MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ŞI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

SISTEM DE MONITORIZARE LA DISTANŢĂ A PACIENŢILOR

LUCRARE DE LICENŢĂ

Absolvent: **Andreea-Dorina PUȘCAȘ**

Coordonator ştiinţific: **Șef lucr.ing. Cosmina IVAN**

2011

MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

DECAN,
Prof. dr. ing. Sergiu NEDEVSCHI

VIZAT,
ŞEF CATEDRĂ,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Andreea-Dorina PUȘCAȘ**

SISTEM DE MONITORIZARE LA DISTANTA A PACIENTILOR

1. **Enunţul temei:** *Proiectul isi propune sa realizeze o aplicatie web prin care medicii sa poata monitoriza starea de sanatate a unor pacienti cu care au avut o prima consultatie fata in fata*

2. **Conţinutul lucrării:** *Pagina de prezentare, aprecierile coordonatorului de lucrare, Introducere,Obiectivele proiectului, Studiu bibliografic, Analiza si fundamentare teoretica, Proiectare de detali si implementare, Testare si Validare, Manual de instructiuni si utilizare,Concluzii, Bibliografie, Anexe.*

3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca, Catedra Calculatoare

4. **Consultanţi:** Șef lucr.ing. Cosmina IVAN

5. **Data emiterii temei:** 1 noiembrie 2010

6. **Data predării:** 24 Iunie 2011

Absolvent: _____

Coordonator ştiinţific: _____

Declaraţie

Subsemnatul *Andreea-Dorina PUSCAS*, student al Facultăţii de Automatică şi Calculatoare, Universitatea Tehnică din Cluj-Napoca, declar că ideile, analiza, proiectarea, implementarea, rezultatele şi concluziile cuprinse în această lucrare de licenţă constituie efortul meu propriu, mai puţin acele elemente ce nu îmi aparţin, pe care le indic şi recunosc ca atare.

Declar de asemenea că, după ştiinţa mea, lucrarea în această formă este originală şi nu a mai fost niciodată prezentată sau depusă în alte locuri sau alte instituţii decât cele indicate în mod expres de mine.

Data: 24 Iunie 2011

Absolvent:Andreea-Dorina PUȘCAȘ
Număr matricol:
Semnătura:_____

Cuprins

1. Introducere

1.1 Contextul proiectului

Internetul este un fenomen spre care toata lumea a migrat, cu pasi mai repezi sau mai inceti. Este mediul de comunicare care a cunoscut cea mai rapida dezvoltare din istoria umanitatii, atragand milioane de utilizatori intr-un timp foarte scurt.

Dezvoltarea aplicatiilor web a devenit astfel o necesitatea pentru orice companie, din orice domeniu de activitate. Aceasta dezvoltare a Internetului si a tendintelor continue de orientare spre acest mediu de comunicare si-a pus amprenta si asupra domeniului medical ducand la aparitia diverselor sisteme menite sa usureze atat munca doctorilor, cat si sa salveze timpul acestora si al pacientilor.

La noi, informatizarea a ajuns cu greu in spitalele si cabinetele de medicina. Acest lucru s-a datorat mai multor factori. Un factor ar putea fi lipsa fondurilor pentru implementarea unui astfel de sistem informatizat si pentru achizitionarea componentelor necesare medicilor pentru utilizarea sistemului. Un al doilea factor ar putea fi reticenta medicilor care pot considera ca un astfel de sistem nu ar aduce nici o imbunatatire in modul de desfasurarea al activitatilor lor zilnice, fiindu-le teama de schimbare si de utilizarea unui domeniu, cel informatic, poate strain lor sau poate, considerand ca educarea pacientului pentru utilizarea sistemului, daca ne gandim de exemplu la un sistem de consultatii online, este o munca mult prea grea si poate fara rezultate, in final.

E-health valoreaza deja cincisprezece miliarde de euro in UE si se incearca eliminarea obstacolelor si accelerarea legislatiei pentru a stimula cresterea in acest sector.
Adoptarea eHelath necesita o retea cu latime de banda extinsa, standarde tehnice obisnuite si standardizate, cumpararea unor programe actualizate, garantarea protectiei pentru datele pacientilor si de formare(invatare) a cadrelor medicale.
Exista o tendinta si in ceea ce priveste dispozitivele mobile de sanatate care ajuta pacientii sa-si monitorizeze sanatatea si sa comunice rezultatele doctorilor la care sunt inregistrati.
Deoarece cererea pentru servicii de consultatie online se astepata sa creasca, in UE exista numeroase tari care ofera dispozitive mobile de sanatate pentru a reduce costurile si timpul pacientilor pe care il petrec in spital sau la doctor.

Prin dezvoltarea unui sistem de consultatie online, sau monitorizare, se pot reduce semnificativ costurile necesare spitalizarii unor pacienti. Un alt avantaj ar fi dat de retinerea in baza de date a istoricului consultatiilor unui pacient. Acest lucru poate avea un impact major in determinarea si diagnosticarea unor boli viitoare.
Asadar, va exista un istoric al bolilor suferite de acestia si al modului in care acestea au evoluat in timp, aceste informatii vor putea fi obtinute cu usurina de catre medic oricand este nevoie de ele.

Un astfel de sistem este ideal pentru stari medicale minore sau acute care sunt tratate mult mai usor printr-o consultatie online decat prin prezenta constanta a pacientului in cabinetul medicului. De asemenea, ii permite pacientului sa fie consultat de medici cu specializari diferite foarte usor.

1.2 Conturarea domeniului

Tema aleasa este „Sistem de monitorizare la distanta al pacientilor”. Este un sistem implementat pentru un spital cu mai multe discipline de activitate (specializari) care isi propune sa-si ajute pacientii prin acest sistem online si sa evite supraincarcarea capacitatii de operare si spitalizare, din spital, pentru cazuri de sanatate care necesita o monitorizare a sanatatii, nefiind neaparat necesara internararea lor in spital.

Pacientul poate sa foloseasca sistemul doar daca a avut deja o consultatie fata in fata cu doctorul. Daca conditia de sanatate a pacientului nu este una ingrijoratoare si nu necesita ingrijirea si supravegherea continua a medicului, acesta poate continua comunicarea starii sanatatii lui prin intermediul sistemului.
Alte avantaje pe care le ofera un astfel de sistem, fata de modul clasic de efectuare al consultatiilor, sunt acelea ca :

- Medicul poate astfel sa monitorizeze starea de sanatate a mai multor pacienti, intr-un timp foarte scurt.
- Exista si avantajul retinerii consultatiilor in baza de date, lucru care ii poate crea o siguranta atat pacientului cat si medicului.
- Medicul ii poate adresa intrebari in mod constant pacientului pentru a observa modul de eveolutie a problemei lui.
- Posibilitatea utilizarii unor intrebari predefinite, sub forma unor chestionare pe care le trimit pacientilor.
- Pacientul nu este nevoit sa mearga constant la cabinetul medicului pentru a i se face evaluarea starii de sanatate.
- Se economiseste timp atata din partea pacientului, dar si al medicului.
- De asemenea, pacientul poate fi scutit de unele cheltuieli materiale necesare prezentarii lui la cabinetul medicului care se ocupa de monitorizarea sanatatii, mai ales daca pacientul se afla si la o distanta considerabila fata de cea unde medicul efectueaza consultatiile.
- Un alt beneficiu care merita mentionat este ca se pot aduce imbunatatiri semnificative in modul de desfasurare al activitatilor din unitatea medicala si de asemenea, se pot reduce costurile de intretinere al pacientilor, unii dintre ei pot fi consultati si de la distanta pentru a realiza un grafic al evolutie bolilor de care acestia sufera,fara a necesita spitalizarea lor.

Dezvoltarea unui sistem de genul acesta care sa poata imbunatatii modul de desfasurare a activitatii intr-un spital poate fi destul de costisitore, nu doar datorita necesarului de componente necesare medicilor, calculatoare si infrastructura retelei, ci si implementarea sistemului. Este nevoie de o retea de Internet care sa functioneze in permanenta, anagajatii unitatii medicale pot necesita perioade de pregatire pentru a putea folosi sistemul pus la dispozitie.

Un astfel de sistem este foarte util si dincolo de costurile pe care le implica, ar putea fi folosit cu succes de catre toata lumea care are acces la Internet si are un cont de email.

Datorita faptului ca utilizarea Internetului a devenit o constanta in viata noastra, tot mai multi oameni aleg sa realizeze diverse operatiuni(de platire a facturilor, rezervari online,etc) utilizand acest mediu de comunicare in schimbul alegerii metodelor vechi, clasice, care pot implica pierderea timpului asteptand la diverse ghisee pentru a-si achita facturi, asteptand la un spital pentru a-si face o consultatie, etc.

Se obseva foarte clar ca reticenta oamenilor in ceea ce priveste noile tehnologii,a utilizarii Internetului a disparut si mai mult decat atat exista o tendinta evidenta de crestere a folosirii servicilor oferite prin intermediul Internetului.

Aceste fapte ar putea garanta intr-un fel si utilizarea de catre asiguratii medicali(pacienti) a unui astfel de sistem de consultatii online, cu atat mai mult cu cat ar slava din timpul nostru.

Acest sistem ar putea oferi un mod ideal de comunicare intre pacient si specialist, un mediu standardizat si in acelasi timp foarte flexibil, un sistem sigur care sa-i confere pacientului incredre si medicului posibilitatea de a-si monitoriza pacientii cu usurinta, fara a le rapi prea mult timp.

2. Obiectivele proiectului

Acest capitol are ca scop definirea unor obiective clare pentru tema aleasa stabilite prin specificarea cerintelor de functionare a sistemului, dar si a atributelor de calitate pe care sistemul ar trebui sa le indeplineasca.

2.1 Obiective generale

Principalul obiectiv al acestui proiect este realizarea unui sistem software prin care sa poata fi monitorizati pacientii care au nevoie de ingrijiri de baza, a caror conditie medicala nu este foarte grava si nu necesita ingrirea din partea personalului specializat al unitatii in care acesta isi face consultatia.

Institutia care va beneficia de acest sistem este o institutie medico-sanitara (spital) avand cabinete cu diverse specializari in care se acorda asistenta medicala bolnavilor, fara ca acestia sa fie internati in spital.

Aplicatia isi propune sa construiasca un mediu de comunicare intre specialist si pacient, dupa ce pacientul a avut o intalnite fata in fata cu medicul specialist.

Rolul utilizarii acestui sistem este de a oferi pacientilor un mediu sigur de comunicare prin care vor putea fi monitorizati in ceea ce priveste starea lor de sanatate. Pacientul nu va fi nevoit sa revina periodic la cabinet pentru a comunica reactiile pe care le are in urma tratamentului prescris de specialist, ci poate pastra o relatie de comunicare prin intermediul sistemului.

In urma numeroaselor consultatii cu un pacient, specialistului ii va fi mult mai usor sa stabileasca un diagnostic final.

2.2 Tema propriu-zisa

Obiectivul principal al acestui proiect, cu titlul „Sistem de monitorizare la distanta al pacientilor” , este realizarea unui sistem software care isi propune sa ofere doctorilor dintr-un spital un mediu de comunicare cu pacientii dupa ce acestia au avut o prima intalnire fata in fata cu doctorul lor.

Aplicatia va fi utilizata pentru a monitoriza unele cazuri de sanatate care nu necesita neaparat spitalizare, ci pot fi foarte usor monitorizate de catre pacienti, de acasa fara a fi nevoie de revenirea lor periodica la cabinetul medicului. Ne propunem implementarea unei aplicatii web complexe care sa permita doctorilor dintr-un spital sa efectueze consultatii prin intermediul aplicatiei, folosind Internetul.

Avand ca scop principal implementarea unui astfel de sistem este necesara o definire clara a tuturor obiectivelor pe care trebuie sa le indeplineasca sistemul pentru a putea realiza in final implementarea proiectului. Astfel, trebuie specificate atat atat cerintele functionale cat si cele nefunctionale.

Obiectivul: atingerea cerintelor functionale si nefunctionale.

2.1.1 Cerinte non-functionale

Cerintele non-functionale descriu cerinte la nivel de utilizator care nu sunt direct legate de cerintele functionale. Includ utilizabilitate, performanta, implementare, suportabilitate, scalabilitate, etc.

Cerinte non-functionale pe care trebuie sa la indeplineasca sistemul sunt legate de disponibilitate, utilizabilitate, scalabilitate, mentenanta, performanta, posibilitatea de intelegere, securitate.

Disponibilitate

- Se refera la probabilitatea unui sistem de a fi operational cand este nevoie de el
- Sistemul trebuie sa functioneze in permanenta, fara intrerupere pentru a putea fi utilizat la orice ora.

Utilizabilitatea

- Cuprinde cerinte legate de usurinta de folosire a interfetei sistemului de catre utilizator
- Interfata trebuie sa fie estetica, intuitiva, consistenta
- Sistemul trebuie sa fie usor de utilizat atata de catre medici, cat si de catre pacienti. Acest lucru se va asigura prin utilizarea unor meniuri cu nume sugestive, si existenta unei pagini de ajutor unde se va explica cum trebuie utilizata aplicatia, pas cu pas.

Performanta

- Se refera la timpul necesar unui sistem sa raspunda la un stimul. Evenimentul poate fi initiat de un utilizator, alt sistem sau sistemul insusi.
- Cuprinde cerinte legate de timpii de lucru, precum timp de raspuns (response time), timp de revenire (recovery time), timp de lansare (startup time). Performanta unui sistem poate fi afectata de un numar prea mare de utilizatori, resursele distribuite, tranzactiile, dimensiunea bazei de date.

Scalabilitatea

- Scalabilitatea este proprietatea unui sistem de a suporta un numar mai mare de cereri, a unui numar mai mare de utilizatori, de exemplu, sau de a permite extinderea sistemului cu eforturi minime.
- Un sistem este considerat scalabil daca este capabil sa ofere rezultate imbunatatite in cazul adaugarii unor resurse aditionale.

Mentenanta

- Se refera la usurinta de efectuare a modificarilor facute unui sistem software dupa lansarea initiala. Modificarea unei parti a software-ului pentru a face o imbunatatire. Intretinerea poate fi facuta prin ajustarea aplicatiei la schimbari de mediu sau pentru a-i imbunatatii unele calitati sau caracteristici.

Posibilitatea de intelegere (*understandability*)

- Intretinerea software depinde foarte mult de intelegerea programelor. Inginerii de intretinere isi petrec cel mai mult timp incercand sa descopere logica aplicatiei si o parte mai mica din timp pentru aplicarea modificarilor necesare.

Securitatea

- este o masura a abilitatii sistemului de a rezista utilizarilor neautorizate in timp ce ofera servicii utilizatorilor legitimi. (1)
- Utilizatorilor trebuie sa li se permita accesul la anumite resurse in functie de rolul pe care acestia il au in sistem. Autentificarea si controlul accesului sun tehnici utilizate pentru asigurarea confidentialitatii datelor utilizatorilor.

2.1.2 Cerinte functionale

Se vor descrie obiectivele functionale care trebuie indeplinite pentru realizarea sistemului software:

1) Aplicatia va trebui sa permita logarea utilizatorilor in sistem
2) Vor exista trei tipuri de utilizatori, cu permisiuni diferite: specialistul, pacientul si administratorul.
3) Specialistul va putea crea un consult pentru pacientul pe care il monitorizeaza.

4) Specialistul trebuie sa poata vizualiza consultatiile in desfasurare sau cele terminate.
5) Se vor implementa mai multe tipuri de intrebari pentru a-i oferi specialistului posibilitatea alegerii tipului de intrebari care se potriveste cel mai bine situatiei pacientului :
    a. De exemplu, intrebari cu mai multe variante de raspuns
    b. Intrebari fara variante de raspuns
    c. Sau intrebari la care pacientul trebuie sa raspunda cu o data
    d. Intrebari din baza de date
6) Terminarea unei consultatii se realizeaza de catre specialist.
7) Specialistul va trebui sa-i poata comunica pacientului data urmatoare in care va comunica cu el.
    - Specialistul va trebui, in acest tip de comunicatie, sa-i trimita intr-un mesaj, orice anunt doreste acesta, impreuna cu data la care va reveni pentru a continua comunicatia cu pacientul.
8) Chestionarele vor putea fi trimise de catre specialist. Acestea vor fi trimise sub forma unor documente. Ele vor fi create de adminstrator si activate de acesta.
9) Notificarile vor fi disponibile prin prin e-mail securizat.
    a. Notificarile vor fi facute de exemplu la terminarea unei consultatii
    b. La fiecare programare noua pe care specialistul i-o face unui pacient care are deschisa o consultatie la el
10) Pacientul trebuie sa aiba un cont creat pentru a putea utiliza aplicatia.

Prin acest mod de inregistrare online se vor reduce numarul de de acte administrative necesare pentru inscrierea unui pacient in organizatia de ingrijire. Odata ce pacientul este inscris in programul de ingrijire, el va avea acces la functionalitatile orferite de aceasta aplicatie. Dupa logare, pacientul va avea urmatoarele functionalitati puse la dispozitie:

a. Va putea reveni la pagina de start unde va avea la dispozitie o pagina de ajutor cu informatii despre aplicatia „Sistem de monitorizare la distanta al pacientilor”.
b. Isi va putea vizualiza toate consultatiile.
    - Pacientul trebuie sa-si poate vedea consultatiile cu toate informatiile referitoare: data crearii, specialistul care a facut consultatia, ultima data cand pacientul sau specialistul au avut o reactie si un status care sa indice cine trebuie sa faca urmatoarea actiune:specialistul-status:”Asteptati raspunsul medicului” sau, pacientul-status:”Se asteapta raspunsul pacientului”

11) Pentru o consultatie din lista care are statusul indicand o consultatie terminata, pacientul va putea vizualiza concluzia acestei consultatii
12) De asemenea, pacientului trebuie sa i se permita sa vizualizeze istoria comunicarii dintre el si specialistul unei anumite consultatii.
13) Pacientul trebuie sa-si poata vizualiza lista de intrebari pe care medicul i le-a adresat si trebuie sa raspunda la toate intrebarile.
14) Administratorul va trebui sa poata efectua operatii de creare, modificare si sterge a specialistilor

Obiectivul principal este dezvoltarea unei aplicatii care sa-i ajute pe specialisti sa efectueze consultatii care sa respecte cerintele functionale impuse si care, in urma numeroaselor consultatii, ii vor putea ajuta in ceea ce priveste stabilirea unui diagnostic final, pacientii vor avea un sentiment de siguranta prin faptul ca vor putea comunica cu usurina si foarte repede orice probleme.

Unitatea sanitara care beneficiaza de acest sistem prezinta indicii de avantaje atat pentru pacienti, precum si pentru organizarea asistentei medicale. Aceasta aplicatie se va adresa in special cazurilor de sanatate care nu necesita ingrijirea asistata a unui specialist,

pentru cazuri de boli care nu sunt acute, nu au o evolutie rapida, ci pentru boli a caror evolutie este constanta, sau care prin tratament administrat de catre pacient, poate stagna sau poate fi tratabila.
Intr-o astfel de unitate, rata de neprezentare si de aglomerare a pacientilor trebuie sa scadea deorece pacientul are nevoie doar de o consultatie fata in fata cu specialistul, dupa care acesta va putea fi monitorizat prin utilizarea sistemului, fara a fi nevoit sa se prezinte periodic la consultatie.
Ingrijirea este mai accesibila, sistemul va functiona 24 ore pe zi , sapte zile pe saptamana.

3. Studiu bibliografic

In acest capitol vor fi prezentate referintele bibliografice studiate pentru o mai buna fixare a referentialului in care se situeaza tema. Vor fi descrise cartile, decumentele, articolele sau orice alte materiale folosite sau studiate pentru conturarea finala a proiectului.

3.1 Informatii generale din domeniul medical

Medicii unei unitati sanitare sunt principalii furnizori de servicii medicale . Pentru ca orice unitate sanitara sa functioneze corect atat din punct de vedere administrativ cat si al modului de desfasurare al activitatilor profesionale ale medicilor si ale personalului medical, exista o serie de regulamente impuse prin legi si ordonante emise de Guvernul Romaniei Conform (2) , Cap. 1, Articolul 1, 2) serviciile de sanatate ale cabinetelor medicale se realizeaza de medici din domeniul medicinei generala – medici de familie, medici stomatologi, medici specilaisti si alte categorii de personal medical autorizat. Cap. 3 din (1) descrie modul de organizare, functionare si finantarea cabinetelor medicale; art. 6 al acestui capitol apar obligatiile personalului medical dupa cum urmeaza: fiecare medic sau personal medical care desfasoara activitati medicale in cadrul cabinetului medical trebuie sa raspunda in mod individual, conform legii, de deciziile profesionale pe care le iau, in cazul unor prejudicii aduse pacientilor.

Pentru o functionare buna a oricarei institutii medicale, trebuie sa existe o modalitate de monitorizare a activitatii profesionale a medicilor. Acest lucru se realizeaza de catre Colegiul Medicilor din Romania si Ministerul Sanatatii Publice, denumite autoritati competente.

Cateva dintre obligatiile importante pe care un medc le are sunt descrise in articolul 374 din capitolul 1 , sectiunea 1 al (3):

1) Profesia de medic are ca principal scop asigurarea starii de sanatate prin prevenirea imbolnavirilor, promovarea, mentinerea si recuperarea sanatatii individului si a colectivitatii.

2) In vederea realizarii acestui scop, pe tot timpul exercitarii profesiei, medicul trebuie sa dovedeasca disponibilitate, corectitudine, devotament, loialitate si respect fata de fiinta umana.

3) Deciziile si hotararile cu caracter medical vor fi luate avandu-se in vedere interesul si drepturile pacientului, principiile medicale general acceptate, nediscriminarea intre pacienti, respectarea demnitatii umane, principiile eticii si deontologiei medicale, grija fata de sanatatea pacientului si sanatatea publica.

4) Personalul medical raspunde civil pentru prejudiciile produse din eroare, care includ si neglijenta, imprudenta sau cunostinte medicale insuficiente in exercitarea profesiunii, prin acte individuale in cadrul procedurilor de preventie, diagnostic sau tratament.

De asemenea, si in celelalte articole se descriu o serie de indatoriri ale acestora in relatia lor cu pacientii, amintim cateva mai importante:

1) Medicul are dreptul sa-si exercite profesia independent si liber, sa puna in aplicare deciziile asuprea hotararilor cu caracter medical, in scopul asigurarii in orice imprejurare a intereselor pacientului.

2) Conform capitolului 4, articolul 24 din (4) ,medicul are obligatia prin lege de a nu divulga informatiile medicale sau informatii privind viata privata a pacientului.

3) Pacientul are dreptul,conform capitolului 6, art. 35 din (4), la ingrijiri medicale pana la ameliorarea starii sale de sanatate sau pana la vindecare.

Drepturile pacientilor conform (3):

- Pacientii au dreptul la ingrijiri medicale de cea mai inalta calitate de care societatea dispune, in conformitate cu resursele umane, financiare si materiale.
- Pacientii au dreptul sa fie informati cu privire la serviciile medicale disponibile si la modul de utilizare al acestora.
- Dreptul la confidentialitatea informatiilor: toate informatiile privind starea pacientului, rezultatele investigatiilor, diagnosticul, prognosticul, tratamentul, datele personale sunt confidentiale atata timp cat pacientul nu isi da consimtamantul explicit.
- Pacientul are acces la datele medicale personale.

Obligativitatea asigurarii asistentei medicale prezentata in (5) contine cateva idei importante de care este bine sa se tina cont atunci cand va fi creat sistemul si anume:

1) Atunci cand medical, asistentul au acceptat pacientul, relatia poate fi intrerupta in urmatoarele cazuri:
   a) odata cu vindecarea bolii;
   b) de catre pacient;
   c) de catre medic, in urmatoarele situatii:
      (i) atunci cand pacientul este trimis altui medic;
      (ii) pacientul manifesta o atitudine ostila si/sau ireverentioasa fata de medic.
2) Medicul va notifica pacientului, in situatia prevazuta la alin. 1) lit. c) pct. (ii), dorinta terminarii relatiei, inainte cu minimum 5 zile, pentru ca acesta sa gaseasca o alternativa, doar in masura in care acest fapt nu pune in pericol starea sanatatii pacientului.

3.2 Referinte tehnologii utilizate

Pentru a putea implementa acest proiect, utilizand framework-ul LLBLGen, de un mare ajutor a fost cartea (6), atat pentru notiunile teoretice referitoare la arhitectura acestui framework, pentru modul de instalare si operare cu programul LLBLGen, dar si prin exemplele concrete de utilizare.
Documentatia (7) a reprezentat o resursa mai sigura de intelegere prin exemple, deoarece documentatia corespunde ultimei versiuni a programului LLBLGen Pro utilizat si pentru acest proiect.

O alta sursa utilizata a fost (8) care cuprinde o sinteza a ceea ce reprezinta un mapper, descrie foarte succint care sunt obiectele create de LLBLGen Pro si o comparatie intre tipurile de proiecte(„template”) Self Servicing si Adapter.

In primul capitol al (6) , se prezinta avantajele utilizarii acestui instrument de mapare Obiect-Relatie. Acest mapper va lua schema unei baze de date existente si va auto-genera nivelul de date (eng. Data layer) .

Un mapper obiect-relatie creaza clase care definesc obiecte care vor corespunde structurii bazei de date create.
Conform (6) si (8) fiecare inregistrare din baza de date, sau rand, devine o entitate. Fiecare tabel din baza de date devine un „EntityCollection”. Campurile tabelelor bazelor de date devin proprietati publice ale obiectului entitate (eng. Entity). Framework-ul construieste si alte clase si motode ajutatoare pentru a gasi entitatile, pentru a le salva inapoi in baza de date si pentru a le seta proprietatile.
Typed Views reprezinta vederi in baza de date care pot fi „impachetate” ca obiecte „strongly-typed” Data Table. Typed Views pot fi doar citite.

In (6) sunt descries avatajele proiectarii pe 3 nivele si a utilizarii instrumentului LLBLGen Pro dupa cum urmeaza:

- Utilizarea metodei de proiectare pe trei nivele va permite schimbarea interfetei utilizator fara a pierde celelalte doua nivele ale aplicatie.
- Regulile de business sunt salvate intr-o locatie centrala. Astfel, daca se doreste schimbarea procesului de validare a unui utilizator, schimbarea va trebui efectuata intr-un singur loc si anume in metoda care efectueaza operatia respectiva.
- Schimbarea bazei de date, ar trebui, in mod teoretic, sa duca doar la modificarea nivelului de acces la date. Din pacate, in cele mai multe situatii acest lucru nu este posibil.

LLBLGen ajuta la indeplinirea unei proiectari ideale in urmatoarele feluri:

- LLBLGen Pro va auto-genera layerul de acces la date. In acest fel, programatorul nu mai trebuie sa scrie codul de mana in acest layer, astfel poate economisii destul de mult timp.

- LLBLGen Pro va genera optional un schelet de nivel logica de business pentru a “porni” layerul de business pentru programator. Nu trebuie sa se inteleaga complexitatea de mostenire si crearea unor clase personalizate pentru a profita de avatajele claselor din business logic; in LLBLGen Pro aceste clase sunt organizate intuitive, puternic si infinit extensibile.

- Colectiile LLBLGen Pro sunt “bindable”, adica pot fi legate de structuri de date, ceea ce le face sa fie extreme de usor de adaugat in controale .NET. Acest lucru poate reduce dimensiunea nivelului de UI.

- Schimbarea aplicatiei bazei de date si a nivelului de acces la date fara a afecta logica de business si nivelul de interfata cu utilizatorul este posibila cu LLBLGen Pro. Daca ati migrat schema bazei de date si procedurile stocate la o noua aplicatie a bazei de date, abtinandu-va sa folositi orice trasaturi specifice ale bazei de date si schema s-a potrivit exact, se poate regenera layerul dumneavoastra de acces la date cu LLBLGen fara a se face nici o alta modificare in aplicatia.

Cartea (6) ofera un exemplu foarte complex pentru a exemplifica modul de utilizare a instrumentului LLBLGen, incepand de la modul de creare a unui nou proiect utilizand acest program, generarea nivelului de acces la date si apoi modul de utilizare a claselor generate si a altor proprietati ale acestora in nivelul logica de business.

LLBLGen pune la dispozitie o serie de constructori de filtrare care tind odata cu evolutia versiunilor sa semene tot mai mult cu sintaxa C# normala.

Pentru implementarea codului au fost utilizate urmatoarele tutoriale:

Tutorialul (9) prezinta modalitatea de utilizarea a serviciilor web pentru a implementa functionalitati AJAX intr-o aplicatie.
Notiuni utilizate din acest tutorial:

- Tutorialul prezinta modalitati de apelare a Serviciilor Web utilizand framework-ul ASP.NET AJAX.
- Utilizare control AutoCompleteExtender disponibil in ASP.NET AJAX Toolkit
- Folosirea serviciilor oferite de proprietatea ScriptManager. Aceasta proprietate permite definirea a unuia sau mai multor servicii pe care o pagina ASP.NET AJAX o poate apela asincron pentru a trimite sau pentru a primi date fara a necesita operatii postback. Tutorialul prezinta modul de utilizare a proprietatii ScriptManager printr-un exemplu explicat pas cu pas. Se defineste un serviciu utilizand controlul ASP.NET AJAX ServiceReference in ScriptManager si I se asigneaza acestuia URL-ul Serviului Web la proprietatea Path a controlului.

In implementarea acestui proiect, o parte din exemplul descris in site-ul lui (9) a fost folosit pentru a popula un control TextBox utilizand AutoCompleteExtender care afiseaza, in mod asincron, date utile din baza de date care vor fi selectate si utilizate de catre textbox-ul respectiv.

Urmatoarele doua tutorial: (10) si (11) au fost utilizate pentru implementarea logarii, in special pentru exemplificarea modului de lucru cu FormsAuthenticationTicket si pentru realizarea unei logari securizate utilizand roluri.

In tutorialul (10) se prezinta modul de setare a fisierului web.config pentru a adauga tag-urile authentication si authorization in <system.web> care vor realiza redirectionarea utilizatorilor logati spre pagina indicata intre tag-urile authentication si, pagina unde vor fi redirectionati daca incerca sa intre intr-o pagina/folder securizat al aplicatiei web.
Tagurile authorization vor ajuta la crearea unui folder securizat; intre aceste tag-uri se specifica un alt tag numit deny care va specifica ca acest folder este inaccesibil pentru toti utilizatorii anonimi.
In tutorialul (11) se prezinta relizarea securizarii aplicatiie in functie de rolurile utilizatorilor, pentru folderele aplicatiei care corespund cu rolurile utilizatorilor, de exemplu admin si specialist.

In cartea (12) , se descriu caracteristicile ASP.NET AJAX si beneficiile pe care acesta tehnologie le poate aduce in proiect prin utilizarea unor pagini care utilizeaza aceasta modalitate de apelare asincrona.
Este descris modul tipic de apel AJAX si beneficiile pe care acest mod de lucru doreste sa il aduca aplicatiilor. AJAX vine cu solutia echilibrarii sarcinii dintre client si server, permitandu-le acestora sa comunice in fundal in timp ce utilizatorul lucreaza cu pagina.
Pe langa benefiile aduse unei aplicatii, sunt descrise si problemele care pot sa apara cand se utilizeaza AJAX cu un scop gresit.

Cartea (13) prezinta in partea I, infrastructura AJAX ASP.NET si arhitectura AJAX, informatii care au fost utilizate pentru partea de fundamentare teoretica in ceea ce priveste tehnologiile utilizate pentru aplicatie.
De asemenea, este descrisa infrastructura ASP.NET AJAX care incerca sa simplifice modul de utilizarea al AJAX in aplicatii si introduce un container control numit UpdatePanel utilizat pentru a “inconjura” portiuni din pagina existenta.

In ASP.NET , controlul ScriptManager este un control server care gestioneaza cea mai mare parte de script din jurul unei pagini AJAX. El gestioneaza si livreaza resursele obisnuite de script, cum ar fi biblioteca client Microsoft JavaScript.

Capitolul 1 din (14) prezinta framework-ul .NET ca un grup de mai multe tehnologii pe care le descrie teoretic. Doua component fundamentale ale .NET framewok sunt biblioteca de clase .NET si The Common Language Runtime. Compilarea limbajelor in .NET se realizeaza utilizand un limbaj intermediar, de nivel mai jos, inainte de executia codului initial.

4. Analiză şi fundamentare teoretica

In acest capitol se vor explica pentru inceput principiile de functionare ale sistemului si se va descrie la nivel teoretic solutia propusa, urmand cu o analiza teoretica a protocoalelor utilizate (frameworkuri, arhitecturi) si modele abstracte, iar in cele din urma se va realiza o structura logica a aplicatiei in functie de solutia aleasa.
Vor fi explicate: arhitectura conceptuala a sistemului si concepte de baza ale tehnologiilor care vor fi in cele din urma utilizate pentru implementarea sistemului.
Se realizeaza asadar o analiza a tehnologiilor care vor putea fi utilizate, structura arhitecturala pe care o va avea sistemul pentru a indeplini cerintele functionale si nonfunctionale intr-un mod cat mai convenabil pentru sistemul nostru.

4.1 Analiza cerintelor

Se va face o analiza a cerintelor functionale pe care trebuie sa le indeplineasca sistemul, urmand ca dupa descrierea solutiei propuse, din punct de vedere functional, sa se poata determina arhitectura acestui sistem.

In primul rand ar trebui sa determinam care vor fi actorii acestui sistem, adica cine va folosi sistemul, cunoscand faptul ca domeniul este cel medical. Asadar, se pot identifica 3 actori principali (sau roluri): specialistul (sau medicul), pacientul si administratorul aplicatiei.

De asemenea, trebuie sa determinam cui ii este adresata aceasta aplicatie, institutia care va beneficia de acest sistem. Vom presupunem ca destinatarul sau beneficiarul este o institutie medico-sanitara (spital) cu cabinete avand diverse specializari in care se acorda asistenta medicala bolnavilor, fara ca acestia sa fie internati in spital.

Aplicatia isi propune sa construiasca un mediu de comunicare intre specialist si pacient, dupa ce pacientul a avut o intalnite fata in fata cu medicul specialist.

O alta cerinta foarte importanta pentru sistem si absolut necesara ar fi oferirea unei logari sigure si securizare in functie de roluri (de exemplu, pacientul sa nu aiba disponibilitate la paginile specialistului).

4.1.1 Cerinte functionale

Cerintele functionale sunt urmatoarele, in functie de rolul pe care utilizatorul il are:
Specialistul trebuie:

I. Sa poata adauga o consultatie noua pentru un pacient.

Specialistul trebuie, in acest caz, sa poata cauta pacientul pentru care se doreste sa se creeze o noua consultatie. Daca acesta exista in baza de date, atunci specialistul va trebui sa aleaga un specialist din aceeasi disciplina ca si el, care nu are deja o consultatie deschisa cu acest pacient. In caz contrar, daca pacientul are o consultatie neterminata la specialistul selectat, ar trebui sa se afiseze un mesaj de atentionare; la fel se va intampla si daca pacientul cautat nu exista in sistem.

II. Sa poata vizualiza pacientii care au consultatii deschise (neterminate) la el.

Pentru fiecare dintre aceste consultatii ale pacientilor trebuie sa existe posibilitatea de a vedea toate mesaje schimbate in timp intre specialist si pacient (pentru a avea o istorie a tuturor intrebarilor si raspunsurilor de-a lungul comunicarii dintre cei doi; acest istorie a conversatiei il poate ajuta pe specialist sa nu mai puna eventual aceeasi intrebare pacientului daca i-a adresat-o odata), dar sa nu le poata modifica, sau, specialistul trebuie sa poata trimite o alta comunicare sau intrebare pacientului.

III. Sa poata vizualiza toate consultatiile saptamanii curente si a saptamanilor trecute, incepute sau nu, in tabele diferite. Pentru fiecare dintre aceste consultatii va trebui sa poata vizualiza informatii privind istoricul comunicarii pentru o consultatie selectata, programarile consultatiei, sa deschida consultatia sau sa inchida o consultatie. Daca nu exista consultatii pentru saptamana curenta sau pentru saptamanaile anterioare, trebuie sa se afiseze un mesaj de atentionare.

- Exista posibilitatea cautarii dupa numele unui pacient.
- Deschiderea paginii de continuare a consultatiei pentru un anumit pacient, care nu are consultatia cu status de consultatie terminata, va consta in apasarea unui buton situat in capatul drept al tabelei in care apar toate consultatile, deschiderea unei pagini (numita open consult) si posibilitatea compunerii unui set de intrebari.
- Pentru pagina de „open consult" trebuie sa existe un dropdown list pentru a selecta actiunea pe care specialistul vrea sa o faca. Aceste actiuni vor fi:
    i. „ Pune o intrebare". In acest caz, specialistul va trebui sa aleaga din lista care apare in cel de-al doilea dropdown tipul de intrebare. Sunt 4 tipuri de intrebari pentru acest sistem :
        1. „Open Question" (Intrebare deschisa) - in aceasta situatie, specialistul poate sa puna o intrebare simpla pe care o scrie intr-un textbox) si daca vrea, poate adauga intrebarea respectiva in tabelul bazei de date cu intrebari predefinite.
        2. „Multiple Choice Question" (Intreabare cu variante de raspuns) - daca specialistul alege acest tip de intrebare, acesta este capabil sa compuna intrebarea (intr-un textbox) precum si raspunsul intrebarii (tot intr-un textbox). La fel ca la i. , i se ofera posibilitatea adaugarii intrebarii in baza de date.
        3. „Question with Date" („Intrebare cu data") - Intrebare la care se asteapta ca raspuns tipul Date.
        4. „Question from Database" („Intrebare din baza de date") – In cazul in care specialistul alege acest tip de intrebare, o lista de radiobutton va fi populata cu intrebarile disponibile in baza de date. La sfarsitul fiecarei intrebari va aparea tipul intrebarii (dintre cele mentionate mai sus la punctul i, ii s iii)
        5. Specialistul poate selecta sa trimita un chestionar intreg catre un pacient, acest lucru este posibil doar daca consultatia nu este incheiata(inchisa).
- Dupa ce specialistul pune o intrebare noua se va trimite un mail de notificare spre pacient pentru a sti ca i-au fost adresate noi intrebari si ca pate intra in sistem pentru a raspunde la ele.
- Daca specialistul va dori sa inchida consultatia pentru pacient, va apasa butonul din tabel care il va redirectiona spre pagina de „open consult" unde va putea scrie o concluzie sau un rezumat pentru a i le trasmite pacientului odata cu incheierea consultatiei.
- La fiecare inchidere a unei consultatii se va trimite un e-mail de notificare spre pacientul respectiv.
- Specialistul ii poate comunica unui anumit pacient data urmatoarei consultatii. Specialistul va alege o data pentru consultatie, iar intr-un textbox va fi afisat un mesaj predefinit care va contine un text de anuntare a actiunii pe care o va face specialistul, la care se adauga data selectata anterior.

IV. Specialistul poate vizualiza consultatiile viitoare, adica cele care au data consultatiei mai mare decat data curenta plus 7 zile. Se va putea face si o filtrare dupa numele pacientului introdus intr-un textbox.

V. Vizualizarea consultatiilor terminate se va face oarecum identic cu celelalte modalitati de vizualizare. Se vor afisa toate consultatiile cu statusul „Closed".
Va exista si in acest caz posibilitatea cautarii dupa numele pacientului. Pentru fiecare inregistrare din tabel va exista un buton de vizualizare a detaliilor legate de comunicarea cu pacientul selectat. Daca se va apasa butonul, se va face o redirectionare catre pagine cu detalii constand in:
   - Afisarea ultimului contact sau a ultimei comunicari dintre specialist si pacient si concluzia trimisa la inchiderea consultatiei
   - Toate programarile: urmatoarea data a consultatiei impreuna cu descrierea pentru consultatia selectata
   - O istorie a tuturor mesajelor transmise intre specialist si pacient pe toata durata consultatiei. Se va afisa data mesajului, data in care a aparut o reactie la intrebarile puse de specialist, tipul de intrebare, raspuns.

   Nu se va putea redeschide consultatie din acest tabel.

Functionalitatile pacientului vor fi urmatoarele:

- Pacientul trebuie sa aiba un cont creat in sistemul de monitorizare pentru a putea accesa functionalitatile puse la dispozitie pentru pacienti
- Pacientul va trebui sa fie logat in sistem pentru a-si accesa pagina
- Dupa logare, pacientul trebuie sa-si poata vizualiza toate consultatiile.

Pentru fiecare consultatie se va afisa data consultatiei, specialistul care a efectuat consultatia, data ultimei reactii,status. De asemenea, pentru fiecare consultatie in parte trebuie sa se poata afisa , daca statusul consultatiei indica o consultatie terminata, atunci se va putea vizualiza, pe langa istoria mesajelor si toate programarile, si data incheierii consultatiei impreuna cu un o concluzie.

- Pacientul va putea sa-si vada intrebarile puse de specialist si sa raspunda la ele sau poate sa deschida un chestionar pentru completare.

Adminstratorul sistemului trebuie sa :

- Poata adauga un nou specialist in sistem, sa modifice datele unui specialist, sau sa-l stearga.
- De asemenea, administratorul trebuie sa poata crea un nou chestionar.

4.1.2 Cerinte non-functionale

Sistemul ar trebui sa indeplineasca urmatoarele cerinte non-functionale si de calitate descrise mai jos.

Sistemul trebuie sa fie scalabil, sa permita extinderea sa daca este nevoie, intretinerea sa fie usoara, oricand sa se poata adauga noi functionalitati sau sa se modifice cele existente fara un efort prea mare. De asemenea , sistemul trebuie sa fie usor de inteles de catre orice programator, sa fie disponibil in orice situatie. Fiabilitatea este o alta cerinta obligatorie care va asigura functionarea sistemului fara a genera erori; fiabilitatea poate fi crescuta prin doua metode: evitarea erorilor care rezulta din practici de proiectare si toleranta la esec care presupune functionarea sistemului chiar daca apar esecuri sau erori.

Securitatea sistemului trebuie asigurata prin realizarea unei autentificari sigure, accesul controlat la anumite parti din sistem. De exemplu, pacientul nu ar trebui sa poata avea acces la pagina dministratorului sau a specialistului.

4.2 Solutia propusa

Solutia propusa pentru realizarea acestui sistem consta in utilizarea unei model arhitectural structurat pe 3 nivele, prezentate teoretic mai jos. Acest tip de arhitectura elimina dependentele dintre componente, imbunatatind astfel scalabilitatea sistemului. De asemenea, prin separarea celor trei componente, sistemul va deveni mult mai usor de intretinut si de inteles, sistemul devine mai sigur, utilizatorul neavand acces direct la nivelul de acces la baza de date in mod direct.

Implementarea bazei de date:

- Baza de date va fi implementata in SQL Server 2008 R2

Maparea la Baza de Date

- Se va utiliza LLBLGen Pro care este un sistem complex de mapare-generare O/R (Object-Relational). LLBLGen Pro va auto-genera in totalitate layer-ul de acces la date usurand astfel foarte mult munca programatorului.

Cerinte de implementare a aplicatiei:

- Proiectul va fi realizat in Microsoft Visual Studio 2010
- Se va utiliza .NET Framework 4 care va oferi un mediu consistent pentru programarea orientata obiect, permitandu-ne sa cream aplicatia web dorita.
- Pentru realizarea interfetei utilizator se vor folosi pagini web realizate in ASP.NET
- Pentru implementarea unor functionalitati AJAX se va utiliza setul de extensii ASP.NET AJAX dezvoltate de firma Microsoft
- Logica de business a aplicatiei se va scrie utilizand clase cu extensia .cs, limajul de programare C#.
- Se vor utiliza Servicii Web pentru realizarea functionalitatii aplicatiei cum ar fi trimiterea unor notificari prin e-mail.

4.3 Arhitectura pe 3 nivele

Ideea proiectarii pe 3 nivele este ca functionalitatea unei aplicatii poate fi impartita in trei nivele principale (ex. Figura 4.1 prezinta diagrama conceptuala a arhitecturii pe 3 nivele).

Arhitectura pe trei layere este o arhitectura in care interfata utilizator, regulile de business, stocarea datelor si accesul la date sunt dezvoltate si mentinute ca module independente.

- Nivelul de prezentare
  - Furnizeaza interfata utilizator a aplicatiei
  - Este primul nivel al arhitecturii
  - Afisaza controale, primeste si valideaza intrarile utilizatorilor. Toti manipulatorii de evenimente din pagina web sunt in primul nivel.
- Nivelul de business
  - Al doilea nivel este nivelul de business, unde se gaseste logica specifica aplicatiei.
  - Acest nivel, numit si domain layer este de obicei compus dintr-un numar de componente implementate utilizand limbaje de programare permise de .NET. Pentru aplicatia noastra s-a folosit limbajul C#.
  - Nu implica detalii .NET generice cum ar fi conectarea la o baza de date.
- Nivelul de acces la date
  - Al treilea nivel este nivelul de acces la date care va furniza acces la sisteme externe cum ar fi bazele de date (13)
  - Acest nivel conține două componente: Data Acces și Data Source. Data Access este realizat prin intermediul tehnologiei LLBLGEN Pro Runtime

Framework. Cu ajutorul acesteia se realizează accesul la baza de date. Data Source reprezintă baza de date propriu-zisă. Realizarea bazei de date s-a facut cu Microsoft SQL Server 2008 R2.

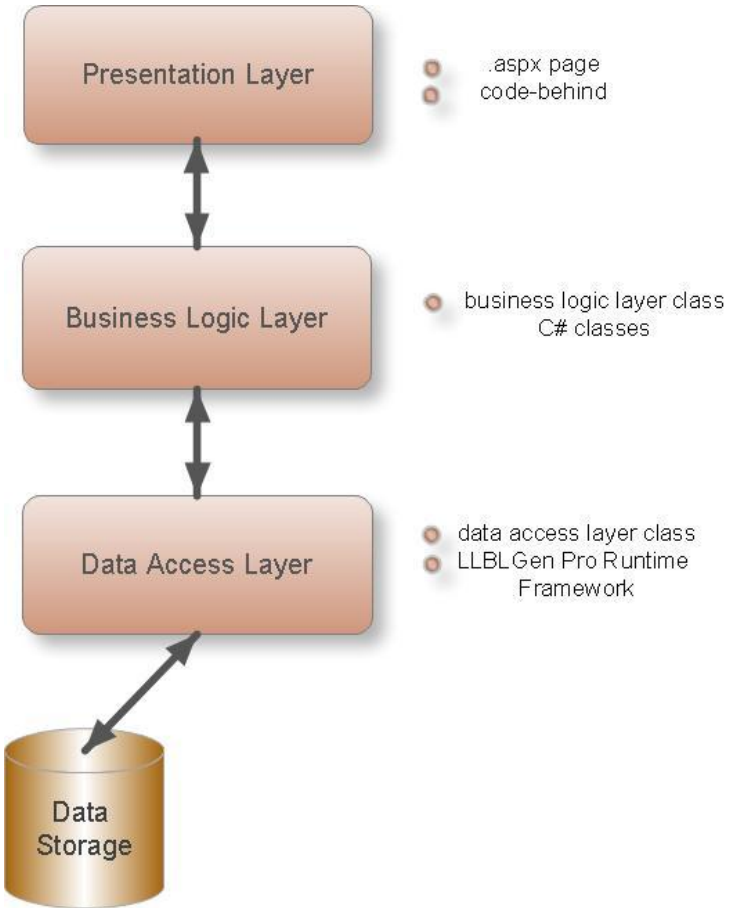

Figura 4.1 – Arhitectura pe 3 nivele

Un detaliu important in ceea ce priveste proiectarea pe 3 nivele este ca informatia circula doar de la un nivel la un altul adiacent. Asadar, codul paginii web nu ar trebui sa se conecteze direct la baza de date sau nivelul de acces la baza de date pentru a sustrage informatii. In schimb, ar trebui sa treaca printr-o componenta din nivelul de business care se conecteaza la nivelul de acces la date si sa returneze data.

4.4 .NET Framework

.NET Framework este un grup de mai multe tehnologii (14) :

- Limbajele .NET : Acesta include Visual Basic, C#, F# si C++. Limbajul folosit de aplicatia de consultatii online va fi C#.
- Common Language Runtime (CLR): Acesta este motorul care executa toate programele .NET si ofera servicii automate pentru aceste aplicatii, cum ar fi verificarile de securitate, managementul memoriei si optimizare.
- Biblioteca de clase .NET Framework: Colecteaza mii de bucati de functionalitati predefinite care pot fi adaugate in aplicatia proprie. Aceste caracteristici sunt uneori organizate in seturi de tehnologii (de exemplu WPF, tehnologie pentru crearea interfete utilizator desktop).

- ASP.NET: Acesta este motorul care gazduieste aplicatiile web create cu .NET si suporta aproape orice functie din clasa de biblioteci .NET Framework. ASP.NET include, de asemenea, un set de servicii specifice pentru web, cum ar fi autentificare securizata si stocare de date.
- Visual Studio: Acest instrument de dezvoltare contine un set bogat de caracteristici de productivitate si debugging. Include framework-ul .NET complet, fara a necesita download separat.

Uneori separarea dintre aceste componente nu este clara. De exemplu, termenul ASP.NET este folosit uneori intr-un sens ingust pentru pentru a face referire la portiunea de biblioteci .NET folosite pentru paginile web. Pe de alta parte, ASP.NET se refera la intregul subiect al aplicatiilor web .NET care includ limbajele .NET si o multime de piese fundamentale ale bibliotecii de clase care nu sunt specifice web.
.NET Framework 4 este proiectat astfel incat sa satisfaca urmatoarele obiective:

- Sa ofere un mediu de programare orientat-obiect consistent chiar daca codul obiectului este stocat si executat local, executat local, dar distribuit prin Internet sau executat la distanta.
- Sa ofere un mediu de executie a codului care: sa minimizeze implementarea software si conflictele de versiune, sa promoveze executia sigura a codului, inclusiv a codului creat de o parte semi-sigura sau necunoscuta, elimina probleme de performanta a mediilor de script sau interpretate
- Sa faca experienta programatorului consistenta peste o varietate de tipuri de aplicatii
- Sa construiasca toate comunicatiile la standardele industriei pentru a asigura codul bazat pe .NET Framework ca poate fi integrat cu orice alt cod

4.4.1 Limbajul C#

Limbajul C# este un limbaj dezvoltat de firma Microsoft. .NET Framework se imbarca cu doua limbaje de baza utilizate pentru construirea de aplicatii ASP.NET: C# si VB. Limbajul C# este utilizat in aplicatia noastra pentru descrierea operatiilor apartinand nivelului logicii de business si in parte de code-behind a nivelului de prezentare, dar si in nivelul de acces la date autogenerat de instrumentul LLBLGen Pro.
Limbajul C# permite programarea orientata obiect pe care se bazeaza si aplicatia de monitorizare a pacientilor, dar , la modul genereal, acest limbaj permite si programarea structurata si modulara.

Principiile de baza ale programarii obiectuale, cum ar fi incapsularea, mostenira , polimorfismul, sunt elemente fundamentale ale programarii C#.
Structura unui program C# conform (15) :

- O aplicatie C# este formata din una sau mai multe clase, grupate in spatii de nume(namespaces). .NET Runtime are un namespace unificat pentru toate informatiile programului(metadata). Clauza” using System” este o metoda de a referi clasele care sunt sistemul namespace astefel incat sa poate fi utilizate faa a fi nevoie sa se puna System in fata numelui tipului.
- Doua clase pot avea acelasi nume cu conditia sa fie definite in namespace-uri diferite.
- O clasa este formata din date si metode (sau functii). Apelarea unei metode in cadrul clasei in care a fost definita aceasta presupune specificare anumelui metodei. O metoda definita intr-o clasa poate fi invocata dintr-o alta clasa prin specificarea clasei urmata de caracterul ‘.’(punct), iar dupa acest caracter apere numele metodei.

- Pentru a facilita cooperarea mai multor programatori la realizarea unei aplicatii complexe, exista posibilitatea de a segmenta aplicatia in mai multe fisiere numite assemblies.

4.4.2 Common Language Runtime (CLR)

Este motorul care accepta toate limbajele .NET. CLR ruleaza doar codul limbajelor intermediare (toate limbajele .NET sunt compilate intr-un alt limbaj de nivel mai jos inainte de executia codului), ceea ce inseamna ca nu stie care limbaj .NET a fost utilizat original. Runtime-urile nu sunt nimic nou insa, CLR este cel mai ambitios runtime al Microsoft pana in prezent. Nu doar ca executa cod CLR dar ofera si un set de servicii conectate cum ar fi verificarea codului, optimizare si destionarea de obiecte.
Toate codurile .NET ruleaza in interiorul CLR indiferent daca se ruleaza o aplicatie Windows sau un serviciu web. De exemplu, cand un client cere o pagina ASP.NET, servicul ASP.NET ruleaza in interiorul mediului CLR, executa codul si creaza o pagina HTML finala pentru a o trimite clientului.
Erori mai putine: Toate categoriile de erori sunt imposibile cu CLR. De exemplu poate preveni multe greseli de memorie care sunt posibile pentru limbajele de nivel mai jos ca C++.
Trei aspecte care sunt adesea invocate de programatori:

- Performanta: O aplicatie ASP.NET tipica este mult mai rapid decat o aplicatie comparabila ASP, deoarece codul ASP.NET este compilat pe masina inainte de a fi executat.
- Transparenta codului: Limbajul intermediar (IL) este mult mai usor de dezasamblat, ceea ce inseamna ca daca se distribuie o aplicatie compilata sau o componenta, alte programe vor avea un timp mai usor de determinat cum functioneaza codul.
- Suport cross-platform discutabil: Datorita proiectului Mono dezvoltatorii pot folosi o implementare libera de .NET care ruleaza pe Linux, Unix, Mac OS si Windows.

4.4.3 Biblioteca de clase .NET

Biblioteca de clase .NET este un depozit de clase care ofera functionalitati prefabricate pentru orice, de la citirea unui fisier XML pana la trimiterea unui mesaj prin e-mail. Biblioteca de clase .NET este mai ambitios si mai cuprinzator decat orice alt framework de programare.

Cele mai multe clase nu sunt specifice doar unui anumit tip de implementare: web sau Windows, ci pot fi folosite in diverse scenarii de programare. Aceasta include un set de clase de baza care definesc tipuri de variabile comune si clasele pentru accesul la date.

.NET Framework se ocupa, de exemplu, cu probleme dificile cum ar fi tranzactii la baze de date si concurenta, asigurandu-se ca sute de mii de utilizatori sa poata accesa simpultan aceeasi pagina web, deodata.

4.4.4 Visual Studio

Ultima parte a .NET este instrumentul de dezvoltare (implementare) Visual Studio care ofera un mediu bogat unde se pot crea aplicatii avansate.
Cateva caracteristici pe care le include Visual Studio sunt:

- Construirea paginilor: Se pot construi pagini atractive foarte usor cu drag-and-drop utilizand web form designer-ul integrat in Visual Studio. Nu este necesara intelegerea HTML.

- Detectarea automata a erorilor: Este o calitate de real folos oferita de Visual Studio care detecteaza si raporteaza erorile inainte de a rula aplicatia. Acest mod de lucru poate imbunatatii considerabil timpul necesar finalizarii unui proiect. Aceasta detectare se manifesta prin sublinierea problemelor potentiale.
- Instrumentul de debugging: Este poate cel mai important instrument oferit de Visual Studio care ne permite sa vedem codul in actiune si sa urmarim continutul variabilelor. Aplicatia web poate fi testata la fel de usor ca orice alt tip de aplicatie deoarece Visual Studio are un server web „incorporat" care lucreaza doar pentru depanare.
- IntelliSense: Visual Studio ofera completarea declaratiilor pentru obiectele recunoscute si listeaza automat informatii cum ar fi parametrii functiilor ca ponturi ajutatoare.

4.4.5 ASP.NET 4

ASP.NET este tehnologia Microsoft care permite dezvoltarea de aplicatii web si servicii web , utilizand platforma Microsoft .NET cu toate beneficiile sale.

Caracteristici ASP.NET:

- Paginile cu extensia .aspx si codul acestora din fisierul aspx.cs sunt dezvoltate intr-unul din limbajele platformei .NET (toate limbajele compatibile cu CLR) dintre care cele mai utilizate sunt C# si VB
- Este o tehnologie de tip server
- Orice pagina ASP.NET este formata din doua componente
    - Un fisier cu extensia .aspx – contine controale, stiluri, etc, intr-un limbaj declarativ foarte bogat care poate sa contina cod HTML, XML si cod client: Javascript, apeluri AJAX
    - Code-behind – Un fisier cu extensia aspx.cs care contine cod executabil in limbajul de programare C#; Codul este executat pe server la fiecare cerere catre pagina respectiva
- ASP.NET ruleaza cod compilat, ceea ce creste performantelei aplicatiei web. (16)
- Aplicatiile ASP.NET lucreaza intotdeauna in conjunctie cu un server web – o piesa software care accepta cereri prin protocolul HTTP (Hypertext Transport Protocol) si continutul serverului.
- La rularea aplicatiei web in Visual Studio, se utilizeaza serverul de test care este construit(incorporat) in Visual Studio. Cand lansati website-ul catre un public mai larg, este nevoie de un web server adevarat , ca IIS.

4.4.5.1 Pagini Master ASP.NET

Paginile master permit crearea unei scheme consistente pentru paginile din aplicatie. O astfel de pagina este un fisier ASP.NET cu extensia .master cu o structura predefinita care poate contine text static, elemente HTML si controale server. Un master page defineste aspectul si un comportament standard pe care il vor avea toate paginile dorite din aplicatie. Se creaza apoi content pages individuale care vor contine continutul care se doreste sa se afiseze.

Cand se face o cerere la un content page, se realizeaza fuziunea dintre schema master page si continutul din content page.

Un master page ofera o functionalitate pe care programatorii obisnuiau sa o creeze prin copierea codului existent, text, si elemente de control in mod repetitiv.

Permit sa se centralizeze functionalitatea comuna a paginilor, astfel incat sa se realizeze actualizarile intr-un singur loc.

4.4.5.2 ASP.NET AJAX

AJAX se refera la utilizarea unui set specific de tehnologii browser pentru a construi pagini web. AJAX este acronimul pentru Asyncronous JavaScript si XML.
In esenta, ofera o tehnica pentru Javascript client-side sa faca apeluri ,in spate,la server si obtinerea datelor aditionale de care este nevoie, modificand anumite portiuni din pagina fara a cauza reincarcarea intregii pagini.

Beneficii aduse de AJAX in proiect:

- Face posibila crearea unor aplicatii web mai bune si mai receptive.
- Incurajeaza dezvoltarea framework-urilor si pattern-urilor, cum ar fi ASP.NET AJAX care ajuta proiectantii sa evite reinventarea rotii cand se efectueaza o sarcina obisnuita.
- Incurajeaza utilizarea tehnologiilor si caracteristicilor suportate de toate browser-ele web modern

Cateva probleme care pot sa apara :

- Butoanele “Back” si “Forward” din browser nu produc acelasi rezultat ca in cazul website-urilor clasice, doar daca aplicatia AJAX este programata sa suporte incarcarea si salvarea starilor.
- JavaScript poate fi dezactivat pe partea de client, ceea ce face ca aplicatia AJAX sa fie nefunctionala.

ASP.NET AJAX – este un framework AJAX puternic produs de Microsoft pentru dezvoltatorii ASP.NET.

- Cuprinde o multitudine de functionalitati testate permitand astfel construirea unor interfete solide
- Este un framework complex care include controale server ASP.NET (de exemplu un CheckBox) cu AJAX activat, la fel ca o biblioteca pe partea de client, foarte puternica.

“Universul” ASP.NET AJAX,dupa cum este prezentat in (12) ,este compusa din :

1. Microsoft AJAX Library – biblioteca Javascript client-side care ofera un API comun pentru toate browserle modern si suporta orice tehnologie web backend.
2. Extensii ASP.NET AJAX – include biblioteca Microsoft AJAX si un set de controale server cu AJAX activcare se integreaza bine cu biblioteca(, UpdatePanel, Timer, UpdateProgress, and AsyncPostbackTrigger)
3. ASP.NET AJAX Control Toolkit – este de controale si component gratuite care ajuta la obitinera CELEI mai marI valoarI de la extensiile ASP.NET AJAX.

Refresh partial

Tehnica cheie in aplicatiile web Ajax este reimprospaterea partiala. Folosind reimprospatarea partiala nu va fi nevoie ca intreaga pagina sa fie “posted back” (sau ceruta in totalitate) si reimprospatata in browser.

ScriptManager

Pentru a utiliza ASP.NET AJAX trebuie plasat acest controler web in pagina noastra care este creierul al ASP.NET AJAX. ScriptManager efectueaza urmatoare sarcina- adauga legaturile la bibliotecile Javascript ASP.NET AJAX .

Update Panel

Ideea de baza este impartirea paginii web in una sau mai multe regiuni distinct, fiecare fiind impachetata intr-un UpdatePanel invizibil.

Cand apare un eveniment intr-un control continut in UpdatePanel, si acest eveniment ar declasa (eng. trigger) in mod normal un postback (pagina se retransmite serverului) al intregii pagini, UpdatePanel intercepteaza evenimentul si efectueaza in schimb un callback asincron.

4.5 Concepte LLBLGen Pro Runtime Framework

LLBLGen Pro este un sistem care ajuta la construirea unei aplicatii n-tier foarte usor si repede. Pentru a putea face acest lucru, este necesara o familiarizare cu unele concepte care formeaza baza sistemului si a codului sursa produs de sistem.
In sistemul LLBLGen Pro se definesc prima data definitiile pentru Entitati, Typed List, Typed Views si , daca se doreste se pot include proceduri stocate existente, apeluri Stored Procedure. La sfarsitul proiectarii aceste definitii vor fi folosite pentru a produce cod.

Concepte de baza :

- Entitati, typed lists si typed views – Aceste elemente sunt blocurile construite ale codului pe care il proiectam cu sistemul LLBLGen Pro. Proiectantul sistemului LLBLGen Pro permite proiectarea acestor elemente inainte de a fi utilizate pentru generarea codului.

- Maparea O/R – Maparea Obiect-Relationala este termenul general pentru conceptul de creare a maparilor intre tabele sau vederi si campurile lor si, clasele Orientate-Obiect (OO) si campurile lor, pentru a putea reprezenta o inregistrare din tabele/ inregistrare din vedere intr-un program OO printr-o clasa. LLBLGen Pro utilizeaza tehnica de Mapare O/R in codul generat. Framework-urile de Mapare O/R mai sunt numite „ framework-uri de obiecte persistente”

- Mostenirea enititatii si modele relationale – Utilizarea constructorilor supertip/subtip in modelele relationale abstracte ofera, de asemenea, abilitatea de utilizare a mostenirii in LLBLGen Pro.

- Persistenta Stateless – Stateful si stateless sunt termeni care au o multime de definitii diferite, adesea contradictorii. Prin conventie, toate aplicatiile pe n-nivele ar trebui sa fie „stateless”. Asta inseamna ca aplicatia in totalitate nu este intr-un fel de „stare” prin tinerea informatiei in memorie, sau mai bine: nivelurile separate nu ar trebui sa fie intr-o anumita „stare” unde acestea tin informatii in memorie. „Starea” aplicatiei este detinuta de catre baza de date utilizata. In modelul „stateful”, asa numitii clienti-„fat” obisnuiti sa lucreze: utilizatorul lecreaza cu aplicatia completa pe masina lui, si starea aplicatiei a fost retinuta in memorie.
  Motivul pentru abordarea stateless este destul de simpla: multi-utilizator.

- Generarea codului bazata pe sarcina – LLBLGen Pro utilizeaza un framework de generare a codului consumator de template, bazat pe sarcina. Acest sistem este foarte

puternic si flexibil, tocmai de aceea poate duce la confuzie si intrebari despre unde sa se modifice si ce pentru a obtine rezultatele dorite.

- Template si grupuri de template – Framework-ul LLBLGen Pro se opreste la doua Template grupuri ale fisierului template: „SelfServicing" si „Adapter „.

- In cele ce urmeaza va fi descris doar template-ul SelfServicing deoarece acesta a fost folosit pentru proiect:
    - Denumirea „SelfServicing" vinde de la faptul ca, clasele entity, typed list si typed view contin toata logica si datele care sa le ajute sa interactioneze cu mediul de stocare persistent. Asta inseamna ca nu este nevoie de o instanta obiect in plus, cum ar fi un broker, pentru a aduce data entity din stocarea persistenta sau salvarea datei in baza de date persistenta.
    - Cand sa se utilizeze SelfServicing :
        - Cand proiectul are ca tinta un singur tip de baza de date
        - Cand este nevoie de navigare prin modelul de obiecte intr-un scenariu de databinding si se doreste incarcarea datelor utilizand incarcarea intarziata(eng. Lazy loading)
        - Cand nu se doreste extinderea framework-ului
        - Cand se doreste sa se aiba logica de persistenta in clasele entity
        - Atunci cand nu se cere control al accesului la baza de date cu granulatie fina, cum ar fi directionarea de apel al altei baze de date.

4.6 Modelul bazei de date

SQL Server este un sistem de gestiune al bazelor de date relational (RDBMS-Relational Database Management System) de la firma Microsoft care este proiectat pentru mediul de afaceri. In acest model relational datele sunt organizate sub forma de tabele. SQL Server ruleaza pe T-SQL(Transact – SQL), un set de extensii de programare de la Sybase si Microsoft care adauga o multime de caracteristici standardului SQL, incluzand controlul tranzactiilor, exceptiilor si erori lor de manipulare, de prelucrare a iregistrarilor si variabile declarative. **(17)**

SQL(Structured Query Language) este un limbaj standard utilizat pentru interogari, modificari si gestionarea bazelor de date, iar mai tarziu a devenit factorul standard pentru accesarea bazelor relationale. SQL nu este de fapt un limbaj, pe cat un instrument de interogare a bazelor de date. Ne permite sa definim o baza de date relationala prin crearea si modificarea tabelelor (in acest sens, SQL este un limbaj de definire a datelor, sau DDL). De asemenea, ne permite sa spunem SQLServer-ului care informatii dorim sa le selectam(obtinem), inseram, modificam sau sterge (SQL – limbaj de manipulare a datelor (DML)). (18)

4.6.1 Microsoft SQL Server 2008 R2

In esenta sa, SQL Server 2008 R2 este un sistem de gestiune al bazei de date orientat spre clasa intreprinderilor, capabil sa execute de la o baza de date personala de doar cativa Megabytes ca marime, tinuta pe un dispozitiv Windows Mobile, pana la un sistem de gestiune al informatiilor unei baze de date multiserver, de ordinal terabytes. (19)
Componenta principal a SQLServer 2008 R2 este motorul bazei de date. Acesta ofera:

- Stocare sigura a datelor
- Rapiditate in obtinerea acestor date
- Acces consistent la date

- Controlul accesului la date prin securitate
- Forteaza regulile de integritate a datelor sa se asigure ca datele sunt fiabile(de incredere) si consistente.

Ideea generala in ceea ce priveste accesul consistent la date este de a permite unui singur client la un moment dat sa schimbe date si sa previna citirea datelor de catre altii atat timp cat au loc modificari la datele respective.
Controlul accesului ofera securitate la nivele multiple. Securitatea este executat la server, baza de date, schema si la nivelul obiectelor.

4.6.2 SQL Server Management Studio (SSMS)

Este un instrument oferit de SQL Server 2008 R2 pentru gestionarea si administrarea "mororului" bazelor de date SQL Server si a altor componente. SSMS ofer o interfata utilizator de unde se poate efectua sarcini de gestionare a bazelor de date si de unde pot fi coordonate.
Furnizeaza o singura interfata de unde orice server poate fi gestionat. O noutate in SSMS, descrisa in (19), pentru SQL Server 2008 R2 este ca aduce imbunatatiri care sunt axate pe proiectant si include: IntelliSense in Query Editor, un debugger integrat Transact-SQL (T-SQL) si o optiune de executie Multiserver Query. IntelliSense ajuta la completarea codului T-SQL pe masura ce se scrie.

4.7 Structura logica a aplicatiei

In cele ce urmeaza (figura 4.2) este prezentata structura logica a aplicatiei respectand modelul de arhitectura format din 3 nivele, amintite si la inceputul capitulului, dupa cum urmeaza: un nivel pentru interfata utilizator, un al nivel pentru logica de business a aplicatiei si un nivel de acces la date.

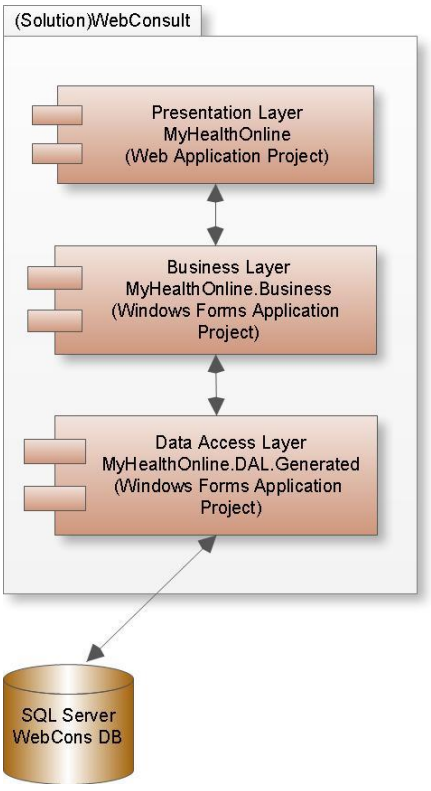

Figura 4.2 Structura logica a aplicatiei

Nivelul de prezentare va fi un proiect Web Application care va contine toata partea de interfata a sistemului. Nivelul al doilea este format dintr-un proiect de tipul Windows Forms Application care va contine toate operatiile de logica ale sistemului, iar nivelul de acces la date este un proiect auto-generat de catre programul LLBLGen Pro si contine clase de entitati, colectii si alte clase ajutatoare, de relatie, etc

Cele 3 proiecte independente vor stabili urmatoarele dependente intre ele:

- Proiectul MyHealthOnline depinde de proiectul MyHealthOnline.Business
- Proiectul MyHealthOnline.Business depinde de proiectul MyHealthOnline.DAL.Generated

Asadar, se poate observa ca nivelul de prezentare nu stie de nivelul de acces la date, datele procesate ajung la acesta prin intermediul nivelului de business care foloseste nivelul de acces la date pentru a realiza toate operatiile necesare nivelului de prezentare.
Similar, nivelul de acces la date nu stie nimic de nivelul de prezentare, datele ajung la acesta tot cu ajutorul nivelului de business.

5. Proiectare de detaliu si implementare

Capitolul 5 are ca scop documentarea sistemului software dezvoltat astfel incat dezvoltarea si mentenanta sistemului in timp sa se poata realiza cu usurinta.

5.1 Tehnologii utilizate

Pentru implementarea acestei aplicatii am utilizat in cea mai mare parte tehnologii Microsoft.

Accesul la date s-a realizat prin utilizarea framework-ului LLBLGen Pro pentru a simplifica modul de acces la date.

LLBLGen Pro este cel mai avansat instrument de mapare Object-Relational (O/RM) pentru .NET fiind foarte flexibil prin directionarea catre 4 tipuri de framework-uri:

- Entity Framework (v1 si v4)
- Nhibernate
- Linq to SQL
- LLBLGen Pro Runtime Framework

Pentru proiectul nostrum am ales framework-ul LLBLGen Pro Runtime Framework deoarce:

- Este cel mai matur mapper O/R (pe piata din 2003)
- Suporta un numar foarte mare de baze de date : SQL Server (MSDE, SqlServer 2000/2005/2008/2008R2/Express, SqlCE Desktop), Oracle, MySQL, etc)
- Genereaza cod pentru .NET 2.0, .NET 3.0, .NET 3.5 or .NET 4.0
- Fisierele proiectelor generate VisualStudio.NET sunt compatibile cu Visual Studio.NET 2005/2008/2010

Utilizand LLBLGen Pro Runtime Framework timpul necesar dezvoltarii este drastic scazut, un alt avantaj este ca suporta atat dezvoltarea Model-first, cat si Database-first.
Pentru proiect am folosit cel mai mult dezvoltarea Model-first, prin crearea bazei de date prima data si apoi generarea codului sursa.

Baza de date este realizata in SQL Server 2008 R2 care vine impreuna cu SQL Server Management Studio (SSMS). Motivul principal pentru utilizarea acestei versiuni a fost compatibilitatea cu Windows 7, dar si pentru noutatile pe care le aduce SSME care include IntelliSense in Query Editor si care pot usura munca dezvoltatorului in ceea ce priveste scrierea scripturilor pentru baza de date.

Alte tehnologii utilizate au fost :

- .NET Framework
- ASP.NET
- limbajul utilizat este C#

Toate aceste tehnologii utilizate in realizarea aplicatiei au fost detaliate in capitolul 4.
Aplicatia utiizeaza, de asemenea, servicii web si functionalitati oferie de ASP.NET AJAX.

5.2 Modelul cazurilor de utilizare

Un caz de utilizare este o tehnica de modelare folosit pentru a descrie ce va face un sistem nou sau ce face deja un sistem existent. (21)

O diagrama a cazurilor de utilizare prezinta o colectie de cazuri de utilizare si actori care:

- ofera o descriere generala a modului in care va fi utilizat sistemul
- realizeaza o descriere din punct de vedere functional , a sistemului, ansamblul tuturor cazurilor de utilizare şi utilizatorii acestora (actorii) formând modelul cazurilor de utilizare
- arata cum interactioneaza unul sau mai multi actori
- pot constitui o baza pentru realizarea testelor de verificare a aplicatiei

Actorii sunt entitati care interactioneaza cu sistemul, adica realizeaza schimburi de informatii cu acesta.

In sistemul acesta au fost identificati trei actori principali : adminstratorul, specialistul si pacientul.

Utilizatorii autentificati sunt, de fapt, actorii principali ai sistemului. Acestia au acces la functionalitatile oferite de sistem, in functie de rolul fiecaruia.

Utilizatorii neautentificati, si cei care nu au un cont creat in sistem, nu pot beneficia de functionalitatile puse la dispozitie de sistemul in cauza.

5.2.1 Diagrama cazurilor de utilizare

Vor fi prezentate diagrame de cazuri de utilizare pentru fiecare actor care interactioneaza cu acest sistem.
Diagramele cazurilor de utilizare prezentate sunt concepute pentru utilizatorii inregistrati in sistem si autentificati.

Pentru un pacient care nu are un cont creat, poate fi inregistrat in acest sistem pentru a beneficia de functionalitatile oferite de sistemul de monitorizare al pacientilor in urmatoarele cazuri:

1. Un pacient poate beneficia de crearea unui cont, daca are stabilita o consultatie fata in fata cu specialistul si cere in mod explicit sa fie inregistrat
2. Specialistul poate decide daca pacientul respectiv necesita o monitorizare prin intermediul acestui sistem si il inregistreaza daca este nevoie .
3. Pacientul poate fi inregistrat in sistem si de catre administratorul sistemului, daca este nevoie.

Diagrama cazurilor de utilizare pentru administrator

Primul actor este Administratorul care se ocupa in sistem de managementul utilizatorilor si de crearea unor chestionare noi. In figura 5.1 se descriu cazuri de utilizare pentru administrator.

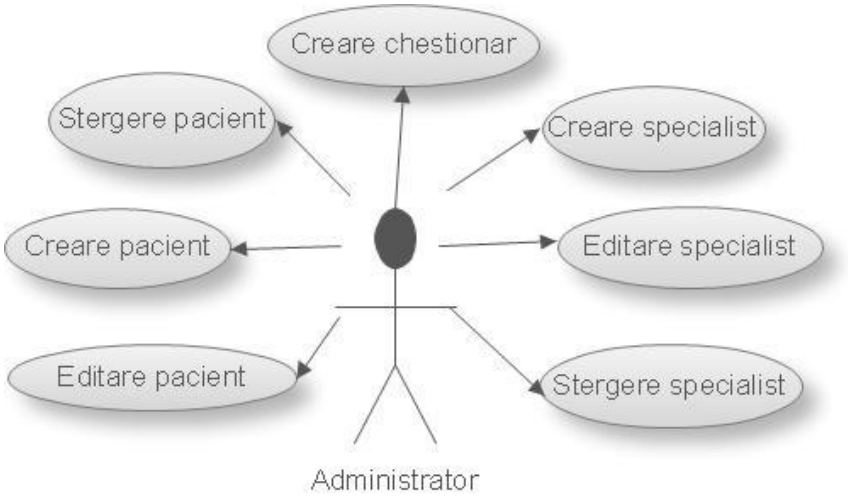

Figura 5.1 Cazuri de utilizare Administrator

Cazul de utilizare “Creare chestionar” : Administratorul are posibilitatea construirii unui nou chestionar care sa poata fi apoi utilizat de catre specialist
Cazul de utilizare “Creare pacient” si “Creare specialist” : Administratorul are posibilitatea de a adauga noi pacienti si specialisti in baza de date;
Cazurile de utilizare “Editare pacient” si “Editare specialist” : Administratorului ii este permis sa modifice informatiile pacientului sau specialistului, fie daca acestia cer acest lucru, sau daca administratorul detecteaza eventuale greseli ale profilurilor acestora.
Cazurile de utilizare “Sterge pacient” si “Sterge specialist” : Doar administratorului ii este permis sa stearga un specialist sau un pacient.

Diagrama cazurilor de utilizare pentru Pacient

Un alt actor al sistemului este pacientul care este inregistrat in sistem si autentificat. Acesta are puse la dispozitie toate functionalitatile pentru un astfel de utilizator. In figura urmatoare (fig. 5.2) se vor putea vedea cazurile de utilizare pentru un pacient inregistat in sistem si logat.

Figura 5.2 Cazuri de utilizare pentru Pacient

Pacientul autentificat are puse la dispozitie urmatoarele cazuri de utilizare: isi poate vizualiza toate consultatiile, poate raspunde la intrebarile puse de specialistul care il consulta, isi poate vedea tot istoricul de mesaje pentru fiecare dintre consultatiile avute.
In cazul in care specialistul i-a trimis un chestionar, acesta trebuie sa-l completeze in totalitate, si de asemenea isi va putea vedea programarile pentru consultatii.

Diagrama cazurilor de utilizare pentru Specialist

Al treilea actor al sistemului este Specialistul. In figura 5.3 sunt prezentate cazurile de utilizare disponibile pentru acest utilizator. Rolul lui in sistem este foarte important deoarece el se ocupa de minitorizarea pacientilor, de initierea consultatiilor si totodata de inchiderea lor pentru fiecare pacient in parte. Acesta realizeaza cele mai multe actiuni in sistem.

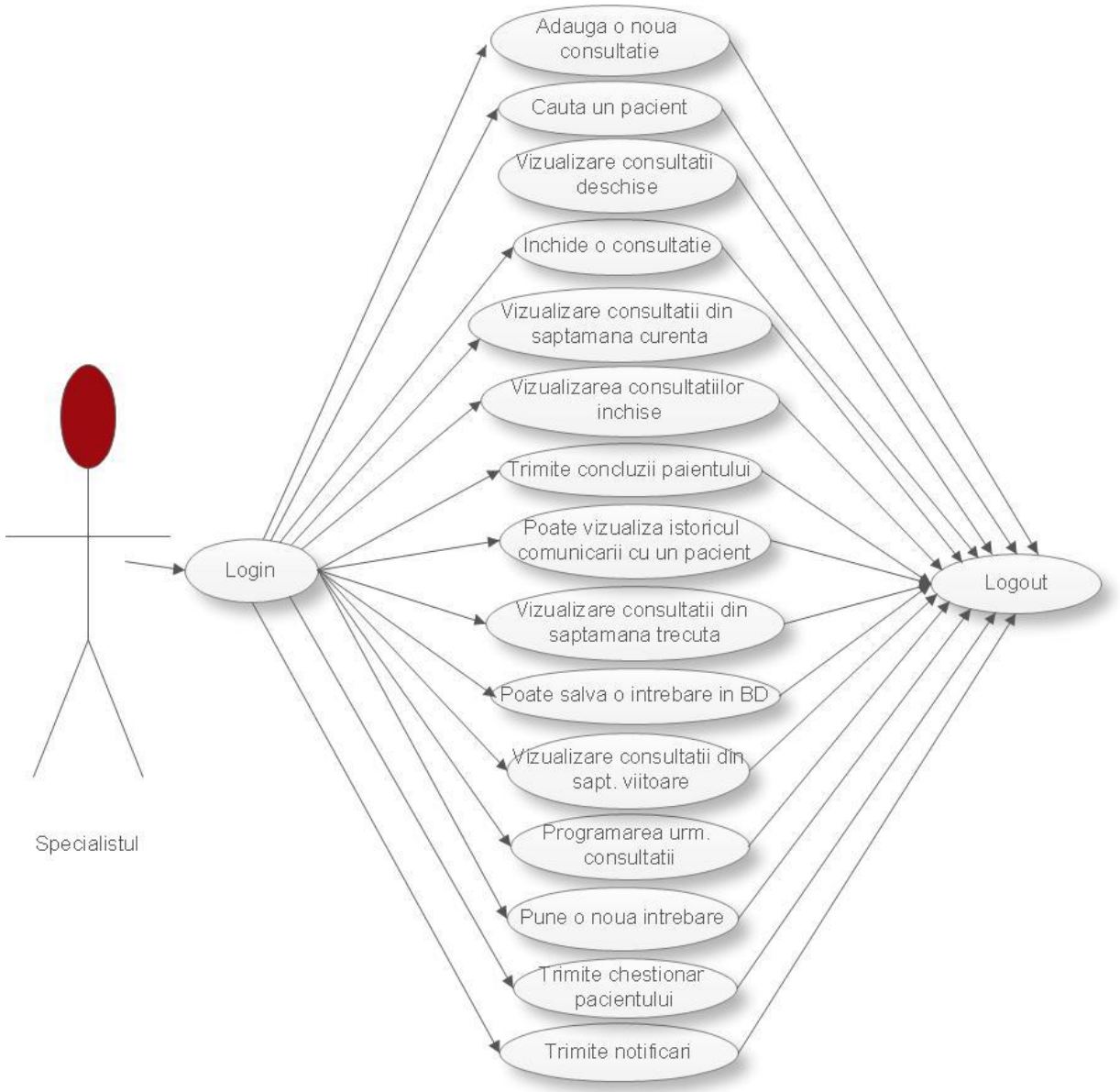

Figura 5.3 Diagrama cazurilor de utilizare pentru Specialist

5.3 Detalii de implementare

Pentru a descrie detaliile de implementare, vom utiliza Figura 4.2 care arata structura logica a aplicatiei. Pentru fiecare nivel din structura aplicatiei vor fi explicate in detaliu structurile folosite.

5.3.1 Structura bazei de date

Baza de date a aplicatiei este creata in SQL Server, modelul bazei de date este unul relational in care vom avea tabele legate intre ele prin intermediul unor constrangeri referentiale, numite si chei straine(eng. foreign key). Fiecare tabel are o cheie primara care va identifica in mod unic fiecare inregistrare din tabel.
Vom prezenta diagram bazei de date (figura 5.4) si apoi vom detalia rolul fiecarui tabel present in diagram cat si relatiile stabilite intre acestea.

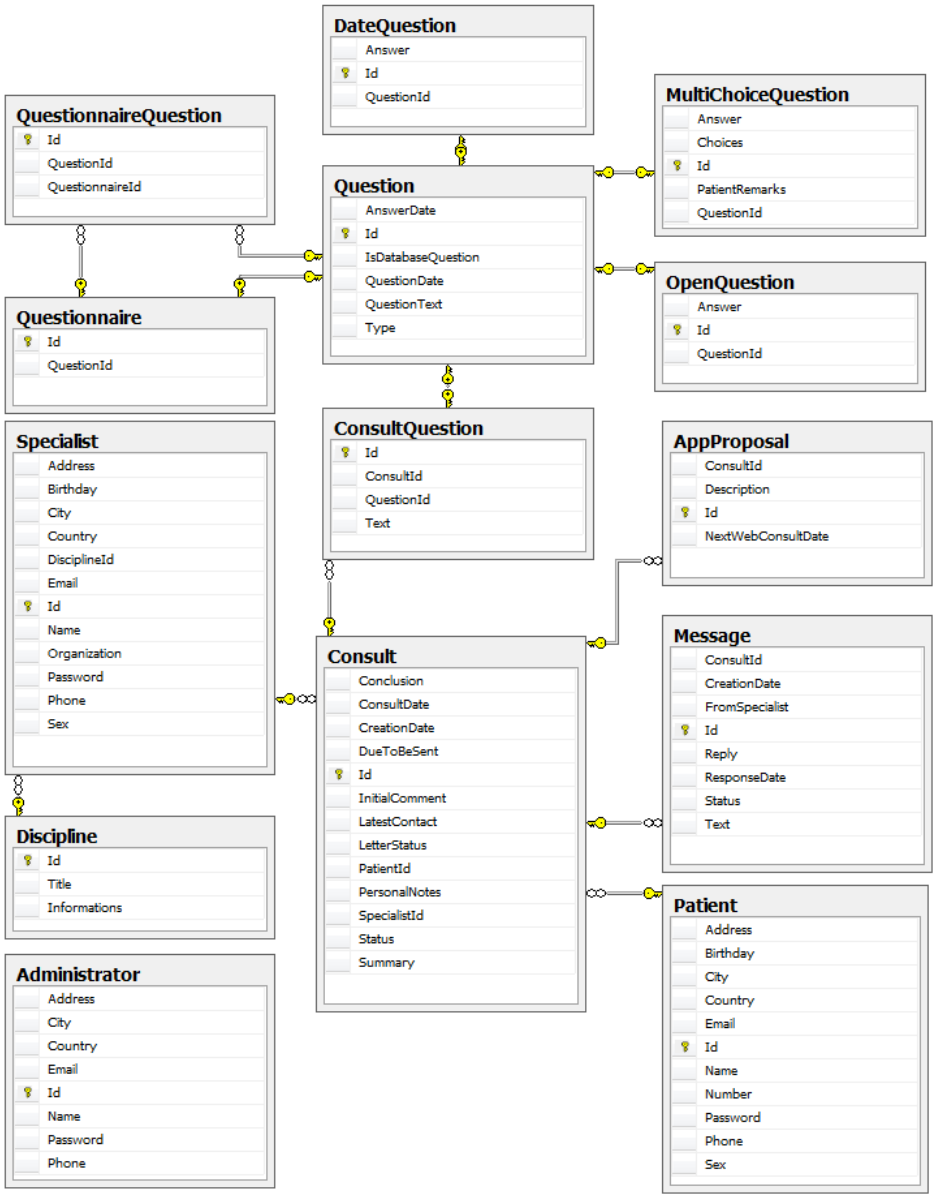

Figura 5.3 Diagrama bazei de date

Pentru fiecare dintre cele 3 tipuri de utilizator (Specialist, Pacient si Administrator) vor exista 3 tabele separate deoarece acestia corespund unor permisiuni diferite in aplicatie, iar campurile tabelelor sunt diferite pentru acestia si nu se putea utiliza , de exemplu, o tabela numita Utilizator si fiecarei inregistrari sa-i corespunda un anumit rol in functie de tipul utilizatorului. De asemenea, specialistul are o legatura cu tabela Discipline, iar celelate nu au nevoie de acest camp.

Tabela Administrator

Tabela Administrator este o tabela independent, neavand nici o constrangere referentiala cu alte tabele din baza de date.
Rolul acestei tabele este de a retine informatiile dorite si necesare pentru administratorul sau administratorii sistemului. In figura urmatoare apare tabela Administrator a bazei de date impreuna cu toate campurile acesteia.

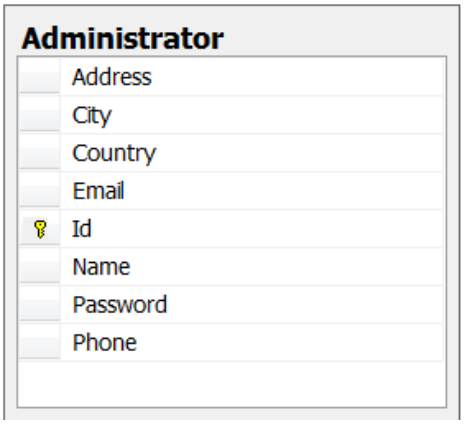

Figura 5.4 Tabela Administrator

Scriptul pentru crearea tabelei in baza de date este urmatorul:

```
CREATE TABLE [dbo].[Administrator]
(       Address varchar(255) NULL,  -- Adresa administratorului, campul poate fi null
        City varchar (255),  -- Orasul in care locuieste administratorul
        Country varchar(255),  --Tara in care locuieste administratorul
        Email varchar (255),  --Email-ul administratorului, acest camp este obligatoriu
        Id int IDENTITY PRIMARY KEY,  --id-ul este o cheie primara, unica
        Name varchar(255),  --numele administratorului
        Password varchar(255),  --parola
        Phone varchar (255),  --numarul de telefon al administratorului
)
GO
```

Campurile Email si Password sunt campuri obligatorii, deoarece acestea vor fi utilizate pentru logarea administratorului in sistem.
Campurile pentru adresa, oras si tara nu sunt neparat obligatorii, deoarece acestea nu se utilizeaza de catre sistem.

Tabela Specialist

Tabela Specialist este utilizata pentru a salva informatiile pentru toti medicii inscrisi in acest sistem. In figura 5.5 de mai jos se pot vedea toate campurile acestei tabele.

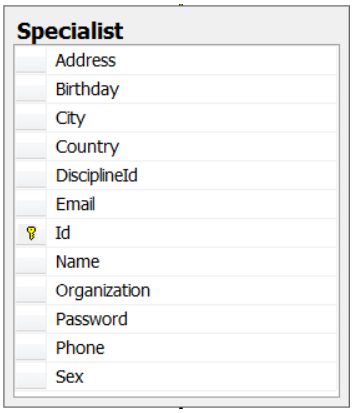

Figura 5.5 Tabela Specialist

Scriptul utilizat pentru crearea tabelei Specialist este:

```
CREATE TABLE [dbo].[Specialist]
(       Address varchar(255),-- adresa cabinetului specialistului
        Birthday varchar(10,   --data nasterii a specialistului
        City varchar(255),     -- orasul unde se afla specialistul
        Country varchar (255),-- tara in care locuieste specialistul
        DisciplineId int,      -- (FK)disciplina careia ii apartine specialistul
        Email varchar(255),    -- adresa de email a specialistului
        Id varchar(50) PRIMARY KEY, -- (PK) Id-cod unic parafa medic
        Name varchar(255),     -- numele specialistului (Nume Prenume)
        Organization varchar(255) , -- institutia din care face parte
        Password varchar(255), -- parola specialistului
        Phone varchar(255),    -- numarul de telefon
        Sex varchar(8)         -- sexul specialistului
)
GO
```

Campurile Email si Password sunt utilizate pentru logarea specialistului in sistem.
Intre tabela Disicipline si Specialist se stabileste o relatie de 1:n (o disciplina poate corespunde la mai multi specialisti, fiecare specialist apartine unei singure discipline).
Scriptul pentru crearea acestei constrangeri intre tabele se poate vedea mai jos:

```
ALTER TABLE [dbo].[Specialist]
        ADD CONSTRAINT fk_disciplineId  FOREIGN KEY
        ([DisciplineId])  REFERENCES [dbo].[Discipline] ( [Id]  )
GO
```

Tabela Discipline

Tabela Discipline este utilizata pentru a retine toate discipline pentru institutia care utilizeaza acest sistem si informatii despre fiecare disciplina existenta dupa cum se poate vedea si in tabelul 5.6.

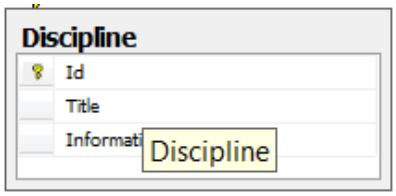

Figura 5.6 Tabela Discipline

Scriptul utilizat pentru crearea acestui tabel este:

```
CREATE TABLE [dbo].[Discipline]
(       Id int IDENTITY PRIMARY KEY,  -- (PK) un numar unic pentru fiecare inregistrare
        Title varchar(50) NOT NULL,          -- titlul sau numele disciplinei
        Informations text                    -- informatii despre fiecare disciplina
) GO
```

Tabela Patient

Tabela Patient este utilizata pentru a retine informatii despre pacientii care utilizeaza sistemul de consultatii online . Informatiile salvate in baza de date pentru fiecare pacient se pot vedea in figura 5.7 care reprezinta tabela Patient.

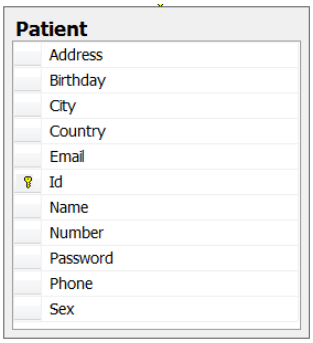

Figura 5.7 Tabela Patient

Scriptul utilizat pentru crearea tabelei Patient este:

```
CREATE TABLE [dbo].[Patient]
(       Address varchar(255), -- Adresa pacientului
        Birthday datetime, -- Data nasterii
        City varchar(255), -- Orasul in care locuieste
        Country varchar(255), -- Tara in care locuieste
        Email varchar(255), -- Emailul pacientului (gmail)
        Id int IDENTITY PRIMARY KEY, -- (PK) cheia primara
        Name varchar(255), -- numele pacientului (Nume Prenume)
        Number int NOT NULL, -- CNP-ul pacientului
        Password varchar(255), -- parola pacientului
        Phone varchar(255), -- numarul de telefon al pacientului
        Sex varchar(8)  -- sexul pacientului
) GO
```

Tabela Consult

Este o tabela care tine minte toate consultatiile facute pentru toti pacientii. Se stabileste o relatie de n:m intre tabela Specialist si Patient. Intre tabelele Specialist – Consult se stabileste o relatie de 1:n, iar intre Patient – Consult se stabileste o relatie de 1:m. Altfel spus, un specialist poate sa aiba mai multe consultatii (sau mai multe consultatii pot corespunde unui specialist) si, un pacient poate sa aiba mai multe consultatii (mai multe consultatii pot corespunde unui pacient).
In tabela 5.8 de mai jos se pot vedea toate campurile tabelei Consult.

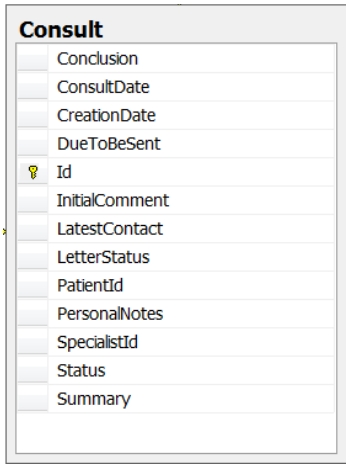

Figura 5.8 Tabela Consult

Scriptul pentru crearea tabelei :

```
CREATE TABLE [dbo].[Consult]
(       Conclusion varchar(50) NOT NULL,  -- retine concluzia pe care specialistul o trimite la inchiderea unei consultatii sau la crearea unei noi programari
        CreationDate datetime,  -- data crearii consultatiei de catre specialist
        Id int IDENTITY PRIMARY KEY,  --PK
        LatestContact datetime,  -- ultimul contact pe care specialistul l-a avut cu pacientul, acest camp va fi modificat la fiecare intrebare si raspuns
        PatientId int,  --id-ul pacientului care trebuie consultat
        PersonalNotes varchar(1024),  --note personale pe care le poate scrie specialistul la crearea unei consultatii
        SpecialistId varchar(50),  -- id-ul specialistului care face consultatia
        SpecialistName varchar(255),  --numele specialistului care face consultatia
        Status varchar(255),   --statusul consultatiei
) GO
```

Adaugarea referintelor la tebelele Patient si Specialist:

```
ALTER TABLE [dbo].[Consult]
        ADD CONSTRAINT fk_specialist_id FOREIGN KEY
        ([SpecialistId])        REFERENCES [dbo].[Specialist] ([Id])
GO
ALTER TABLE [dbo].[Consult]
        ADD CONSTRAINT fk_pacient_id FOREIGN KEY
        ( [PatientId] ) REFERENCES [dbo].[Patient] ([Id])
GO
```

Tabela AppProposal

Acesta tabela este utilizata pentru crearea unei noi programari pentru o consultatie. Campurile tabelei se pot vedea cu usurinta in figura 5.9.

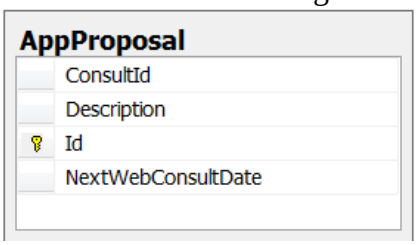

Figura 5.9 Tabela AppProposal

Scriptul utilizat pentru crearea tabelei este :

```
CREATE TABLE [dbo].[AppProposal]
(       ConsultId [int] NULL, -- (FK); Id-ul consultatiei pentru care se face o programare
        Description varchar(1024), -- o descriere pe care o trimite specialistul atunci cand face o noua consultatie
        Id int IDENTITY PRIMARY KEY,
        NextWebConsultDate datetime –data pe care o comunica specialistul pacientului pentru continuarea consultatiei
)
GO
```

Constrangerile de chei straine:

```
ALTER TABLE [dbo].[AppProposal]
        ADD CONSTRAINT FK_CONSULT FOREIGN KEY
        ( [ConsultId]) REFERENCES [dbo].[Consult] ( [Id] )
GO
```

Intre tabela Consult si AppProposal se stabileste o relatie de 1:n (unu la mai multi), altfel spus, o consultatie poate avea mai multe programari (appProposal) sau pot exista mai multe programari pentru o consultaie.

O vedere pentru modelul Intrebari este descries in urmatoarea imagine (Figura 5.10), diagrama generate cu programul LLBLGen Pro..

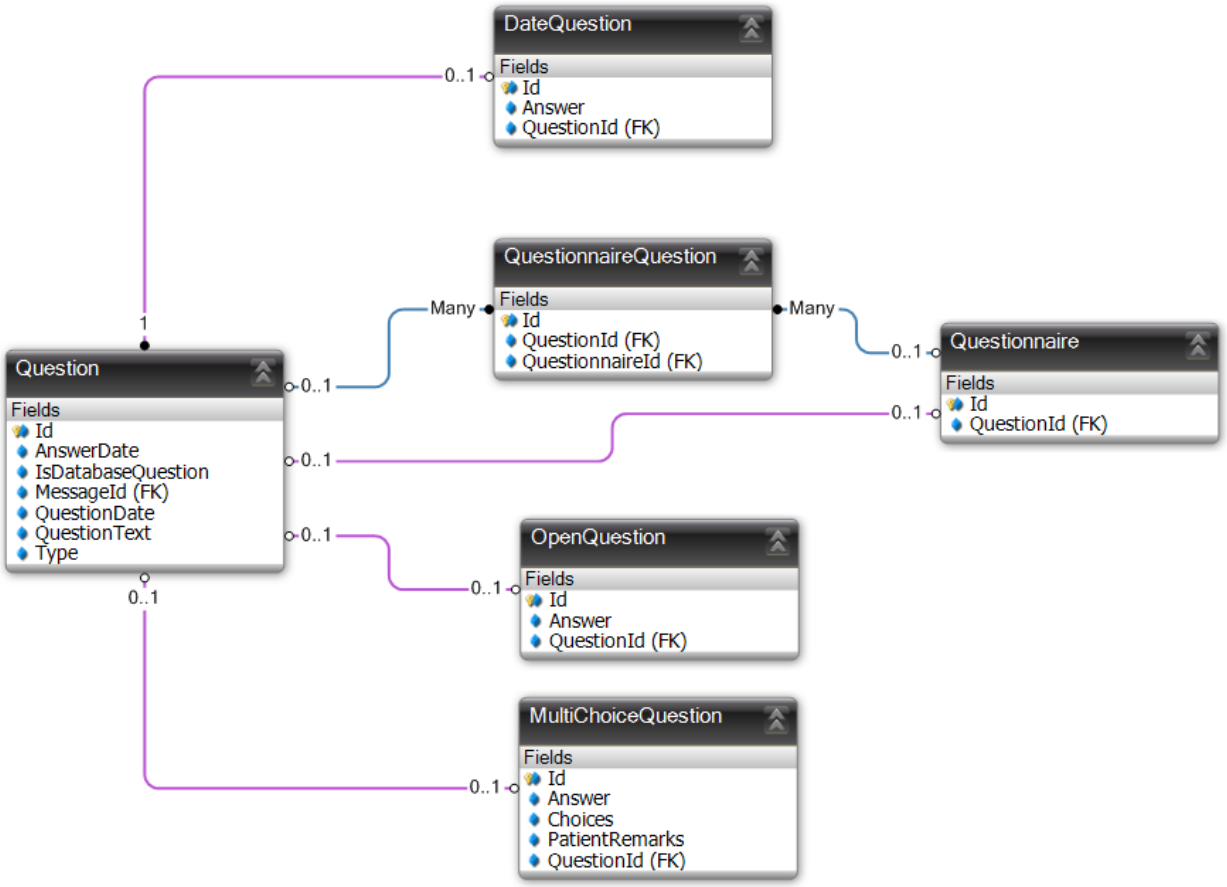

Figura 5.10 Vedere a modelului Intrebari

Tabela ConsultQuestion

Tabela ConsultQuestion prezentata si in figura 5.10 este utilizata pentru a adauga intrebari unui consult.
Intre tabela Consult si Consult Question se stabileste o relatie de 1:n, o consultatie poate sa aibamai multe intrebari sau , altfel spus, pot exista mai multe intrebari pentru o consultatie. Tabelul este prezentat mai jos in figura 5.11:

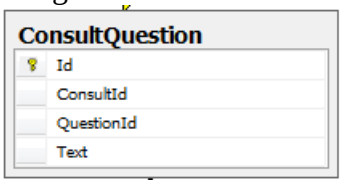

Figura 5.11 Tabela ConsultQuestion

Codul pentru crearea tabelei si a relatiei cu tabela consult :

```
CREATE TABLE [dbo].[ConsultQuestion]
(  Id  int IDENTITY PRIMARY KEY,--(PK)Identificatorul unic al fiecarei rand
   ConsultId int,  --(FK) Id-ul consultatiei pentru care exista intrebari
   QuestionId int, --(FK) Id-ul intrebarii
   Text varchar(1024) -- o descriere care poate fi adaugata
)
GO
```

Definirea constrangerilor de cheie referntiala:

```
ALTER TABLE [dbo].[ConsultQuestion]
        ADD CONSTRAINT [FK_Consult_id] FOREIGN KEY
        (QuestionId) REFERENCES [dbo].[Question] ([Id])
GO
```

Tabela Question

Tabela Question retine toate intrebarile care sunt adresate pacientilor.
Stabileste o relatie de 1:n cu tabela Consult. O consultatie poate sa aiba mai multe consultQuestion-uri. Intre tabela ConsultQuestion si Question se stabileste o relatie de 1:1, fiecarei entitati din ConsultQuestion ii coespunde o singura entitate din Question.
In figura urmatoare (fig. 5.12) este prezentata tabela Question cu toate campurile sale.

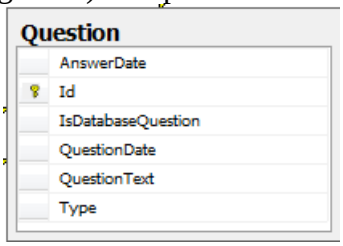

Figura 5.12 Tabela Question

Scriptul pentru crearea tabelei si explicarea semnificatiei fiecarui camp:

```
CREATE TABLE [dbo].[Question]
(       Id int IDENTITY PRIMARY KEY, -- (PK) Id-ul care identifica fiecare intrebare in mod unic
        IsDatabaseQuestion bit NOT NULL, -- Atunci cand specialistul adauga o intrebare, acesta are posibilitatea sa o salveze sau nu in baza de date( utilizand un checkbox). Daca salveaza intrebarea (IsDatabaseQuestion=true), atunci cand va selecta sa vizualizeze "Question from Database" ii vor aprea aceste intrebari cu IsDatabaseQuestion=true(sau 1)
        QuestionText varchar(50), -- textul intrebarii (intrebarea in sine)
        Type varchar(50)        -- se va retine tipul fiecarei intrebarii: Open, MultiChoice, Date, ...
)
GO
```

Tabela OpenQuestion

Acesta tabela retine toate raspunsurile la intrebarile compuse de Specialist pentru pacienti.
Intre tabela Question in care se afla toata colectia de intrebari si tabela OpenQuestion care contine raspunsurile la intrebarile care au type="Open" in tabela Question, se stabileste o relatie de 1:1.
O entitate(inregistrare) din tabela OpenQuestion corespunde unei singure inregistrari din tabela Question. Altfel spus, fiecare raspuns continut in tabela OpenQuestion corespunde unei singure intrebari retinute in tabela Question.
Campurile tabelei OpenQuestion se pot vedea in figura 5.13

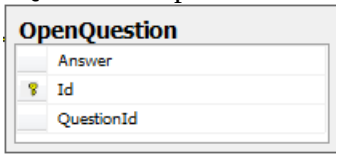

Figura 5.13 Tabela Open Question

Scriptul pentru crearea tabelei impreuna cu constrangerile de cheie straina:

```
CREATE TABLE [dbo].[OpenQuestion]
(       Answer varchar(255),  --raspunsul intrebarii deschise
        Id int IDENTITY PRIMARY KEY, --(PK) identificat inregistrare
        QuestionId int –(FK) referinta tabela Question
)
GO
ALTER TABLE [dbo].[OpenQuestion]
        ADD CONSTRAINT [FK_OpenQ] FOREIGN KEY
        ( [QuestionId] )REFERENCES [dbo].[Question]([Id])
GO
```

Tabela MultiChoiceQuestion

Acesta tabela este uilizata pentru intrebarile care au mai multe variante de raspuns, prestabilite de specialist.
Intre tabela Question si MultiChoiceQuestion exista o relatie de 1:1, pentru fiecare intrebare din tablea Question exista o inregistrare cu raspunsuri in tabela MultiChoiceQuestion.
In figura 5.14 se pot vedea campurile acestei tabele, iar mai jos constrangerile de cheie straina.

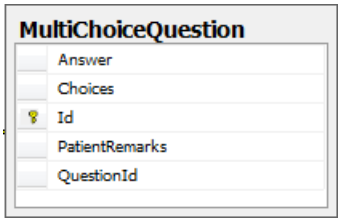

Figura 5.14 Tabela MultiChoiceQuestion

```
CREATE TABLE [dbo].[MultiChoiceQuestion]
(
        Answer varchar(255),  -- raspunsul dat
        Choices varchar(255),  --variantele de raspun
        Id int IDENTITY PRIMARY KEY,  -- (PK) identificatorul fiecarei inregistrari
        PatientRemarks varchar(255),  --pacientul poate sa adauge o remarca
        QuestionId int  -- (FK) cheie straina catre intrebarea din tabela Question
)
GO
ALTER TABLE [dbo].[MultiChoiceQuestion]
        ADD CONSTRAINT [FK_MultiChoiceQ] FOREIGN KEY
        ( [QuestionId] ) REFERENCES [dbo].[Question] ([Id])
GO
```

Tabela DateQuestion

Tabela DateQuestion este utilizata pentru a salva raspunsurile unor intrebari sub forma unei date. Intre tabela DateQuestion si Question se stabileste o relatie de 1:1; o intrebare din tabela Question poate avea ca raspuns o singura entitate din DateQuestion.
Scriptul pentru acesta tabela impreuna cu constrangerile referentiale:

```
CREATE TABLE [dbo].[DateQuestion]
(
```

```
        Answer datetime NOT NULL, -- raspunsul la intrebare sub forma unei date
        Id int IDENTITY PRIMARY KEY, -- (PK) identificatorul inregistrarilor
        QuestionId int – (FK) id-ul intrebarii din tabela Question
)
GO
ALTER TABLE [dbo].[DateQuestion]
        ADD CONSTRAINT [FK_DateQuestion] FOREIGN KEY
        ([QuestionId]) REFERENCES [dbo].[Question] ([Id])
GO
```

Tabela Questionnaire

Aceasta tabela contine intrebarile utilizate pentru chestionar. Tabela Questionnaire isi ia intrebarile din tabela Question cu care realizeaza o relatie de 1:1 (o intrebare corespune unei entitati din Questionnaire), iar intre tabela Questionnaire si QuestionnaireQuestion se stabileste o relatie de 1:n (un QuestionnaireQuestion contine mai multe inregistrari din Questionnaire, ceea ce inseamna mai multe intrebari din Question). Un questionnaire este in esenta tot o intrebare (ca si OpenQuestion, etc), adica e adaugat in aceeasi pagina ca si celelalte intrebari, utilizand tot un textbox si un checkbox pentru salvarea in baza de date.

```
CREATE TABLE [dbo].[Questionnaire]
(       Id int  IDENTITY PRIMARY KEY,
        QuestionId int   -- (FK) referinta la
)
GO
ALTER TABLE [dbo].[Questionnaire]
        ADD CONSTRAINT [FK_Question] FOREIGN KEY
        ([QuestionId] )  REFERENCES [dbo].[Question]  (  [Id]  )
GO
```

Tabela QuestionnaireQuestion

Tabela QuestionnaireQuestion retine toate intrebarile necesare pentru un chestionar. Intre tabela Question si QuestionnaireQuestion se stabileste o relatie de 1:n, iar intre tabela Questionnaire si QuestionnaireQuestion se stebileste o reatie de 1:n descrisa mai sus. Semnificatia pe care o are relatia dintre Question si QuestionnaireQuestion este ca o intrebare poate sa apara in mai multe chestionare.
Codul pentru crearea tabelei QuestionnaireQuestion si a constrangerilor de cheie:

```
CREATE TABLE [dbo].[QuestionnaireQuestion]
(
        Id int IDENTITY PRIMARY KEY,
        QuestionId int,
        QuestionnaireId int
)
GO
ALTER TABLE [dbo].[QuestionnaireQuestion]
        ADD CONSTRAINT [FK_Questionnaire] FOREIGN KEY
        ([QuestionnaireId]) REFERENCES [dbo].[Questionnaire]([Id]),
        ADD CONSTRAINT [FK_Question] FOREIGN KEY
        ( [QuestionId])REFERENCES [dbo].[Question]([Id])
GO
```

5.3.2 Nivelul de acces la date

Nivelul de acces la date este un proiect Windows Forms Application, autogenerat de programul LLBLGen Pro.
Proiectul LLBLGen a fost creat dupa modelul de dezvoltare Database-first, adica prima data am creat baza de date in SQL Server, dupa care proiectul LLBLGen utilizand baza de date existenta.
Framework-ul ales pentru proiect este LLBLGen Pro Runtime Framework care se specifica chiar la inceputul crearii proiectului , iar template-ul este SelfServicing.
Pentru a crea proiectul utilizand modelul de dezvoltare database-first trebuie urmati cativa pasi:

1. Din Catalog Explorer se alea “Add Relational Model Data from a database” prin click dreapta pe Relation Model Data.
2. Se realizeaza conexiunea cu baza de date , dupa care se selecteaza baza de date care se doreste sa fie importata.
3. Dupa importarea bazei de date cu success, tot din Catalog Explorer, se va selecta prin click dreapta, reverse-engineer Tables to Entity Definitions. Aici se vor selecta toate tabelele din aceasta baza de date si se vor adauga proiectului.
4. Se vor crea astfel toate entitatile proiectului dupa care se va genera codul sursa prin apasarea tastei F7 unde trebuie configurat procesul de generare. In acest pas se alege limbajul codului sursa, in cazul nostru C# si template group SelfServicing.
5. Se specifica folder-ul destinatie unde se genereaza proiectul .NET.

Structura proiectului auto-generat este prezentata in figura 5.15 si contine 7 foldere in care se gasesc clasele entitati , colectii, dao, clase de ajutor,factory, relatii si pentru procedurile stocate.
Avanataelej pe care le ofera utilizarea acestui program este ca:

- Favorizeaza structurarea pe N-nivele a proiectului
- Permite sortare, filtrare, grupare, paginare, toate interogarile bazei de date doar prin cateva apeluri de clase.
- Colectiile si entitatile create de LLBLGen Pro sunt atasabile (eng. Bindable) , ceea ce le face sa fie extrem de usor de adaugat controalelor .NET.

Folder-ul *CollectionClasses* – pentru fiecare clasa entitate, LLBLGen Pro creaza de asemenea si o entitate colecti. Fiecare “tabel” devine o “EntityCollection”. Aceste clase Collection elimina nevoia de obiecte DataTable. Acest lucru este foarte util in special pentru controalele interfetei utilizator.

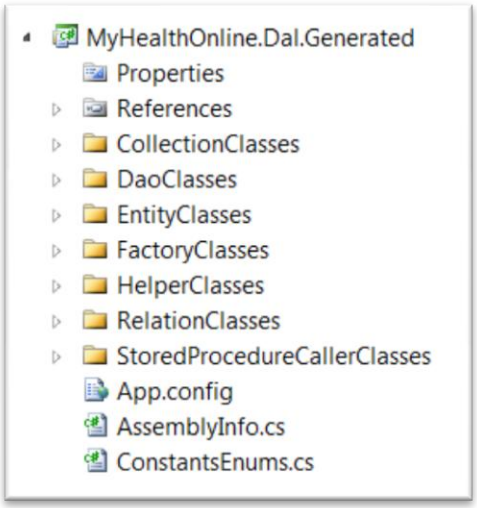

Figura 5.15 Structura proiectului auto-generat de LLBLGen Pro

Folderul *EntityClasses* – Contine toate clasele entitate. Pentru fiecare tabel adaugat proiectului, LLBLGen va crea o clasa entitate specificacare sa reprezinte tabelul. Fiecrae entitate corespunde unei linii dintr-un tabel specific din baza de date.
Folderul *DaoClasses* – acest folder contine obiecte care gestioneaza optiuni legate de baza de date in spatele ‘scenei”. De exemplu, aceste clase iau obiectele entitate si utilizeaza motorul de interogari dynamic pentru a construi propozitiile SQL, precum si a executa query-ul actual.
Folderul *RelationClasses* – Aceste clase contin definitii pentru toate relatiile care exista intre entitati. Aceste relatii ne ajuta sa navigam intre tabelele legate. Se va obtine fie o singura entitate, fie o entitate colectie, in functie de tipul relatiei.
Folderul *FactoryClasses* – Aceste clase contin “fabrici” care va ajuta sa creati criterii pentru interogari si obiecte care vor define modul in care doriti sa fie elementele sortate. Aceste clase nu se vor modifica.
Folderul *HelperClasses* – Aceste clase sunt apelate de data layer. Aceste clase creeaza conexiuni la baza de date, ofera sprijin pentru tranzactii si definesc valorile implicite ale tipurilor, printer alte actiuni.
Folderul *StoredProcedureCallerClasses* – acest folder expune procedurile stocate pe care le-am selectat

Setari necesare in App.config

Cand LLBLGen Pro doreste sa citeasca informatii din baza de date, utilizeaza un string pentru conexiune pentru a-si da seama unde sa gaseasca baza de date si ce credentiale sa utilizeze.

```
<?xml version="1.0"?>
<configuration>
	<appSettings>
		<add key="ConnectionString.SQL Server (SqlClient)" value="data
source=DELL-PC\SQLEXPRESS;initial catalog=WebConsult;integrated
security=SSPI;persist security info=False;packet size=4096"/>
	</appSettings>
</configuration>
```

Fisierul *ConstantsEnums.cs* – acest fisier contine un index pentru fiecare camp din oricare tabel/entity, typed list si typed view din proiect.

5.3.3 Nivelul de business logic

Acest nivel va realiza toate operatiile logice ale aplicatiei si este un proiect . Nivelul de business logic utilizeaza nivelul de acces la date pentru a-si indeplini sarcinile.
Acest nivel este compus din clase cu extensia .cs, fiecare clasa realizeaza operatii de logica specifice numelui dat clasei.
Clasele acestui proiect sunt : Utility.cs, SpecialistBusiness.cs, QuestionnaireBusiness.cs, QuestionBusiness.cs, PatientBusiness.cs, MessageStatusEnum.cs, MessageBusiness.cs, DisciplineBusiness.cs, ConsultBusiness.cs,AppProposalBusiness.cs, AdminBusiness.cs

Clasa Utility.cs

In clasa Utility.cs contine metode pentru calcularea numarului saptamanii curente si pentru obtinerea datei de inceput a saptamanii cu numarul weekNo.

Clasa SpecialistBusiness.cs

Clasa SpecialistBusiness.cs realizeaza operatiile necesare pentru utilizatorul cu rol de Specialist.
Metoda cu numele GetSameDisciplineSpecialists avand ca parametru de intrare id-ul specialistului logat, va returna toti specialistii care apartin aceleiasi discipline cu specialistul al carui id este specificat.

Metoda cu numele GetSpecialists va returna colectia de specialisti existenti in baza de date.

Metoda cu numele CreateSpecialist adauaga o noua entitate Specialist in baza de date.

Metoda cu numele UpdateSpecialist primeste ca parametrii de intrare id, name, password, organization, discipline, email, address, city, country, phone, birthday, sex si disciplineId, cauta in baza de date specialistul cu id-ul primit ca parametru si, daca exista modifica datele existente cu noile date specificate ca parametrii ai acestei metode.

Metoda cu numele DeleteSpecialist primeste ca parametru id-ul specialistului, il cauta in baza de date si apoi il sterge.

Clasa QuestionnaireBusiness.cs

Clasa QuestionnaireBusiness.cs realizeaza operatii pentru chestionare.
Metoda GetQuestionnaires care obtine toate intrebarile din QuestionCollection, cu care este in relatie, care au campul type=”Questionnaire” si returneaza aceasta colectie de intrebari.

Metoda GetUnansweredQuestionnairesByConsult primeste ca parametru id-ul unei consultatii si returneaza o colectie de intrebari (QuestionCollection) reprezentand intrebarile care au campul AnswerDate null (adica nu s-a raspuns la ele) din lista de intrebari ale unui chestionar pentru un consult anume.

Metoda SendQuestionnaire primeste ca parametrii de intrare id-ul chestionarului care trebuie trimis si id-ul consultatiei careia i se va trimite chestionarul. Se selecteaza toate intrebarile care corespund unui anumit chestionar intr-o colectie QuestionCollection. Daca exista intrebari in QuestionCollection pentru chestionarul cu id-ul dat, atunci se va crea un nou chestionar avand ca intrebari entitatile din QuestionCollection filtrate anterior. Metoda AddQuestionToConsult din QuestionBusiness.cs primeste ca parametrii entitatea intrebare si id-ul consultatiei pentru a adauga intrebarea la consult.

Metoda CreateQuestionnaire primeste ca parametru un titlupentru chestionar si returneaza un intreg reprezentand id-ul acestuia dupa ce a fost creat. Se creeaza o noua intrebare (un nou QuestionEntity) cu tipul “Questionnaire”, textul intrebarii si setarea sau nu a campului IsDatabaseQuestion.

Metoda CreateQuestionnaire primeste ca parametru un QuestionEntity. Aceasta metoda este utilizata si de metoda SendQuestionnaire din aceasta clasa. Se creaza o noua intrebare in tabela Question si o noua entitate Questionnaire careia i se seteaza id-ul, legatura catre entitatea Question, iar in entitatea Question se va salva legatura catre entitatea Questionnaire.

Metoda CreateNewQuestionnaireQuestion va crea o noua inregistrare in tabela QuestionaireQuestion .

Metoda GetQuestionnaire primeste ca parametru id-ul un chestionar si returneaza entitatea gasita.

Metoda AddQuestionToQuestionnaire va adauga o noua entitate Question unui anumit chestionar cu id-ul transmis ca parametru. Asta implica si crearea unei noi entitati QuestionnaireQuestion cu metoda prezentata mai sus CreateNewQuestionnaireQuestion.

Metoda AddMultiQuestion adauga o intrebare de tipul raspunsuri multiple unui chestionar cu id-ul dat ca parametru metodei.

Metoda AddDateQuestion va adauga o intrebare la care se poate raspunde cu o data unui anumit chestionar a carui id este transmis ca parametru.

Metoda AddDatabaseQuestionnaire va adauga unui Questionnaire o noua intrebare de tipul database si in tabela Questionnaire noua intrebare chestionarlui cu id-ul transmis ca parametru.
Metoda DeleteQuestion va sterge o intrebare din tabela Question care are acelasi id cu cel transmis acestei metode ca parametru.

Clasa QuestionBusiness.cs

Metoda AddOpenQuestion primeste ca parametrii de intrare id-ul unei consultatii, textul unei intrebari si o valoare booleana care va specifica daca intrebarea va fi adaugata la setul de intrebari ale bazei de date. Se salveaza in tabela Question o intrebare de tipul “Open” , iar in tabela OpenQuestion se salveaza datele de relatie cu noua intrebare.

Metoda UpdateOpenQuestion este utilizata pentru a salva un raspuns dat unei intrebari de tipul “Open’ si totodata realizeaza si modificarea statusului unei consultatii, setandu-l la valoarea “Waiting For Specialist Respone”.

Metoda CreateMultiQuestion primeste ca parametrii de intrare variantele de raspuns sub forma unui string si o entitate Question reprezentand intrebarea si, va crea o noua intrebare cu variante de raspuns si va stabili legaturile dintre tabelele care sunt in relatie MultiChoiceQuestion si Question.

Metoda UpdateMutiQuestion primeste ca parametrii o entitate MultiChoiceQuestionEntity, raspunsul si remarca pe care o face pacientul si va edita in tabela MultiChoiceQuestion si Question campurile necesare, dar va modifica si statusul consultatiei.

Metoda AddDateQuestion si UpdateDateQuestion , dar si AddDatabaseQuestion
sunt similar cu AddOpenQuestion si UpdateOpenQuestion sau CreateMultiQuestion si UpdateMutiQuestion.

Metoda CreateNewQuestion va adauga o noua intrebare la colectia de intrebari existente si va returna entitatea respectiva.

Metoda CreateNewQuestionAndRelatedQuestionType primeste ca parametru o entitate intrebare. Se creaza in prima faza o noua intrebare sau QuestionEntity si in functie de tipul intrebarii se vor apela metodele de creare a acestor intrebari.

Metoda CreateNewQuestion primeste ca parametrii o entitate Question si va returna noua entitate create.
Metoda CreateDateQuestion primeste ca parametru o entitate Question, creaza o noua entitate DateQuestion si salveaza campurile si referintele catre entitatea Question. Metoda returneaza o entitate DateQuestion.
Metoda CreateOpenQuestion este similara cu metoda CreateDateQuestion.

Metoda AddQuestionToConsult va adauga o noua intrebare unui consult cu id-ul specificat ca parametru. Aceasta metoda utilizeaza tabela de legatura ConsultQuestion care salveaza atat id-ul de la intrebare cat si id-ul de la consultatia careia ii corespunde intrebarea . Se realizeaza si un apel al metodei UpdateConsultStatus care va modifica statusul la “Waiting For Patient Reaction”. Metoda returneaza true daca operatiunea de adaugare s-a efectuat fara erori, false altfel.

Metoda GetQuestionFromDatabase() returneaza o colectie de intrebari care reprezinta toate intrebarile a caror camp IsDatabaseQuestion este true in tabela Question.

Metoda GetQuestionByConsult returneaza lista tuturor intrebarilor care au id-ul consultatiei la fel cu cel transmis ca parametru acestei metode.
Metoda GetAnsweredQuestionsByConsult va returna doar acele intrebari, pentru un anumit consult cu acelasi id al consultatiei egal cu cel primit ca parametru al metodei, care au campul AnswerDate din tabela Question diferit de null.

Metoda GetUnansweredQuestionsByConsult va returna o colectie de intrebar a caror camp AnswerDate este null.Cautarea acestor intrebari la care nu sa raspuns se face dupa id-ul consultatiei transmis ca parametru.

Metoda GetQuestionnaireQuestions primeste ca parametru id-ul chestionarului si cauta toate intrebarile care corespund chestionarului cu id-ul respective. Se returneaza o colecte de intrebari care reprezinta intrebarile chestionarului cu id-ul specificat.

Metodele clasei PatientBusiness.cs

Metoda GetPatients returneaza colectia de pacienti reprezentand toate inregistrarile existente in tabela Patient.
Metoda GetPatient care primeste ca parametru id-ul unui pacient va returna primul element al entitatii Patient daca exista in baza de date.
Metoda GetPatients realizeaza filtrarea pacientilor dupa mai multe campuri : patientNo, name, birthday, gender si se returneaza colectia de pacienti filtrati.

Metoda FilterPatients utilizata la adaugarea unei consulatii pentru un pacient (Subscribe), primeste ca parametru o colectie de pacienti si un id pacient si returneaza o entitate pacient a carui id corespunde cu id-ul primit ca parametru.

Metoda CreatePatient va crea o noua entitate pacient in baza de date avand informatiile egale cu cele transmise ca paramtru acestei metode.

Metoda DeletePatient primeste ca parametru id-ul pacientului care se va sterge din baza de date.

Metoda UpdatePatient va realiza modificarea informatiilor pentru un pacient cu id-ul egal cu cel transmis ca parametru. Informatiile care trebuie modificate pentru pacient sunt transmise ca parametrii metodei: name, password, etc.

Metodele clasei DisciplineBusiness.cs

Aceasta clasa contine metoda cu numele GetDisciplines care returneaza toate disciplinele existente in baza de date.

Metodele clasei ConsultBusiness.cs

Acesta clasa defineste urmatoarele metode:
Metoda GetSpecialistConsults are urmatorii parametrii: o colectie consult, id-ul specialistului si id-ul pacientului. Va returna tot o colectie ConsultCollection, rezultatul filtrarii va intoarce toate consultatiile care apartin unui anumit specialist cu id-ul specificat si pacientului cu id-ul specificat ca parametru. Metoda este folosita la adaugarea unei consultatii pentru un pacient unde trebuie sa se verifice ca specialistul current sa nu aiba alte consultatii deschise cu acest pacient.

Metoda GetConsult primeste ca parametrul id-ul unei consultatii si returneaza o entitate Consult rezultata in urma filtrarii consultatiilor dupa id-ul dat ca parametru.
Metoda GetOpenedConsults returneaza toate consultatiile a caror status este in baza de date “Open”.
Metoda GetClosedConsults returneaza toate consultatiile a caror status in baza de date este “Closed”.

Metoda GetConsults() returneaza toate consultatiile din tabela Consult a bazei de date sau null in cazul in care nu exista consultatii.
Metoda GetFutureConsults returneaza o colectie de consultatii care vor avea loc in saptamanile urmatoare. Se face o filtrare utilizand metoda GetStartOfWeek din clasa Utility.cs si campul ConsultDate din Tabela Consult.

Metoda GetCurrentConsults va returna colectia consultatiilor saptamanii curente. Se determina numeric saptamana din an utilizand metoda GetWeek din Utility.cs dupa care apeleaza metoda mai jos definita GetConsults care va returna colecta de consultatii care au data consultatiei intre startOfWeek si startOfWeek plus 7 zile.

Metoda SaveConsult salveaza o concluzie si statusul pentru o consultatie transmisa ca parametru.
Metoda CreateConsult creaza o noua consultatie , cu informatiile transmise ca parametrii metodei si adauga o programare pentru consultatia curenta in AppProposal.
Metoda FilterConsults primeste ca parametrii o colectie de consultatii (ConsultCollection) si numele pacientului. Va realiza filtrarea colectiei primite dupa numele pacientului.

Metodele clasei AppProposal.cs

Acesta clasa este utilizata pentru a crea noi programari pentru o anumita consultatie si pentru a afisa toate programarile facute pentru o consultatie.

Metoda AddAppointment salveaza o noua programare pentru o consultatie. Primeste ca parametrii de intrare id-ul consultatiei pentru care se doreste sa se faca o programare noua, un mesaj, data urmatoarei consultatii si statusul consultatiei.
Metoda GetAppointmentsForConsult primeste ca parametru id-ul unei consultatii si si returneaza o colectia de programari pentru consultatia cu id-ul respective.

Clasa MessageStatusEnum.cs

Contine o enumerare a statusurilor numita MessageStatusEnum.
Metoda GetMessageStatus utilizeaza enumerarea MessageStatusEnum pentru a determina statusul care trebuie returnat.

5.3.4 Nivelul de prezentare

Nivelul de prezentare este construit utilizand un proiect Web Application. Motivele pentru care am ales acest tip de proiect si nu WebSite Project sunt urmatoarele :

- Un website nu are fisier de proiect, un site ii doar o colectie de fisiere in directorul site-lui
- Proiectul/Referintele binare si alte setari de configuratie sunt stocate in fisierul web.config
- Un proiect Web Application are un fisier de proiect, este tratat ca un proiect de biblioteca de clase.
- Deoarece exista un fisier proiect, cateva scenarii devin mult mai usor de pus in practica:
  - Se poate diviza o aplicatie ASP.NET in mai multe proiecte Visual Studio
  - Se pot exclude fisiere din fisierul proiect si din codul sursa-control cu usurinta
  - Pentru rulare si debug de pagini, trebuie construit(eng.buid) intregul proiect Web. Construirea (building) intregii aplicatii Web deodata este de obicei mai rapid, deoarece Visual Studio foloseste un model de creare foarte util care construieste doare fisierele care s-au schimbat.

Nivelul de prezentare se ocupa de toata partea de interfata cu utilizatorul. Pentru acest proiect al nivelului de prezentare s-au separate in foldere Web Formurile ,contreoalele utilizator si master Pages pentru a putea securiza accesul la aceste pagini in functie de rolul utilizatorilor, dar si pentru a favoriza intretinerea sistemului deoarece proiectul creeaza o structura mult mai logica care permite foarte usor sa se identifice formurile care trebuie modificate. De asemenea s-a tinut cont ca aceste formuri web, pagini master, etc, sa fie cat mai sugestiv numite pentru functiile pe care le indeplinesc.

Asadar, s-au creat urmatoarele foldere, pe langa cele existente de la crearea proiectului (Account si Scripts), folder-ul Images salveaza imagini care vor fi afisate pe controale, folderul Styles contine imagini si fisierul cu extensia .css care sunt folosite pentru stilizarea paginilor web. Celelalte foldere vor fi descrise mai jos deoarece acestea sunt mai importante, avand legatura cu functionalitatile proiectului.

Proiectul contine 4 Master Pages. Primul master Page se numeste Site.Master si este pagina care apare prima data la rularea aplicatiei. Acest master este diferit de celelalte deoarece el

contine in partea de sus un meniu care ne redirectioneaza la anumite Content Page-uri cum ar fi About.aspx si Help.aspx. Content Page-ul care se incarca in master page initial este Default.aspx care contine cateva linii de text, de intampinare a utilizatorilor.

De aici utilizatorii trebuie sa se logheze, apasand pe link-ul [Log In], care va duce la incarcarea continutului paginii Login.aspx in Master Page-ul Site.Master arata in figura 5.16.

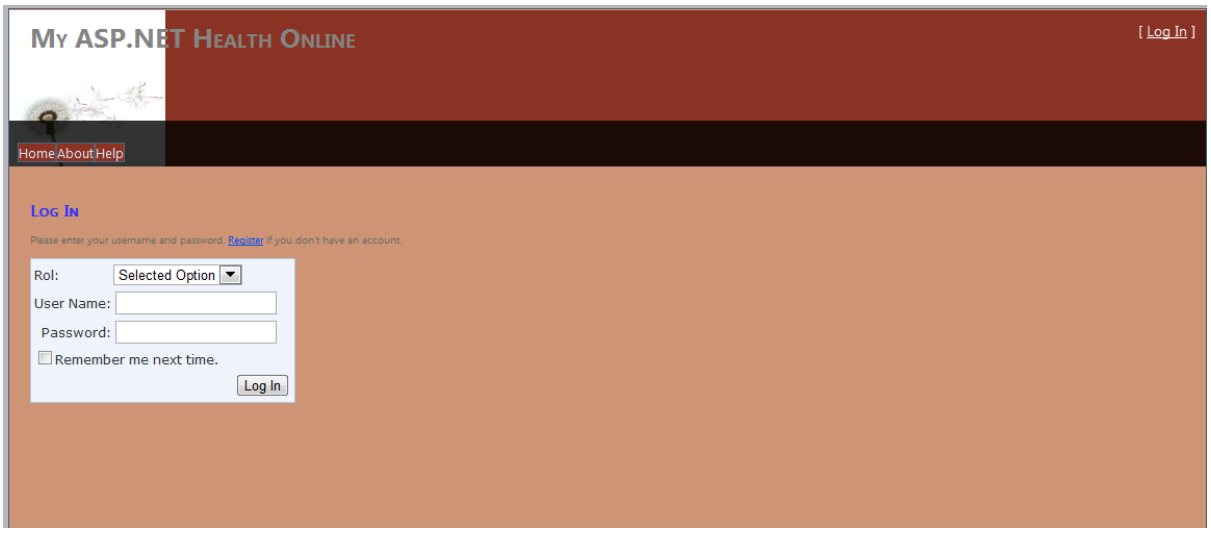

Figura 5.16 Pagina de logare

Pentru fiecare dintre cei 3 utilizatori s-a creat cate un folder in care se gasesc toate controalele necesare pentru a realiza functionalitatile cerute pentru fiecare dintre aceste 3 roluri in parte.

Folder-ul Common contine 4 Controale Web User care contin functionalitati redundante si evita incarcarea inutila a altor Web Formuri si a claselor lor atasate de code-behind cu secvente de cod identice(care pot ocupa un spatiu mare) care afecteaza foarte mult lizibilitatea codului,afectand implicit intelegerea acestuia.
WebUserControl-ul cu numele Appointments.ascx (incluzand codul din spate) afiseaza toate programarile pentru o consultatie cu un anumit Id.
WebUserControl-ul cu numele ConsultHistory.ascx este utilizat atat pe partea de Pacient, cat si de catre Specialist pentru a afisa istoria intrebarilor si raspunsurile acestora pentru o anumita consultatie selectata pentru vizualizare sau continuare a comunicarii.
WebUserControl-ul cu numele EndConsult.ascx si code-behind realizeaza afisarea in paginile Pacientului a datei in care s-a terminat o anumita consultatie impreuna cu concluzia de terminare.
WebUserControl-ul cu numele ViewQuestionList.ascx si partea de code-behind vor afisa toate intrebarile pentru ConsultHistory.

Structura folder-ului Administrator
Exista un master page pentru Administrator numit AdminMaster.Master care contine in partea stanga meniul corespunzator operatiilor pe care administratorul le poate realiza.
Prima pagina care se va incarca in continutul paginii master este AdminPageContent.aspx.

Administratorul realizeaza operatiile de creare, citere, editare si steregere a pacientilor si a specialistilor si, de asemenea, el poate vizualiza si crea chestionare noi.Pagina de adaugare a unui nou chestionar si editare se pot vedea in figura 5.17.

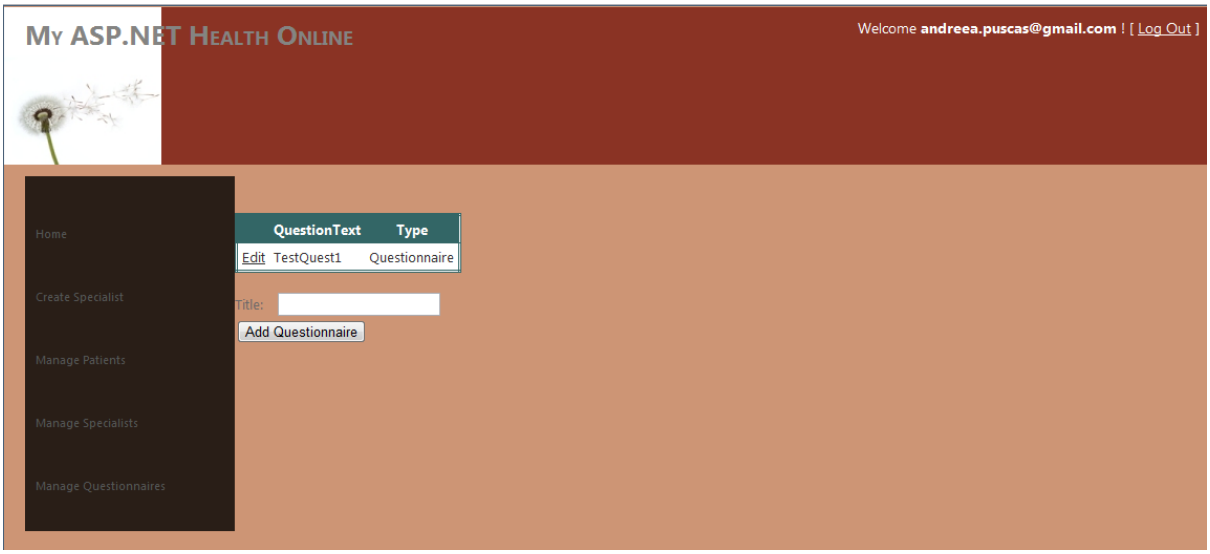

Figura 5.17 Adaugare-Editare Chestionar

Folder-ul Administrator contine mai multe content page-uri pentru a-si realiza sarcinile. In paginile ManagePatient.aspx si ManageSpecialist.aspx se realizeaza operatiile de citire, creare, modificare si stergere a pacientilor, respective a specialistilor. Informatiile sunt afisate in tabele cu posibilitatea de editare, stergere si adaugare la pacient si, stergere si editare la specialist, iar pentru adaugare exista un tabel separate.

Pagina OpenQuestionnaire.aspx contine un user control definit in folderul Specialist numit Questions.ascx care va afisa chestionarul creat prin selectarea (apasare Edit) a unui chestionar din tabel.

Pagina ManageQuestionnaire.aspx afiseaza tabelul cu toate chestionarele si permite editarea lor sau adaugarea unui nou chestionar.

Structura folder-ului Patient

Folder-ul Pacient contine si el un Master Page pentru Pacient numit PatientMaster.Master care contine in partea stanga un meniu propriu.Pacientul isi poate vizualiza toate consultatiile in pagina ViewPatientConsultations.aspx, intr-un tabel prevazut cu un buton la capatul fiecarei linii. Apasarea unui buton va deschide pagina ViewConsult.aspx care contine urmatoarele referinte la controale user din folderul Common: EndConsult.ascx, Appointments.ascx si ConsultHistory.ascx explicate mai sus, dar si un user control definit in folderul Patient, QuestionList care afiseaza o lista cu o serie de intrebari la care pacientul trebuie sa raspunda sau daca are un chestionar, atunci va trebui sa-l deschida si va fi redirectionat la pagina OpenQuestionnaire.aspx care contine user control-ul QuestionList.ascx pentru a putea afisa toate intrebarile din chestionar.

Structura folder-ului Specialist

Si utilizatorul cu rolul Specialist, ca si ceilalti doi, are o pagina master de unde isi va coordona toata activitatea. Acesta pagina este numita SpecialistMaster.Master si are in partea stanga un meniu propriu pentru operatiile pe care le poate efectua.
De asemenea, contine un subfolder cu controale proprii pentru a evita scrierea aceluiasi si pentru a-si indeplini sarcinile .

WebUserControl-ul cu numele EndConsult.ascx ii permite specialistului sa incheie o consultatie.
WebUserControl-ul NextAppointment.ascx ii va permite Specialistului sa creeze o noua programare pentru o anumita consultatie.Acest Control este format din 2 label-uri si doua text box-uri si un buton de salvare a programarii.
WebUserControl-ul cu numele Questionnaire.ascx permite selectarea unui chestionar dintr-un dropdown list si trimiterea lui unui pacient (unei consultatii pentru un pacient).
WebUserControl-ul cu numele Questions.ascx permite trimiterea unei intrebari catre pacinetul a carei consultatii a fost selectata pentru vizualizare. Specialistul trebuie sa selecteze tipul intrebarii (OpenQuestion, multiChoiceQuestion, etc) dintr-un dropdown list si i se ofera posibilitatea de a putea salva intrebarea in baza de date prin bifarea unui checkbox.
Specialistului i se ofera posibilitatea vizualizarii consultatiilor inchise , a consultatiior in desfasurare, din saptamana curenta si saptamnaile trecute si viitoare in pagini diferite de tipul Web Form using Master Page. Pacientul poate fi inregistrat in sistem de catre Specialist.
WebUserControl-ul Consultations.ascx este utilizat in toate content page-urile acestui folder care afiseaza consultatiile.

5.*3.5 Utilizarea Serviciilor Web*

Serviciile Web sunt parte integrata a framework-ului .NET care ofera o soultie de cross-platform (depasire a limitelor impuse de anumite platforme) pentru a schimba date intre sisteme distribuite. Serviciile web pot fi folosite si pentru a injecta date dinamic intr-o pagina ASP.NET AJAX sau pentru a trimite date dintr-o pagina unui sistem back-end.

Am utilizat servicii web pentru metode generice, foarte generale, cum ar fi un serviciu de notificare prin Email , dar si pentru injectarea unor date, in mod dinamic, intr-o pagina care utilizeaza ASP.NET AJAX, mai precis un serviciu de cautare a pacientilor dupa nume care sa poata fi utilizat la cautarea acestora utilizand functionalitati AJAX .

6. Testare si Validare

Testarea este un process prin care se executa un sistem pentru a-i descoperi erorile. Erorile care apar intr-un sistem sunt in cea mai mare parte cauzate de erorile de proiectare sau de intelegerea gresita a cerintelor de functionare. Testarea nu asigura ca functionalitatea sistemului este cea dorita, dar poate detecta erorile de functionare sau de implementare.

6.1 Cerinte functionale

Testarea aplicatiei s-a realizat prin rulari dupa implementarea fiecarei noi functionalitati adaugate proiectului. In acest sens, s-au utilizat instrumentul oferit de Visual Studio, de debugging (ro. Depanare) care ne permite sa urmarim continutul variabilelor din momentul rularii (testarii) aplicatiei. Acest mod de rulare a permis detectarea mult mai usoara a erorilor de functionare, deoarece am putut sa vad pas cu pas daca continutul anumitor variabile era cel dorit. Acest mod de testare consta in introducerea unor puncte de intrerupere (numite Breakpoint) in metoda sau linia care se doreste a fi verificata.

De exemplu, s-au efectuat testari pentru logarea in sistem a unui utilizator cu rolul de Specialist. S-a introdus un breakpoint in metoda care verifica daca specialistul cu e-mailul si parola date nu returneaza o colectie goala (cu un numar de entitati=0) in urma efectuarii unei filtrari cu PredicateExpression asupra colectiei de specialisiti.

Cu toate ca testarea este un proces de durata, care necesita nu doar atentie, dar si multa rabdare, am incercat sa testez scenarii cat mai diferite pentru a detecta eventualele problem de proiectare.

La sfarsitul implemetarii proiectului, s-au facut teste care sa ateste functionarea corecta a aplicatiei in totalitatea ei, pentru diverse cazuri de utilizare.
Si in acest caz s-a utilizat tot instrumental de debugging pus la dispozitie de Visula Studio, pentru a verifica daca, de exemplu o data citita din interfata ajunge cu valoare dorita la nivelul de business unde trebuia procesata (de exemplu formatul pentru un tip Data).

Validarea sistemului are ca scop confirmarea faptului ca produsul indeplineste asteptarile cerute. Activitatea de validare se concentreaza pe produsul final, care este testat din punct de vedere al utilizatorului (clientului). Validarea stabileste daca produsul indeplineste asteptarile utilizatorilor globali. (22)

6.2 Cerinte non-functionale

Testarea joaca un rol important in mentinerea si imbunatatirea calitatii produselor software.
In urma numeroaselor testari efectuate, se poate spune ca sistemul este unul fiabil deoarece functioneaza conform cerintelor functionale stabilite, fara sa genereze erori.
Fiabilitatea poate fi crescuta prin doua metode: evitarea erorilor care rezulta din practici de proiectare si toleranta la esec care presupune functionarea sistemului chiar daca apar esecuri sau erori.

Securitatea sistemului este asigurata prin realizarea unei autentificari sigure, accesul controlat la anumite parti din sistem.
S-a realizat securizarea accesului la anumite fisiere ale aplicatiei prin construirea folderelor pentru utilizatorii sistemului si securizarea acestora astfel incat doar utilizatorul cu rolul specificat sa aiba acces la fisierele din interiorul folderelor. Se realizeaza un acces controlat la paginile unui anumit rol pentru a evita eventuale atacuri asupra sistemului.
De exemplu, pacientul nu ar trebui sa poata avea acces la pagina dministratorului sau a specialistului.

Scalabilitatea sistemului este asigurata de structurarea proiectului pe 3 nivele care permit adaugarea unor module noi fara prea mare efort si a unor noi functionalitati.

7. Manual de instalare si utilizare

In acest capitol se va detalia procesul de instalare,necesarul de resursele software si hardware, precum si modul de utilizare al aplicatiei din punct de vedere al utilizatorului.

7.1 Instalare

7.1.1 Resurse software si hardware necesare

Resursele software necesare pentru a puteainstala si rula aceasta aplicatie sunt:

- SQL Server 2005, SQL Server 2008 sau SQL Server 2008 R2
- Programul Visual Studio 2010
- Programului LLBLGen Pro v3.1
- ASP.NET AJAX Control Toolkit
- Microsoft .NET Framework 4
- Sisteme de operare pe care poate fi instalat VS2010:
    - Windows XP (x86) with Service Pack 3 - all editions except Starter Edition
    - Windows Vista (x86 & x64) with Service Pack 2 - all editions except Starter Edition
    - Windows 7 (x86 & x64)

Resursele hardware necesare pentru instalarea si rularea aplicatiei sunt:

- Calculator cu un processor de 1.6GHz sau mai rapid
- 1 GB (32 Bit) sau 2 GB (64 Bit)
- Spatiu disponibil pe hard disk : 3GB
- 5400 RPM hard disk drive
- DVD-ROM Drive

7.1.2 Procesul de instalare

Trebuie sa se instaleze LLBLGen Pro v3.1 de pe site-ul (23) conform instructiunilor oferite pe site.
Pentru a putea rula aplicatia trebuie urmati cativa pasi:

1. Rulare script cu numele DDLSQL_CreateScript pentru crearea bazei de date cu numele WebConsult in SQLServer 2008 R2.
2. Deschiderea proiectului in Visual Studio 2010
3. Adaugare referinta din proiectul MyHealthOnline AjaxControlToolkit.dll daca acesta nu este gasit la compilare

7.2 Manual de utilizare

- Un utilizator poate utiliza acest sistem doar daca au fost inregistrati deja, in caz contrar, un specialist trbuie sa-i ceara administratorului sa-l inregistreze, iar pacientul poate fi inregistrat atat de catre administrator, cat si de catre specialist cand are o consultatie fata in fata cu acesta
- Un utilizator care este inregistrat in sistem, va putea beneficia de functionalitatile puse la dispozitie in functie de rolul sau.
- Formul pentru logare este acelasi pentru toti utilizatorii. Pentru logare, trebuie sa se apese pe hyperlink-ul din coltul din dreapta sus cu textul [Log In] si va aparea pagina pentru logare care se poate vedea si in figura 5.16 .

- In functie de rolul sau, utilizatorul va fi redirectionat catre una dintre paginile :
    i. AdminPageContent.aspx
    ii. Specialist.aspx
    iii. Patient.aspx
- Daca veti fi redirectionat spre pagina Specialistului, atunci pagina va arata ca in figura 7.1 care prezinta pagina de Subscribe pentru un pacient

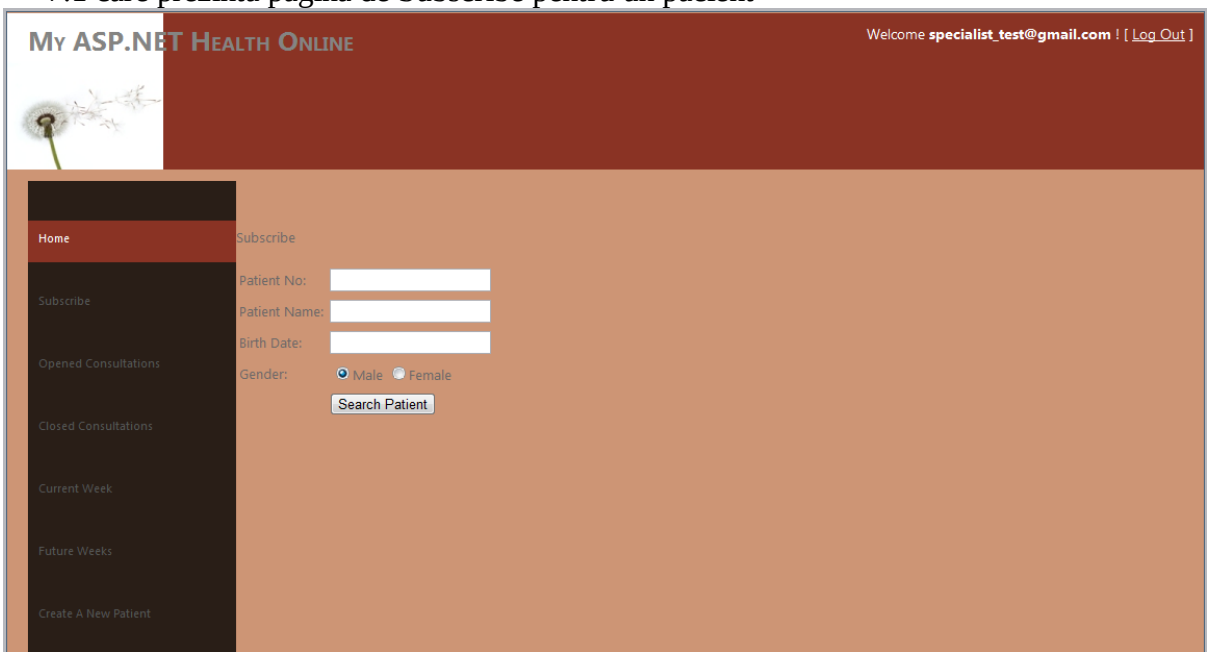

Figura 7.1 – Pagina Subscribe.aspx

Dupa cum se vede, Specialistul are urmatorul meniu pus la dispozitie:

- Home
- Subscribe – de aici se va putea cauta un pacient si crea o noua consultative
- Opened Consultations – afisare consultatii care nu au status “Closed” si posibilitate de a inchide o consultatie, adauga o programare pentru o consultatie, pune intrebari, trimite chestionare, vizualizare programari
- Closed Consultations – afisare consultatii terminate, posibilitate cautare dupa un anumit pacient, penttru o consultative selectata prin apasarea butonului din dreapta fiecarei linii, se va putea vedea data terminarii consultatiei, programarile pentru acea consultatie si un istoric al tuturor intrebarilor si raspunsurilor
- Current Week – va afisa consultatiile din saptamana curenta si saptamanaile trecute in tabele diferite si va avea aceleasi functionalitati ca si pentru meniul Opened Consultations
- Future Weeks – afiseaza consultatiile saptamanilor viitoare,se poate cauta dupa un anumit pacient, se poate deschide o consultative pentru vizualizare detalii consultatie, data terminarii consultatiei, daca e cazul, toate programarile, se pot trimit e chestionare, intrebari, se poate termina consultative(inchide), programa pentru urmatoare intalnire online
- Create A New Patient – permite adaugarea unui pacient nou in sistem

- Daca se va face redirectionare catre pagina administratorului, atunci acesta va putea vedea urmatorul meniu:
    - Home

- Manage Patient – va aparea un table cu toti pacientii din sistem si se va putea adauga un pacient nou, edita sau sterge dupa cum se va putea vedea si in figura 7.2
- Manage Specialists – se vor afisa toti specialistii din sistem intr-un table, se vor putea edita sau apasand pe butoanele din partea stanga din table sau din tabelul afisat mai jos se vor putea adauga noi specialist
- Manage Questionnaire – permite vizualizarea chestionarelor si adaugarea unora noi, vezi figura 5.17 Adaugare-editare chestionar

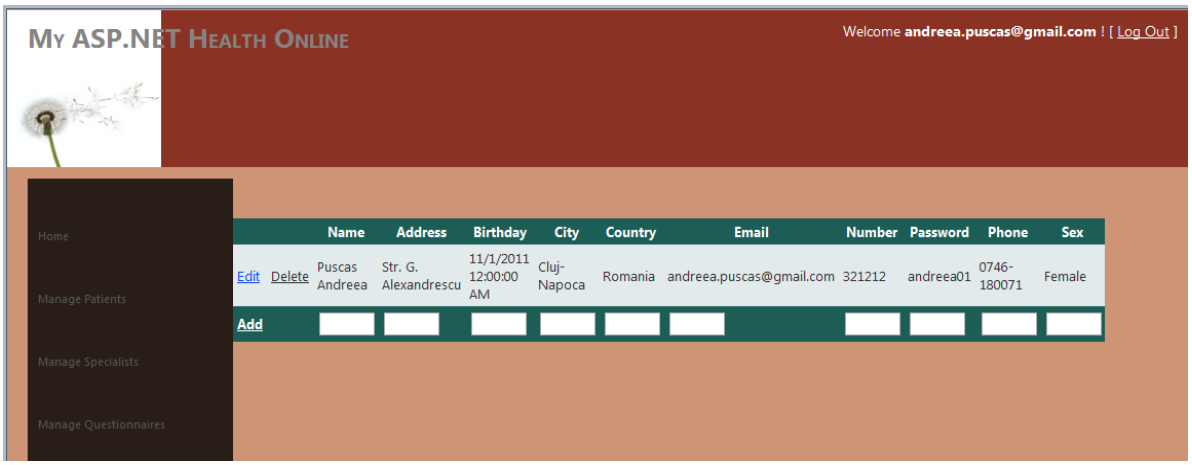

Figura 7.2 ManagePatients.aspx

- Daca utilizatorul are rol de pacient, atunci se va face redirectionare catre pagina pacientului care contine urmatorul meniu:
  - Home
  - Consultations – se afiseaza un table cu toate consultatiile pacientului logat si pe fiecare linie din table va avea un buton View; prin apasarea butonului pentru o anumita consultatie, pacientul va putea vedea,data terminarii consultatiei (daca statusul acesteia este “Closed”), o lista a tuturor programarilor, o istorie a tuturor intrebarilor-raspunsurilor pentru consultatia respective, va putea deschide un chestionar pentru completare, daca este disponibil sau va putea raspunde la intrebari.

8 . Concluzii

8.1 Contributii personale

Proiectul “Sistem de monitorizare la distanta a pacientilor” se bazeaza pe structura arhitecturala pe 3 nivele.Nivelul de prezentare l-am construit astfel incat sa fie cat mai usor de inteles de catre utilizatori si sa fie usor de extins.

Nivelul de business logic este un proiect care realizeaza toate operatiile de care are nevoie sistemul pentru a functiona conform specificatiilor impuse inaintea inceperii proiectarii si implementarii propriu-zise.

Nivelul de acces la date este un proiect auto-generat de LLBLGen Pro pe care l-am folosit in solutia Web Consult pentru a putea indeplinii cu usurinta sarcina de acces la baza de date a sistemului.

8.2 Rezultate obtinute

Rezultatul final este un proiect care se doreste sa vine in ajutorul pacientilor, dar si a medicilor. Scopul crearii acestui proiect este de a oferi pacientilor si medicilor un mediu de comunicare in care sa poata schimba informatii standardizate, utilizand tehnologii avansate si de actualitate.

Este un proiect usor de extins atat datorita arhitecturii pe 3 nivele utilizate, cat si a modului de implementare care elimina redundanata codului, mareste lizibilitatea codului scris in C# si datorita structurarii cu atentie a proiectelor.

8.3 Dezvoltari ulterioare si imbunatatiri

Sistemul de fata poate fi vazut ca o componenta a unui sistem mai mare si caruia i se pot adauga o serie de noi functionalitati si imbunatatiri.

In primul rand un astfel de sistem ar putea migra cu succes pe partea de dispozitive mobile, implementandu-se o aplicatie similara cu aceasta.

Cateva dintre dezvoltarile si imbunatatirile care se pot adauga acestui sistem sunt :

- Sistemul ar putea implementa noi functionalitati pentru roluri noi, cum ar fi asistentii medicali.
- Pacientului i s-ar putea adauga noi facilitati, cum ar fi posibilitatea trimiterii unor imaginii specialistului.
- Posibilitatea de a cere inchiderea unei consultatii.
- Integrarea unui chat pentru comunicari de urgenta.
- Adaugarea unui nou tip de notificari prin SMS.
- Crearea de rapoarte periodice

1. **Dinsoreanu, Mihaela.** *Lecture 3 - Analysis.* Cluj-Napoca : s.n., 2010.
2. **Parlament.** Ordonanta Nr. 124/1988 actualizata prinvind organizarea si functionarea cabinetelor medicale. *Casa Nationala de Asigurari de Sanatate.* [Online] Noiembrie 13, 2001. http://www.cnas.ro/legislatie/organizarea-cabinetelor-medicale.
3. Exercitarea profesiei de medic - partea 1. *Casa Nationala de Asigurari de Sanatate.* [Online] Parlamentul, Ianuarie 14, 2006. http://www.cnas.ro/legislatie/exercitarea-profesiunilor-medicale/medic.
4. **Parlament.** *Legea Nr. 46 - Legea drepturilor pacientului.* Bucuresti : Monitorul Oficial Nr. 51 din 29 ianuarie 2003.
5. *Legea nr. 95 din 14 aprilie 2006 - privind reforma in domeniul sanatatii – extras –.* s.l. : Monitorul Oficial, 2006.
6. **Chancellor, Joseph.** *Rapid C# Windows DEVELOPMENT Visual Studio 2005, SQL Server 2005 & LLBLGen Pro.* 1st Edition. s.l. : Lulu.com, 2006. p. 141.
7. **Solutions Design bv.** hh_start.htm. *http://www.llblgen.com/documentation/3.1/Designer/hh_start.htm.* [Online] 2011. http://www.llblgen.com.
8. **Phyu, May Hnin.** *http://www.mayvelous.com/2007/03/08/introducing-or-mapper-and-llblgen-pro/.* [Online] March 8, 2007. http://www.mayvelous.com.
9. **Wahlin, Dan.** Understanding ASP.NET AJAX Web Services. *Microsoft ASP.NET Web Site.* [Online] http://www.asp.net/ajax/tutorials/understanding-asp-net-ajax-web-services.
10. **Dutta, Raja.** Forms Authentication in ASP.NET with C#: Basic. *.net Funda - The fundamentals of .NET.* [Online] http://www.dotnetfunda.com/articles/article114.aspx.
11. —. Forms Authentication in ASP.NET with C#: Advance. *.net Funda - The fundamentals of .NET.* [Online] http://www.dotnetfunda.com/articles/article141.aspx.
12. **Brinzarea, Bogdan and Darie, Cristian.** *Microsoft AJAX Library Essentials.* Birmingham : Packt Publishing Ltd., 2007. 978-1-847190-98-7.
13. **Esposito, Dino.** *Microsoft ASP.NET AJAX: Architecting Web Applications.* s.l. : Microsoft Press, 2009.
14. **MacDonald, Matthew.** *Beginning ASP.NET 4 IN C# 2010.* New York : Paul Manning, 2010. 978-1-4302-2609-3.
15. **Microsoft.** Three-Layered Services Application. [Online] Microsoft. http://msdn.microsoft.com/en-us/library/ff648105.aspx.
16. **Nita, Adrian, et al.** *Introducere in .NET Framework - Suport de curs pentru elevi.* [PDF] s.l. : Microsoft Romania, 2008.
17. **Unknown.** ASP.NET. *Wikipedia.* [Online] http://ro.wikipedia.org/wiki/ASP.NET.
18. SQL Server. *SearchSQLServer.com.* [Online] Ianuarie 2006. http://searchsqlserver.techtarget.com/definition/SQL-Server.
19. **Bagui, Sikha and Earp, Richard.** *Essential SQL on SQL Server 2008.* s.l. : Jones and Barlett Publishers, 2011.
20. **Rankins, Ray, et al.** *Microsoft SQL Server 2008 R2.* 2011.
21. **Giurca, Adrian.** Modelarea cazurilor de utilizare. [Online] http://inf.ucv.ro/~giurca/courses/CB3105/resources/Modelarea%20cazurilor%20de%20utilizare.pdf.
22. **FAGARASAN, Ioana.** Testarea software si asigurarea calitatii. [Online] http://www.shiva.pub.ro/PDF/TEST/Testarea%20software%20si%20asigurarea%20calitatii%20-%20curs2.pdf.
23. LLBLGen Pro v3.1. [Online] http://www.llblgen.com/defaultgeneric.aspx.

Anexe

Lista de figuri