectuate<br>**Afişare:** Pentru fiecare test se va prezenta opțiune dacă s-a efectuat precum si note din partea doctorului. |
| | Plan | |
| Medicament | Diagnostic | **Rol în consult:** face parte din tratamentul pacientului<br>**Afişare:** text simplu + grupare de doze |
| | Indiciu (tip) | |
| | Î Variantă răspuns | |
| | Plan | |
| | Tratament | |
| Nume | Plan | **Rol în consult:** Numele planului care include o serie de concepte<br>**Afişare:** fără afişare |

| Concept | Concept părinte | Descriere |
|---|---|---|
| Operație | Listă operații | **Rol în consult:** operație care trebuie efectuată<br>**Afişare:** Se va prezenta opțiune dacă s-a efectuat precum si note din partea doctorului. |
| Plan | | **Rol în consult:** modalitate de grupare de concepte care corespund unei boli sau unor boli similare<br>**Afişare:** fără afișare |
| | Schimbare plan | |
| Presupunere | Plan | **Rol în consult:** Un fapt care este prezent si care trebuie luat în considerare pentru a răspunde la întrebările următoare<br>**Afişare:** text simplu |
| Procedeu | Lista procedee | **Rol în consult:** procedeu care trebuie efectuat de către medic în cazul în care este nevoie<br>**Afişare:** text simplu |
| Risc (listă) | Risc mediu | **Rol în consult:** un risc posibil pentru un anumit nivel de risc precizat<br>**Afişare:** text simplu |
| | Risc ridicat | |
| | Risc slab | |
| Risc mediu | Listă riscuri | **Rol în consult:** grupare de riscuri care corespund unui risc mediu pentru determinarea medicației care va fi prescrisă unui pacient<br>**Afişare:** text simplu + grupare de riscuri |
| Risc ridicat | Listă riscuri | **Rol în consult:** grupare de riscuri care corespund unui risc mediu pentru determinarea medicației care va fi prescrisă unui pacient<br>**Afişare:** text simplu + grupare de riscuri |
| Risc slab | Listă riscuri | **Rol în consult:** grupare de riscuri care corespund unui risc mediu pentru determinarea medicației care va fi prescrisă unui pacient<br>**Afişare:** text simplu + grupare de riscuri |
| Schimbare plan | Indiciu (tip) | **Rol în consult:** într-un anumit moment în consult se poate trece la o altă serie de întrebări, plan medicamentos etc. după care se revine la consultul inițial<br>**Afişare:** grupare de concepte. Vezi Plan |
| | Î Variantă răspuns | |
| | Tratament | |
| Scop | Diagnostic | **Rol în consult:** scopul pentru care se fac următoarele acțiuni din consult<br>**Afişare:** text simplu |
| | Î Variantă răspuns | |

| Concept | Concept părinte | Descriere |
|---|---|---|
| Simptom | Listă simptoame | **Rol în consult:** un anumit simptom / semn pe care îl prezintă pacientul<br>**Afişare:** text simplu |
| Sugestie | Î Variantă răspuns | **Rol în consult:** sugestii pentru continuarea consultului, posibile boli, cauze, alte tipuri de consult etc.<br>**Afişare:** text simplu |
| Terminare plan | Î Variantă răspuns | **Rol în consult:** determină oprirea ramurii de consult respective<br>**Afişare:** fără afişare. |
| Test | Listă teste | **Rol în consult:** reprezintă un test care trebuie efectuat asupra pacientului şi care poate da anumite răspunsuri privind starea de sănătate sau să rezulte simptome noi.<br>**Afişare:** Se va prezenta opțiune dacă s-a efectuat precum si note din partea doctorului. |
| Tratament | Diagnostic | **Rol în consult:** reprezintă o modalitate de tratare a pacientului pe baza datelor obținute până în acel punct<br>**Afişare:** text simplu |
| | Î Variantă răspuns | |
| | Plan | |

Fiecare concept poate fi afişat în mod separat, dar conceptele similare se pot grupa şi modalitățile de afişare se reduc la următoarele posibilități:

| Modalitate afişare | concepte |
|---|---|
| Afişare opțiune răspuns (se poate alege o opțiune sau alta) | Î Variantă răspuns |
| Afişare text: „Va rugam verificati daca exista urmatoarele *nume_concept*:” + lista concepte | Listă cauze, Listă riscuri, Listă simptome |
| Afişare text: „Recomandăm următoarele *nume_concepte* sa fie realizate + listă editare concepte | Listă operații, Listă teste, Listă procedee |
| Afişare text simplu fără să fie încapsulat într-un element | Cauză, Risc, Simptom |

### 5.3.2 Structura Spatiului de nume (namespace)

| Nr. | Namespace | Description |
|---|---|---|
| **1.** | PATIENTDIAGNOSIS.WEB | **Conține fişierele aspx/ascx şi codul pentru procesarea datelor (C#)** |

| | | |
|---|---|---|
| **2.** | **PatientDiagnosis.Web.Utils** | **Conține clase suplimentare pentru accesul la resurse, formatare date etc.** |
| **3.** | **PatientDiagnosis.Web.Controls** | **Conține controalele utilizator** |
| **4.** | **PatientDiagnosis.Entities** | **Conține obiectele Entitate** |
| **5.** | **PatientDiagnosis. BusinessLayer** | **Conține clasele nivelului aplicație (business)** |
| **6.** | **PatientDiagnosis. DataLayer** | **Conține clasele nivelului de date** |

## 5.4 Detalii de proiectare

### 5.4.1 Nivelul de prezentare (UI)

Design-ul interfeței utilizator (UI) este simpl,u pentru a veni în ajutorul medicului şi a-i uşura munca.

Modulul de client web (UI) constă în mai multe fişiere aspx şi clasele de cod corespunzătoare. De obicei există o pereche de fişiere .aspx şi aspx.cs pentru fiecare pagină. În aceste fişiere vor fi codate doar conținutul specific acelei pagini. Conținutul comun tuturor paginilor va fi creat in masterpage şi nu va fi duplicat. Peste tot pe unde a fost posibil am creat componente independente (controale de tip utilizator), iar codul lor va fi plasat în namespace-ul Controls.

Toate textele existente în modulul UI se pot ulterior internaționaliz folosind uneltele existente în Visual Studio. Modulele GUI web client constau din fişiere aspx cu clasele C# corespondente.

**Masterpage:**

Aici sunt implementate toate componentele vizuale care sunt comune aplicației. Design-ul primar este stabilit aici şi pagina este împărțită în zone.

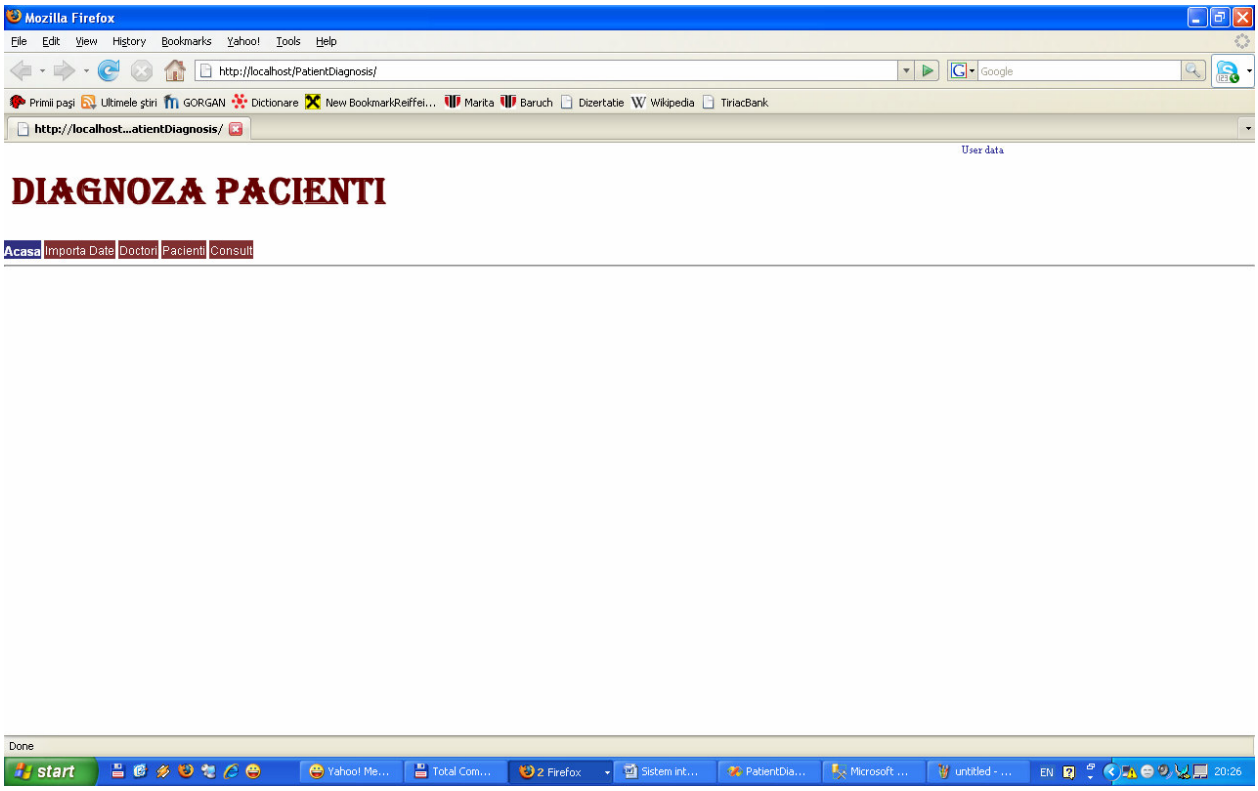



**Fig 21: Design-ul site-ului**

Se pot determina următoarele zone:

1. Header
   i. În partea stângă conține logo-ul site-ului. Dedesubtul acestuia se regăseşte meniul cu ajutorul căruia se realizează navigația între paginile site-ului.
   ii. În centrul există posibilitatea de a afişa mesaje pentru utilizator.
   iii. În dreapta se vor afişa date despre utilizatorul autentificat în aplicație, posibilitatea de autentificare, opțiune de ieşire.
2. Content
   i. Zona din pagină care se află sub Header şi în care se va afişa conținutul specific fiecărei pagini.

**Pagina de pornire**

Pe această pagină se vor afişa informații specifice utilizatorului: setări individuale, informații grupate despre consultații, rețete (în cazul unui pacient), listă de pacienți şi urgențe (în cazul unui doctor) sau informații administrative (în cazul unui administrator).

**Importă date**

Pe această pagină se va oferi o interfață pentru un utilizator de tip administrator pentru a importa diverse planuri, medicamente şi întrebări prin intermediul fişierelor tipizate xml.

**Doctori**

Pe această pagină avem o lista a tuturor medicilor care pot utiliza sistemul.

**Pacienți**

Pe această pagină avem o lista a tuturor pacienților introduşi în sistem.

**Consult**

Pe această pagină are loc consultul asistat al unui pacient. Tot aici are loc vizualizarea istoricului consulturilor efectuate de doctorul logat.

**Interfața cu celelalte module**

Acest modul se interfațează cu utilizatorul prin intermediul navigatoarelor (browserelor) prin care acesta (utilizatorul) se conectează la aplicația web. Totodată există o interfațare cu modulul Business prin intermediul căruia face operații asupra datelor existente în aplicație (cerere/scriere/transformare date) şi cu nivelul Backend care ține tot de serverul web şi cu ajutorul căruia realizează diferite operații cum ar fi: validare date, autentificare utilizator, verificare acces utilizator, transformare date într-un format date pentru afişare.

### 5.4.2 Nivelul de business

**Design**

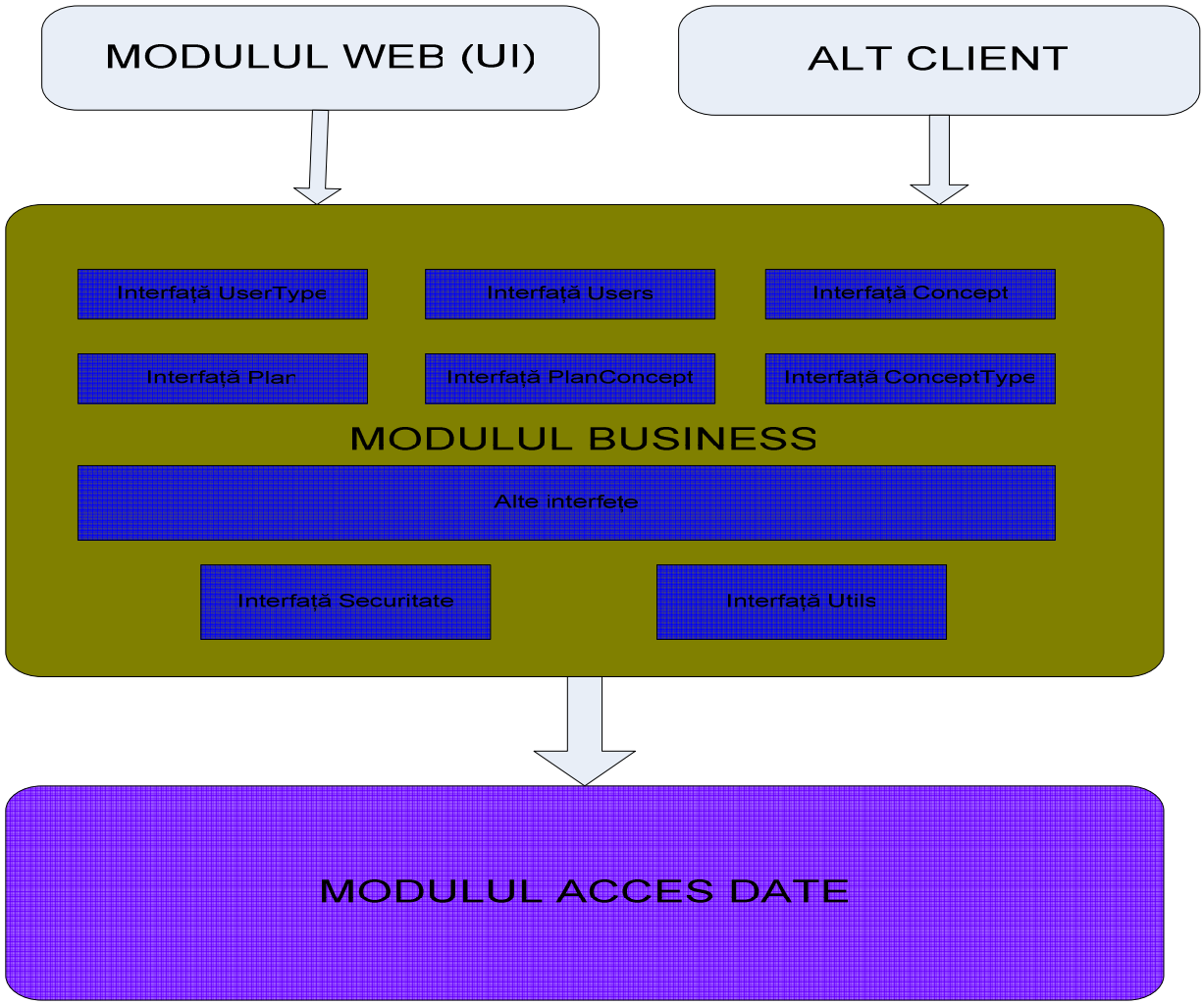



Fig 22: Nivelul business

**Descriere**

Nivelul Business acționează ca și partea de logică a aplicației. El poate oferi functionalitate (prin intermediul interfețelor) pentru citire și scriere date către mediul de stocare, autentificare utilizator, validator acces utilizator la date, validare date, procesare și transformare date, trimitere de email, obținere și formatare date în scopul generării de rapoarte etc.

**Interfața cu celelalte module (I/O)**

Nivelul Business definește o serie de interfețe prin intermediul cărora expune toată funcționalitatea de care modulele externe au nevoie pentru a interacționa cu clientul. Pentru moment acest nivel interacționează cu nivelele prezentare și acces date. Funcționalitatea lui poate fi accesată prin apeluri locale sau prin servicii remote.
Un exemplu de interfață pentru Consult este:

```
using System;
using System.Collections.Generic;
using System.Text;

using PatientDiagnosis.Entities;
using PatientDiagnosis.Entities.Parameters;


namespace PatientDiagnosis.BusinessLayer.Interfaces
{
      /// <summary>
      /// Business Interface for 'Consult'
      /// </summary>
      public interface IConsultBLL
      {
            /// <summary>
            /// Remove selected Consult element from the database
            /// </summary>
            /// <param name="user">User making the request</param>
            /// <param name="consultId">Id of Consult to be
deleted</param>
            /// <returns>true if the Consult was successfully
removed from
           database, false otherwise</returns>
           bool RemoveConsult(Users user, int consultId);
           /// <summary>
           /// Remove selected Consult elements from the database
           /// </summary>
           /// <param name="user">User making the request</param>
           /// <param name="idList">Id List of Consult elements
to be
           deleted</param>
           /// <returns>Number of removed elements</returns>
           int RemoveConsults(Users user, String idList);
           /// <summary>
```

```
        /// Fetch Consult elements from the database into a
list
        /// </summary>
        /// <param name="user">User making the request</param>
        /// <param name="filters">Filters for the selection
sent as
        stored procedure parameters</param>
        /// <returns>Consult Entity</returns>
             List<Consult> GetConsults(Users user,
    List<DALParameter>
        filters);
        /// <summary>
        /// Fetch selected Consult element from the database
into an
        internal entity
        /// </summary>
        /// <param name="user">User making the request</param>
        /// <param name="consultId">Id of selected Consult
        element</param>
        /// <returns>Consult Entity</returns>
        Consult GetConsult(Users user, int consultId);
        /// <summary>
        /// Saves Consult element to database. Returns new
entity id (or
        old id if it's an update)
        /// </summary>
        /// <param name="user">User making the request</param>
        /// <param name="consult">Instance of Consult
entity</param>
        /// <returns>Consult id</returns>
        int SaveConsult(Users user, Consult consult);
    }
}
```

### 5.4.3 Nivelul acces la date

**Design**

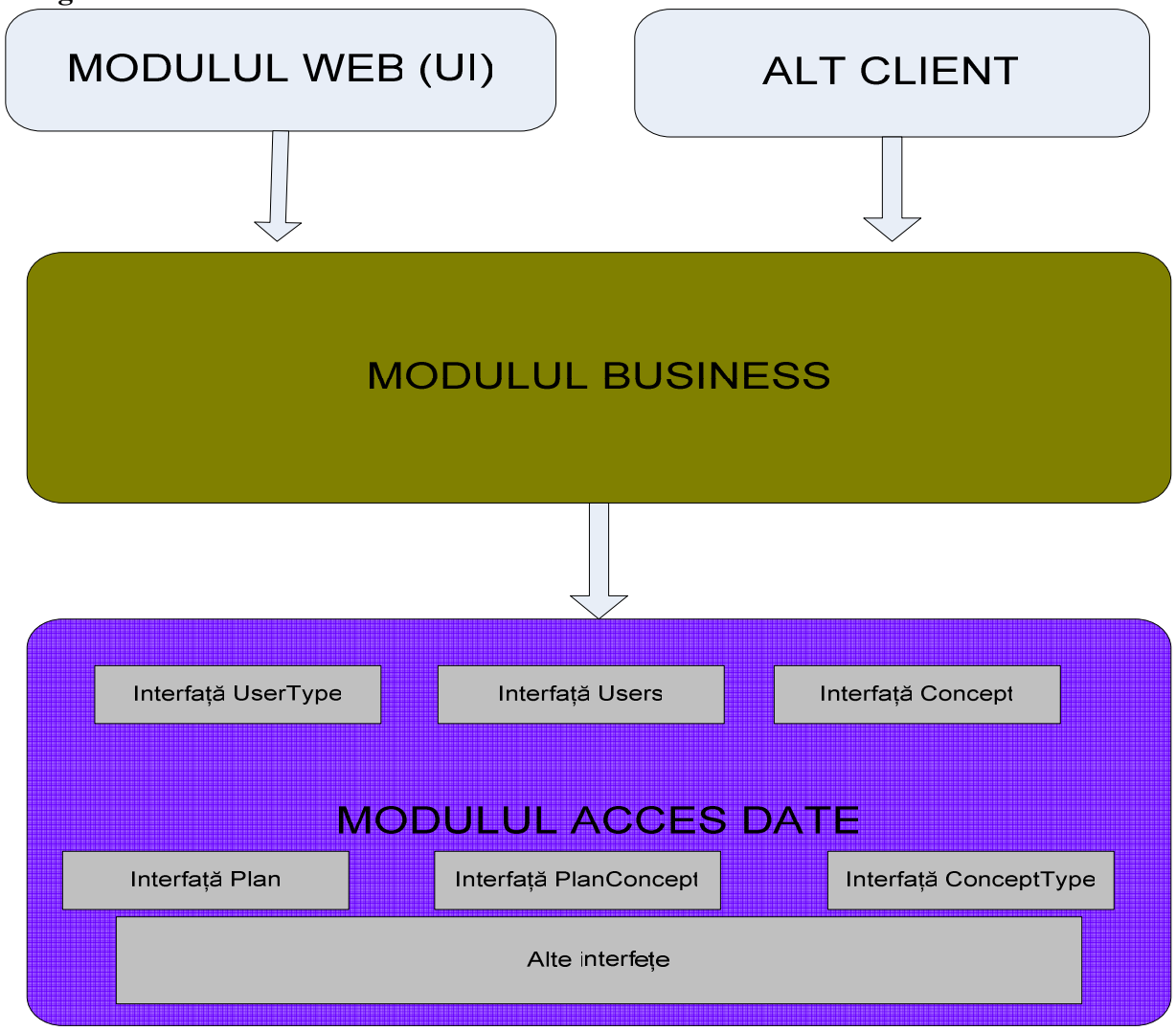



Fig 23: Acces date

**Descriere**

Oferă funcționalitate pentru accesul la date independent de mediul de stocare. Accesul la date se face prin posibilitatea de citire şi scriere date pentru fiecare dintre entitățile din aplicație care corespunde unei tabele în baza de date. Implementarea curentă va folosi Microsoft SQL Server 2005 Express pentru stocarea datelor. Pentru accesul la date for fi folosite proceduri stocate şi vederi.

**Interfața cu celelalte module (I/O)**

Funcționalitatea este oferită prin interfețe, în acest mod implementarea modului de acces la baza de date este ascuns celorlalte nivele oferindu-le independență față de modul de stocare al datelor. Un exemplu de interfață pentru Consult este:

```
using System;
using System.Collections.Generic;
using System.Text;

using PatientDiagnosis.Entities;
using PatientDiagnosis.Entities.Parameters;


namespace PatientDiagnosis.DataLayer.Interfaces
{
     /// <summary>
     /// Data Access Interface for 'Consult'
     /// </summary>
     public interface IConsultDA
     {
          /// <summary>
          /// Remove selected Consult elements from the database
          /// </summary>
          /// <param name="UserId">Id of user making the
request</param>
          /// <param name="idList">Id List of Consult elements
to be
          deleted</param>
          /// <returns>Number of removed elements</returns>
          int RemoveConsults(Int32 UserId, String idList);
          /// <summary>
          /// Fetch Consult elements from the database into a
list
          /// </summary>
          /// <param name="UserId">Id of user making the
request</param>
          /// <param name="filters">Filters for the selection
sent as
          stored procedure parameters</param>
          /// <returns>Consult Entity</returns>
          List<Consult> GetConsults(Int32 UserId,
List<DALParameter>
          filters);
          /// <summary>
          /// Saves Consult element to database. Returns new
entity id (or
          old id if it's an update)
          /// </summary>
          /// <param name="UserId">Id of user making the
request</param>
          /// <param name="consult">Instance of Consult
entity</param>
```

```
            /// <returns>Consult id</returns>
            int SaveConsult(Int32 UserId, Consult consult);
        }
}
```

### 5.4.4 Colaborarea între module

În imaginea următoare este prezentată modalitatea prin care interacționează modulele:

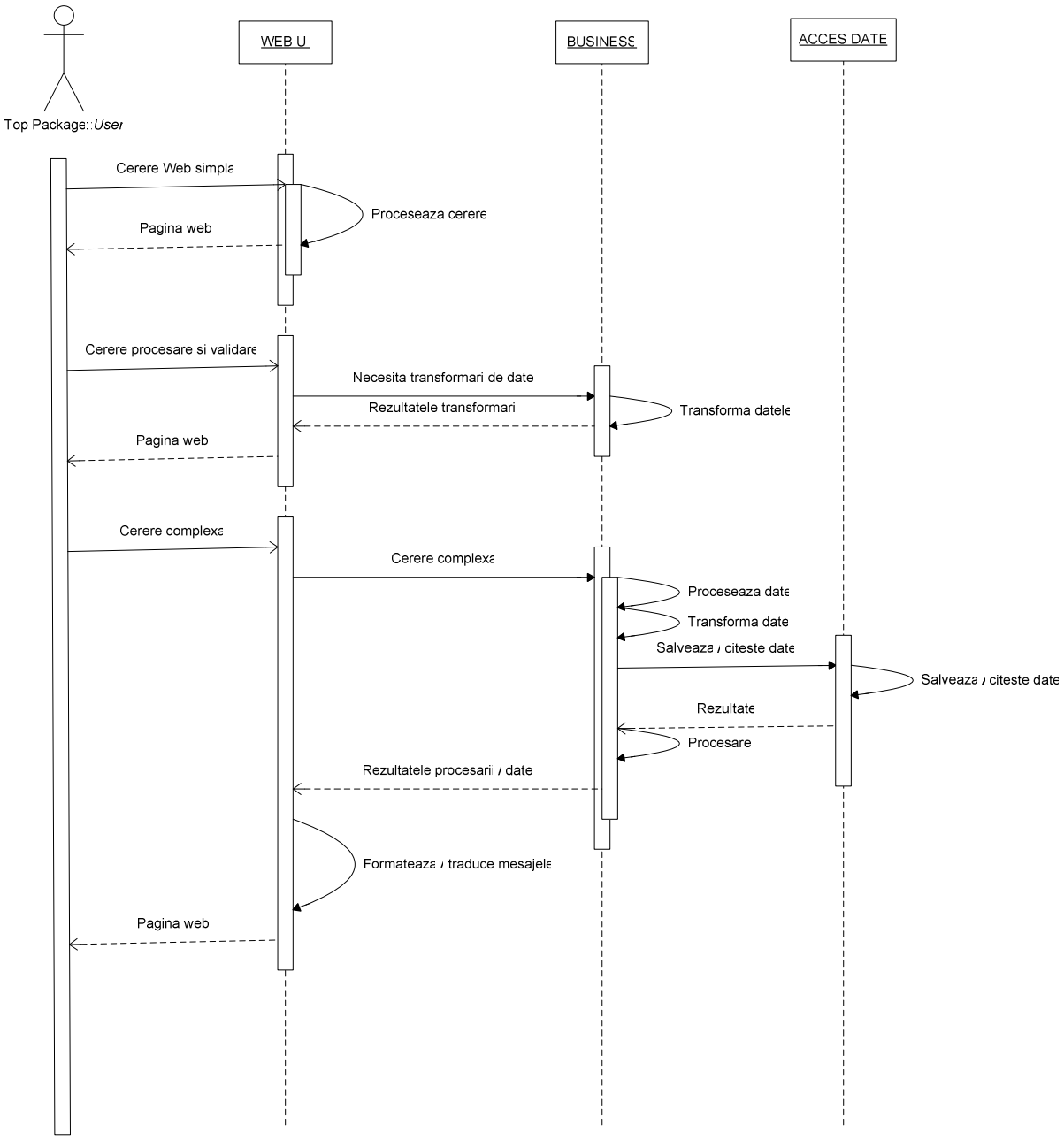



**Fig 24: Colaborarea între module**

Operațiile pe care le pot efectua un utilizator al aplicației sunt mai mult sau mai puțin complexe şi pot fi catalogate în trei categorii mari:

a) Operații care necesită procesare exclusiv pe serverul web. De obicei aceste operații se cer când utilizatorul are deja datele încarcate şi efectuază operații fără să salveze datele. Deoarece Ajax va fi utilizat pentru implementarea interfeței utilizator aceste operații se vor produce destul de des datorită comunicării dintre navigator (browser) şi serverul web.
b) Operații care necesită procesări sau validări de date atât pe serverul web cât şi în cadrul modulului de Business. Aceste operații sunt rare deoarece validările inițiale şi o parte din procesare este efectuată în cadrul nivelului Prezentare.
c) Cea mai comună operație care este cerută este când utilizatorul cere date pentru afişare sau trimite date pentru modificare. Acest tip de operație necesită colaborarea tuturor modulelor deoarece o parte din validare şi procesare se produce la nivelul Prezentare, după care datele sunt trimise nivelului Business, care după (posibile) alte procesări, rearanjări şi validări de date, trimite cererea mai departe nivelului Acces Date care va efectua operațiile de citire / scriere date.

### 5.4.5 Constrângeri de implementare

Aici se regăsesc toate constrângerile de implementare care au fost luate în considerare pentru implementarea aplicației:

- Microsoft .NET Framework Versiunea 3.5 va fi utilizată pentru implementarea aplicației
- ASP .NET va fi utilizat pentru implementarea interfeței web a sistemului
- Limbajul C# va fi utilizat pentru scrierea codului
- Navigatorul (browserul) pentru care se dezvoltă aplicația este Internet Explorer 7. Aplicația va funcționa şi pe Internet Explorer 6 precum şi în Firefox 2 şi 3, dar nu va oferi neapărat o interfață identică.

### 5.4.6 Dependențe

Componentele externe care vor fi utilizate în sistem:

| Nr. | Nume componentă | Descriere | De unde se poate obține | Tipul de licență |
|---|---|---|---|---|
| 1. | Log4NET | Componentă utilizată pentru salvarea mesajelor de tip log. | http://logging.apache.org/log4net/ | Apache Software License Version 2.0 |

## 5.5 Manual de instalare

### 5.5.1 WEBSITE

a) Se copiază folderul PatientDiagnosis de exemplu pe d:\PatientDiagnosis (în cazul în care se foloseste altă cale veți modifica path-ul corespunzător)

b) Se porneşte IIS: Start -> Control Panel -> Administrative Tools -> Internet Information Services

c) Se navighează pe serverul local astfel:

IIS -> LOCAL_SERVER -> Web Sites -> Default Web Site

d) Se dă click dreapta pe Default Web Site şi se alege properties. Se selecteaza tab-ul ASP.NET şi se verifică să fie selectată versiunea 2.0. Trebuie să aveţi instalat minim versiunea 2.0 (preferabil 3.5). In IIS versiunea maximă selectabilă este 2.0.

e) Se dă ok şi se închide fereastra.

Se dă din nou click dreapta pe Default Web Site şi se alege: New -> Virtual Directory

f) Se scrie ca şi alias PatientDiagnosis iar la path se alege calea către proiectul web. pentru exemplul dat aceasta este: d:\PatientDiagnosis\Website

Se dă ok şi se iese. Site-ul este navigabil la adresa: **http://localhost/PatientDiagnosis**

### 5.5.2 BAZA DE DATE

ATAŞARE BAZA DE DATE

a) Se porneşte aplicaţia SQL Server Management Studio (Express)

b) Se creează o conexiune la baza de date utilizată de către aplicaţie

c) Se navighează până la SERVER \ Databases se dă click dreapta pe Databases şi se alege Attach.

d) In fereastra nou deschisă se dă click pe Add şi se navighează până în locaţia unde aveţi baza de date. In exemplul dat, ea este: d:\PatientDiagnosis\Database\ şi se alege fişierul PatientDiagnosis.mdf. Se dă click pe OK.

e) Daţi OK şi baza de date va fi ataşată serverului de baze de date.

STABILIRE ACCES BAZA DE DATE

f) Se navigheazlă până la SERVER \ Security \ Logins

g) Se dă click dreapta pe Logins şi se alege optiunea New login

h) Login name: med. Totodată se selecteaza modalitatea de conectare: SQL Server Authentication şi se introduce parola: med. Se debifeaza Enforce password expiration.

i) În meniul din stânga se alege USER MAPPING şi se selectează baza de date PatientDiagnosis. Dupa selecţie în partea de jos vor fi afişate opţiunile de acces. Selectaţi db_owner şi daţi OK.

# 6 Sistem portabil de culegere si prelucrare de date medicale

## 6.1 Obiective

**Obiectivul general** a fost definirea, proiectarea si implementarea unui sistem mobil pentru de culegerea de date medicale de la pentru pacientii suferinzi de afectiuni cardiovasculare.

**Obiectivele specifice** ale sistemului au fost:

- definirea contextului in care va opera sistemul in cadrul proiectului CardioNET
- stabilirea tehnologiilor si arhitecturilor ce vor fi incluse in sistem
- definirea unei unitati medicale pacient generice compuse din trei module (modulul de achizitie de date, modulul de stocare in baza de date, modulul de interfatare)
- implementarea unitatii medicale pacient generice
- integrarea unitatii medicale pacient in arhitectura proiectului CardioNET prin stabilirea protocolului de comunicatie intre entitatile implicate in dialogul cu sistemul definit in acest capitol
- asigurarea securitatii sistemului de culegere a datelor (atat din punct de vedere al accesului la aplicatie, dar si din punct de vedere al accesului la informatiile generate de sistem)
- folosirea unor mijloace si metode de identificare simple si eficiente ce necesita interventie minima din partea pacientilor monitorizati

Se va prezenta in cele ce urmeaza, utilizarea tehnologiei RFID pentru identificarea pacientilor in sistemul CardioNET. Sectiunea 6.3 prezinta in detaliu implementarea sistemului de achizitie de date, pentru ca urmatoarele subcapitole sa prezinte aplicatia de monitorizare in timp real a pacientilor, respecitv seviciile web pentru expunerea informatiilor stocate.

## 6.2 Tehnologia RFID in cadrul sistemului CardioNET

RFID - Radio Frequency Identification- Identificarea prin radiofrecventa face parte din categoria tehnologiilor de identificare automata AutoID (Automatic ID). RFID reprezinta tehnologia de identificare a unei entitati folosind undele radio pentru localizarea si citirea identificatorului acestei entitati inscris intr-un cip (tag. La ora actuala este cea mai avansata metoda tehnologica de achizitie automata a datelor de identificare a unor entitati.

Sistemele de identificare bazate pe tehnologia RFID sunt compuse, în general, din trei componente:

−un cititor (reader);

−un transponder (tag sau eticheta de radiofrecventa);

−un sistem de prelucrare a datelor citite (PDA, PC, calculator specializat, ...).

Exista mai multe tipuri de tag-uri, clasificate în functie de diferite criterii, ( sursa de energie, banda de frecventa etc.)

*Aplicarea tehnologiei RFID în domeniul medical*

Pe plan international în domeniul medical, urmatoarele sectoare au realizari prin aplicarea tehnologiei RFID:
−gestiunea articolelor medicale – ofera posibilitatea de localizare rapida a articolelor medicale mobile;
−îngrijirea pacientilor – posibilitatea de identificare rapida a pacientilor si de localizare a acestora în orice moment;
−managementul medicamentelor si al substantelor periculoase – este foarte importanta trasabilitatea
medicamentelor pentru gestionarea in sigura a acestora;
−managementul inventarului – posibilitatea de identificare rapida a elementelor de inventar.
In cadrul sistemului a fost propusa identificarea pacientilor pe baza informatiilor stocate într-un tag RFID (microcircuit electronic cu memorie - CIP) de tip bratara - fig.6.1. In acest sens cipurile vor memora elementele de identificare ale persoanei, eventuale specificatii necesare în actul medical de urgenta etc.

Pentru implementare s-au folosit tag-uri pasive, de cost scazut. În cadrul sistemului, s-au folosit mai multe tipuri de cititoare RFID prezentate in tabelul 6.1.

| Dispozitiv RFID | Standard RFID Implementat | Frecventa | Interfata conectare | Imagine Dispozitiv |
|---|---|---|---|---|
| SdiD 1020 | ISO 15693 (8cm)<br>ISO 14443A (6cm) | 13.56 MHz | SDIO – card SD |  |
| SdiD 1212 | ISO 11784<br>ISO 11785 | 124/134.2 kHz<br>half duplex(12.5cm)<br>full duplex(9cm) | SDIO – card SD |  |
| idTronic USB Stick | ISO 18000-6 | 125 kHz<br>13.56Mhz | USB |  |

Tabel 6.1 Dispozitive RFID mobile folosite

Cititoarele ce vor fi utilizate pentru citirea/scrierea informatiilor reprezinta vor putea comunica wireless cu echipamente de tip PC sau PDA, constituind astfel o retea cititoare fixe sau mobile. Pentru asigurarea unui grad ridicat de mobilitate în cadrul sistemului se vor utiliza cititoare mobile, la nivelul de culegere de date.

La dezvoltarea sistemului se vor implementa componente pentru asigurarea securitatii datelor, precum si pentru respectarea standardelor medicale (HL7, EHR etc.).


Figura 6.1. Identificarea pacientilor folosind taguri de

In cele ce urmeaza vom prezenta un exemplu de implementare a procedurii de citire de tag-uri de radiofrecventa folosind un dispozitivil mobil de tip PDA echipat cu cititor RFID SdiD 1020. Procedura de citire este implementata in mediul de programare C#:.
Prima parte a secventei de cod descrie structura si formatul datelor ce sunt stocate pe tag-urile pasive:

```
//TAG CARD ATTRIBUTES
#define NUM_BLOCKS      28
#define UID_SIZE        8
#define BLOCK_SIZE      4
```

Urmatoarea secventa de cod citeste intr-un ciclu infinit informatiile stocate pe taguri. Se poate alege atat formatul hexadecimal cat si formatul ASCII pentru formatarea informatiilor. Functia de citire este prezentata mai jos:

```
        /****** READ THE DATA ******/
        time = GetTickCount();
        for(; errorcode == ERR_NONE; blocksRead++)
        {
            time = GetTickCount();
            errorcode = isoCard.ReadBlock(blocksRead,data);
            time = GetTickCount() - time;

            //Read OK display data
            if (errorcode == ERR_NONE)
            {
                switch(mode)
                {
                case STATE_HEX:
                   swprintf(copyBuffer,_T("Block %03i: %02X %02X
%02X %02X
%05i\r\n\0"),blocksRead,(data[0]&0xFF),(data[1]&0xFF),(data[2]&0
xFF),(data[3]&0xFF),time);
```

```
                    printBuffer = wcscat(printBuffer,(const
wchar_t*)copyBuffer);
                    SetDlgItemText(IDC_15693_READ_RX,
printBuffer);
                    break;

                case STATE_CHAR:
                    swprintf(copyBuffer,_T("Block
%03i:"),blocksRead);
                    printBuffer = wcscat(printBuffer,copyBuffer);

                    for(int i = 0; i < BLOCK_SIZE; i++)
                    {
                       if (data[i] == NULL)
                         swprintf(copyBuffer,_T("  \0"));
                       else
                         swprintf(copyBuffer,_T("
%c\0"),data[i]);
                       printBuffer =
wcscat(printBuffer,copyBuffer);
                    }

                    swprintf(copyBuffer,_T("\r\n\0"));
                    printBuffer = wcscat(printBuffer,copyBuffer);

                    SetDlgItemText(IDC_15693_READ_RX,
printBuffer);
                    break;
                }
            }

            //Read failed
            else
                blocksRead--;

            //Check if done
            if (blocksRead >= (NUM_BLOCKS-1))
                return;
        }
    }
```

## 6.3 Retele de Senzori Wireless

In cadrul sistemului de achizitie de date se remarca trei tipuri diferite de senzori:

- senzori pentru monitorizarea semnelor vitale ale pacientului – in aceasta categorie includem senzorii conectati la electrozii EKG, clemele Spo2, etc – si circuitele

hardware necesare pentru a asigura achizitia si transmisia semnalelor medicale in format electronic

- senzorii ambientali – au rolul de a monitoriza parametri fizici din

Figura 6.2 prezinta topologia generala a retelei de senzori wireless din sistemul de monitorizare si culegere de date.

Componentele retelei sunt prezentate

Figure 6.2. Topologia Retelei de Senzori Wireless

- Senzorii – reprezinta dispozitivele efective de culegere a datelor si sunt „hardwired" pe placile de extensie ale senzorilor wireless. Prin intermediul canalelor ADC (analog to digital), a conectorilor digitali integrati la nivelul senzorilor se poate asigura interconectivitatea senzorilor cu dispozitivele medicale existente pe piata (Ex: electrozi EKG, cleme SPo2, etc)
- Wireless mote – modul wireless pentru achizitia de date si transmisia spre baza de date personala p-DB. Acest mote (nod) comunica intr-o retea de senzori wireless folosind standardul IEEE 802.15.4, cunoscut si ca ZigBee.
- Gateway – Este un nod wireless care are in plus functia de rutare a mesajelor pe portul serial RS232
- Base Station-pDB – reprezinta statia (computer sau PDA) care va gazdui baza de date cu informatii medicale

In Figura 6.2 se remarca deasemenea prezenta utilizatorilor P-DB. Acestia vor interactiona cu baza de date prin nivelul intermediar al serviciilor web puse la dispozitie de statia pacientului.

Mobilitatea pacientului este data de folosirea dispozitivelor PDA pentru stocarea datelor receptionate din reteaua de senzori wireless. Fiecare nod din reteaua de senzori wireless va fi montat pe pacientii monitorizati impreuna cu bateriile necesare asigurarii unei autonomii crescute a dispozitivului. Mentionam ca functionalitatile unui nod pot fi compuse astfel ca, de exemplu: pacientul va purta un singur nod wireless indiferent de numarul de parametri fizici monitorizati.

### 6.3.1 Platforma Hardware

Figure 6.3. Tinynode – Wireless Mote

Pentru implementarea sistemului de achizite s-au folosit microcontrollerele TinyNode. Exista numeroase motive pentru aceasta alegere:

- consum redus de energie(microcontroller MSP430, transciever radio XE1205)
- prezenta unui sistem de operare la nivelul de programare a microcontrolerului
- exista mai multe porturi pentru interfatarea nodului cu diverse periferice de intrare/iesire (6 intrari analogice, 2 iesiri digitale)
- raza mare de acoperire pentru reteaua wireless (pana la 2km)
- dimensiuni fizice reduse(30x40 mm)
- rate mari de transfer in reteaua wireless (153kbps)
- prezenta unei memorii flash 512kb onboard
- fiabilitate
- performanta
- cost redus

Pentru faza incipienta a procesului de achizitie a datelor semnalele vitale monitorizate au fost simulate in laborator, intr-un mediu controlat. In acest sens, prezentam un exemplu de structura programata care evidentiaza un canal EKG preluat de la un dispozitiv medical:

```
int[] ECGArray = new int[]
{ -5, -14, -10, -8, -9, -7, 0, 121, 108, -24, -34, -18, -19, -16, -19, -17,
-11, -11, -10, -1, 13, 33, 45, 47, 46, 37, 12, -12, -24, -23, -25, -21,-10,
-17, -19, -14, -12, -13, -12, -13, -8, -9, -5, -8, 10, 21, 19, 4, -10,
-11, -11, -12, -8, -12, 0, 121, 108, -24, -23, -18, -21, -16, -15, -14,
-10, -12, -7, 5, 26, 43, 46, 46, 41, 21, -1, -18, -24, -19, -22, -21,
-18, -18, -16, -13, -10, -12, -11, -8, -10, -8, -7, 0, 18, 24, 9, -7,
-13, -11, -9, -8, -10, 0, 121, 108, -24, -33, -17, -20, -19, -17, -14,
-13, -11, -8, 2, 19, 35, 47, 50, 43, 29, 7, -15, -29, -22, -25, -20,
-16, -19, -17, -16, -12, -11, -11, -10, -11, -9, -6, 0, 13, 23, 15, -2,
-11, -10, -10, -8, -7, -15, 0, 121, 108, -24, -20, -19, -20, -16, -12,
-14, -15, -11, -2, 12, 30, 41, 48, 44, 34, 16, -5, -21, -22, -21, -20,
-20, -13, -15, -15, -17, -14, -13, -9, -10, -10, -11, -8, 5, 23, 22, 4,
-9, -10, -12, -10, -10, -11, 0, 121, 108, -24, -28, -18, -19, -16, -16,
-16, -15, -13, -8, 4, 21, 39, 44, 46, 38, 45, 24, -1, -23, -28, -27, -25,
-20, -21, -18, -18, -14, -14, -13, -8, -9, -11, -9, 0, 16, 21, 11, -9,
-11, -9, -8, -9, -9, 0, 121, 108, -24, -32, -17, -21, -19, -16, -18, -13,
-10, -12 };
```

Pentru a simplifica trasarea grafica, valorile din sirul ECGArray au fost traslatate, si reprezinta valoarea functiei pentru un anumit punct. Astfel functia de trasare grafica aplecata cu parametrul in intervalul [1..lungimea sirului] va returna valorile mentionate mai sus in ordine.

### 6.3.2 Formatul pachetelor wireless

Pachetele de date formatate ca active messages(conform Tabelului 1) sunt transmise in cadrul retelei wireless folosind Collection Tree Routing Protocol descris pe larg in [1].

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P | C | rezervat | | | | | | THL | | | | | | | |
| ETX | | | | | | | | | | | | | | | |
| Origine | | | | | | | | | | | | | | | |
| Number Secventa | | | | | | | | Collect Id | | | | | | | |
| Date… | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | |

Tabel 6.1 – Structura unui pachet CTP

Unde campurile sunt definite dupa cum urmeaza:

- P: Bitul de rutare. Bitul P permite nodurilor sa ceara informatii de rutare de la alte noduri. Daca un nod care detine o cale de rutare valida primeste un pachet cu bitul P setat, acesta va raspunde in cel mai scurt timp posibil cu o secventa de rutare
- C: Notificare pentru congestie. Daca un nod pierde un pachet de date CTP, it MUST acesta va trebui sa seteze bitul C in imediat urmatoarul pachet de date transmis.
- THL: Numarul de noduri pentru propagare(echivalent TTL in cadrul retelelor descrise IEEE 802.3). De fiecare data cand un nod genereaza un pachet CTP, acesta va avea bitii THL setati la 0. La fiecare receptionare a unui pachet de date CTP, nodul va incrementa valoarea campului THL. Daca un nod receptioneaza un pachet cu THL mai mare de 255, acesta va fi incrementat la 0 – asigurand astfel retransmisia pachetelor – exista totusi metode de a evita retransmisia infinita a pachetelor.
- ETX: este o metrica pentru transmitatorul single-hop. Cand un not transmite un pachet CTP trebuie sa puna valoarea ETX a rutei sale ca single-hop. Daca un nod receptioneaza un pachet cu valoare mai mica, nodul trebuie sa retransmita pachetul.
- Originea: Adresa de origine a pachetului. Nodurile care transmit mai departe un pachet primit nu au voie sa modifice valoarea acestui camp.
- Numarul de secventa: este un numar dat de nodul de la care a fost generat mesajul. Indiferent de nodul care retransmite mesajul, acest camp nu poate fi modificat.
- Collect_id: Identificator de nivel inalt. Nodul care genereaza mesajul seteaza acest camp. Nu poate fi modificat de nodurile care retransmit.
- date: informatia efectiva de transmis, zero sau mai multi biti. Nodurile care retransmit pachetele nu au voie sa modifice acest pachet.

### 6.3.3 Exemple de implementare pentru TinyOS

Urmatoarea secventa de cod prezinta modalitatea in care un mesaj poate fi transmis folosind CTP :

```
event void Timer.fired() {
```

```
        if (reading == NREADINGS) {
          if (!sendbusy) {
      oscilloscope_t *o = (oscilloscope_t *)call Send.getPayload(&sendbuf,
sizeof(oscilloscope_t));
      if (o == NULL) {
        fatal_problem();
        return;
      }
      memcpy(o, &local, sizeof(local));
      if (call Send.send(&sendbuf, sizeof(local)) == SUCCESS)
        sendbusy = TRUE;
            else
              report_problem();
          }

          reading = 0;
          /* Part 2 of cheap "time sync": increment our count if we didn't
             jump ahead. */
          if (!suppress_count_change)
            local.count++;
          suppress_count_change = FALSE;
        }

        if (call Read.read() != SUCCESS)
          fatal_problem();
      }
```

Data fiind o perioada de esantionare, pentru fiecare esantion, metoda prezentata se declanseaza si genereaza procedura de citire a senzorilor. Valorile efective citite sunt memorate intr-un buffer si ulterior – cand continutul bufferului este umplut – acestea sunt transmise pe reteaua wireless conform protocolului de rutare prezentat.

Este de asemenea importanta urmatoarea secventa de cod, implementata la nivelul nodului Gateway, care faciliteaza conexiunea retelei de senzori cu Base Station (calculator sau PDA) care au porturi RS232 :

```
    event void SerialSend.sendDone(message_t *msg, error_t error) {
        uartbusy = FALSE;
        if (call UARTQueue.empty() == FALSE) {
          // We just finished a UART send, and the uart queue is
          // non-empty.  Let's start a new one.
          message_t *queuemsg = call UARTQueue.dequeue();
          if (queuemsg == NULL) {
            fatal_problem();
            return;
          }
          memcpy(&uartbuf, queuemsg, sizeof(message_t));
          if (call UARTMessagePool.put(queuemsg) != SUCCESS) {
            fatal_problem();
            return;
          }
          post uartSendTask();
        }
      }
```

## 6.4 Aplicatia de monitorizare

Dupa autentificarea cu succes a pacientului in sistemul CardioNET, acestuia i se ofera posibilitatea de a urmari evolutia propriilor semne vitale monitorizate conform figurii 6.4.

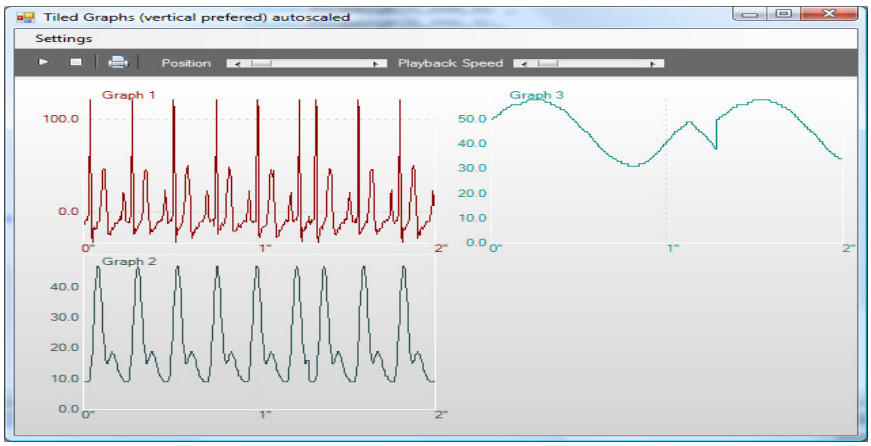



Figura 6.4. Aplicatia de monitorizare

In figura 6.4 sunt prezentate 3 grafice pentru 3 categorii diferite de semnale medicale: semnal EKG, semnal pentru rata respiratorie, respectiv semnalul generat de puls oximetru.

Aplicatia foloseste informatiile stocate in baza de date de catre o aplicatie terta care are rolul de a citi datele transmise pe portul serial si de a le stoca.

Totodata, aplicatia permite introducerea datelor manual (ex: temperature) pentru a oferi pacientului posibilitatea de a introduce in sistem valori masurate de catre acesta.

Aceasta aplicatie de monitorizare este presenta si la medicul specialist. In functie de rolul utilizatorului (pacient/medic) se vor putea introduce temperaturi monitorizare noi, respectiv limite pentru alertele de temperaturi. Folosind managerul de evenimente (implementat ca tabele in baza de date si triggere pe aceste tabele) medicul specialist poate fi notificat in permanenta de evolutia (critica) a pacientului aflat sub monitorizare continua. Se remarca deasemenea prezenta unui sistem interactiv de comunicare de tip video-chat intre pacient si medic specialist, prin care pacientul poate interactiona atat in scris cat si verbal folosind microfoane si camere video. Toate mesajele transmise sunt stocate in baza de date. In cazul in care pacientul/medicul transmite mesaje interlocutorului care nu este logat in sistem(indisponibil) aceste mesaje vor fi afisate interlocutorului atunci cand acesta se va loga in sistem.

Aplicatia a fost dezvoltata folosind mediul de programare C# din platforma Microsoft .NET Framework.

## 6.5 Baza de date

Figura 6.5 prezinta o diagrama(partiala) a bazei de date implementata.la nivelul sistemului de culegere si prelucrare informatii medicale, urmand ca baza de date completa sa fie documentata in etapa urmatoare.

Diagrama prezentata descrie relatii de tip 1 la N intre un pacient si alergii, bolile cronice de care sufera. Este evidentiata posibilitatea ca un pacient sa aiba mai multe acte de identitate dar si relatia 1-N intre pacient si consultatii.

## 6.6 Servicii Web

Pentru a asigura un grad maxim de interoperabilitate cu unitatile din sistem, BaseStation(Figura 6.2) va expune datele folosind serviciile web prezentate in aceasta sectiune.

Serviciile Web au fost dezvoltate folosind mediul de programare C# din platforma Microsoft .NET Framework, structura proiectelor este prezentata in figura 6.6.

Se remarca structurarea pe trei nivele a proiectului:

- Nivelul datelor – identificat in clasa “Dataservice.cs” class – asigura functionalitati de baza si operatii CRUD asupra bazelod de date
- Nivelul de logica business – reprezentat de existenta claselor de tip DataObject in cadrul directorului DO din solutie. Pentru o mai buna gestionare si pastrarea consistentei in toate aplicatiile sistemului, acest nivel va fi compilat ca o librarie dinamica, astfel ca aceasta librarie va putea fi refolosita in toate aplicatiile care vor lucra cu tipurile de obiecte definite in baza de date
- Nivelul de interoperabilitate – este reprezentat defapt de implementarea efectiva a serviciilor web

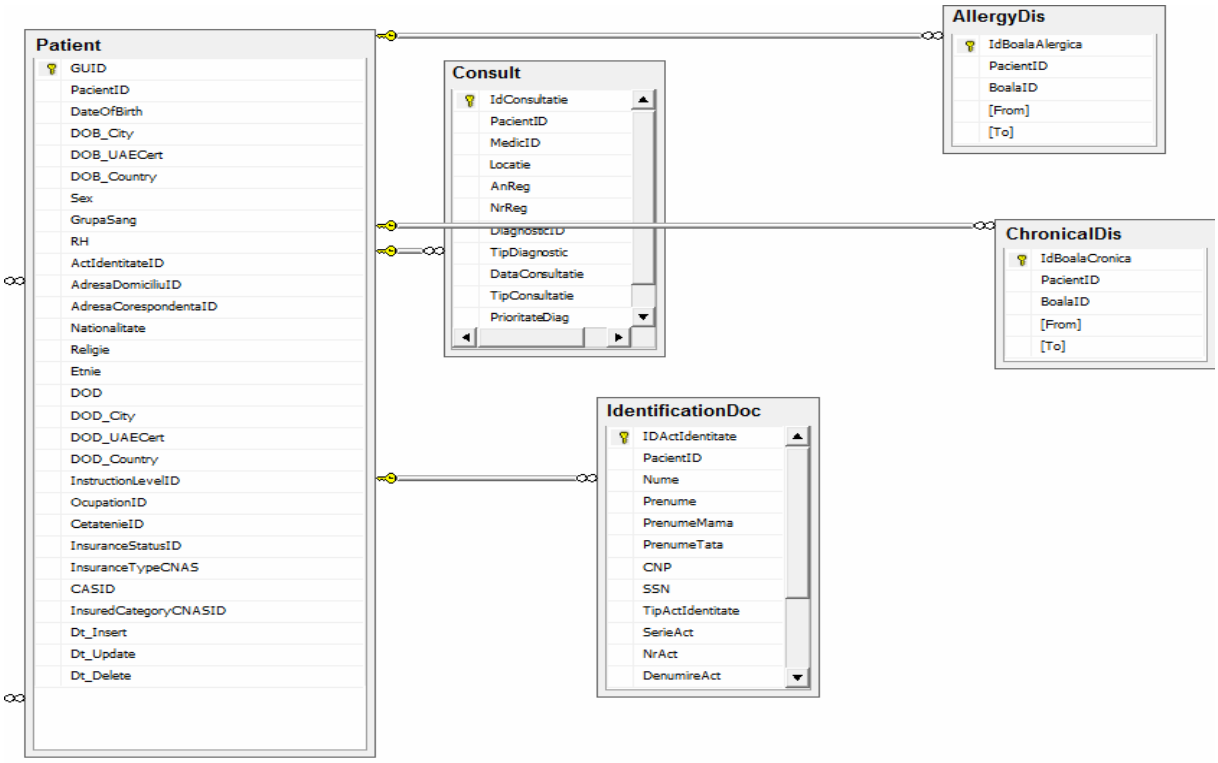



Figura 6.5. Baza de date p-DB

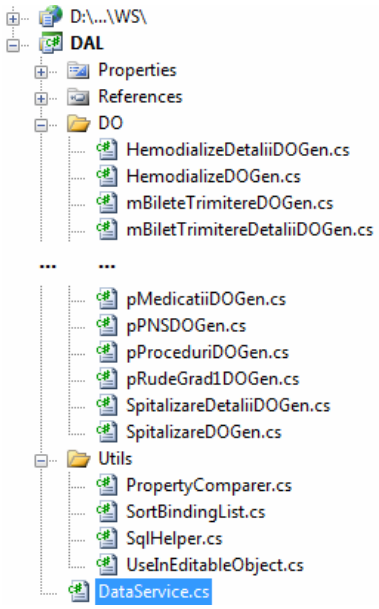



Figura 6.6 Structura proiectului Implementat pentru ServiciileWeb

In figura 6.7 sunt prezentate metodele web ale serviciului WEB implementat pentru accesul la informatiile de interes general ale pacientului precum asigurari de sanatate, retete, bilete de trimitere, etc. Metodele serviciului web respecta formatarea SOAP[2].

In figura 6.8 este prezentat un exemplu de format de mesaj pentru interactiunea cu metoda PacientByID a serviciului web mai sus mentionat. Aceasta metoda returneaza inregistrarea unica din sistemul CardioNET ce identifica un pacient.

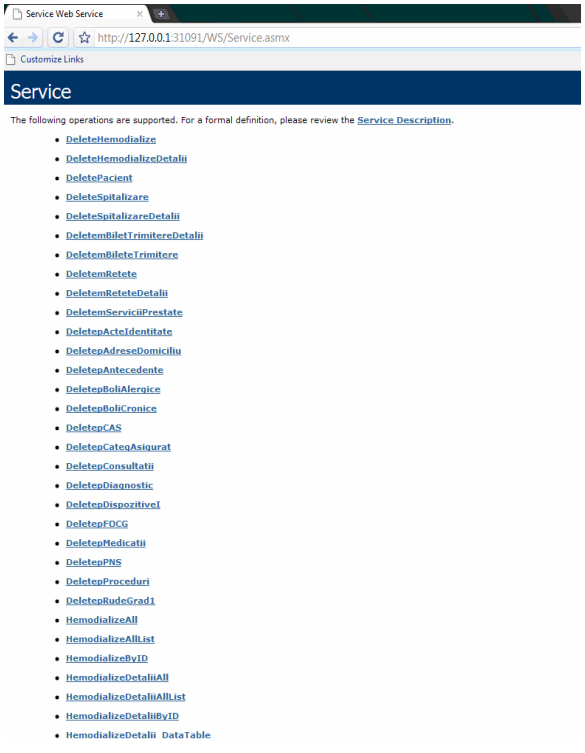



Figura 6.7 Metodele in cadrul serviciului WEB

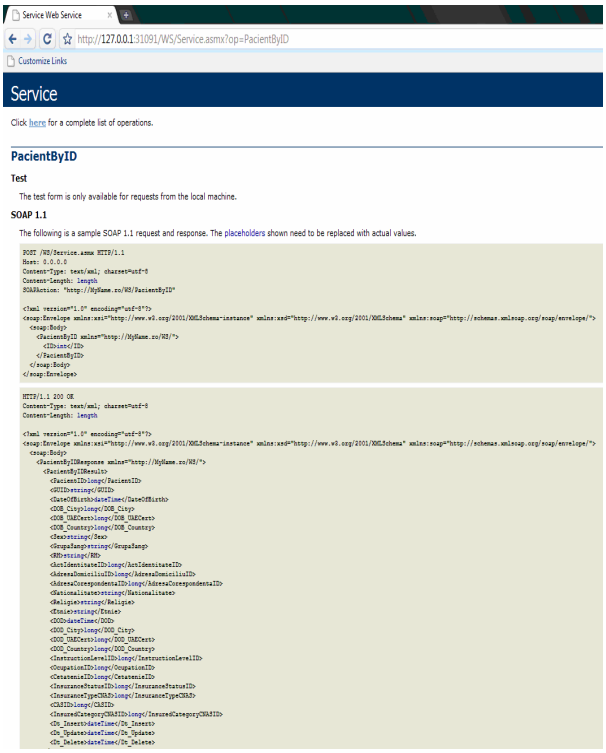



Figure 6.8 Formatul SOAP al mesajelor

In continuare este prezentat un exemplu de implementare a serviciului web descris mai sus:

```
#region Pacient
    #region GET
      [WebMethod]
      public PacientDO[] PacientAll(String sWhere, String sOrderBy)
      {
            return DataService.PacientAll(sWhere,sOrderBy);
      }
      [WebMethod]
      public List<PacientDO> PacientAllList(String sWhere, String sOrderBy)
      {
            return DataService.PacientAllList(sWhere,sOrderBy);
      }
      [WebMethod]
      public PacientDO PacientByID(int ID)
      {
            return DataService.PacientByID(ID);
      }
      [WebMethod]
      public PacientDO[] PacientBypCASID(int pCASID)
      {
            return DataService.PacientBypCASID(pCASID);
      }
      [WebMethod]
      public PacientDO[] PacientBypCategAsiguratID(int pCategAsiguratID)
      {
            return DataService.PacientBypCategAsiguratID(pCategAsiguratID);
```

```
        }
        [WebMethod]
        public DataTable Pacient_DataTable(String sWhere, String sOrderBy)
        {
              return DataService.Pacient_DataTable(sWhere,sOrderBy);
        }
        #endregion

        #region UPDATE
        [WebMethod]
        public int InsertPacient(PacientDO PacientDO)
        {
              return DataService.InsertPacient(PacientDO);
        }

        [WebMethod]
        public int UpdatePacient(PacientDO PacientDO)
        {
              return DataService.UpdatePacient(PacientDO);
        }

        [WebMethod]
        public int DeletePacient(PacientDO PacientDO)
        {
              return DataService.DeletePacient(PacientDO);
        }
        #endregion
#endregion
```

Se observa faptul ca toata logica este preluata si asigurata de catre obiectele de business. In exemplul de mai sus a fost prezentata metoda web `StergerePacient`. In mod evident un pacient nu poate fi sters dintr-un sistem medical, insa un dispozitiv medical care nu mai este in folosinta ar putea fi sters. Asemenea constrangeri si specificatii fac subiectul modificarilor ulterioare motiv pentru care ele vor fi gestionate in cadrul nivelului de logica business.

Disemniarea ultimei variante de librarii DLL in cadrul aplicatiilor CardioNET este o sarcina simpla intrucat este vorba de un singur fisier ce trebuie copiat in toate aplicatiile sistemului.

In continuare de prezinta descrierea WSDL a serviciului mai sus mentionat :

```
<wsdl:definitions targetNamespace="http://MyName.ro/WS/">
<wsdl:types>
<s:schema elementFormDefault="qualified" targetNamespace="http://x.ro/WS/">
<s:element name="pDispozitiveIByID">
     <s:complexType>
     ……
     <s:complexType name="pDispozitiveIDO">
     <s:complexContent mixed="false">
     <s:extension base="tns:NotifyValidatableBase">
     <s:sequence>
     <s:element minOccurs="1" maxOccurs="1" name="IdDispozitiv"
type="s:long"/>
     <s:element minOccurs="1" maxOccurs="1" name="PacientID" nillable="true"
type="s:long"/>
     <s:element minOccurs="0" maxOccurs="1" name="DispozitivID"
type="s:string"/>
```

```
     <s:element minOccurs="1" maxOccurs="1" name="DispozitivProducator"
nillable="true" type="s:long"/>
     <s:element minOccurs="1" maxOccurs="1" name="DataAcordare"
nillable="true" type="s:dateTime"/>
     <s:element minOccurs="1" maxOccurs="1" name="DataImplant" nillable="true"
type="s:dateTime"/>
     <s:element minOccurs="1" maxOccurs="1" name="Cantitate" nillable="true"
type="s:decimal"/>
     <s:element minOccurs="1" maxOccurs="1" name="MedicID" nillable="true"
type="s:long"/>
     <s:element minOccurs="1" maxOccurs="1" name="Observatii" nillable="true"
type="s:long"/>
     </s:sequence>
     </s:extension>
     <s:complexType name="NotifyValidatableBase" abstract="true">
     …
     <s:complexType name="DataErrorInfoValidatableBase" abstract="true">
     …
     <s:extension base="tns:ValidatableBase"/>
     …
     <s:complexType name="ValidatableBase" abstract="true"/>
     …
     <s:element name="InsertpDispozitiveI">
     …
     <s:element name="InsertpDispozitiveIResponse">
     …
     <s:element minOccurs="0" maxOccurs="1" name="pDispozitiveIDO"
     …
     <s:element name="UpdatepDispozitiveIResponse">
     …
     <s:element name="DeletepDispozitiveIResponse">
     …
     <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="DeletepDispozitiveIResult"
      type="s:int"/>
     </s:sequence>
     </s:complexType>
     </s:element>
     …
     …
     </wsdl:binding>
     –
     <wsdl:service name="Service">
     –
     <wsdl:port name="ServiceSoap" binding="tns:ServiceSoap">
     <soap:address location="http://0.0.0.0:31091/WS/Service.asmx"/>
     </wsdl:port>
     –
     <wsdl:port name="ServiceSoap12" binding="tns:ServiceSoap12">
     <soap12:address location="http://0.0.0.0:31091/WS/Service.asmx"/>
     </wsdl:port>
     </wsdl:service>
     </wsdl:definitions>
```

Toate serviciile CardioNET expun descrieri WSDL formatate XML in conformitate cu standardul W3C[3]. Aceasta particularitate este fundamentala pentru viitoare dezvoltari a unor

operatii precum: descoperirea, compozitia si invocarea in vederea orchestrarii si coreografiei serviciilor web.

## 6.7 Integrarea sistemului de culegere date in proiectul CardioNET

Sistemul implementeaza servicii specializate:

- servicii de interfata – care sa furnizeze utilizatorului o interfata grafica pentru a putea initia cautari sau pentru vizualizarea rezultatelor unor interogari;
- serviciu gestiune resurse (broker) –care va cunoaste toate resursele sistemului; prin intermediul lui utilizatorul va putea realiza o cautare generala pentru apelarea unei resurse;
- servicii de administrare – vor implementa administrarea resusselor sistemului;
- servicii de acces la date – vor controla accesul la o baza de date cu înregistrarile medicale despre pacient;
- servicii de securitate – comunicatiile dintre acest agent, doctor si pacient vor fi criptate.

**Referinte bibliografice:**
**Capitolul VI:**

[1] Collection Tree Protocol, TinyOS TEP 123, http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html
[2] Simple Object Access Protocol, http://www.w3.org/TR/soap/
[3] www.w3.org/TR/wsdl

# 7 Portalul Cardionet –interoperabilitate, servicii web si structuri de date

## 7.1 Portalul CARDIONET – componenta a sistemului distribuit Cardionet

Analizand tendintele actuale de globalizare in domeniile procesarii datelor, in care IT&C are un rol preponderent si anticipand generalizarea lor si in domeniul sanatatii am proiectat, in cadrul proiectului Cardionet, un model de SIM - Sistem Informatic Medical distribuit, cu structuri de date, componente software si aplicatii informatice care sa permita abordarea problemelor de sanatate intr-o maniera globala, cu un grad ridicat de interoperabilitate , dincolo de limitele unei organizatii administrative (locale) de "healthcare". In acest scop am creat un modelul de sistem informatic medical SIM, distribuit, cu structuri de date, distribuite la partenerii de proiect, care sa permita un grad ridicat de interoperabilitate bazat pe un model global de inregistrare electronica de tip "Universal Patient Record" si care sa permita de asemenea interactiunea automata intre diferite SIM-uri, independet de platforma pe care acestea au fost implementate. Schimburile de date intre partenerii sistemului distribuit se va realiza in cadrul proiectului pilot, prin intermediul portalului Cardionet.

### 7.1.1 Cerinte specifice sistemelor medicale interoperabile cu inregistrari electronice de date

Un SIM proiectat pentru interoperabilitate, cu inregistrari medicale despre pacienti de tip UPR -Universal Patient Record sau "IMU- Inregistrari Medicale Universale" trebuie sa indeplineasca urmatoarele conditii:

-sa permita comunicarea intre diferite institutii din domeniul sanatatii- UPR trebuie sa fie interoperabile;

-sa permita totodata cerintele speciale de prelucrare ale informatiilor, caracteristice institutiilor implicate- UPR trebuie sa poata interopera cu diverse formate de inregistrari si de documente medicale;

-UPR trebuie sa foloseasca standarde globale si "reglementari de exploatare" intre operatori (institutii) din industria "healthcare";

-UPR trebuie sa permita stocarea informatiilor medicale "oriunde" in infrastructura partenerilor din domeniu si sa poata fi regasita de "oriunde" si "oricand";

- in abordarea UPR, SIM trebuie sa implementeze masuri si tehnologii adecvate de securitate si confidentialitate a informatiilor;

-sistemul trebuie sa permita si "interoperabilitate lingvistica", pemitand operarea in orice limba folosita de partenerii din domeniu implicate;

-este posibil ca in perioada de stabilizare si generalizare a unui astfel de sistem sa fie necesara o coordonare din partea unei institutii dupa modelul american al HMO - ***Health Maintenance Organization*** [1-HMO] care sa impuna regulile de dezvoltare si apoi de exploatare a unui astfel de sistem.

In cadrul proiectului se creaza modelul conceptual si infrastructura unde are loc derularea etapei de implementare a acestui sistem cu grad ridicat de interoperabilitate. In acest system un rol important in privinta interoperabilitatii il are componenta denumita Portalul Informatic Medical Cardionet. Acest portal a fost dezvoltat pe durata etapei III si se va continua ata dezvoltarea cat si implementarea si in etapele urmatoare, IV si V, ale proiectului Cardionet.

Bazat pe cerintele minimale prezentate mai sus, strict necesare pentru un nivel ridicat de interoperabilitate, se pot anticipa modelele de date si de aplicatii care vor permite o asfel de abordare “universala”. Modelele de inregistrari “pe hartie” ale datelor medicale sunt diferite in institutii diferite, de aceea organizatiile implicate trebuie sa convina de comun accord asupra standardelor folosite in dezvoltarea unor inregistrari electronice de date, despre pacienti, cat mai generale si apoi sa stabileasca si conventiile organizationale necesare in exploatarea unor astfel de sisteme.

### 7.1.2 Cerinte organizationale necesare prelucrarilor de date in sisteme interoperabile

O prima cerinta strict necesara in vederea asigurarii gradului solicitat de interoperabilitate este asigurarea automatizarii inregistrarii datelor despre pacienti la nivelul propriei organizatii. Idea crearii UPR presupune ca toate organizatiile de sanatate vor opera in aceasta maniera, moment in care operatorii din domeniul sanatatii vor avea la dispozitie o imagine completa a starii de sanatate a pacientului, a evolutiei in timp a acesteia si date despre toate serviciilor de specialitate suportate de pacient si detalii despre furnizorii care le-au asigurat.
Informatiile clinice vor trebui inregistrate de catre operator (“encounter system”) acolo unde “se produc”, in sistemul propriu (source document repository) si in mod automat in depozitul centralizator CPR (paragraf 7.3.2.). Pentru maximizarea beneficiilor partajarii informatiilor si a schimburilor automate de informatii se recomanda operatorilor din industria “healthcare” folosirea standardului HL7 (the standard healthcare applications level network protocol – Health Level 7) [2-HL7] asa cum am mentionat si in etapele anterioare ale proiectului Cardionet.
Ca un rezultat direct al constituirii acestor depozite CPR’s de catre operatori se va putea obtine un “patient Clinical Summary” care va evidentia evenimentele medicale semnificative ale pacientului, furnizorii de servicii medicale care le-au efectuat, etc.

Localizarea fiecarui document sursa va putea fi realizata prin intermediul CPR repository, daca documentele au fost generate in format electronic si chiar daca au fost generate pe suport “hartie”, se va obtine “calea” spre aceste documente. De pe aceste documente primare se vor putea obtine detalii, daca vor fi solicitate de entitati fizice sau juridice eligibile si autorizate.

## 7.2 Portalul Cardionet si tehnologia serviciilor-web

In continuare sunt prezentate pe scurt detalii privind tehnologia seriviciilor-Web, si implementarea acestora in cadrul componentelor portalului Cardionet.

Un serviciu web este o componenta software bazata pe o colectie de protocoale si standarde care asigura o modalitate de schimb de date intre aplicatii sau sisteme de aplicatii distribuite proiectate conform tehnologiei SOA (Service Oriented Architecture)[3-SOA].
Aceste aplicatii pot fi scrise in limbaje de programare diferite, pot rula pe diverse platforme si pot folosi serviciile web pentru a face schimb de date intr-o retea locala sau prin Internet. Interoperabilitatea este asigurata atat prin design-ul serviciilor web cat si prin aplicarea de standarde publice specializate pentru comunicare intre aplicatii.
De obicei serviciile web nu furnizeaza o interfata grafica, in schimb ele impart logica de procesare, date si proceduri de busines, dupa functionalitati si le ofera spre exterior prin intermediul unor metode publice, vizibile in retea.
Programatorii pot folosi aceste metode si pot adauga oricate servicii web unei interfete utilizator (asemenea unei pagini web) pentru a oferi noi functionalitati de busines.

Serviciile web permit aplicatiilor de pe platforme diferite sa comunice unele cu altele, fara interventia operatorului uman si pentru ca toate comunicatiile sunt în XML este asigurata si "autodescrierea" datelor care se schimba intre aceste aplicatii.
"Furnizorul" unui serviciu web defineste un format pentru cererile catre serviciul sau si pentru raspunsurile care vor fi generate de catre acesta. Aceste formate sunt vizibile catre "consumatorul" serviciului si respectandu-le acesta va putea interactiona automat cu "furnizorul".

Tehnologia Serviciilor-Web este o modalitate standardizata de integrare a aplicatiilor bazate pe web, si cuprinde: XML (Extensible Markup Language)[4], SOAP (Simple Object Access Protocol)[5], WSDL (Web Services Description Language)[6] si UDDI (Universal Description, Discovery and Integration)[7].

### 7.2.1 Cardionet – tehnologii web pentru protocoale de mesaje pentru eHealth

SOAP -Simple Object Access Protocol este un protocol de trimitere a mesajelor bazat pe XML, care specifica toate regulile necesare pentru localizarea serviciile Web, integrarea lor in aplicatii si modalitatile de comunicatie dintre ele.

Altfel spus protocolul SOAP defineste modalitatea de comunicare dintre apelant si furnizorul serviciului. WSDL-ul asigura „descrierea" serviciului oferit. Aceasta descriere se face folosind XML si ofera, practic, documentarea necesara aplicatiilor pentru a comunica intre ele în mod automat. WSDL descrie ce poate face serviciul respectiv, unde este localizat si cum poate fi invocat. De fapt, descrierea unui serviciu web se face printr-un document XML in a carui structura pot fi definite si incluse mai multe tipuri de elemente ce pot fi grupate astfel:

**- definitii abstracte** – contin informatii despre tipurile de date folosite de serviciu (integer, string, float,etc.), mesajele pe care serviciul le poate accepta si "port"-urile, definitii care formeaza metodele si procedurile serviciului;

**- definitii concrete** - specifica prin legaturi tipul de accesare permis (de exemplu, SOAP) si serviciul, care nu este altceva decat o „publicare" a porturilor definite anterior.

Pentru a avea valoare practica, un serviciu web trebuie sa fie cunoscut eventualilor sai utilizatori. UDDI este un standard al carui rol este de a oferi un "catalog" cu serviciile disponibile, astfel incat orice aplicatie sa poata gasi serviciul adecvat necesitatilor sale.
Acest "catalog" ofera informatii despre localizarea geografica, categoria serviciului, informatii de contact, precum si informatii tehnice despre serviciile web prezentate.
Pe scurt, XML este folosit pentru a eticheta datele, SOAP la transferul de date, WSDL pentru descrierea disponibilitatilor serviciului si UDDI este folosit pentru a lista serviciile disponibile.

### 7.2.2 Cardionet- tehnologia UDDI – specializare registre in domeniul proiectului

UDDI -Universal Description, Discovery and Integration, este registru public, gratuit, unde orice organizatie isi poate publica propriile servicii Web. UDDI este de fapt un framework independent de platforma, creat special pentru descrierea serviciilor, descoperirea si integrarea lor in aplicatii de business distribuite:

-UDDI inseamna Universal Description, Discovery and Integration;

-UDDI este un catalog pentru stocarea de informatii desptre servicii web;

-UDDI este de asemenea un catalog pentru descrierile WSDL ale interfetelor serviciilor web;

-UDDI schimba informatii via SOAP;

-UDDI este construit pe platforma Microsoft .NET.

UDDI poate fi privit ca fiind un catalog universal al resurselor prezente in Internet impreuna cu serviciile Web pe care aceste resurse le ofera. Furnizorii de servicii trebuie sa fie inscrisi in *Registrul Universal de Afaceri UDDI* (*Universal Business Registry*), furnizand informatii de „contractare" si „apel" al serviciilor publicate. Consumatorii serviciilor vor avea posibilitatea sa localizeze companiile care furnizeaza produse si/sau servicii in diverse domenii si sa le apeleze. Versiunea prezenta a **UDDI** se constituie ca un „meta serviciu" pentru descoperirea si integrarea serviciilor web. **UDDI** permite actorilor prezenti pe Internet:

-sa descopere serviciile (resursele) existente (publicate) ;

-sa defineasca modul cum vor interactiona pentru folosirea acestor resurse.

Protocolul pentru descriere, descoperire si integrare universala este unul dintre suporturile majore ale folosirii serviciilor web. UDDI creeaza o platforma interoperabila care permite aplicatiilor sa gaseasca si sa utilizeze in mod dinamic serviciile web oferite prin Internet. UDDI permite de asemenea operarea si mentinerea de diverse registre pentru diverse scopuri domenii de activitate. Efortul pentru implementarea registrelor UDDI este realizat de furnizorii majori de software de aplicatii precum si de operatorii din cadrul consortiului de OASIS[8].

UDDI foloseste standardele W3C[9] si IETF[10] ca de exemplu limbajul XML, protocolul HTTP[11] si serviciul DNS [12]. UDDI reprezinta tehnologia care va permite afacerilor sa interactioneze una cu cealalta in mod automat, folosind fiecare aplicatiile si procedurile proprii de procesare a datelor schimbate cu partenerii de business.

UDDI este rezultatul unei initiative a unui consortiu de firme din domeniul tehnologic, cu un numar ce depaseste 200 de membri dintre care amintim: Dell, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Oracle, SAP si SUN . UDDI foloseste standarde informatice agreate international, dintre care cele mai importante sunt XML pentru reprezentarea datelor si SOAP (Simple Object Access Protocol) pentru transportul lor. UDDI foloseste de asemenea WSDL (Web Services Description Language) pentru descrierea interfetelor serviciilor web

La UDDI se poate inscrie orice companie, indiferent de marimea ei si indiferent de locatia geografica unde se afla. Punctul de contact cu aceasta organizatie este *www.uddi.org*.

De asemenea, folosind aceleasi tehnologii dezvoltatorii isi pot construi propriile structuri pentru interoperabilitate, la nivelul dorit: local, regional, etc.

Toate domeniile de afaceri vor putea beneficia de pe urma protocolului UDDI deoarece specificatiile acestuia rezolva problemele de interactiune de tip B2B (*Business to Business*) si de folosire a serviciilor web. Standardul UDDI raspunde unei probleme fundamentale generate de dezvoltarea fara precedent a afacerilor pe Internet si anume dezvoltarea unor metode globale de interactiune intre aplicatii.

In prezent numarul firmelor care ofera produse sau servicii pe Internet, numite generic „servicii Web" (Web services) a ajuns de ordinul milioanelor.

Standardul UDDI (Integrare, Descoperire si Descriere Universala) este cea mai promitatoare initiativa in acest domeniu al afacerilor de tip B2B. UDDI se bucura de un larg suport din partea industriei si de un numar mare de aderenti care sa-i asigure succesul.

Un pas important l-au constituit cataloagele online si pietele electronice. In general acestea sunt limitate la o anumita industrie sau la o anumita regiune. UDDI rezolva problema oferind un singur registru central pentru orice afacere, din orice domeniu si localizata in orice punct de pe glob. In plus registrul universal ofera facilitati de cautare pe baza de parametri, care includ locatia geografica, categoria de afaceri, detaliile serviciului oferit, specificatiile serviciului, etc.

### 7.2.3 Cardionet- Registries si nivele de vizibilitate

Registrul in sensul UDDI, este de fapt un index cu grad de vizibilitate local sau global (universal), prin care se pot oferi in esenta trei categorii de informatii, ce ar putea fi prezentate si astfel:

**- *Pagini Albe (White Pages)*** care contin informatii de identificare: nume, adresa, persoane de contact, coduri fiscale, precum si alti identificatori acceptati la nivel international.

**- *Pagini Galbene (Yellow Pages***) care contin informatii despre domeniul de activitate, cum ar fi clasificarea industriala sau localizarea geografica. Aceste informatii pot fi prezentate in taxonomii de uz international, sub forma unor perechi denumire/valoare. Exista astfel de taxonomii publice care pot fi folosite in aplicatii comerciale, cum ar fi:

-*NAICS US Government standard industry codes* (standardul SUA de taxonomie industriala)

-*UN/SPSC (ECMA) product and services codes* (standardul ONU de taxonomie industriala)

-*Geographic taxonomies* (taxonomie geografica).

***-Pagini Verzi (Green Pages)*** care contin descrierea informatica a modalitatilor de comunicare electronica cu societatea respectiva. Informatiile includ: platforma de operare, programele folosite, metode, etc. De asemenea paginile verzi vor descrie aplicatiile cu care sistemul informatic respectiv se poate conecta pentru a comunica automat.

In cadrul proiectului se vor contrui registri specializati pentru domeniul . Dezvoltarea si testarea lor se va continua si in etapele urmatoare ale proiectului.

Ca si organizare**, arhitectura Cardionet propusa** pentru interactiunile dintre partenerii de business ai aplicatiilor de medicina se incadreaza foarte bine in modelul general al arhitecturii orientate spre servicii, de tip SOA. In acest model de arhitectura interactioneaza trei entitati:

-Furnizorul de servicii

-Beneficiarul serviciului

-Repertoriul (registrul) de servicii

Furnizorul de servicii poate fi unul dintre partenerii de pe lantul de servicii medicale. Repertoriul de servicii este sistemul/sistemele care inmagazineaza informatii despre serviciile oferite de furnizorii de servicii si care poate/pot fi interogat(e) de beneficiarii serviciilor in procesul de descoperire a serviciilor.

Beneficiarul de servicii este consumatorul final, unul dintre partenerii de business.

“Mesajul de business” este modelul de formalizare al unei entitati de date ce va fi schimbata intre parteneri, dupa logica domeniului (in privinta construirii si schimbului de mesaje se vor respecta recomandarile HL7).

Schimbul de informatii la acest nivel se va face prin intermediul unor astfel de mesaje, incluzind la nevoie documente tipizate. Din multitudinea mesajelor procesului de business, consideram ca prim exemplu, urmatoarele:

1. Identificarea furnizorilor de servicii medicale – prin schimbul unor informatii cum ar fi: identificatorul partenerului (codFurnizor), numele partenerului, tipul, adresa, persoana de contact, cod fiscal, cont banca, orele si zilele de operare/functionare a business-ului, adresa de e-mail, URL;
2. Notificarea actiunilor dorite– prin expedierea spre parteneri a unor informatii cum ar fi: documentul de business
3. Publicarea - datelor referitoare la serviciile oferite;
4. Abonarea - unui partener la un serviciu sau grup de servicii;
5. Confirmarea abonarii si furnizarea informatiilor solicitate

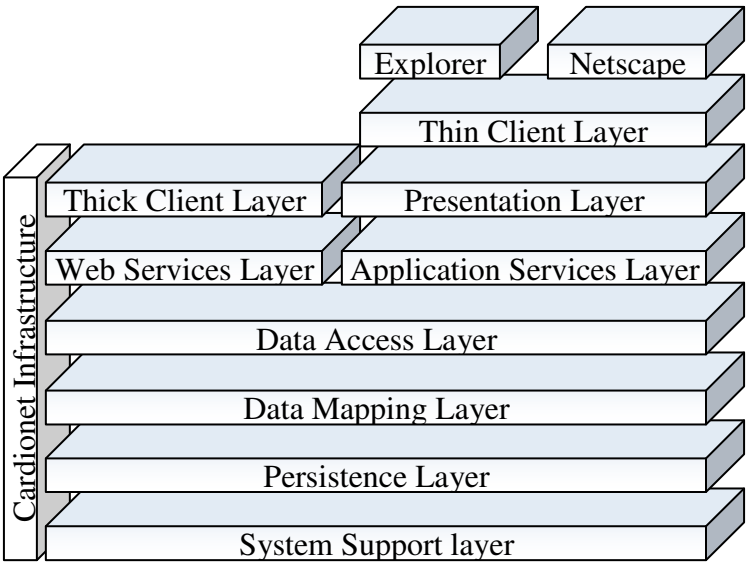
6. Interogarea - din partea unui partener cu privire la caracteristicile/starea/istoricul unui serviciu sau grup de servicii;
7. Raportarea - spre un partener specializat a datelor solicitate prin intermediul unor interogari, eventual cu expedierea documentelor aferente acestei interogari;
8. Erori si exceptii – anunta partenerii de existenta unor situatii/conditii de eroare sau care impiedica indeplinirea sarcinii/operatiunii solicitate.

Mesajele (si unele dintre documentele) de business vor fi reprezentate folosind limbajul XML. Ele vor trebui sa respecte schemele XML corespunzatoare, iar verificarea validitatii mesajelor (documentelor) se va face atat la emitere, cat si la receptie.

### 7.3 Portal Cardionet - arhitectura generala orientata pe servicii

Asa cum a fost prezentata si in etapa a-II-a, arhitectura generala a proiectului CardioNET, prezentata mai jos in figura VII.1., este multi-layer, pe mai multe nivele arhitecturale.
Daca nivelul "System Support Layer" se refera la sistemul de operare propriu-zis si setul de functii oferite de catre acesta, "Persistence Layer" se refera la motorul de baze de date, cataloage si metadate.

Nivelele superioare (Data Access, respectiv Data Mapping) ofera functionalitati de mapare a informatiei din baza de date pe instante de obiecte abstracte definite in modelul si structura claselor sistemului informatic CardioNET. Acest modelofera suport pentru schimbul informational intre diverse entitati de business, folosind diverse modalitati de conectare asigurate de catre nivelul architectural al serviciilor tuturor clientilor: thick clients, thin clients in intranet sau prin intermediul internetului si este modelul respectat in mare de toate componentele sistemului distribuit Cardionet.




CardioNET - Service Oriented Arhitecture

Figura VII.1 – Arhitectura generala Cardionet

Implementarea arhitecturii orientate pe servicii prezentata mai sus in fig. VII.1, va fi facuta prin folosirea tehnologiei SOA si a *serviciilor Web,* instrumentata prin diverse standarde, limbaje si protocoale. Dintre cele mai importante reamintim:

-HTTP (HyperText Transfer Protocol)[11] – reprezinta protocolul de transport al mesajelor in reteaua de comunicatii;

-SOAP (Simple Object Access Protocol)[5] – este un protocol standardizat care specifica modul de impachetare a mesajelor schimbate/partajate de aplicatii/parteneri si permite apelarea serviciilor Web;

-UDDI (Universal Description Discovery and Integration) [7]– specificatie de protocol pentru organizarea, functionarea si administrarea repertoriilor de servicii Web, permitand descoperirea de catre clienti a serviciilor Web publicate;

-WSDL (Web Service Description Language) [6]– defineste o modalitate standard pentru descrirea detaliilor – metode si parametri - serviciilor Web;

-XML (eXtensible Markup Language)[4] – folosit pentru descrierea si codificarea datelor ca obiecte numite documente XML si, partial, pentru descrierea programelor care le prelucreaza.

Accesul atat la serviciile specifice de business cat si la date se va putea face prin:

*-Portalul Cardionet, de prezentare a informatilor* si de *expunere* a serviciilor pentru organizatiile din proiect (inmagazinand informatii de interes general pentru consumatorii de servicii)

-accesarea directa a *serviciilor Web* dupa ce acestea au fost localizate in registrele locale sau repertoriul UDDI, in functie de regulile de securitate impuse de partenerul business care ofera serviciul

*-aplicatii client* de sine statatoare, consumatoare a serviciilor Web specializate publicate.

In toate cazurile descrierea, descoperirea si integrarea serviciilor Web specializate se va face prin intermediul *unor repertorii specializate de servicii(registre) (interne sau externe).* Legatura dintre "furnizorii" de servicii si "consumatorii" acestora este reprezentata mai jos in figura VII.2:

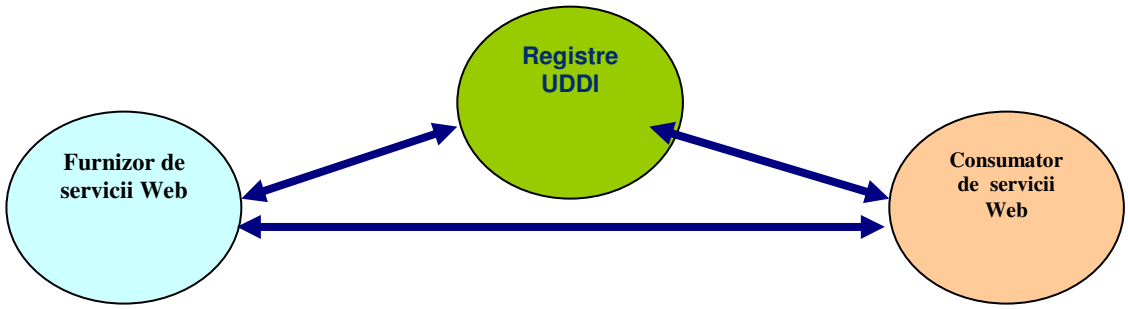
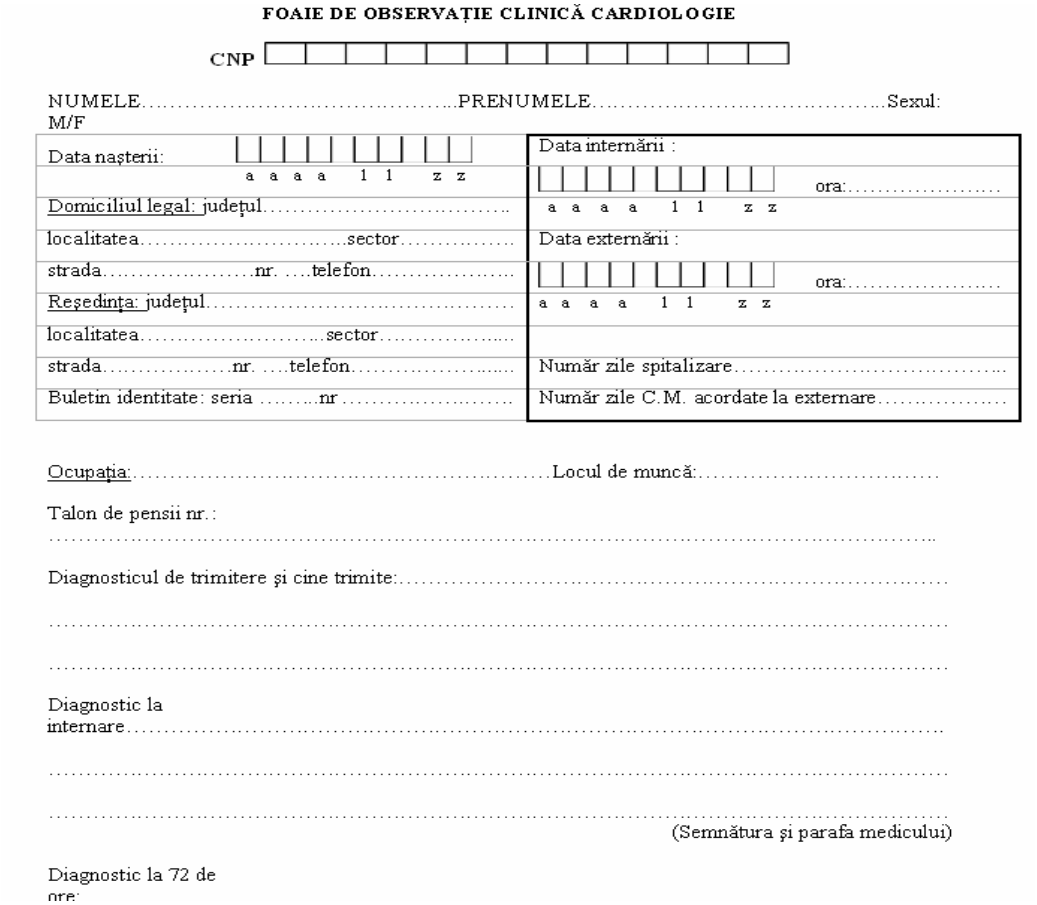



Figura VII.2 – Registre, Furnizori si Consumatori de servicii web

### 7.3.1 Structuri de date ale bazelor portalului Cardionet

Pentru structurile de date ale portalului Cardionet am tinut cont atat de foaia de observatii generala FOCG prezentata in Cap.2, cat si de foaia de observatie specializata pentru cardiologie (model 23.5-A4 t2, cu 7 pagini), a carei prime pagina poate fi vazuta mai jos, in figura VII.3.:



**FOAIE DE OBSERVAȚIE CLINICĂ CARDIOLOGIE**

CNP

NUMELE……..PRENUMELE……..Sexul: M/F

| | |
|---|---|
| Data nașterii: a a a a 1 1 z z | Data internării : |
| | ora:…… |
| Domiciliul legal: județul…….. | a a a a 1 1 z z |
| localitatea……..sector…….. | Data externării : |
| strada……nr. ….telefon…….. | ora:…… |
| Reședința: județul…… | a a a a 1 1 z z |
| localitatea……..sector…….. | |
| strada……nr. ….telefon…….. | Număr zile spitalizare……... |
| Buletin identitate: seria ……..nr…….. | Număr zile C.M. acordate la externare…… |

Ocupația:……Locul de muncă:……

Talon de pensii nr.:
……..

Diagnosticul de trimitere și cine trimite:……

……

……

Diagnostic la internare……..

……

……

(Semnătura și parafa medicului)

Diagnostic la 72 de ore:

Figura VII.3- Model de Foaie Clinica pentru Cardiologie

Din motive de spatiu, din multitudinea de tabele care compun structurie portalului, exemplificam aici doar tabela de “pacienti” cu informatiile generale, fara tabelele asociate pentru informatiile complete  aferente pacientului, atat actuale cat si istorice, asa cum le contine modelul.

| Nume Coloana | Tip | Size | Constraints | Sursa de date si Descriere |
|---|---|---|---|---|
| id_Pacient | | | PK, not null, identify | Cheie primara (interna) |
| guid_Pacient | | | NOT NULL, Identity | global unique identifier, cheie secundara (externa) |
| DataNastere | DateTime | | not null | Data de nastere a pacientului |
| LocalitateNastere_id | | | FK, NOT NULL, in catalogul de localitati | |
| UAECertNastere_id | | | FK, NOT NULL, in tabela Unitati Administrative Emitente(UAE) | de creat catalogul Unitati Administrative Emitente |
| TaraNastere_id | | | FK, NOT NULL, in catalogul Tari | |
| Sex | | | | |
| GrupaSang | | | | |
| RH | nvarchar | 10 | | |
| ActIdentitate_id | | | FK, NOT NULL, in tabela ActeDeIdentitate | |
| AdresaDomiciliu_id | | | FK, NOT NULL, in tabela AdreseDomiciliu | |
| AdreseCoresp_id | | | FK, NOT NULL, in tabela AdreseCorespondenta | |
| Nationalitate | | | | |
| Religie | nvarchar | 50 | | |
| Etnie | nvarchar | 50 | | |
| DataDeces | DateTime | | null | not null = decedat |
| LocalitateDeces_id | | | | |
| UAECertDeces_id | | | FK, NOT NULL, in tabela Unitati Administrative Emitente | |
| TaraDeces_id | | | FK, NOT NULL, | |
| Dt_INS | DateTime | | NOT NULL | data INSERT |
| Dt_UPD | DateTime | | NOT NULL | data UPDATE |
| IdUtilizator | | | | userID_ul logat la update |
| Dt_DEL | DateTime | | null | not null = deleted (stergere logica) |
| IdNivelInstruire | | | fk | campul IdNivelInstruire din nomenclatorul Lista_NivelInstruire |
| IdOcupatie | | | fk | campul IdOcupatie din nomenclatorul Lista_Ocupatii |
| IdTipCetatenie | | | fk | campul IdTipCetatenie din nomenclatorul Lista_TipCetatenie |
| IdCetatenie | | | fk | campul IdCetatenie din nomenclatorul Lista_Cetatenii; se completeaza doar cand IdTipCetatenie este 2 sau 3 |
| IdStatutAsigurat | | | fk | campul IdStatutAsigurat din nomenclatorul Lista_StatutAsigurare |
| IdTipAsigurareCNAS | | | fk | campul IdTipAsigurareCNAS din nomenclatorul Lista_TipAsigurareCNAS; se completeaza cand IdStatutAsigurat este 1 sau 3 |
| IdCAS | | | fk | campul IdCAS din nomenclatorul Lista_CAS; se completeaza cand IdTipAsigurareCNAS este 1 sau 2 |
| IdCategorieAsiguratCNAS | | | fk | campul IdCategorieAsiguratCNAS din nomenclatorul Lista_CategorieAsiguratCNAS; se completeaza cand IdTipAsigurareCNAS este 1 sau 2 |

Modelul de date proiectat permite inregistrarea tuturor datelor din foaia de observatii pentru cardiologie.
Odata cu inregistrarea datelor in bazele de date operative, la unitatile care le produc se va completa si depozitul (CPR) de date al portalului Cardionet, cu datele relevante despre pacient si episodul curent, sau legaturi spre aceste date in functie de optiunile de exploatare. Se genereaza astfel depozite de date care vor cumula in timp informatii despre pacienti, depozite care vor putea fi apoi consultate atat pentru informarea operativa a specialistilor cat si pentru cercetare sau trialuri specializate. Modelul complet si exemplificarea de inregistrari in aceste structuri de date vor fi prezentate in detaliu in documentatia etapei urmatoare a proiectului, cea din decembrie 2009.

### 7.3.2 Cardionet Portal Repositories

Ca un rezultat direct al constituirii acestor depozite CPR de catre operatori se va putea obtine un "patient Clinical Summary" care va evidentia evenimentele medicale semnificative ale pacientului, furnizorii de servicii medicale care le-au efectuat, etc.

Localizarea fiecarui document sursa va putea fi realizata prin intermediul CPR repository, daca a fost generate in format electronic si chiar daca a fost generat pe suport "hartie" se va obtine "calea" spre acest document. De pe aceste documente primare se vor putea obtine detalii, daca vor fi respectate politicile de acces stabilite prin rolurile si regulile de acces si vor fisolicitate de entitati fizice sau juridice eligibile si autorizate.

## 7.4 Portal Cardionet - sevicii Web specializate pentru domeniul eHealth

*Serviciile Web* propuse sunt in fond aplicatii (obiecte si metode) care sunt accesibile (si pot fi invocate) de catre orice program client folosind protocoale standard de Internet intr-o maniera independenta de platforma. *Serviciile Web* sint construite pe baza protocolului SOAP care permite o functionalitate de mesagerie peste protocolul de transport HTTP (pe portul 80 pentru majoritatea serverelor de Web) si foloseste mijloace standard pentru descrierea datelor.

*Serviciile Web* pot fi implementate in mai multe feluri. Una dintre cele mai utilizate tehnici de implementare este prin folosirea tehnologiilor si uneltelor de dezvoltare de la Microsoft: sitemele de operare Windows (cu variantele 2003 server si XP intr-o prima instanta), serverul de Web IIS (Internet Information Service), cu ASP.NET, mediul de dezvoltare Visual Studio .NET 2005 sau 2008, cu limbajele C# si Visual Basic, .NET Framework 3.5, WSDL pentru descierea serviciilor, UDDI pentru publicarea si descoperirea serviciilor Web si HTTP ca protocol de transport.

*Serviciile Web* vor pune la dispozitia beneficiarilor informatiile utile, atat cele actuale cat si informatii din "istoricul" inregistrarilor pacientilor, informatii stocate in bazele de date operationale, sau in depozitele centralizate, situate fie pe un server intern (in Intranetul organizatiei furnizoare de date medicale) fie pe serverele depozitelor centralizatoare.

Etapele constructiei si utilizarii unor servicii Web sunt schitate in diagrama de mai jos:

**Furnizor de servicii Web** **Registre de servicii Web** **Consumator de servicii Web**

1. Construieste serviciul Web (C#, VB dinVS.NET sau Java/J2EE)
2. Publicarea serviciului Web (IIS cu ASP.NET sau server de aplicatie cu JAX-WS)
3. Publicarea serviciului Web --→ Registre de servicii Web (UDDI) (UDDI) ←-----4. Localizarea serviciului Web(UDDI)
5. Preluarea descrierii serviciului Web (WSDL)

6. Construirea unui “proxy” pentru serviciul Web si a clientului serviciului

Web (C#, VB, VS.NET, Java/J2EE)

8. Extragerea, codificarea si validarea datelor (XML) ←---------- 7. Apelul serviciului Web (SOAP)

9. Furnizarea datelor (SOAP/HL7) ----------→ 10. Receptia, validarea si consumarea (prelucrarea) datelor (XML

Transportul mesajelor SOAP prin stratul de comunicatie (retea) se poate face folosind un protocol de transport cum ar fi de exemplu protocolul HTTP sau SMTP.
Un exemplu de implementare a unui serviciu Web folosind tehnologiile de la Microsoft este prezentat mai jos in figura VII.4:

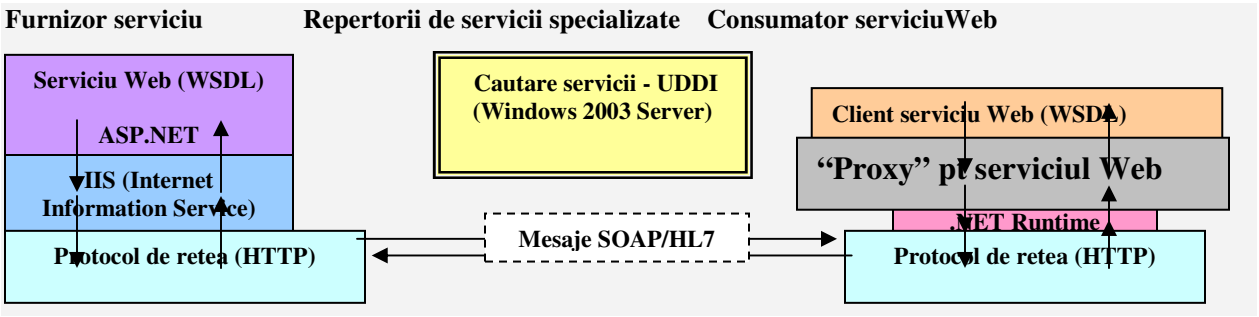



Figura VII.4. Transport de „mesje” prin servicii web

### 7.4.1 Repertoriul de servicii

Repertoriul de servicii(UDDI) este unul dintre cele mai populare mecanisme de cautare a serviciilor, mai ales prin prisma proceselor de afaceri pe care le modeleazã.

UDDI are o functie asemanatoare motoarelor de căutare disponibile pe Web, permitand utilizatorilor căutarea serviciilor Web pentru necesitatile proprii. UDDI foloseste informatiile de descriere a serviciului Web stabilite prin intermediul limbajului WSDL, în scopul oferirii potentialilor utilizatori a unei modalitati efciente de cautare, completata cu un set de informatii de utilizare a serviciului Web.

Data fiind arhitectura UDDI, a tipurilor de informatii stocate si a intimitatilor privind functionarea, prezentam solutii practice de constituire a unui registru UDDI propriu testate. Amintim: open source jUDDI, BEA Aqualogic Service Registry. Pentru exemplificare am ales Microsoft UDDI Services a carui implementare se poate face cu ajutorul aplicatiilor UDDI din Windows 2003 Server (Standard sau Enterprise Edition). Acestea constituie o componenta optionala a Windows 2003 Server care ofera facilitatile UDDI (publicarea, descoperirea, partajarea si reutilizarea serviciilor Web) in cadrul intreprinderii sau intre partenerii de business.
Pentru instalarea serviciilor UDDI se procedeaza dupa cum urmeaza:

1. Se da clic pe butonul START si, apoi, se acceseaza Control Panel
2. Se da click pe Add/Remove Windows Components

3. In fereastera Windows Components Wizard se bifeaza caseta din dreptul intrarii ”UDDI Services” (care contine doua componente ”UDDI Services Administration Console” si ”UDDI Services Database and Web Server Components” vizibile prin clic pe butonul ”Details...”)
4. Se da clic pe butonul OK, apoi pe Next> pina la Finish

**Nota**: Utilizarea serviciilor UDDI este conditionata de existenta unei instalari SQL Server pe serverul Windows 2003 respectiv sau pe unul accesibil si de instalarea si functionarea a IIS (Internet Information Service) pe serverul Windows 2003 respectiv.

Pentru administrarea serviciilor UDDI:

1. Se da clic pe butonul START si, apoi, se acceseaza Administrative Tools
2. Se selecteaza UDDI Services din lista de scule de administrare.
3. Se deschide UDDI Services Console care permite

De principiu, un partener de business poate sa publice, intr-un repertoriu de servicii UDDI, trei tipuri de informatii grupate dupa cum urmeaza:

- Pagini albe – informatii de baza pentru contact si identificatori pentru partenerul de business cum ar fi: numele partenerului, adresa, informatii de contact, identificatori unici (numere DUNS, identificatori de impozitare sau GLN)
- Pagini galbene – informatii care descriu serviciile Web folosind diferite categorisiri (taxonomii) pe domenii de business, cum ar fi: fabricatie/productie, vinzare de autovehicole etc.
- Pagini verzi – informatii tehnice care desciu comportarea si functiile sustinute de un serviciu Web gazduit de partenerul de business, cit si locul unde este localizat serviciul Web..

**Nota**: o alternativa la folosirea unui repertoriu UDDI pentru descoperirea serviciilor Web este utilizarea fisierelor DISCO (.disco si .vsdisco). DISCO este un protocol Microsoft care nu este un standard, asa incit folosirea lui prezinta cel putin doua dezavantaje:

- Limitarea la medii centrate pe tehnologii si produse Microsoft,
- Limitarea la servere locale dintr-un LAN

### 7.4.2 Protocoale de mesaje si structura mesajelor SOAP/HL7

Specificatia SOAP[5] a fost dezvoltata de IBM, Microsoft si UserLand Software, Inc. si a fost pusa pe piata in mai 2000. Aceasta specificatie defineste structura completa de impachetare a mesajelor, bazata pe o schema XML si expediata intr-o cerere HTTP. Specificatia a fost apoi depusa la W3C[9], care are acum sarcina de a gestiona pe mai departe aceasta specificatie. In cadrul proiectului a fost adoptata tehnologia SOAP impreuna cu standardul de mesaje specializate pentru medicina, HL7. Descrierea detaliata a mesajelor acestui standard depaseste cadrul acestei lucrari, dar mesajele specializate pentru domeniul ales vor fi prezentate totusi in documentatia etaei urmatoare.

#### 7.4.2.1 Mesaj SOAP fara atasamente

Figura VII.5, prezentata mai jos, ilustreaza structura mesajelor SOAP 1.1, fara atasamente. Remarcam faptul ca in afara antetului SOAP, toate celelalte parti sint cerute pentru fiecare mesaj

SOAP. Antetul SOAP contine informatii despre partenerii emitatori si cei receptori ai mesajului. Continutul mesajului se va afla in corpul SOAP.

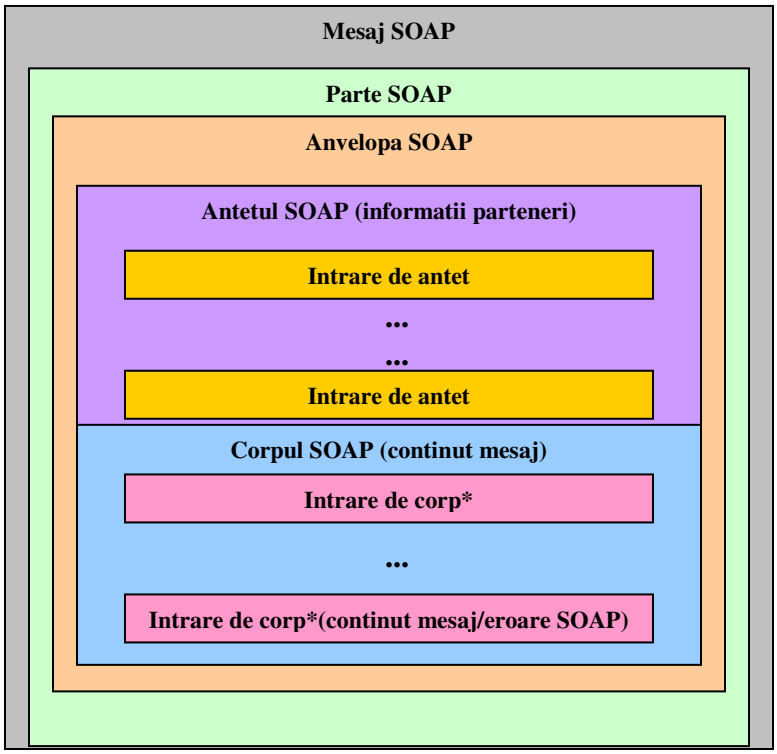
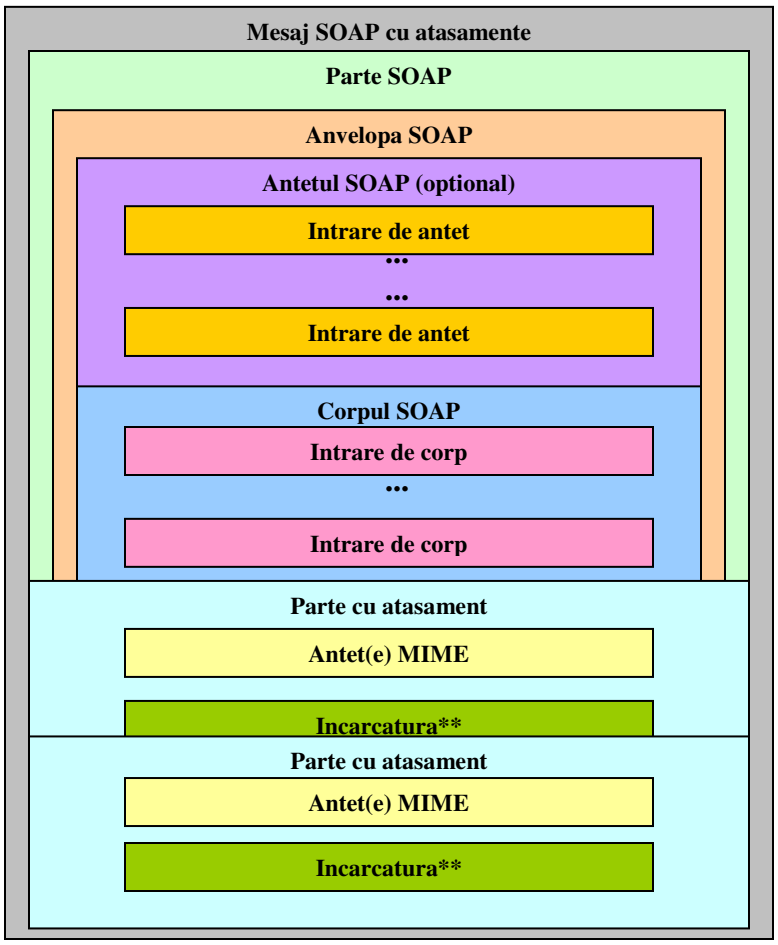



Figura VII.5. Structura mesaje SOAP fara atasamente

### 7.4.2.2 Mesaj SOAP cu atasamente

Specificatia urmatoare SOAP 1.2 a fost pusa pe piata, fiind numita si SwA (SOAP with Attachments), deoarece contine o extindere majora fata de versiunea 1.1: crearea legaturilor SOAP care transmit atasamente impreuna cu anvelopa SOAP si ofera posibilitatea referirii acestor atasamente din anvelopa. Atasamentele SOAP sint descrise folosind notiunea de structura compusa de document, constind dintr-o parte de mesaj SOAP primara si zero sau mai multe parti de documente inrudite cunoscute sub numele de atasamente.




* Intrarea de corp poate sa poarte fie un continut XML al mesajului, fie o eroare SOAP.
** Incarcatura atasamentului poate sa poarte continut XML sau non-XML al mesajului

#### 7.4.2.3 Mesajele SOAP/HL7

Spre deosebire de arhitectura Client-Server, HL7 prezintă o arhitectură B2B, aceasta însemnând că aplicația, în care are loc un eveniment, va trimite mai degrabă un mesaj altei aplicații decât să răspundă unei cereri.

Structura unui mesaj HL7 este:

Mesaj ⟹ Segmente ⟹ Date compuse ⟹ Alte tipuri de date compuse sau date primitive

Setul complet de mesaje pentru domeniul ales va fi prezentat in documentatie etapei din decembrie 2009. Dam acum,ca exeplu, un mesaj HL7 tipic, de forma ADT^A04 („Admitting”. Acest mesaj este trimis atunci când un pacient ajunge la spital. Datele demografice ale pacientului sunt introduse în sistemul HIS (hospital information system ) şi trimis tuturor sistemelor pentru a evita mai multe intrări ale datelor demografice ale pacientului.

MSH|^~\&|EPIC|EPICADT|SMS|SMSADT|199912271408|CHARRIS|ADT^A04|1817457|D|2.3|

EVN|A04|199912271408|||CHARRIS

PID||0493575^^^2^ID 1|454721||DOE^JOHN^^^^|DOE^JOHN^^^^|19480203|M||B|254

E238ST^^EUCLID^OH^44123^USA|| (216)731-4359 |||M|NON|400003403~1129086|

NK1||CONROY^MARI^^^^|SPO|| (216)731-4359 ||EC|||||||||||||||||||||||||

PV1||O|168 ~219~C~PMA^^^^^^^^^|||277^ALLEN FADZL^BONNIE^^^^|||||||||

||2688684|||||||||||||||||||||||||199912271408||||||002376853

Mesajele HL7 sunt mesaje ASCII (spre deosebire alte protocoalele specializate, cum ar fi DICOM), iar standardele solicită ca acestea să fie descifrabile pentru om.

Mesajele - reprezintă o secvență de segmente definită sau grupuri de segmente. Fiecare segment, grup sau mesaj în interiorul unui mesaj poate fi opțional şi/sau repetat.

Fiecare mesaj este format din segmente care sunt delimitate de caractere "carriage return" ( "\ r" sau 0x0D). Fiecare segment este plasat pe o altă linie.

Fiecare linie dintr-un mesaj este menționată ca un segment, fiecare având propriul său scop semantic. Aceasta înseamnă că el conține informații de un anumit tip. De exemplu:

- **MSH** conține informații despre cel care trimite sau primeşte mesajul, tipul mesajului, timpul de timbru, etc.
- **EVN** conține informații despre tipul de mesaj; de exemplu, A04 (înregistrarea pacientului. Informația inclusă în **EVN** este duplicată în **MSH**, astfel încât începând cu versiunea HL7 2.3, acest segment este exclus din toate definițiile mesajului.
- **PID** conține informații demografice cu privire la pacient; de exemplu: nume, cod, cod ID, adresă, etc.
- **PV1** conține informații legate de şederea pacientului în spital, cum ar fi, locația, doctorul recomandat, etc.

În versiunea HL7 2.3 sunt definite peste 120 de segmente.

Segmentul – reprezintă unitățile care comprimă un mesaj. Un segment este definit ca şi o secvență de câmp care se poate repeta sau nu. Definiția unui mesaj HL7 este exprimată indiferent dacă un segment este obligatoriu sau nu.

Segmentele sunt formate din câmpuri, care sunt compuse. Datele compuse sunt delimitate de caracterele tip "|" (pipe). Fiecare câmp îşi are propriul scop şi este definit de standardul HL7 pentru fiecare segment.

În versiunea 2.3, segmentul PID conține 30 de câmpuri. Doar două dintre ele sunt obligatorii: PatientName şi PatientID. Unele câmpuri conțin o singură valoare (de exemplu, câmpul nr. 8), pe când alte câmpuri pot să conțină mai multe valori delimitate de caracterul „^” (de exemplu, câmpul 5 - PatientName).

#### *7.4.2.3.1 Gramatica segmentelor HL7*

Există o notație a standardului HL7 folosită pentru a documenta structura unui mesaj HL7, pe care o vom găsi în definiția standardului HL7, în descrierea interfețelor celor mai mulți furnizori.

Exemplu de definiție de mesaj

```
MSH PID PV1
```

Exemplul de mai sus reprezintă un mesaj care trebuie să conțină doar segmente MSH, PID şi PV1, în această ordine.

Segmente opționale

Un segment este opțional atunci când apare între paranteze pătrate. De exemplu, mesajul **MSH PID PV1 [PD1]** indică faptul că segmentele MSH, PID şi PV1 trebuie să apară în mesaj în această ordine, şi că un segment PD1 ar putea apărea, după cum se poate observa din cele două paranteze pătrate, după PV1.

Segmente repetitive

Când un segment s-ar putea repeta, acesta va apărea între două acolade, de genul **{NTE}**., care înseamnă că cel puțin o apariție a segmentului este prezentă, şi că se poate repeta de nenumărate ori.

În definiția mesajului, fiecare segment poate fi obligatoriu sau opțional. Fiecare mesaj începe cu un segment care este întotdeauna obligatoriu. La primirea unui mesaj HL7, se parsează segmentului MSH pentru a determina ce fel de mesaj este.

Un alt exemplu de câmp obligatoriu este PID (Identificare pacient), fără de care mesajele de genul ADT^A04 (înregistrare pacient), nu au relevanță.

Alte segmente, precum AL1 (alergii), sunt opționale, deoarece pacienții pot fi alergici sau nu. Exemplul de mesaj HL7, de mai jos, conține segmentele MSH, EVN, PID, NK1 si PV1

Ținând cont de definiția versiunii 2.2 a protocolului HL7, segmentele MSH, EVN, PID şi PV1 sunt necesare într-un mesaj ADT^A04 iar segmentul NK1 este opțional. De asemenea, şi segmentele DG1, PR1, AL1 sunt opționale care ar putea face parte din mesaj dar nu sunt prezente.

Ordinea este importantă pentru ambele, segment şi câmp, din interiorul segmentului.

Segmente din mesajul HL7 se pot de asemenea repeta. De exemplu, NK1 (Next of Kin/Associated Parties) se va repeta de mai multe ori în cazul în care o persoană are mai multe relații de rudenie. În exemplul de mai jos segmental NK1 se repetă de două ori:

MSH|^~\&|EPIC|EPICADT|SMS|SMSADT|199912271408|CHARRIS|ADT^A04|1817457|D|2.3
|

EVN|A04|199912271408|||CHARRIS

PID||0493575^^^2^ID 1|454721||DOE^JOHN^^^^|DOE^JOHN^^^^|19480203|M||B|254

E238ST^^EUCLID^OH^44123^USA|| (216)731-4359

|||M|NON|400003403~1129086|999-|

NK1||CONROY^MARI^^^^|SPO|| (216)731-4359 ||EC|||||||||||||||||||||||||

NK1||DOE^JOHN ^^^^|SPO|| (216)731-4222 ||EC|||||||||||||||||||||||||

NK1||DOE^ROBERT ^^^^|SPO|| (216)731-4222 ||EC|||||||||||||||||||||||||

PV1||O|168 ~219~C~PMA^^^^^^^^^^||||277^ALLEN FADZL^BONNIE^^^^|||||||||

||2688684|||||||||||||||||||||||||||199912271408||||||002376853

Un grup de segmente este o colecție de segmente care pot fi considerate ca un singur segment în scopuri repetitive. De exemplu, un grup de segmente poate fi opțional ca și grup, astfel încât se poate repeta ca și grup.

Considerăm următorul exemplu:

Dacă mesajul ORU^R01 are un grup care conține un OBR (Observation request) și segmente de la 0 până la N OBX (Observation result), acest grup este opțional în mesaj. Mesajul va arăta astfel:

```
MSH PID PD1
```

De asemenea un mesaj poate conține un număr de grupuri de observație. De exemplu când grupul OBR-OBX se repetă mesajul va arăta astfel:

```
MSH PID PD1 OBR OBX OBX OBX OBR OBX OBX
```

Primele trei rezultate aparțin primei cereri de observație, și următoarele două aparținând celei de-a doua:

```
MSH PID PD1 - header and patient demographics
OBR OBX OBX OBX  - first observation group (e.g. height and weight)
OBR OBX OBX  - second observation group (e.g. lab results)
```

Dacă se doreste parsarea un mesaj care conține grupuri de segmente trebuie creată o structură complexă similară cu tabelele de baze de date relaționale, acest lucru permițând păstrarea unei legături între segmentele din grup, mai degrabă decât o lungă serie de segmente.

#### *7.4.2.3.2 HL7 - Protocolul de Acknowledge*

O altă parte importantă a standardului HL7 îl reprezintă protocolul Acknowledge. De fiecare dată când o aplicație acceptă un mesaj și acesta este consumat, trebuie trimis înapoi la aplicație un mesaj de confirmare. Aplicația va continua să trimită mesajul până ce va primi mesajul de confirmare.

Dacă nu este respectată această regulă, atunci datele se pot pierde.

Conceptul cheie în protocolul Acknowledgement este **Message Control ID**. Acesta este un număr unic pe care fiecare mesaj HL7 îl are în câmpul 10 din segmental său MSA. Un mesaj acknowledge HL7 valid va repeta acest ID în cel de-al doilea câmp din segmentul MSA. Următoarea diagramă arată un mesaj de acknowledge cu cele mai importante câmpuri etichetate:

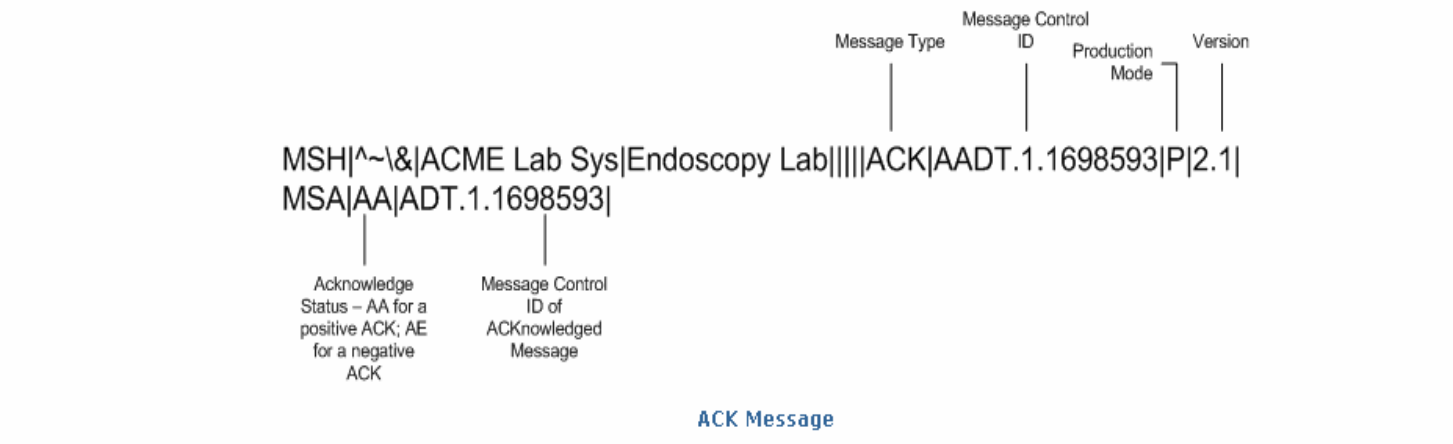



Fig. 9 Mesaj Acknowledgement

Se poate observa faptul că mesajul conține două segmente:

- Segmentul MSH
- Segmentul MSA.

**MSH** este prescurtarea de la **M**e**S**sage **H**eader segment. Fiecare mesaj HL7 începe cu un segment MSH. Acesta are informații despre trimiterea şi primirea aplicațiilor şi a facilităților. Conține de asemenea si versiunea fiecărui mesaj. Cel mai important, el deține Message Control ID al mesajului, care este ID-ul unic al fiecărui mesaj.

Segmentul **MSA** trebuie să aibă message control ID-ul mesajului care este confirmat în cel de-al doilea câmp. Primul câmp trebuie să fie o constanta, AA, aceasta însemnând o confirmare pozitivă fără erori.

Setul de mesaje si segmente precum si testele de interoperabilitate vor fi prezentate in documentatia etapei a IV a, dec 2009.

## 7.5 Securitatea serviciilor Web si a schimburilor de informatii

Pentru asigurarea securitatii serviciilor Web, a mesajelor si documentelor schimbate intre partenerii de business solutia propusa rezolva cele trei cerinte de baza de securitate:

- Autenticitatea
- Integritatea
- Confidentialitatea

Cerintele de securitate pentru serviciile Web sunt autentificarea, controlul accesului, stabilirea unui canal sigur pentru schimbul de mesaje, securitatea la nivelul mesajului si securizarea interactiunilor cu alte componente in timpul prelucrarii cererilor. In consecinta, problemele de securitate trebuie identificate, abordate si rezolvate in toate straturile nivelului interactiunilor dintre parteneri de business, dupa cum urmeaza:

-la nivelul serviciilor Web: prin tehnici de autentificare si autorizare pe baza de nume de utilizator si parola, de acces bazat pe roluri, precum si folosirea certificatelor de server si/sau de utilizator.

-la nivelul canalului de comunicatie si al schimbului de mesaje: prin intermediul SSL (Secure Socket Layer) si, eventual, o infrastructura bazata pe cheie publica (PKI).
-la nivelul documentelor: prin tehnologia semnaturii electronica si criptarea partiala sau totala continutului.

## 7.6 Portal Cardionet- Implementare si testare functionlala

### 7.6.1 Sevicii Web specializate pentru aplicatia Cardionet

Pentru exemplificarea implementarii solutiei distribuite Cardionet, la nivelul schimbului de informatii intre parteneri prin folosirea serviciilor Web am ales ca exemple operatiuni simple, dar semnificative si vom ilustra implementarea interactiunilor folosind uneltele de dezvoltare si tehnologiile Microsoft (IIS si ASP.NET din Windows 2003 Server si Windows XP Professional, limbajul de programare C# din Visual Studio .NET 2008 si sistemul de gestiune a bazelor de date relationale SQL Server 2008)
La nivelul portalului Cardionet vor exista "registries" specializate dupa anumite functionalitati de business (fluxurile medicale de informatii, in cazul nostru). Exemple: registru de servicii medicale disponibile, registru de documente tipizate impuse de MSP, registru medicilor specialisti, registrul unitatilor de asistenta, etc.

**Exemplu de WS specializat pentru obtinerea informatiilor despre un serviciu medical:**
Un prim exemplu de serviciu Web permite obtinerea informatiilor de detaliu despre un serviciu medical:
**Serviciul de cautare: Web *ServiciuMedical*:**
Scheletul serviciului *ServiciuMedical*

```
...
using System.Web;
using System.Web.Services;
namespace WS_Prod
{
        public class ServiciuMedical : System.Web.Services.WebService
        {
                public ServiciuMedical()
                {
                        //CODEGEN: This call is required by the ASP.NET Web Services Designer
                        InitializeComponent();
                }
                ...
         [WebMethod]
                //public string GetServiciu ()
                public DataSet GetServiciu()
                {
                        ...
                }
                [WebMethod]
```

```
        public string GetServiciu(string serviciu)
        {
            ...
        }
        ...
    }
}
```

Descrierea tuturor acestor registre, cu testele aferente va fi prezentata in etapa urmatoare, din dec 2009.

### 7.6.2. Sabloane de documente, completarea si livrarea sabloanelor prin WS

Un alt exemplu de index (registry) si serviciu Web aferent este cel care permite obtinerea informatiilor despre un tip de document (« template » reglementat de MSP), alimentarea (« binding ») lui cu datele unui anumit pacient (gasit dupa identificatorul unic de tip GUID) crearea documentului respectiv si livrarea lui catre apelant.

Din « LISTA FORMULARELOR DIN SISTEMUL INFORMAȚIONAL" publicate de catre MSP, am ales ca exemplu, pentru documentare "sablonul" cu codul: 14.3 ("Bilet de iesire din spital"). Pentru acest document solicitat, regasim macheta in repository-ul de sabloane, prezentata (listata) in continuare (fisierul BiletIesireSpital.dot sau dotx):

| | | |
|---|---|---|
| Judeţul | | anul luna ziua |
| Localitatea | | |
| Unitatea sanitară | | CNP |
| | | |

| **BILET DE IEŞIRE DIN SPITAL** |
|---|
| Numele Prenumele |
| Sexul M / F în vârstă de ani, cu domiciliul în: judeţul |
| localitatea str. nr. |
| Cabinetul medical: |
| A fost internat în secţia: |
| Cu diagnosticul: |
| |
| ESTE / NU ESTE purtător de germeni. |
| Felul: |
| Tipul: |

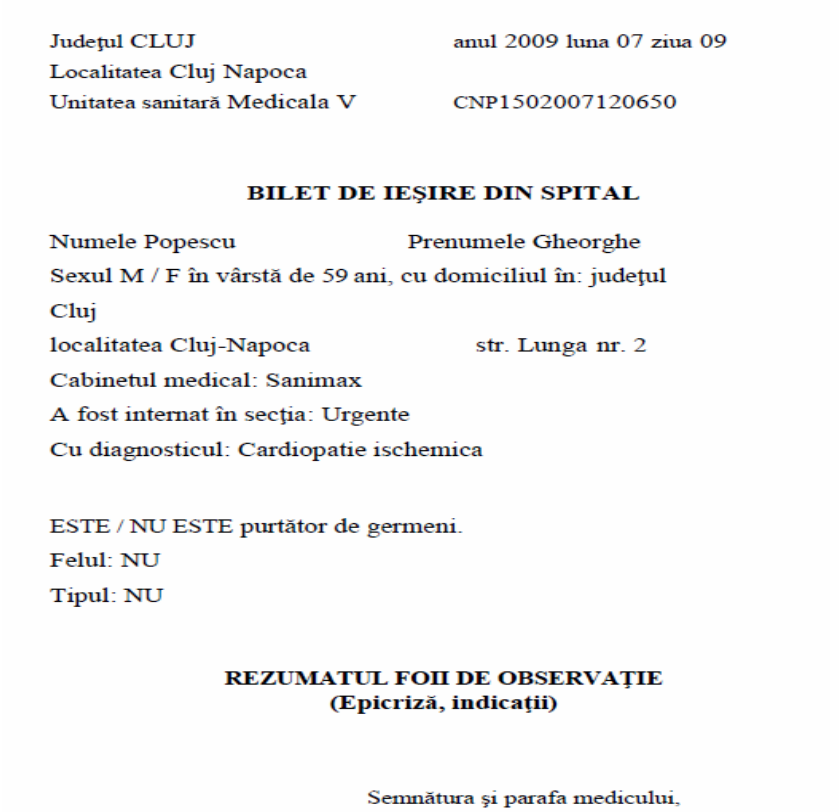
**REZUMATUL FOII DE OBSERVAŢIE**
**(Epicriză, indicaţii)**

Semnătura şi parafa medicului,

Acest template in format xml aste alimentat  din baza de date operativa, cu datele pacientului solicitat (prin GUID), rezultand o instanta de document populata, in fisierul xml « BiletIesireSpital.xml »:

```
<?xml version="1.0" encoding="utf-8" ?>
- <BiletIesireSpital xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Judet>CLUJ</Judet>
  <An>2009</An>
  <Luna>07</Luna>
  <Zi>09</Zi>
  <Localitate>Cluj Napoca</Localitate>
  <UnitateSanitara>Medicala V</UnitateSanitara>
  <Cnp>1502007120650</Cnp>
  <Nume>Popescu</Nume>
  <Prenume>Gheorghe</Prenume>
  <Varsta>59</Varsta>
  <AdresaJudet>Cluj</AdresaJudet>
  <AdresaLocalitate>Cluj-Napoca</AdresaLocalitate>
  <AdresaStrada>Lunga</AdresaStrada>
  <AdresaNumar>2</AdresaNumar>
  <CabinetMedical>Sanimax</CabinetMedical>
  <Sectia>Urgente</Sectia>
  <Diagnostic>Cardiopatie ischemica</Diagnostic>
  <GermeniFel>NU</GermeniFel>
  <GermeniTip>NU</GermeniTip>
    </BiletIesireSpital>
```

Dupa popularea cu datele reale a documentului, acesta este creat intr-o instanta de  fisier pdf : « BiletIesireSpital.pdf » :



Judeţul CLUJ anul 2009 luna 07 ziua 09
Localitatea Cluj Napoca
Unitatea sanitară Medicala V CNP1502007120650

**BILET DE IEŞIRE DIN SPITAL**

Numele Popescu Prenumele Gheorghe
Sexul M / F în vârstă de 59 ani, cu domiciliul în: judeţul Cluj
localitatea Cluj-Napoca str. Lunga nr. 2
Cabinetul medical: Sanimax
A fost internat în secţia: Urgente
Cu diagnosticul: Cardiopatie ischemica

ESTE / NU ESTE purtător de germeni.
Felul: NU
Tipul: NU

**REZUMATUL FOII DE OBSERVAŢIE**
**(Epicriză, indicaţii)**

Semnătura şi parafa medicului,

si apoi livrat apelantului, cu semnatura electronica daca partenerii au stabilit si implementat intre ei acest mod de comunicare.

**Este prezentata in continuare descrierea WSDL a serviciului web de generare documente, dupa sabloane: (aici apelat pentru Bilet de Iesire)**

```
<?xml version="1.0" encoding="utf-8" ?>
- <wsdl :definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://tempuri.org/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" targetNamespace="http://tempuri.org/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
- <wsdl:types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
- <s:element name="HelloWorld">
  <s:complexType />
    </s:element>
- <s:element name="HelloWorldResponse">
+ <s:complexType>
    </s:element>
- <s:element name="GenereazaDocumentPDF">
- <s:complexType>
```

```
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="codTemplate" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="dateBiletIesire" type="tns:BiletIesireSpital"
      />
  <s:element minOccurs="1" maxOccurs="1" name="tipDoc" type="tns:TipDocument" />
    </s:sequence>
    </s:complexType>
    </s:element>
- <s:complexType name="BiletIesireSpital">
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="Judet" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="An" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Luna" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Zi" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Localitate" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="UnitateSanitara" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Cnp" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Nume" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Prenume" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Varsta" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="AdresaJudet" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="AdresaLocalitate" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="AdresaStrada" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="AdresaNumar" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="CabinetMedical" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Sectia" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Diagnostic" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="GermeniFel" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="GermeniTip" type="s:string" />
    </s:sequence>
    </s:complexType>
- <s:simpleType name="TipDocument">
- <s:restriction base="s:string">
  <s:enumeration value="PDF" />
  <s:enumeration value="XPS" />
    </s:restriction>
    </s:simpleType>
- <s:element name="GenereazaDocumentPDFResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="GenereazaDocumentPDFResult"
      type="s:base64Binary" />
    </s:sequence>
    </s:complexType>
    </s:element>
    </s:schema>
```

```
    </wsdl:types>
- <wsdl:message name="HelloWorldSoapIn">
  <wsdl:part name="parameters" element="tns:HelloWorld" />
    </wsdl:message>
- <wsdl:message name="HelloWorldSoapOut">
  <wsdl:part name="parameters" element="tns:HelloWorldResponse" />
    </wsdl:message>
- <wsdl:message name="GenereazaDocumentPDFSoapIn">
  <wsdl:part name="parameters" element="tns:GenereazaDocumentPDF" />
    </wsdl:message>
- <wsdl:message name="GenereazaDocumentPDFSoapOut">
  <wsdl:part name="parameters" element="tns:GenereazaDocumentPDFResponse" />
    </wsdl:message>
- <wsdl:portType name="ServiceSoap">
- <wsdl:operation name="HelloWorld">
  <wsdl:input message="tns:HelloWorldSoapIn" />
  <wsdl:output message="tns:HelloWorldSoapOut" />
    </wsdl:operation>
- <wsdl:operation name="GenereazaDocumentPDF">
  <wsdl:input message="tns:GenereazaDocumentPDFSoapIn" />
  <wsdl:output message="tns:GenereazaDocumentPDFSoapOut" />
    </wsdl:operation>
    </wsdl:portType>
- <wsdl:binding name="ServiceSoap" type="tns:ServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="HelloWorld">
  <soap:operation soapAction="http://tempuri.org/HelloWorld" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GenereazaDocumentPDF">
  <soap:operation soapAction="http://tempuri.org/GenereazaDocumentPDF" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
    </wsdl:binding>
- <wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
```

```
- <wsdl:operation name="HelloWorld">
  <soap12:operation soapAction="http://tempuri.org/HelloWorld" style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GenereazaDocumentPDF">
  <soap12:operation soapAction="http://tempuri.org/GenereazaDocumentPDF" style="document"
     />
- <wsdl:input>
  <soap12:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
    </wsdl:binding>
- <wsdl:service name="Service">
- <wsdl:port name="ServiceSoap" binding="tns:ServiceSoap">
  <soap:address location="http://localhost:8175/WS/Service.asmx" />
    </wsdl:port>
- <wsdl:port name="ServiceSoap12" binding="tns:ServiceSoap12">
  <soap12:address location="http://localhost:8175/WS/Service.asmx" />
    </wsdl:port>
    </wsdl:service>
    </wsdl:definitions>
```

In continuare sunt prezentate cateva exemple de servicii care vor fi disponibile pe portal. Lista serviciilor va fi completata, documentata si adaugata in etapa urmatoare.

**Serviciul Web ServiciuFurnizori**

Scheletul clasei *ServiciuFurnizori*

```
...
using System.Web;
using System.Web.Services;

namespace WS_Part
{
        public class ServiciuFurnizori : System.Web.Services.WebService
        {
                public ServiciuFurnizori()
```

```
                {
                        //CODEGEN: This call is required by the ASP.NET Web Services
Designer
                        InitializeComponent();
                }
                [WebMethod]
                public DataSet GetFurnizori()
                {
                        ...
                }
                [WebMethod]
                public string GetFurnizor(string codFurnizor)
                {
                        ...
                }
                ...
        }
}
```

**Utilizarea serviciului Web *ServiciuFurnizori***

Pentru exemplificarea utilizarii (consumarea) serviciului Web *ServiciuFurnizori* am folosit o aplicatie Windows care afiseaza lista tuturor partenerilor aflati in baza de date si numele partenerului care carespunde GLN-ului specificat de utilizator.

Adaugarea referintei Web la *ServiciuFurnizori*

Apelul unei metode oferite de serviciul Web *ServiciuFurnizori*

```
private void button1_Click(object sender, System.EventArgs e)
{
        localhost_WS_Part.ServiciuFurnizori myService =
                        new localhost_WS_Part.ServiciuFurnizori();
        richTextBox1.Text = myService.IaPartener(tbxGLN.Text);
        ...
```

Scheletul serviciului web *WSMedicFamilie*

```
...
using System.Web;
using System.Web.Services;

namespace WS_Prod
{
        public class ServiceMedicFamilie : System.Web.Services.WebService
```

```
        {
                public ServiceMedicFamilie()
                {
                        //CODEGEN: This call is required by the ASP.NET Web Services Designer
                        InitializeComponent();
                }
                ...
                /// <summary>
                /// met.folosita de catre spital pt a trimite rezultaul unei externari
                /// </summary>
                /// <param name="mesajBiletExternare"></param>
                /// <returns></returns>
         [WebMethod]
                //public string GetServiciu ()
                public string BiletExternare(string mesajBiletExternare)
                {
                        ...
                }
                /// <summary>
                /// met. apelata de catre laborator pt a trimite rezultaul unei analize
                /// </summary>
                /// <param name="rezultatAnalize"></param>
                /// <returns></returns>
                [WebMethod]
                public string RezultatAnalize (string rezultatAnalize)
                {
                        ...
                }
                ...
        }
}
```

Scheletul serviciului web *WSSpital*

```
...
using System.Web;
using System.Web.Services;

namespace WS_Prod
{
        public class ServiceSpital : System.Web.Services.WebService
        {
                public ServiceSpital()
                {
```

```
                    //CODEGEN: This call is required by the ASP.NET Web Services Designer
                    InitializeComponent();
                }
                ...
                /// <summary>
                /// metoda apelata de catre medicul de familie cand trimite un pacient
                /// pentru investigatii la spital
                /// </summary>
                /// <param name="mesajBiletTrimitere"></param>
                /// <returns></returns>
        [WebMethod]
                //public string GetServiciu ()
                public string SolicitareInvestigatie (string mesajBiletTrimitere)
                {
                    ...
                }
                ...
        }
}
```

**Descrierea serviciului web *WSMedicFamilie*(folosind *WSDL)***

```
<?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    …
- <wsdl:types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
- <s:element name="BiletExternare">
- <s:complexType>
- <s:sequence>
 <s:element minOccurs="0" maxOccurs="1" name="mesajBiletExternare" type="s:string" />
    </s:sequence>
    </s:complexType>
    </s:element>
+ <s:element name="BiletExternareResponse">
+ <s:element name="RezultatAnalize">
+ <s:element name="RezultatAnalizeResponse">
    </s:schema>
    </wsdl:types>
- <wsdl:message name="BiletExternareSoapIn">
 <wsdl:part name="parameters" element="tns:BiletExternare" />
    </wsdl:message>
- <wsdl:message name="BiletExternareSoapOut">
 <wsdl:part name="parameters" element="tns:BiletExternareResponse" />
    </wsdl:message>
```

```
- <wsdl:message name="RezultatAnalizeSoapIn">
  <wsdl:part name="parameters" element="tns:RezultatAnalize" />
    </wsdl:message>
- <wsdl:message name="RezultatAnalizeSoapOut">
  <wsdl:part name="parameters" element="tns:RezultatAnalizeResponse" />
    </wsdl:message>
- <wsdl:portType name="ServiceMedicFamilieSoap">
- <wsdl:operation name="BiletExternare">
  <wsdl:input message="tns:BiletExternareSoapIn" />
  <wsdl:output message="tns:BiletExternareSoapOut" />
    </wsdl:operation>
- <wsdl:operation name="RezultatAnalize">
  <wsdl:input message="tns:RezultatAnalizeSoapIn" />
  <wsdl:output message="tns:RezultatAnalizeSoapOut" />
    </wsdl:operation>
    </wsdl:portType>
- <wsdl:binding name="ServiceMedicFamilieSoap" type="tns:ServiceMedicFamilieSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
+ <wsdl:operation name="BiletExternare">
+ <wsdl:operation name="RezultatAnalize">
    </wsdl:binding>
- <wsdl:binding name="ServiceMedicFamilieSoap12" type="tns:ServiceMedicFamilieSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
+ <wsdl:operation name="BiletExternare">
+ <wsdl:operation name="RezultatAnalize">
    </wsdl:binding>
- <wsdl:service name="ServiceMedicFamilie">
- <wsdl:port name="ServiceMedicFamilieSoap" binding="tns:ServiceMedicFamilieSoap">
  <soap:address location="http://localhost:59441/WSMedicFamilie/ServiceMedicFamilie.asmx" />
    </wsdl:port>
- <wsdl:port name="ServiceMedicFamilieSoap12" binding="tns:ServiceMedicFamilieSoap12">
  <soap12:address location="http://localhost:59441/WSMedicFamilie/ServiceMedicFamilie.asmx"
     />
    </wsdl:port>
    </wsdl:service>
    </wsdl:definitions>
```

**Utilizarea serviciului Web *ServiciuMedical***

Pentru exemplificarea utilizarii (consumarea) serviciului Web *ServiciuMedical* am folosit o aplicatie Windows care afiseaza lista tuturor serviciilor oferite de partener cu detalii detalii.

Adaugarea referintei la serviciul Web *ServiciuMedical*
Apelul unei metode oferite de serviciul Web *ServiciuMedical*

```
private void button1_Click(object sender, System.EventArgs e)
{
```

```
localhost_WS_Prod.ServiciuMedical myService =
                        new localhost_WS_Prod.ServiciuMedical();
richTextBox1.Text = myService.GetServiciu(tbxCodServiciu.Text);
...
```

In etapa urmatoare de implementare a solutiei, conform modelului prezentat vom dezvolta servicii specializate pentru toti “actorii” care vor activa in sistemul distribuit Cardionet, pentru a permite un maximum de interconectivitate intre aplicatii si sisteme informatice.

**Bibliografie Capitol VII:**

[1-HMO]- Health Maintenance Organization Act of 1973 (Public Law 93-222)

[2-HL7]- Health Level Seven, www.hl7.org

[3-SOA]- Service-Oriented Architecture (SOA), www.sun.com/products/soa/

[4-XML]-Extensible Markup Language: http://www.w3.org/XML/

[5-SOAP]-Simple Object Access Protocol: http://www.w3.org/TR/soap/

[6-WSDL]-Web Services Description Language: http://www.w3.org/TR/wsdl

[7-UDDI]-Universal Description, Discovery and Integration: *www.uddi.org*.

[8-OASIS]- Organization for the Advancement of Structured Information Standards, www.oasis-open.org/

[9-W3C]- World Wide Web Consortium: http://www.w3.org/

[10-IETF]- Internet Engineering Task Force: http://www.ietf.org/

[11-HTTP]- http://www.w3.org/Protocols/http/

[12-DNS]- Domain Name System: http://www.dns.net/dnsrd/

# 8 Cardionet- Fluxuri de prelucrare si tehnici de analiza a datelor medicale

## 8.1Recomandari ESC (Societatea Europeana de Cardiologie) privind managementul domeniului cardiovascular

Exploatarea unui sistem distribuit, cum este cel propus prin Cardionet, presupune un flux de prelucrare a datelor, respectiv un set de reguli de „comportament IT" acceptate de toti actorii implicati in exploatarea sistemului.
Pentru a putea stabili aceste conditii a fost necesara o analiza interdisciplinara, efectuata de toti actorii importanti implicati in domeniul proiectului, acela al managementului afectiunilor cardiovasculare. Prezentam, ca exemplu in sensul acestei analize a comportamentului „actorilor", concluziile unui raport al Societatii Europene de Cardiologie privind managementul infarctului miocardic: „Recomandări pentru managementul infarctului miocardic acut ale Societății Europene de Cardiologie" [1-ESC] :

*„**Pacientii:***
*Pacienții cu suspiciunea de accident coronarian au dreptul la un diagnostic prompt, terapia durerii, resuscitare sau de tratament de reperfuzie, dacă au indicație.*
*Pacienții cu suspiciune sau cu infarct miocadic acut confirmat trebuie îngrijiți de personal antrenat şi cu experiență în terapia coronariană modernă. Ei trebuie să aibă acces la metode moderne de diagnostic şi tratament atât în primul loc de examinare, cât şi în unitățile specializate în care va fi transferat. Ei trebuie să*
*beneficieze de aceleaşi facilități şi după externare, pentru recuperare şi prevenția secundară. Ei şi însoțitorii lor trebuie informați cum să recunoască şi cum să reacționeze la un viitor accident coronarian.*
***Cardiologii:***
*Cardiologii, în asociere cu medicii de urgență şi cu autoritățile responsabile de sănătate, trebuie să asigure un sistem optim pentru îngrijirea pacienților cu accident coronarian, care să fie operativ în aria locală, în funcție de fondurile disponibile. La un nivel minim, acesta trebuie să includă o educație potrivită pentru personalul de pe ambulanță şi a medicilor de primă linie, organizarea unui departament de urgență eficient pentru diagnosticul şi tratamentul infarctului miocardic acut şi dezvoltarea unui circuit rapid pentru inițierea terapiei de reperfuzie.*
*Cardiologii, în asociere cu anesteziştii şi ceilalți specialişti, trebuie să se asigure că personalul medical şi paramedical al spitalului au pregătirea necesară resuscitării. Trebuie făcute registre cu timpul scurs de la anunțul telefonic al pacientului până la administrarea terapiei trombolitice ('call to needle time') şi de la internare până la inițierea reperfuziei ( 'door to needle time' or 'door to ballon time'). Primul ar trebui să fie sub 90 minute şi, pentru pacienții cu circuit rapid şi indicație clară de reperfuzie, ''door to needle'' nu ar trebui să depăşească 20 minute, iar ''door to baloon'' 60 de minute.*
*Trebuie făcute registre cu pacienții cu infarct miocardic cert, internați la mai puțin de 12 ore de la debutul simptomelor, cu supradenivelare de segment ST sau cu BRS nou apărut, care primesc terapie de reperfuzie medicamentoasă sau mecanică. Procentul acestor pacienți ar trebui probabil să fie > 90%.*
*PCI este privit ca o alternativă la tratamentul fibrinolitic când tehnica este disponibilă imediat.*
*Rezultatele PCI trebuie păstrate în registre locale şi naționale.*
*Mulți pacienți cu infarct necomplicat, în special cei la care terapia de reperfuzie a fost eficientă, pot fi externați după 4-5 zile.*
*Ar trebui implementată o strategie potrivită pentru evaluarea riscului unui nou accident coronarian acut. Aceasta ar include evaluarea funcției ventriculului stâng şi un test de stress precoce (ECG, scintigrafie sau ecocardiografie).*

*Toţi pacienţii trebuie să aibă acces la un program de reabilitare, în concordanţă cu propriile nevoi.*
*Trebuie stabilită o politică pentru întreruperea fumatului, care să cuprindă un program continuu, susţinut de profesionişti din sănătate, care nu numai să încurajeze oprirea, ci să o menţină.*
*Trebuie păstrate registre cu terapia de prevenţie secundară prescrisă la supravieţuitorii unui infarct miocardic. Aspirina, betablocantele, inhibitorii de enzimă de conversie trebuie recomandate dacă nu există contraindicaţii.*
*La toţi pacienţii trebuie determinat profilul lipidic, de preferat în ziua internării. Cei cu nivele crescute ale lipidelor vor primi iniţial sfaturi despre regimul dietetic. Dacă prin dietă nu scade nivelul seric al lipidelor, se vor administra hipolipemiante, conform recomandărilor Societăţii Europene de Cardiologie.*

***Medicii de familie***

*Când sunt primii care vin în contact cu un caz suspectat de infarct miocardic, ei trebuie să fie capabili să intervină imediat şi să îndrume pacientul spre un serviciu de urgenţă.*
*Dacă medicii de famile pot interveni rapid şi sunt antrenaţi şi echipaţi corespunzător, ei ar putea defibrila şi tromboliza pacienţii cu infarct miocardic.*
*Ei trebuie implicaţi în programul coordonat local privind tratarea urgenţelor cardiace.*
*Medicii de familie trebuie să contacteze pacienţii cât mai repede posibil după externarea din spital, să se asigure că programul de recuperare este bine organizat şi să supervizeze cele mai potrivite măsuri de prevenţie secundară.*

***Autorităţiile responsabile cu sănătatea publică***

*Autorităţiile responsabile cu sănătatea publică trebuie să încurajeze educarea publicului în tehnicile de bază ale resuscitării cardiopulmonare şi a personalului de pe ambulanţă cu privire la măsurile de baza şi avansate ale susţinerii funcţiilor vitale.*
*Ele trebuie să se asigure că pacienţii cu infarct miocardic acut sau stop cardiac beneficiază de un sistem operaţional, prin coordonarea activităţiilor serviciului de ambulanţă, medicilor de familie şi a personalului din spital.*
*Ele trebuie să se asigure că serviciile de urgenţă au protocoale corespunzătoare pentru un tratament prompt al pacienţilor suspectaţi de infarct miocardic acut şi că există personal calificat corespunzător disponibil 24 ore din 24.*
*Ele trebuie să asigure suficiente paturi pentru unităţile de terapie intensivă coronariană, pentru pacienţii cu infarct miocardic acut. Specialiştii în cardiologie trebuie să fie disponibili. Autorităţile responsabile cu sănătatea publică trebuie să asigure condiţii de recuperare a pacienţilor externaţi din spital după un infarct miocardic acut. Ele trebuie să asigure facilităţi în propriul spital sau regiune pentru investigaţii şi tratamente speciale, de care să beneficieze pacienţii cu infarct miocardic acut complicat, sau dacă nu există aceste dotări, trebuie să asigure o înţelegere cu un centru terţiar.*

***Procedura realizării ghidului***

*Managementul infarctului miocardic acut a fost creat de Comitetul pentru Iniţiativele Ştiinţifice şi Clinice al Societăţii Europene de Cardiologie în 1999. Fiecare membru a fost invitat să relizeze schiţe privind aria lui de experienţă şi a avut loc o primă discuţie la Bruxelles pe 3 iunie 2000. După câteva revizuiri, membrii s-au reîntâlnit la Amsterdam pe 30 august 2000, apoi la Stockholm pe 5 septembrie 2001. Contribuţii suplimentare s-au obtinut de la K.Malberg, H. Heidbuchel şi F.Rademakers. Documentul a circulat extensiv printre experţi şi a fost discutat deschis cu conducerea Societăţii Europene de Cardiologie şi cu reprezentanţii societăţilor naţionale la o întâlnire ce a avut loc la European Heart House pe 7-8 februarie 2002. Documentul final a fost trimis pentru aprobare la 20 iunie 2002 Comitetului pentru Ghiduri Practice (J.W.Deckers, G.De Backer, A. Parkhomenko, G. Mazzoto, W.Klein). Asistenţa de neprețuit în procesarea documentului a fost acordată de Ms R. Struyven. Ghidurile au fost realizate fără nici o implicare a industriei „* [1-ESC].

Am prezentat aceste recomandari pentru a arata complexitatea cerintelor care trebuie deservite printr-un astfel de sistem si totodata nivelele de responsabilitate si implicare din partea

„actorilor” (persoane fizice sau juridice) solicitati in asistenta operarii in spatiul real „healthcare”, cardiovascular in cazul modelului nostru pilot.
Aceste recomandari privind fluxul de prelucrare si analiza a datelor, respectiv de analiza a nivelului de implicare si comportament al actorilor implicati, au stat la baza definitivarii arhitecturii sistemului Cardionet si la stabilirea principalelor componente de prelucrare si de analiza a datelor medicale, pentru a acoperi cerintele sub-domeniului cardiovascular abordat prin proiect.

## 8.2Sistemului Național Informational de Sanatate –SNIS.

“…Divizia de Informații de Sănătate, Dovezi şi Comunicare a OMS/EURO a inițiat în 2003 unproiect care utilizează o metodologie unanim acceptată, pentru a evalua şi îmbunătăți SIS din Statele Membre. O mai bună înțelegere a sistemului va permite OMS şi altor agenții interguvernamentale, să identifice nevoia Statelor Membre de sprijin în domeniul informațiilor de sănătate şi de informație privind cele mai bune soluții de îmbunătățirea calității şi relevanței datelor pe care le produc.
**Obiectivele** proiectului sunt:
1. Descrierea sistemului național de informații de sănătate, inclusiv fluxurile de date, legăturile dintre diferitele instituții care se ocupă cu date şi informații de sănătate cât şi legislația din domeniu;
2. Definirea utilizatorilor de informații de sănătate, curenți şi potențiali, cât şi a modalităților de diseminare a informațiilor de sănătate către aceşti utilizatori într-o manieră relevantă;
3. Identificarea lipsurilor şi problemelor (duplicări, întârzieri, etc.) în fazele de producere, validare, analiză, interpretare şi/sau diseminare, având în vedere atât nivelul național cât şi cel județean;
4. Recomandarea soluțiilor de îmbunătățire a sistemului național informațional de sănătate, cât şi a rețelei de instituții de sănătate publică şi decidenți din domeniul sănătății interesați de informațiile de sănătate…” [2]

Actorii principali identificati prin acest studiu OMS/EURO si structura sistemului de sanatate din Romania sunt prezentate mai jos, in figura VIII.1:

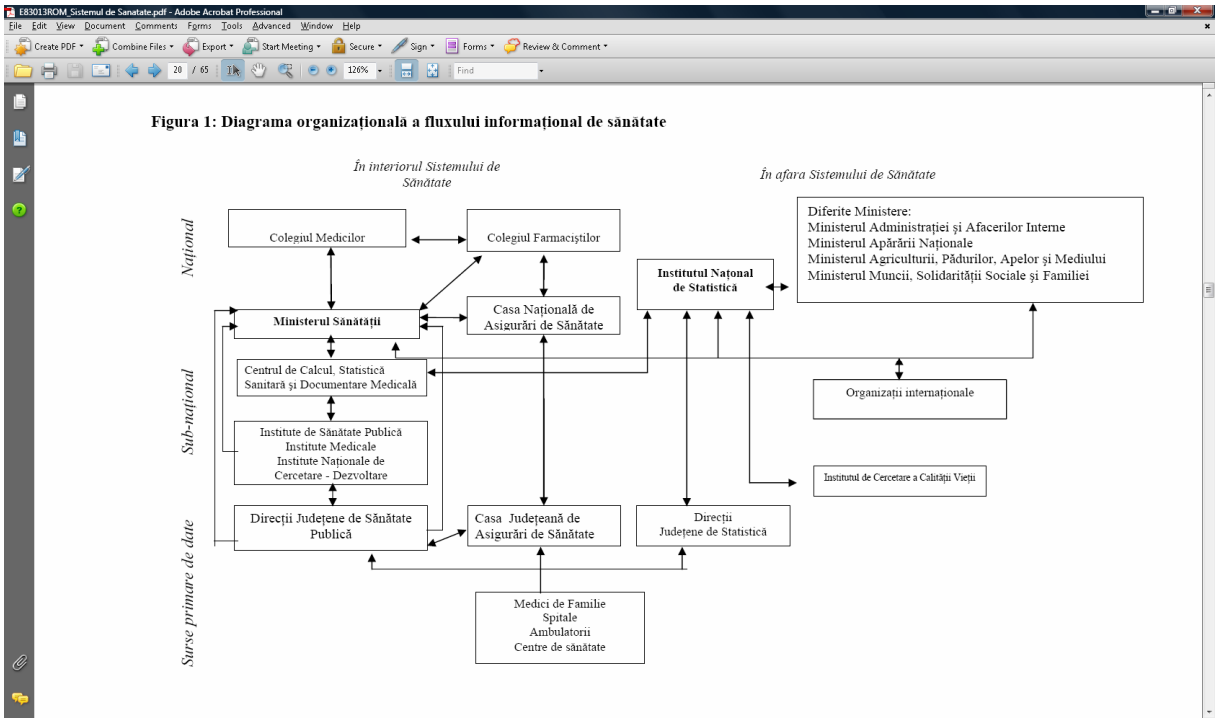



Figura VIII.1. Actorii si structura sistemului de sanatate din Romania @- [2-SNIS]

## 8.3Fluxuri de prelucrare si analiza a datelor specifice

Fluxurile generale de prelucrare intr-un sistem medical trebuie sa permita:
-colectarea informațiilor din foaia de observație clinică a pacientului internat, urmărindu-se în timp real traseul pacientului pe durata spitalizării acestuia în cadrul unui spital de cardiologie;
-gestiunea documentelor implicate în procesul de internare, ingrijire si externare al pacienților - elaborarea documentelor (format electronic si tiparit) la internare si externare respectând legislația şi formularele în vigoare;
-gestiunea inregistrarilor unice de internare şi externare la nivel de spital;
-analiza calității datelor electronice (prevalidarea lor înainte de raportare, pentru eliminarea costurilor de revenire si corectare a erorilor din documentația finala);
-analiza unor indicatori pentru luarea unor decizii de management;
-inregistrarea tuturor dalelor necesare eliberării unui decont pe pacient;
-posibilitatea integrării datelor in alte sisteme folosind standarde de interoperabilitate bazate pe XML;
-asigurarea sigurantei si securitatii datelor prin politici adecvate.

Fluxul informational principal in sistemul de sanatate din Romania, a fost si el identificat tot prin studiul OMS/EURO din 2003, si este prezentat mai jos, in figura VIII.2:

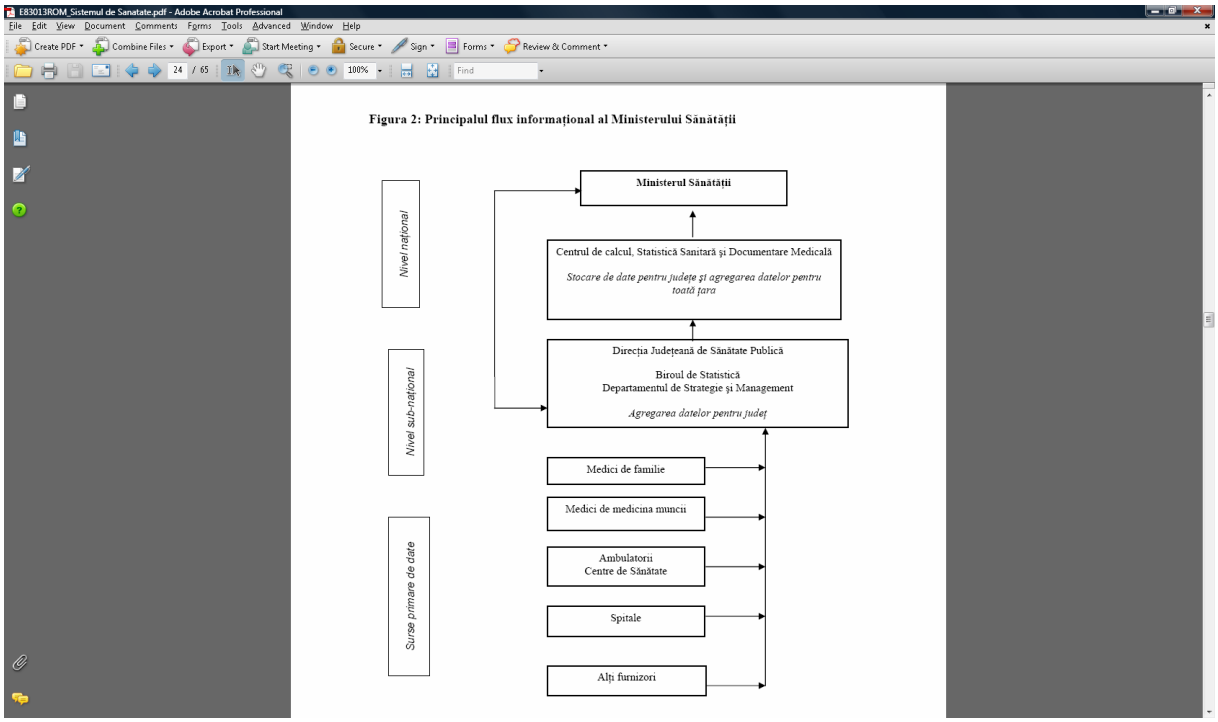



Figura VIII.2. Fluxul de date principal reglementat de MSP @-[2-SNIS]

La proiectarea componentelor si a fluxurilor de prelucrare efectuate prin Cardionet, am tinut cont de acest cadru organizational si de cerintele legale de prelucrare si schimburi de informatii.

Pe langa investigatiile si fluxurile generale exista si o gamă specifica domeniului cardiovascular, lucru de care am tinut cont de asemenea.

Dintre caracteristicile importante pentru cardiologie amintim:

-vârstă, sex (în general calculate automat din CNP), indice de masa corporală, circumferința taliei;

-factorii de risc modificabili: statusul de fumător, HTA, DZ tip II, dislipidemie;

-investigații hemodinamice invazive şi minim invazive;

-investigații metabolice pentru determinarea markerilor riscului cardiometabolic;

-evaluarea calității vieții (metoda chestionarelor): scorul de anxietate-depresie;

(Exemple considerate: scala HAD, Duke Activity Status Index (DASI) - chestionar multidimensional).

Locul spitalului in acest flux informational, reglementat de MSP, identificat tot prin studiul OMS/EURO din 2003, este prezentat mai jos, in figura VIII.2:

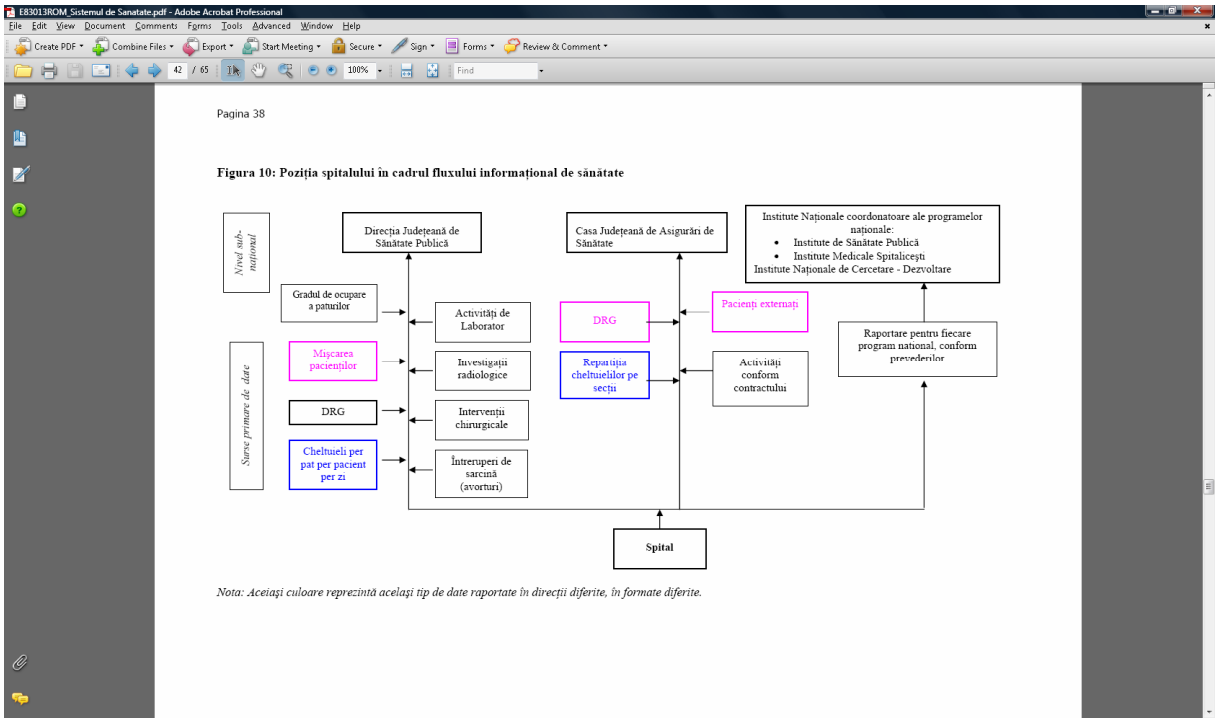



Figura VIII.3. Fluxul informational de sanatate-Pozitia spitalului @-[2-SNIS]

Idea de supraveghere continua in domeniul cardiovascular se coreleaza cu cea de flux informațional integrat (considerand ca starea unui pacient se poate schimba dintr-un moment în altul), cu condiția introducerii în timp real a datelor). Pentru a oferi o solutie la aceasta cerinta a fost necesară construirea infrastructurii Cardionet (inclusiv sistemul „Home Unit"). Structurile de date sunt în conformitate cu SMDPC (Setul minim de date la nivel de pacient impus de MSP), si acopera toate necesitatile specifice domeniului cardiovascular. De asemenea aceste structuri sunt flexibile si modulare si permit adaugarea cu usurinta noi tipuri de date specifice domeniului, care pot apare prin tehnologii sau metode noi de investigatie.
Avand in vedere importanta analizelor de laborator, vom prezenta in continuare cateva consideratii asupra culegerii/prelucrarii si analizei acestor tipuri de date.

## 8.4Considerații generale asupra analizelor de laborator si datelor medicale

### 8.4.1.Obiectivele analizei datelor

Presupunem că datele noastre sunt generate de observații supra unor subiecți umani. Fiecare subiect este investigat prin evaluarea unor caracteristici predefinite ținând cont de obiectivele studiului. Statistica are ca obiect studiul unui număr de subiecți de același fel în vederea stabilirii unor judecăți științifice, economice, etc. Această mulțime de subiecți asupra cărora se vor răsfrânge judecățile noastre se numeşte populație statistică sau pe scurt populație.

Numărul subiecților care compun populația statistică se numeşte volumul populației sau talia populației.

Lucrul cu întreaga populație care ne-ar interesa sau pe care dorim să o evaluăm este de multe ori dificil sau imposibil de efectuat sau pentru că populația este prea mare şi analiza completă ar lua prea mult timp sau prea multe resurse sau pentru că analiza presupune un anumit risc sau un anumit discomfort pentru pacienți. Din acest motiv statisticienii lucrează cu eşantioane care sunt subpopulații extrase aleator din populația originală astfel încât să se poată face studiul exhaustiv al subiecților de acolo.

Eşantionul trebuie să reproducă cât mai exact imaginea populației originale. Fără a intra în detalii principalele metode de eşantionare sunt:

- Eşantioane sistematice la care se ia în studiu tot al 2-lea, al 3-lea, …, al n-lea pacient
- Eşantioane stratificate cu straturi ca de exemplu copii, adulți, vârstnici iar în fiecare strat se face o selecție aleatoare. Desigur mulțimea straturilor şi tipul acestora sunt alese ținând cont de fenomenul investigat.

Deşi vom reveni cu o definiție şi proprietăți mai amănunțite reamintim că o variabilă aleatoare este un ansamblu format din mulțimea valorilor posibile pe care le poate lua o anumită caracteristică împreună cu probabilitățile de apariție a acestor valorirelativ la populația dată. Dacă variabilele studiului sunt cunoscute, atunci datele pacientului se obțin precizând valorile fiecărei variabile. Datorită acestei legături datele vor prelua toate proprietățile variabilelor.

Obiectivul general al unui studiu statistic asupra unei populații este de a cunoaşte variabilele care o definesc împreună cu relațiile care există între variabile. Relațiile între variabile vor fi subînțelese cu atributul statistic şi fac în general obiectul unei modelări elaborate de către statistician împreună cu specialiştii de domeniu.

În mod practic datorită efortului prea costisitor sau a inutilității nu se face o analiză exhaustivă. De obicei se analizează o variabilă aleatoare numită “end point” şi care este în acest mod un substitut sau o parte a obiectivului. De exemplu dacă dorim să analizăm starea de sănătate a populației județului atunci vom analiza variabilele aleatoare care ne indică prezența bolii pentru toate bolile cunoscute. Variabila aleatoare asociată unei boli va fi binară, adică are numai două posibilități: boală prezentă respectiv boală absentă.

Un alt exemplu poate fi chiar boala cardiovasculara cu dependențele acesteia de factorii de risc. În acest caz “enpointul” este tot binar boală prezentă ~ boală absentă. Analiza de această dată se face investigând toate corelațiile posibile cu variabilele aleatoare disponibile pentru toți subiecții. Desigur cum am mai amintit analiza se face de obicei pe eşantioane iar raționamenetele se extind apoi pe întreaga populație.

### 8.4.2. Elemente de teoria probabilităților aplicate datelor medicale

Teoria probabilităților şi întreaga statistică matematică se bazează pe noțiunea de experiment aleator care este privit privit ca un proces sau acțiune care are ca scop dobândirea unora sau mai multor rezultate noi ca urmare a aplicării unei mulțimi de proceduri specifice unui subiect. În cazul laboratorului de analize rezultatele se numesc probe. Vom extinde denumirea şi în alte situații fără alte precizări. Exemple de experimente ar putea fi:

- aruncarea unui zar şi notarea numărului
- aruncarea unei monede şi înregistrarea rezultatului
- dintr-o urnă cu bile albe şi negre extragem la întâmplare o bilă şi notăm culoarea
- determinarea grupei sanguine la o persoana aleasa la întâmplare din populația unui cartier
- înregistrarea momentului decesului la pacienții cu cancer gastric !

Pentru a respecta caracterul aleator după cum se observă şi din exemple va trebui ca:

- rezultatul experimentului sănu fie cunoscut apriori
- rezultatul se cunoaste numai după parcurgerea tuturor procedurilor associate experimentului
- modul de evaluare să nu influențeze rezultatul.

În mod practic analiza statistică cu metodele sale statistice studiază rezultatele unei mulțimi de experimente asupra unui eşantion.

Din definiția experimentului aleator se sunînțelege că sunt interesante numai experimentele care pot produce mai multe rezultate. Un experiment în care avem întotdeauna acelaşi rezultat este lipsit de logică. Prin eveniment vom înțelege un rezultat posibil al unui experiment.

Respecând notația statisticienilor evenimentele vor fi notate cu majuscule de la începutul alfabetului latin.

**Ex**. La aruncarea unui monedei avem două evenimente posibile notate cu A şi B
A= a ieşit "cap"
B= a ieşit "pajură".

Când spunem că "s-a realizat evenimentul A" aceasta este consecința faptului că urmare a desfăşurării procedurilor unui experiment s-a obținut rezultatul indicat de evenimentul A.

Proba prin perspectiva definiției evenimentului este realizarea practică a unui eveniment oricare ar fi rezultatul.

Probabilitatea sau şansa de apariție a unui eveniment este definită prin raportul dintre numărul de cazuri favorabile supra numărul de cazuri posibile. Dacă notăm cu cu

A un eveniment oarecare atunci simbolic definiția de mai sus se retranscrie ca

$$P(A) = \frac{\text{Numar de cazuri favorabile producerii evenimentului A}}{\text{Numar de cazuri posibile ale experimentului}}$$

**Ex.1.** La aruncarea zarului A= evenimentul de a obține cifra 2 sau 3

$$P(A) = \frac{2}{6} = \frac{1}{3} = 0.33 = 33\%$$

**Ex.2.** O urnă cu 6 bile albe şi 4 negre A= evenimentul de a obține bilă neagră are probabilitatea

$$P(A) = \frac{4}{10} = \frac{2}{5} = 0.40 = 40\%$$

**Notații consacrate**

1) Ø = evenimentul imposibil care nu se realizează niciodată. Evident P(Ø)=0.

2) Evenimentul total sau evenimentul sigur sau evenimentul cert se realizează când se realizează oricare dintre probele experimentului. Evident P(E)=1.

3) Evenimentul contrar unui eveniment A este evenimentul care se realizează când nu se relizează A. Se notează cu $\overline{A}$ şi avem $P(\overline{A}) = 1 - P(A)$. Desigur $P(E) = 1 - P(\varnothing)$ adică evenimentul total E şi evenimentul imposibil Ø sunt evenimente contrare.

**Ex**. La aruncarea zarului evenimentul de a ieşi 2, 4 sau 6 este contrar evenimetului de a ieşi 1, 3 sau 5.

De multe ori numărul de cazuri posibile este greu de evaluat şi atunci se evaluează un număr de evenimente stabilindu-se de câte ori a apărut evenimentul de interes. Numărul de apariții ale evenimentului, să zicem A împărțit la numărul total de evenimente analizate se numeşte frecvența. Se arată că frecvența tinde la probabilitatea evenimentului A când numărul de experimente tinde la infinit.

În figura VIII.4, de mai jos, avem o simulare pentru evenimentul A care constă în extragerea unei bile albe dintr-o urnă cu 40 bile albe şi 60 bile negre cu întoarcerea bilei. Se observă că frecvența pentru eşantioanele care sunt mai mari de 1000 de experimente frecvența este foarte aproape de 6/4 care etse probabilitatea teoretică conform definiției de mai sus. Pe această observație se bazează toată teoria estimației din statistica matematică.

Prevalența şi mortalitatea sunt exemple simple de aplicare a noțiunilor de probabilitate şi frecvență.

$$\text{Prevalența} = \frac{\text{Numãrul de cazuri dintr - un an}}{\text{Numãrul de locuitori la mijlocul anului adica la 1 august}}$$

$$\text{Mortalitatea} = \frac{\text{Numãrul de decese dintr - un an}}{\text{Numãrul de locuitori la mijlocul anului adica la 1 august}}$$

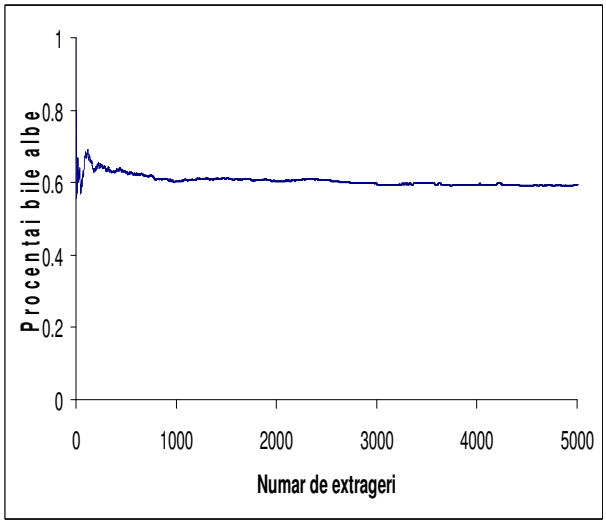



**Fig. VIII.4.** Frecvența extragerilor bilei albe în funcție de mărimea eşantionului.

**Observație**. Ratele nu sunt probabilități, ele sunt viteze de creştere care pot să se refere şi la probabilități (mortalitate, morbiditate, prevalență) ca de exemplu

$$\text{Rata îmbolnăvirilor prin cancer} = \frac{\text{Numărul de îmbolnăviri în cursul anului}}{\text{Numărul de cazuri aflate în evidenþă la începutul anului}}$$

După definiția evenimentelor şi exemplele arătate este evidentă legătura acestora cu teoria mulțimilor. Vom defini operațiile cu evenimente într-un mod simplu, similar cu operațiile pe mulțimi.

Introducem mai întâi noțiunea de incluziune.

Evenimentul A este inclus în evenimentul B şi notăm $A \subset B$ dacă realizarea evenimentului A diferit de B atrage sau implică realizarea evenimentului B. Mai spunem şi că B include A sau B este mai bogat decât A. Altă variantă notată $A \subseteq B$ şi citită "A este inclus sau egal cu B" este definiția de mai sus în care se şi posibila egalitate a lui A cu B.

**Ex.** La aruncarea zarului A= evenimentu că a ieşit unul din numerele {2, 3, 5} şi B= evenimentul că a ieşit unul din numerele {1, 2, 3, 5, 6}, avem $A \subset B$.

**Proprietăți**

1) $A \subset B$ implică $P(A) \subset P(B)$.
   $A \subseteq B$ implică $P(A) \subseteq P(B)$.

2) $\varnothing \subset A$ oricare ar fi evenimentul A.

3) $A \subset E$ oricare ar fi evenimentul A unde E este evenimentul cert.

Evenimentul A este eveniment elementar dacă nu este evenimentul imposibil şi nu poate fi descompus în sensul incluziunii.

**Ex.** La aruncarea zarului evenimentele $A_1$= a ieşit 1, $A_2$= a ieşit 2, $A_3$= a ieşit 3, $A_4$= a ieşit 4, $A_5$= a ieşit 5, $A_6$= a ieşit 6 sunt toate evenimente elementare. Evenimentul A la ieşirea unuia din numerele {1, 2, 3} nu este elementar deoarece $A_1 \subset A$, $A_2 \subset A$ şi $A_3 \subset A$ iar $A_1$, $A_2$ şi $A_3$ sunt diferite între ele.

Spaţiul fundamental al evenimentelor este prin definiţie mulţimea evenimentelor elementare asociate unui experiment.

**Proprietăţi**

1) Evenimentul cert E este format din mulţimea tuturor evenimentelor elementare.

2) Evenimentul vid $\varnothing$ este evenimentul care nu conţine nici un eveniment.

Intersecţia a două evenimente A, B este prin definiţie un eveniment C astfel că numai realizarea simultană a evenimentelor A şi B implică realizarea evenimentului C. În termenii teoriei mulţimilor şi cu ajutorul evenimentelor elementare mulţimea C este mulţimea evenimentelor elementare comune ale lui A şi B. Pentru interseţie avem aceeaşi notaţie ca pentru mulţimi: $A \cap B$ care se citeşte "intersecţia evenimentului A cu B" sau mai scurt "A intersectat cu B".

**Ex.** La experimentul cu aruncarea zarului dacă A este evenimentul de ieşi un număr din mulţimea {1, 2, 3} iar B este evenimentul de a ieşi un număr din mulţimea {2, 3} atunci evenimentul care ne arată intersecţia este evenimentul C care constă din ieşirea evenimentului {2, 3}.

**Proprietăţi**

1) $P(A \cap B) \leq P(A)$ şi $P(A \cap B) \leq P(B)$

2) $A \cap \varnothing = \varnothing$

3) $A \cap E = A$

Evenimentel A şi B sunt incompatibile dacă realizarea evenimentului A implică nerealizarea evenimentului B şi invers sau în termeni de teoria mulţimilor evenimentelor elementare dacă mulţimea evenimentelor comune lui A şi B este vidă sau simbolic $A \cap B = \varnothing$.

**Ex.**

1) La zaruri dacă A este evenimentul realizat când obţinem 1 sau 2 iar B este evenimentul de a ieşi 4.

2) La zaruri evenimentul A realizat când obţinem 1, 2 sau 3 iar nu este incompatibil cu evenimentul B de a ieşi 2, 5 sau 6 pentru că $A \cap B$ este evenimentul de a ieşi numărul 2 care nu este evenimentul incompatibil $\varnothing$.

Reuniunea evenimentelor A şi B notată cu $A \cup B$ este un eveniment C care se realizează când cel puţin unul dintre evenimentele A sau B s-a realizat. O definiţie echivalenă şi mai scurtă

spune că reuniunea evenimentelor A şi B este evenimentul format din mulțimea evenimentelor commune şi necomune celor două evenimente

**Ex.**
La aruncarea zarurilor. A={1, 2, 3}, B={2, 3, 6} => $A \cup B$={1, 2, 3, 6}.

**Proprietăți**

1) P(A $\cup$ B)≥P(A) şi P(A $\cup$ B)≥P(B).

2) P(A $\cup$ B)≤P(A)+P(B).

3) A, B incompatibile => P(A $\cup$ B)=P(A)+P(B).

4) P(A $\cup$ B)=P(A)+P ( B ) – P ( A $\cup$ B ).

5) A $\cup \varnothing$ =A

6) A $\cup$ E=E

Diferența evenimentelor A şi B notată A-B este un eveniment C care se realizează când A este realizat dar B nu s-a realizat sau echivalentul definiției făcând apel la teoria mulțimilor spune că diferența este mulțimea evenimentelor elementare ale lui A din care au fost înlăturate evenimentele lui B.

**Ex.**
La aruncarea zarurilor. A={1, 2, 3}, B={2, 3, 6} => A-B={1}.

**Proprietăți**

1) E - A = $\bar{A}$

2) A - $\varnothing$ = A

3) A - B = A –(A $\cap$ B)

Evenimentul A condiționat de B notat A | B este un eveniment C care se realizează când A este realizat dacă în prealabil s-a realizat B sau folosind noțiuni de teoria mulțimilor A | B este mulțimea evenimentelor elementare comune lui A şi B când a fost realizat B.

**Proprietăți**

$$P(A|B) = \frac{\text{Numar de cazuri favorabile lui } A|B}{\text{Numar de cazuri posibile pentru } A|B} =$$

$$\frac{\text{Numar de cazuri de realizare a lui A daca s-a realizat B}}{\text{Numar de cazuri favorabile lui B}} =$$

$$\frac{\text{Numar de cazuri de realizare a lui A} \cap \text{B}}{\text{Numar de cazuri favorabile lui B}} =$$

$$\frac{\frac{\text{Numar de cazuri de realizare a lui A} \cap \text{B}}{\text{Numar de cazuri totale}}}{\frac{\text{Numar de cazuri favorabile lui B}}{\text{Numar de cazuri totale}}} = \frac{P(A \cap B)}{P(B)}$$

Evenimentul A este independent de B prin definiție dacă realizarea lui A nu depinde în nici un fel de realizarea lui B.

**Observații**

1) Evenimentele incompatibile sunt în general diferit de independente.

2) Evenimentele incompatibile au acelaşi spațiu de evenimente iar intersecția este mulțimea vidă ($\varnothing$).

3) La evenimentele independente spațiul de evenimente este diferit.

**Ex.** Dacă avem două zaruri atunci evenimentul că obținem 1 la primul zar este incompatibil cu cel de a obține 2 tot la primul zar, dar evenimentele de a obține 1 la primul zar şi 2 la al doilea sunt independete.

$A \mid B = A$ în acest caz şi atunci $P(A \cap B)=P(A)P(B)$.

### 8.4.2.1. Aplicații imediate ale teoriei probabilităților: Specificitate, sensibilitate

Pentru a face expunerea mai cursivă introducem aici câteva noțiuni care pot fi deja definite cu noțiunile care le avem deja din teoria probabilităților urmând să revenim acolo unde este cazul cu completări.

Presupunem că avem o mulțime de pacienți cărora le cunoaştem exact starea din punctul de vedere al prezenței sau absenței unei boli precizate anterior. Pacienților li se aplică un test medical care sperăm să ne dea indicații cât mai exacte despre starea pacientului cu privire la absența sau prezența bolii precizate. Suntem interesați în găsirea unor metode de evaluare a testului.

Experimentul constă din analiza fiecărui pacient şi notarea statusului pacientului sub aspectul bolii şi al testului.

**Starea bolii pacientului** comportă două stări: boală prezentă şi boală absentă. Notăm cu B+ evenimentul care se obține când constatăm că pacientul are boala prezentă iar cu B- evenimentul contrar adică pacientul nu are boala sau boala este absentă.

2) **Starea testelor pacientului** este asemănătoare stării bolii dar vom spune că avem un test pozitiv dacă rezultatul testului de indică o posibilă boală şi vom spune că avem un test negativ dacă rezultatul testului de indică posibila absență a bolii. Notăm cu T+ evenimentul că în urma

testului medical pacientul a fost testat iar testul a fost pozitiv iar cu T- evenimentul contrar adică testul a fost negativ.

În limbajul abstract al teoriei probabilităților putem asimila cele de mai sus cu imaginea unei urne din care extragem bile de patru culori. Cele patru culori sau categorii sunt cele patru variante pe care la întâlnim la bolnav:

1. T+ şi B+ când bolnavul a fost testat pozitiv şi boala este prezentă
3. T- şi B+ când bolnavul a fost testat negativ şi boala este prezentă
2. T+ şi B- când bolnavul a fost testat pozitiv şi boala este absentă
4. T- şi B- când bolnavul a fost testat negativ şi boala este absentă

Datele descrise mai sus pot fi sintetizate într-un tabel cum este cel din exemplul de mai jos

| | | **Stare subiecți** | | **Total** |
|---|---|---|---|---|
| | | **B+** | **B-** | |
| **Test** | **T+** | 800 | 100 | 900 |
| | **T-** | 200 | 300 | 500 |
| | **Total** | 1000 | 400 | 1400 |

Sunt 1400 de pacienți iar după investigarea lor avem rezultatele din acest tabel. Avem 1000 pacienți care au boala şi 400 care nu au boala. Din cei 1000 cu boala prezentă 800 au avut test pozitiv şi 200 test negativ iar din cei cu boala absentă 100 au avut test pozitiv şi 300 test negativ.

Făcând apel la definiția clasică a probabilităților ca număr de cazuri favorabile împărțit la numărul de cazuri posibile calculăm probabilitatea ca testul să fie pozitiv P(T+) ca raportul 900/1400= 0.64.

Variante globale posibile de probabilități calculate relativ la întregul lot de pacienți sunt

$$P(T+)=900/1400=0.64$$

$$P(T-)=500/1400=0.36$$

$$P(B+)=1000/1400=0.71$$

$$P(B-)=400/1400=0.29$$

Iar dacă efectuăm intersecții de evenimente avem

$$P(B- \cap T-)=300/1400=0.21$$

$$P(B- \cap T+)=100/1400=0.07$$

$$P(B+ \cap T-)=200/1400=0.14$$

$$P(B+ \cap T+)=800/1400=0.57$$

Un loc special în aceste variante de calcul al probabilităților îl ocupă sensibiltatea care se defineşte ca probabilitatea de a obține un test pozitiv când boala este prezentă. Mai este cunoscută şi sub numele de adevărat pozitiv deoareceîn acest caz testul este pozitiv iar boala este prezentă. Cu notațiile definite mai sus sensibilitatea poate fi notată cu $P(T+ \mid B+)$ şi pentru exemplul nostru

$$P(T+ \mid B+)=P(B+ \cap T+)/P(B+)=800/1000=0.80$$

Al doilea calcul cu un loc special este specificitatea definită ca probabilitatea de a obține un test negativ când boala este absentă. La fel ca mai sus se mai numeşte şi adevărat negativ şi este notată cu $P(T- \mid B-)$. Pentru exemplul nostru

$$P(T- \mid B-)=P(B- \cap T-)/P(B+)=300/400=0.75$$

**Observație**

De multe ori este mai convenabil să calculăm probabilitățile complementare adică fals pozitiv $P(T- \mid B+)$ pentru sensibilitate şi fals negativ $P(T+ \mid B-)$ pentru specificitate. În aceste situații este bine de reținut proprietățile:

$$P(T+ \mid B+)+P(T- \mid B+)=1 \text{ şi } P(T- \mid B-)+P(T+ \mid B-)=1.$$

Metodele de analiza specializate pentru Cardionet vor fi dezvoltate in continuare si exemplificate pe datele culese in etapele urmatoare IV si V ale proiectului.

### 8.4.3. Tipuri de date si teste de laborator

In limbajul medical prin date de laborator vom înțelege conform manualelor de teste de laborator orice examinare efectuată cu scop diagnostic pornind de la teste cum ar fi: probele de sânge, urină sau alte fluide prelevate şi sfârşind cu examinările de computer tomograf sau RMN, cu EKG-urile de specialitate, etc. Practic in aceste cazuri are loc unul sau mai multe schimburi de informatii intre medicul curant si medicul/medicii de laborator

De asemenea trebuie mentionat si rolul medicului de familie care trebuie să însoțească şi consilieze orice modificare în starea noastră de sanatate şi care trebuie să țină la zi întreaga arhivă medicală a fiecăruia.

Baza de date de laborator va fi un instrument la indemana utilizatorilor menționați mai sus: pacient, medic de familie, medic curant.

După analiza manualelor de teste diagnostice şi de laborator, testele se pot clasifica în primul rând după specificul acțiunilor medicale în:

1. Probe de sânge
2. Probe de urină
3. Probe fecale
4. Probe de lichid cerebrospinal

5. Chimie
6. Microbiologie
7. Imunologie
8. Radionuclizi
9. Radiografie
10. Citogenetică
11. Endoscopie
12. Ecografie
13. Funcții pulmonare; Gaze sanguine
14. Probe funcționale pe organe şi sisteme
15. Fluide amniotice

La rândul său orice test se clasifică generic în funcție de rezultatul pe care îl conține în două mari categorii: normal şi patologic care rezultă la rândul lor dintr-o serie de atribute care le posedă fiecare test în parte. În cele ce urmează prezentăm o descriere formalizată, generală orientată spre o prelucrare informatică urmând ca toate cataloagele şi anexele necesare funcționării laboratorului sau laboratoarelor să fie prezentate într-o etapă ulterioară.

Din punctul de vedere al naturii informației date de un test de laborator rezultatele se clasifică în trei tipuri: numerice, categoriale şi de tip text sau redacționale. Detaliem în continuare din punct de vedere semantic aceste tipuri.

Dacă **măsurătoarea este numerică** avem următoarele situații pentru valorile considerate normale ale rezultatelor:

1. **Min** dacă valorile normale sunt considerate în momentul în care valoarea măsurată se situează sub o anumită limită bine precizată.
2. **Max** dacă valorile normale sunt considerate în momentul în care valoarea măsurată se situează peste o anumită limită bine precizată.
3. **Min-Max** dacă valorile normale sunt considerate în momentul în care valoarea măsurată se situează sub între două limite bine precizate.

Pe scurt vom spune despre aceste teste că sunt de tip **Min**, **Max** sau **Min-Max**.

Dacă **valoarea testului este categorială** avem nevoie de lista totală a categoriilor precum şi de cea a categoriilor aşa zise normale. Prin omiterea categoriilor normale vom obține categoriile patologice. Opțional se poate completa un text ataşat cu o eventuală completare din partea medicului iar completarea rezultatului cu una dintre opțiuni este obligatorie. Vom numi un astfel de test de tip **Listă**.

În situația în care **rezultatul testului constă într-un test redacțional** atunci se impune obligativitatea completării acelui text. Opțional pe lângă textul redacțional care constituie corpul rezultatului este bine de considerat şi o listă cu nişte categorii generale la fel ca mai sus la tipul Listă pentru toate posibilitățile împreună cu mulțimea categoriilor normale. Numim un test cu un astfel de conținut test de tip **Text**.

### 8.4.3.1.Atribute biologice folosite în elaborarea rezultatului unui test

Starea de normalitate suferă modificări importante care țin de statusul în care se află subiecul. Am identificat până acum patru posibili parametri:

1. sex în testele pentru hemoglobină, hematocrit, nivelul hormonilor estrogenici şi progesteronici, etc.
2. vârstă sau categorii de vârstă în testele reticulocite (reticulocyte count), colesterol
3. ora recoltării găsită ca momente ale zilei în diferite teste de urină
4. data calendaristică apărută ca anotimp în testele vitamina D şi metaboliți

Dacă pentru **sex** care este o variabilă binară un există nici o problemă în codificare restul sunt variabile de tip interval.

Pentru **vârstă** sunt de multe ori diferite sisteme de apreciere pentru adulți şi copii. În acest caz am considerat intervalul [0,18) ani pentru copii şi [18,999) pentru adulți. Am folosit întotdeauna convenția ca intervalul să fie închis la stânga şi deschis la dreapta adică limita stângă a intervalului să fie considerată iar limita dreaptă să fie omisă. Trebuie făcute precizări pentru termenii comuni din manuale mai puțin exacți cum ar fi: nou născuți, tineri, vârstnici, etc. dar cu siguranță ei pot fi asociați unui interval bine definit.

**Ora recoltării** este formată de asemenea din subintervale ale momentelor zilei adică ale intervalului [00:00,24:00). Şi aici trebuiesc făcute asocieri corecte între indicația de manual pentru denumirile comune cum ar fi dimineță, după masă, seara, etc. şi cele numerice date în limitele intervalului. Sunt posibile corecții datorită legăturii între momentele zilei şi data calendaristică.

**Data calendaristică** a recoltării am utilizat-o pentru a cuantifica ideea de anotimp sau perioadă a anului apărută la anumite teste în aprecierea normalității. Pentru a uniformiza am considerat drept model anul 2008 (anul curent) care este bisect şi acesta a fost divizat în intervale. De exemplu pentru primăvară putem folosi primăvara astronomică adică perioada din 21.03.2008 până în 21.06.2008 desigur dacă aceasta se suprapune peste manualul de laborator.

### 8.4.3.2.Atribute tehnice folosite la interpretarea testelor de laborator

Deşi ar trebui să nu aibă nici un impact, în practică există o variabilitate destul de mare. Aceasta este generată în primul rând de diversele **metode de determinare** pentru acelaşi test. De exemplu testul rata de sedimentare a eritrocitelor are cinci metode de efectuare: Westegren, Cutler, Landon Adams, Wintrobe, Smith. Fiecare metodă presupune comportamenet diferit la execuție, interpretare, preț, cost, etc. Se impune crearea unui catalog al metodelor chiar dacă la majoritatea se impune atribuirea unei valori care să indice faptul că metoda este unică.

Al doilea element de varianță se referă la **aparatul cu care se execută testul**. Deşi aparatele sunt în general standardizate este bine de gestionat un catalog chiar dacă va fi completat lacunar.

Desigur apar şi alte elemente care sporesc sau diminuează gradul de incertitudine şi țin de **încrederea clinicianului într-un anumit laborator, locație sau medic de laborator**. De aici rezultă obligativitatea completării pe orice rezultat a medicului care supervizează şi a laboratorului unde se lucrează împreună cu eventualele locații sau puncte de lucru. Din aceasta rezultă de asemenea necesitatea unor cataloage care să gestioneze laboratoarele împreună cu locațiile şi medicii.

### 8.4.3.3.Atribute economice folosite la interpretarea testelor de laborator

Atributele de natură economică se referă la preţurile şi sistemele de codificare DRG sau OMS. Dacă sistemele de codificare un prezintă nici o dificultate întrucât se desfăşoară după reguli precise, preţul face parte din opţiunile bolnavului şi trebuiesc evidenţiate. Pentru exactitudine înregistrarea preţului este imperativă.

## 8.5. Descrierea funcţională a sistemelor de baze de date de laborator

Bazele de date medicale este de dorit să funcţioneze cu principiul de a un fi eliminat nimic din ele pentru a putea reface istoricul oricărei modificări. Fiecare articol din orice tabel sau bază de date are mai multe stări care reflectă statusul informaţiei pe care îl conţine. Deşi articolele, în cazul general, pot avea mai multe stări şi sunt dependente de semantica informaţiei pe care o conţine am identicat trei stări comune pentru orice articol din bazele de date sau tabele: editare, validare şi anulare.

Pentru editare, validare şi anulare orice articol conţine elemente care ne dau data şi ora la care s-a efectuat trecerea în starea respectivă împreună cu identificatorul utilizatorului care este responsabil de acea tranziţie. Avem aşadar în interiorul oricărui articol trei blocuri de informaţie descriptive auxiliare:

1. data şi ora editării articolului împreună cu identificatorul userului care lansează editarea
2. data şi ora validării articolului împreună cu identificatorul userului care a validat articolul
3. data şi ora anulării articolului împreună cu identificatorul userului care a anulat articolul.

Când un articol nou este creat acesta se crează cu statutul de editare. Data şi ora creării împreună cu identificatoul utilizatorului responsabil sunt înregistrate automat.

Când articolul după completare corespunde cerinţelor se poate lansa validarea informaţiei. Se face mai întâi o verificare a conţinutului articolului iar dacă aceasta este trecută se consemnează validarea adică se schimbă statusul din editare în validare împreună cu data şi ora validării împreună cu identificatorul userului.

Modificarea unui articol validat se face prin înregistrarea articolului respectiv ca anulat, după care se face duplicarea acestui articol şi trecerea lui în starea de editare împreună cu data şi ora efectuării împreună cu identificatorul userului care este responsabil pentru modificare.

Dacă dorim să pornim de la un articol anulat şi eventual să modificăm conţinutul şi să-l validăm vom efectua o duplicare a articolului după care la această copie o să în registrăm parametrii de editare adică data şi ora şi identificatorul userului.

Notând cu E editare, V validare şi A anulare atunci folosind un limbaj formal de tip Gray la care numai ultimul cuvânt se poate retranscrie. Gramatica dată prin reguli prestabilite
generează toate posibilităţile. Cuvintele generate de această gramatică sunt de forma $A^0V$, $A^0E$ sau $A^+$. Unde $A^0$ are semnificaţia unui cuvânt vid sau cuvânt format numai cu simbolul A care poată să apară de oricâte ori iar $A^+$ are semnificaţia unui cuvânt care apare de oricâte ori. Cu alte cuvinte un articol poate să fie în starea editare, validare sau anulare iar stările precedente dacă există trebuie să fie toate de anulare.

Datele accesibile medicului clinician în vederea tratării sau urmăririi pacientului trebuie să se regăsească într-o dază de date care să aibă cerinţele enunţate în punctele de mai sus.

Această bază de date, gândită ca o arhivă, permite fiecărui pacient să şi consulte datele personale şi rezultatele testelor proprii iar medicii de familie şi medicii specialişti pot consulta şi modifica datele atâta timp cât administratorul le permite acest lucru.

Alimentarea bazei de date este făcută direct de către laborator printr-un protocol care va fi stabilit de común acord intre parteneri. Dacă acest lucru nu este posibil atunci sub îndrumarea medicului specialist sau a medicului de familie datele vor trebui să ajungă în baza de date de pe suportul convențional furnizat de către laborator (hârtie, film, etc.).

Administratorul bazei de date gestionează în primul rând utilizatorii. Aceştia au fost menționați mai devreme: pacient, medic de familie, medic specialist şi secretară medicală. Pacientul are numai drepturi de citire ale bazei de date în schimb restul actorilor secretară, medic de familie sau medic specialist au şi dreptul de scriere şi modificare. Catalogul utilizatorilor conține în primul rând codul numeric personal, datele personale adresă, telefon, etc. şi modul de acces al bazei de date.

Grija exclusivă a administratorului este actualizarea cataloagelor. Am identificat următoarele cataloage:

1. catalog al unităților de măsura
2. catalog al aparatelor
3. catalog al laboratoarelor
4. ctalog al locațiilor punctelor de lucru ale laboratoarelor
5. catalog al medicilor de laborator
6. catalog al examinarilor de unde se ia proba (ex. sânge, urină)
7. catalog al analizelor de laborator cu indicarea valorilor normale

Baza de date care conține rezultatele testelor de laborator conține informația despre modul cum s-a efectuat testul, valorile normale ale testului şi valoarea testului. Concluzionând o bază de date de laborator conține:

A. Date de identificare a pacientului
    1. Codul numeric personal din care rezultă sexul şi vârsta
B. Date despre modul în care s-a efectuat prelevarea probelor
    1. Data prelevării
    2. Ora prelevării
    3. Cine a efectuat prelevarea
C. Date despre valorile normale şi patologice ale testului
    1. Grupa de vârstă la care se referă proba din catalogul analizelor
    2. Perioada zilei la care se referă proba din catalogul analizelor
    3. Perioada anului la care se referă testul
    4. Unitatea de măsură
    5. Tipul testului (Min, Max, Min-Max, Lista valori, Text)
    6. Numele testului
    7. Metoda utilizată
    8. Limita inferioară
    9. limita superioară
    10. Lista totală pentru rezultate categoriale
    11. Lista valorilor normale pentru rezultate categoriale
D. Date administrative privind desfăşurarea efectivă a testului
    1. Laboratorul sau indexul din catalogul laboratoarelor
    2. Locația unde se efectuează testul
    3. Medicul în a cărui responsabilitate se va trece testul
E. Valorile propriuzise ale testului aşa cum sunt ele cerute în manuale: valori numerice, categoriale, text sau de tip container cum ar fi imagini de radiografii, EKG, etc.

F. O zonă de observații constând dintr-un text liber în care medicul de laborator poate să expună eventuale completări, abateri, indicații speciale, et.

Majoritatea cataloagelor sunt simple. Ele sunt compuse dintr-un nume pentru fiecare entitate inclusă, un index dat automat de către calculator şi eventual alte informații. În figura de mei jos prezentăm o secvență din secțiunea de actualizare a catalogului cu unități de măsură. Se observă o serie de informații generale grupae. Primul grup este numărul articolului curent şi numărul total de articole împreună cu statusul acestui articol. La mijloc avem informația utilă adică indexul atribuit automatic şi denumirea unității de măsură după care avem informații despre crearea, validare şi eventual anularea acestei poziții. Statusul în care se află utilizatorul la momentul respectiv: răsfoire, căutare, modificare şi în partea de jos o casetă specială de mesaje date utilizatorului pe durata activității sale. Funcțiile de bază ale intervențiilor în cataloage sunt exemplificate în figura VIII.5. Putem căuta sau genera cereri, modifica, anula sau crea o poziție nouă.

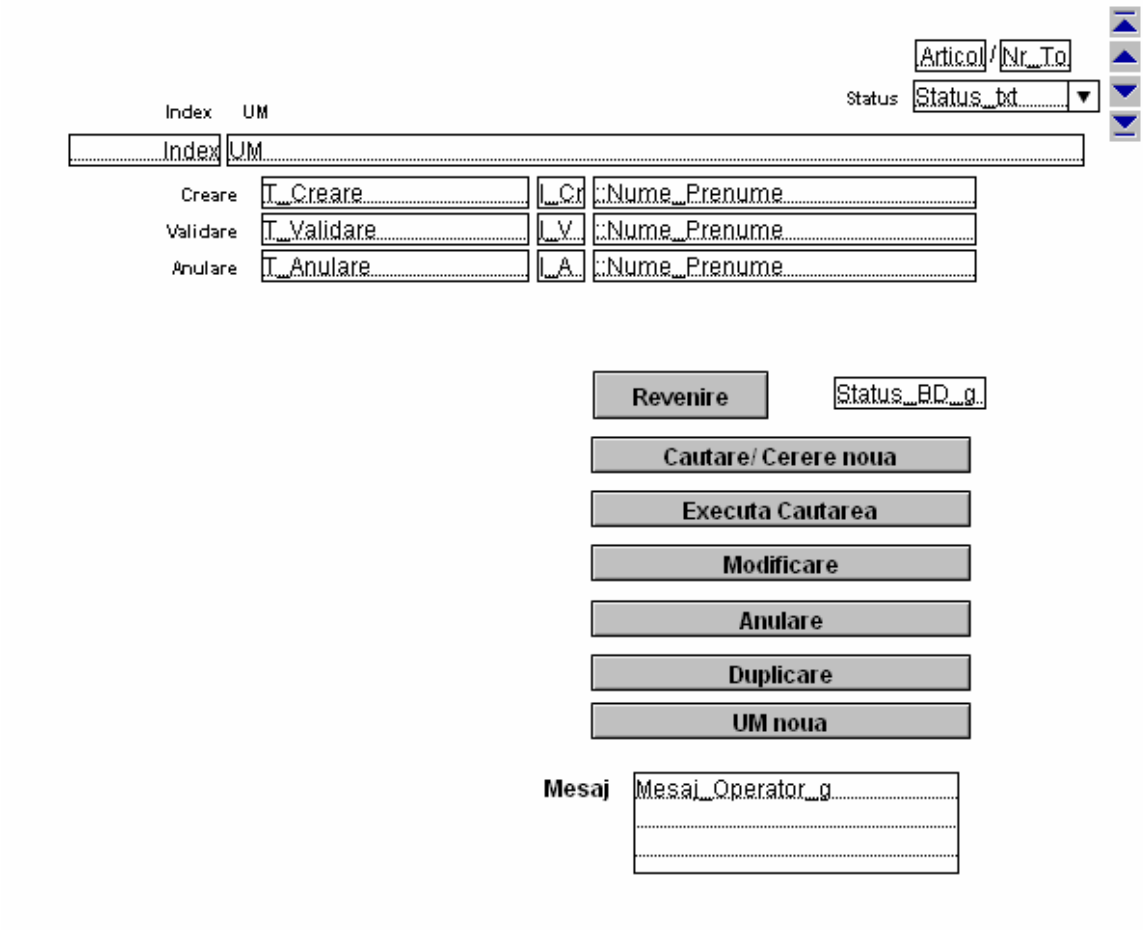



Fig. VIII.5. Cataloage – Functii de baza.

Cataloagele care se referă la laborator şi aparate sunt identice cu catalogul prezentat mai sus dar cu deosebirea că avem altă semnatică. Catalogul cu locul de unde se ia proba este practic

identic cu împărțirea manualului de laborator pe capitole așa cum s-a arătat mai sus (sânge, urină, etc.).

Catalogul cu analize CASA cuprinde nomenclatorul cu analizele casei de asigurări de sănătate. Acest nomenclator poate să un fie identic cu cel al laboratorului şi conține denumirea oficială a casei împreună cu un cod atașat.

Catalogul locațiilor conține deocamdată numele locației şi numele laboratorului la care este subordonată ierarhic. Se mai pot adăuga aici informații administrative în viitor.

Catalogul medicilor de laborator este prezentat mai jos, în figura VIII.6.:

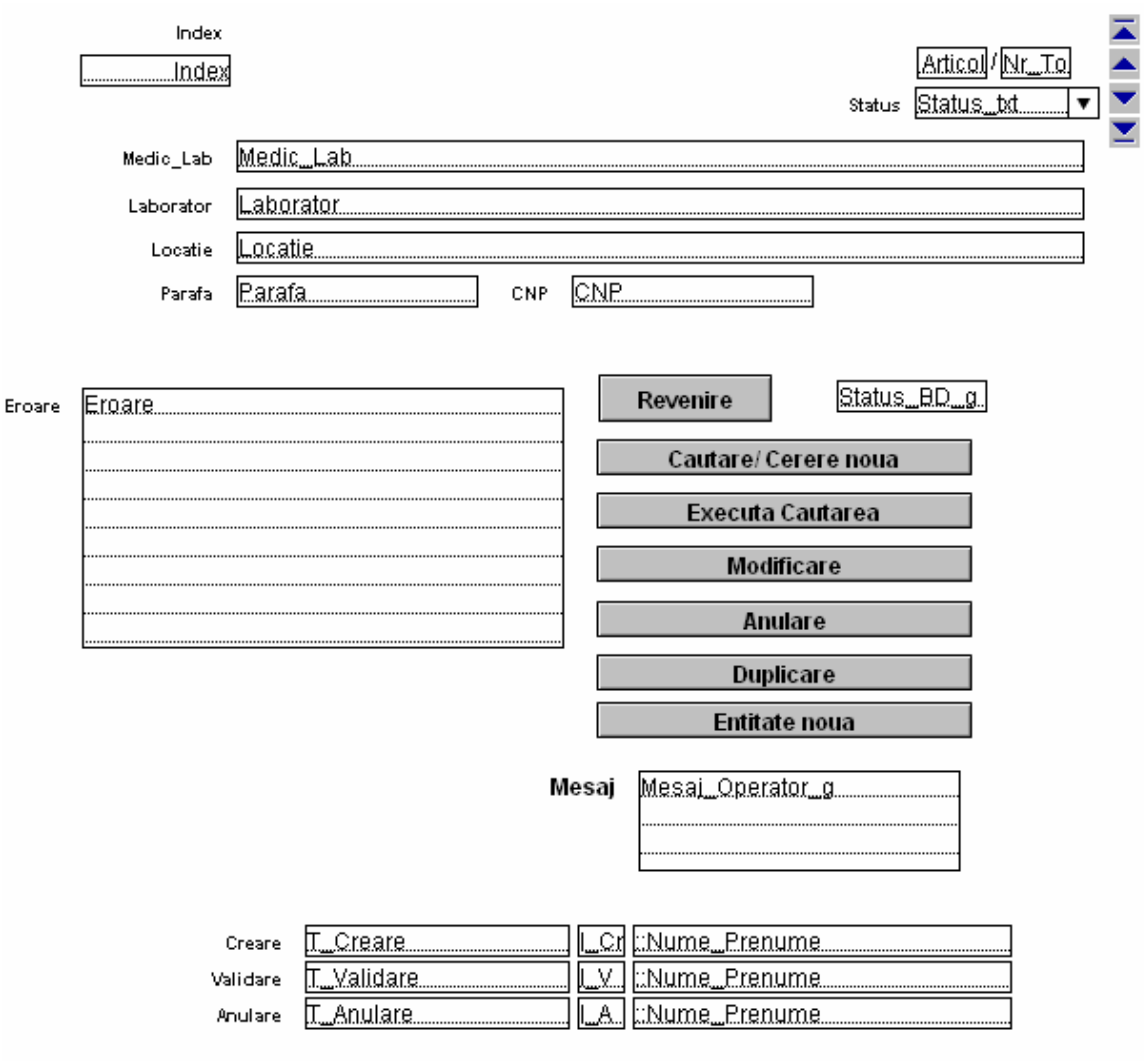



Fig. VIII.6. Catalogul medicilor de laborator

Pe lângă informațiile prezentate mai sus să observăm o detaliere a informațiilor legate de medic cum ar fi numele, laboratorul unde activează, locația unde activează, codul parafei şi CNP-ul pentru identificare.

Catalogul analizelor este cel mai complet. În el găsim toate analizele pe care le face un laborator sau grupul de laboratore asociat împreună cu descrierea acestora. Catalogul trebuie să conțină toate detaliile de execuție inclusiv valorile normale şi patologice iar clasificarea în analiză normală sau patologică să fie efectuată automat. În baza de date cu analizele reale de laborator solicitările şi valorile testelor vor trebui să fie conforme acestei descrieri. Figura VIII.7. conține descrierea acestui catalog exceptând informațiile administrative prezentate deja la celelalte cataloage.

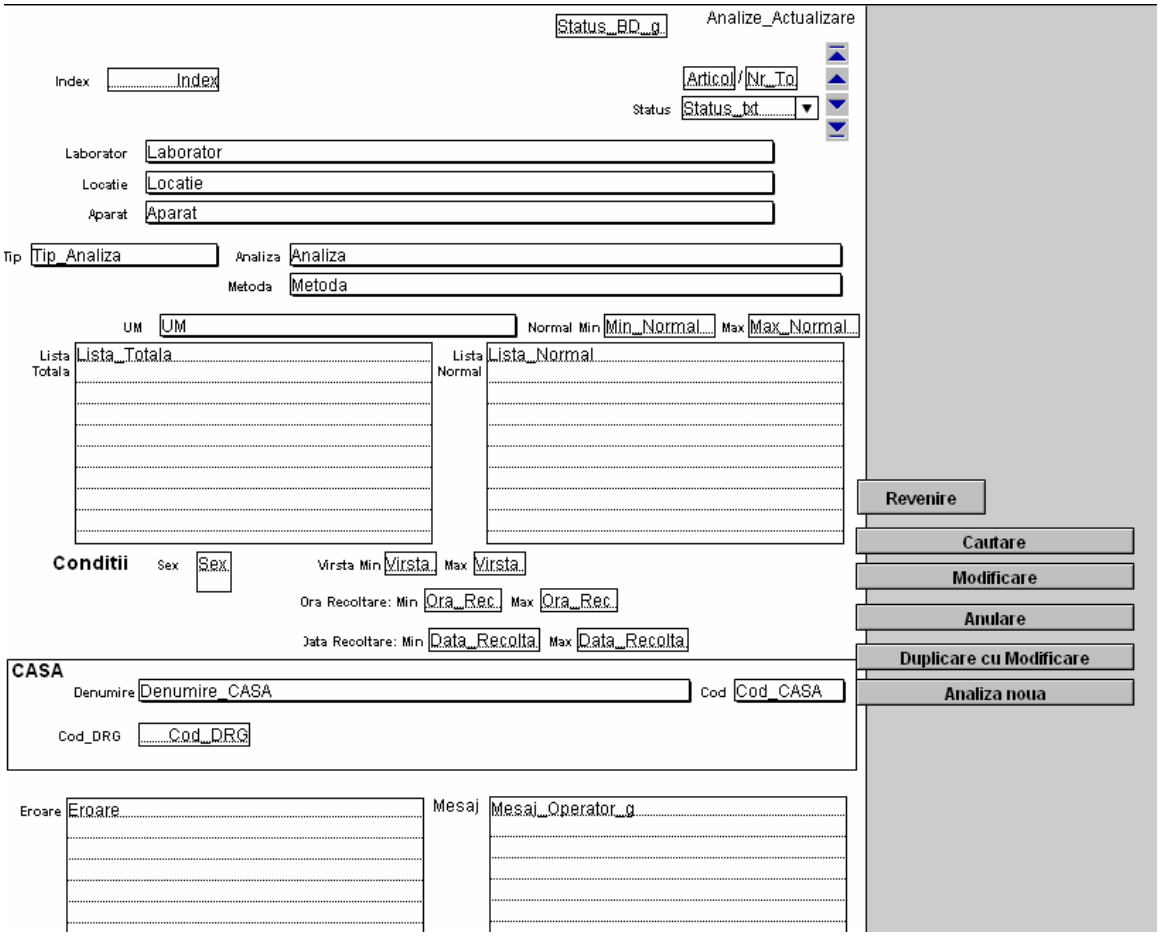


Fig. VIII.7. Catalogul analizelor

Analizele propriuzise conțin în plus față de figura de mai sus partea de identificare a bolnavului şi alte informații administrative. Pentru simplificare putem considera numai codul numeric personal şi medicul care face prescripția. Desigur, în mod practic un laborator trebuie să fie interesat şi de aspectele economice şi atunci implementarea bazei de date trebuie să fie adaptată cu informații despre decontul analizei, tipul de asigurare, etc. Aceste informații le vom adăuga în viitorul imediat chiar în situația când nu este important fie regăsite în ontologie şi să fie disponibile de exemplu clinicianului.

Dezvoltarea sistemului de  laborator va continua in etapele urmatoare, in paralel cu integrarea acestei componente in sistemul Cardionet.

**Referinte bibliografice:**
**Capitolul VIII:**

[ 1-ESC] *European Heart Journal (2003) 24, 28-66-* Managementul infarctului miocardic acut la pacienţii cu supradenivelare de segment ST - Frans Van de Werf, preşedinte, Diego Ardissino, Amadeo Betriu, Dennis V.Cokkinos, Erling Falk, Keith A.A. Fox, Desmond Julian, Maria Lengyel, Franz-Josef Neumann, Witold Ruzyllo, Christian Thygesen, S. Richard Uderwood, Alec Vahaian, Freek W.A. Verheugt, William Wijns; - *Traducere de: prof. dr. Doina Dimulescu, dr. Andreea Catarina Popescu, dr. Gianina Bercu, dr. Ionuţ Stancă, dr. Laura Aramă, dr. Teodora Avram, dr. Mihaela Ruxandra Popescu, dr. Sergiu Bârsan, dr. Daniel Cristea, dr. Mihai Şotcan.*

[2-SNIS] Description of the National Health Information System in Romania: Results of a participative evaluation conducted in June 2003. © World Health Organization 2004, Traducere: © Institutul de Sănătate Publică Bucureşti.

# 9 Concluzii

Prin proiectul CardioNet s-a inceput implementarea în cadrul etapei aIIIa, cu termen de predare 30 iunie 2009, unui sistem distribuit de urmărire, monitorizare şi inregistrare a pacienţilor cu afectiuni cardio-vasculare. Sistemul a fost conceput cu scopul de a facilita interacţiunea dintre pacienţi şi medici, utilizând în acest scop cele mai avansate modele şi tehnologii informatice şi de comunicaţie.

La elaborarea proiectului s-au luat în considerare rezultatele analizelor efectuate în etapele anteriare privind:

- soluţii şi sisteme similare de telemedicină şi telecardiologie
- cadrul legislativ şi practica medicală curentă din ţară
- standardele existente în domeniu.

Având în vedere comlexitatea sistemului medical ce trebuia să se modeleze, s-a decis adoptarea unei soluţii distribuite, în care mai multe aplicaţii autonome, implementate la nivelul diferitelor entităti medicale, cooperează în vederea îndeplinirii funcţiilor de inregistrare şi monitorizare a pacienţilor. Ca urmare a acestei abordări s-au implementat mai multe componente autonome, care împreună formează sistemul CardioNet.

Astfel, s-a implementat o aplicaţie de tip Medic de familie, ce oferă suport pentru interacţiunea dintre pacienţi şi medici de familie, atât în mod direct (la cabinetele medicale) cât şi indirect prin intermediul Internetului.

Cu ajutorul aplicaţiei pacientul poate să dialogheze cu medicul de familie prin intermediul Internetului; astfel consultaţiile de rutină pot fi efectuate de la domiciliul pacientului fără să fie necesară deplasarea acestuia la cabinet.

Medicul, pe baza celor declarate de pacient si a unor observatii de specialitate va genera un diagnostic şi un tratament adecvat.

In vederea realizării schimubului de informaţii cu alte aplicaţii medicale similare, aplicaţia medic de familie expune un set de servicii web prin care baza de date poate fi interogată.

A doua componentă importantă a sistemului este aplicaţia de asistenţă pentru diagnoză şi tratament. Această componentă, accesibilă opţional de către medic pe parcursul unui consult, ajută medicul în investigarea şi diagnosticarea corectă a pacienţilor.

A treia componentă a sistemului este un portal medical. Acest portal are rolul de a coagula componentele distribuite ale sistemului, oferind o cale securizata de acces la informaţii şi servicii medicale. Portalul are mai multwe roluri:

- rol de informare atât a pacienţilor cu afecţiuni cardio-vasculare cât şi a medicilor;
  Sunt postate informaţii privind realizări recente în domeniul cardio-vascular,
  informaţii despre entităti medicale şi despre servicii medicale oferite de medici
  înregistraţi în sistem.
- rol de cale unica de acces din exterior in sistemul distribuit;
- rol de centralizare a informatiilor medicale in depozite specializate de date;
- rol de diseminare de informatii din domeniu.

Cu scopul monitorizării de la distanţă a pacienţilor s-a conceput şi implementat un sistem de achiziţie a parametrilor medicali bazat pe o reţea de senzori de tip wireless. Cu ajutorul retelei se pot culege atât valori de parametrii ce caracterizează starea pacientului (ex.

temperatură, ECG) cât şi parametri ai mediului în care pacientul îşi desfăşoară activitatea (temperatură ambientală, umiditate, etc.).

Toate aceste componente concură la îndeplinirea funcțiilor de bază ale sistemului: facilitatea dialogului pacient-medic, monitorizarea de la distanță a pacienților şi informarea pacienților şi a personalului medical.

Un aspect important ce s-a avut în vedere a fost achiziția şi procesarea statistică a datelor medicale. Un capitol al raportului prezintă tehnicile de prelucrare şi analiză a datelor utilizate în proiect şi câteva rezultate preliminare obținute.

In această fază a priectului s-a realizat implementarea sistemului şi testarea pe componente. In faza următoare se are în vedere integrarea acestor componente şi efectuarea de teste funcționale. De asemenea, se are în vedere încărcarea bazei de date cu informații medicale reale culese de la cele 3 entități medicale cuprinse în consorțiul proiectului.

Rezultatele ştiințifice obținute în etapa curentă au fost sintetizate în 4 articole ştiințifice, prezentate în cadrul unor conferințe internaționale din domeniul informaticii aplicate şi din domeniul cardiologiei.

In ceea ce priveste achizițiile de echipamente prevăzute inițial în proiect, datorită diminuării fondurilor alocate pe anul 2009, acestea au fost amânate până în anul următor. Activitătile de cercetare-dezvoltare au fost reorientate astfel încât aceste întârzieri să nu afecteze obiectivele inițiale ale proiectului.

Consideram ca obiectivele generale si specifice ale etapei au fost atinse.

# 10 BIBLIOGRAFIE


1. "Practical Experiences in Building Ontology-based Retrieval Systems" - Elena Paslaru Bontas : http://userpage.fu-berlin.de/~paslaru/papers/swcase2005.pdf
2. "Introduction to OWL Web Ontology Language for Medical and Biosciences Applications" - Alex Amies: http://www.medicalcomputing.net/owl1.html, http://www.medicalcomputing.net/owl4.html
3. "Medical Ontology Research": http://lhncbc.nlm.nih.gov/cgsb_site/servlet/Turbine/template/research%2Cmedterm%2COntology.vm
4. "Medical ontologies for computer assisted clinical decision support": http://www.freshpatents.com/Medical-ontologies-for-computer-assisted-clinical-decision-support-dt20070426ptan20070094188.php.
5. Collaborative Ontology Construction for Information Integration - Adam Farquhar, Richard Fikes, Wanda Pratt, James Rice: http://www3.isrl.uiuc.edu/~junwang4/langev/localcopy/pdf/farquhar95collaborativeOntology.pdf
6. American Telemedicine Association: http://www.atmeda.org/ . "Core Standards for Telemedicine Operations " din documentul de pe pagina: http://www.atmeda.org/ICOT/Standards/11-20-07%20Standards%20Framework%20final%20for%20public.pdf
7. Proiectul Epi-Medics de pe: http://epi-medics.univ-lyon1.fr/statico/epimedica.htm. "Toward personal eHealth in cardiology. Results from the EPI-MEDICS telemedicine project" din documentul de pe pagina: http://cbbp.thep.lu.se/pub/Preprints/05/lu_tp_05_39.pdf
8. Proiectul HealthPal: "An Intelligent Personal Medical Assistant for Supporting the Self-Monitoring of Healthcare in the Ageing Society" din documentul de pe pagina: http://www.pervasivehealthcare.dk/UbiHealth2006/papers/2006/2.a.Komninos.HealthPal.pdf
9. Institutul Național de Studii şi Cercetări pentru Comunicații – INSCC de pe pagina: http://www.inscc.ro/ . Prezentare eHealth Pitesti: http://www.inscc.ro/Materiale/eHEALTH_prez2006_PITESTI.pdf
10. [bTemplate] – Brian Lozier 2002 - http://www.massassi.com/bTemplate/ - template engine-ul de baza folosit in pagina (implementat in PHP )
11. [Script.aculo.us] – Script.aculo.us – Librarii javascript cross-browser http://script.aculo.us/
12. [tiny_mce] – web based javascript text editor – http://tinymce.moxiecode.com/

# 11 ANEXA 1. Diagrama Bazei de date pentru aplicatiile Medic de familie si Spital: