**UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAŞI**

FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENŢĂ

**MediArch**

Aplicație Web

Propusă de

**Moroșanu C.D. George-Cosmin**

Sesiunea: **Iulie 2018**

Coordonator știinţific

**Drd. Colab. Florin Olariu**

**UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAŞI**

FACULTATEA DE INFORMATICĂ

**LUCRARE DE LICENȚĂ**

MediArch

Aplicație Web

**Moroșanu C.D. George-Cosmin**

Sesiunea: **Iulie 2018**

Coordonator știinţific

**Drd. Colab. Florin Olariu**

*DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR*

Prin prezenta declar că Lucrarea de licenţă cu titlul „***MediArch – Arhiva Medicală***” este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau din străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;

- reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;

- codul sursă, imaginile etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;

- rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi,

Absolvent Moroșanu George-Cosmin

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „***MediArch – Arhiva Medicală***”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică. De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi,

Absolvent Moroșanu George-Cosmin

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil şi sursă, să aparţină autorului prezentei lucrări, *Moroșanu George-Cosmin.*

Încheierea acestui acord este necesară din următoarele motive:

*[Se explică de ce este necesar un acord, se descriu originile resurselor utilizate în realizarea*

*produsului-program (personal, tehnologii, fonduri) şi aportul adus de fiecare resursă.]*

Iaşi, *data*

Decan *Adrian Iftene* Absolvent *Moroșanu George-Cosmin*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original) (semnătura în original)

Cuprins

[Introducere 7](#_Toc517308553)

[Motivație 7](#_Toc517308554)

[Scopul 7](#_Toc517308555)

[Alegerea temei 7](#_Toc517308556)

[Alegerea tehnologiei 8](#_Toc517308557)

[Context 8](#_Toc517308558)

[Cerințe funcționale 9](#_Toc517308559)

[Gradul de noutate 10](#_Toc517308560)

[Uzabilitate 10](#_Toc517308561)

[1. Contribuții 11](#_Toc517308562)

[2. Proiectare și Implementare 12](#_Toc517308563)

[3. Manualul utilizatorului 21](#_Toc517308564)

[4. Concluzii 22](#_Toc517308565)

[5. Bibliografie 23](#_Toc517308566)

[6. Tehnologii Folosite 25](#_Toc517308567)

[6.1. C# (.Net, MVC, ORM, ASP.NET, Entity) 25](#_Toc517308569)

[6.2. SQL (SQL Server Express) 25](#_Toc517308570)

[6.3. Source Controlling -> GitHub (GitKraker) 25](#_Toc517308571)

[6.4. Bootstrap 25](#_Toc517308572)

[6.5. HTML, CSS & JavaScript 25](#_Toc517308573)

[6.6. Adobe Illustrator 25](#_Toc517308574)

# Introducere

## Motivație

În ultima vreme am observat o dezvoltare din ce în ce mai accentuată a domeniului medical, și nevoia tot mai mare a unei platforme prin intermediul căreia pacienții să își acceseze datele într-un mod sigur, de oriunde. Astfel a apărut subiectul acestui proiect de licență, **MediArch**, locul perfect unde vei avea acces la orice informație utilă din istoricul tău medical.

Această aplicație va facilita atât pacienții, cât și doctorii, deoarece acestora le va fi foarte usor și rapid să aibă acces la informațiile consultațiilor pe care le-au avut până acum. Și va fi și foarte ușor de gestionat, deoarece, ca și utilizator special (Owner sau Moderator) voi avea acces la o listă cu tot ce s-a întâmplat pe platformă in ultimile 2 sătpămâni, și voi putea imediat să iau măsuri dacă observ că ceva este în neregulă(voi dezvolta în următoarele pagini acest subiect).

### Scopul

Scopul principal al acestei aplicații este menținerea sigură a tuturor informațiilor medicale ale unei persoane pe o singură platformă. precum și accesarea rapidă a acestora. Astfel, dacă cineva va avea nevoie la un momentdat de o informație dintr-o consultație din trecut, să nu fie nevoit să intre pe mai multe platforme online, cu conturi separate, sau să caute consultația respectiva prin zecile de foi aruncate în colțul unui sertar.

### Alegerea temei

De mic copil am avut suficiente probleme medicale și am fost la numeroase consultații la care am primit și prescripții. Bine înțeles că s-a întamplat să mai am nevoie de unele dintre acele prescripții, dar acele foi erau pur și simplu de negăsit de cele mai multe ori. Chiar recent am pățit ceva asemănător. Așa că am decis ca tema pe care o voi aborda pentru licență să aibă utilizare și în domeniul medical, iar această aplicație va rezolva problema specificată anterior, pe care sunt sigur că mulți oameni au întâlnit-o până acum.

### Alegerea tehnologiei

Pe parcursul anului II la “Facultatea de Informatică Iași” am fost implicat în diverse proiecte, care mai de care mai grele. Unul dintre aceste proiecte a fost cel de la Tehnologii Web, unde deși nu știam mai nimic la început, am ajuns să mă descurc, și a început să îmi surâdă idea dezvoltarii unei aplicații web.

În primul semestru din anul III am participat la cursul de „Introducere in .NET”, în cadrul căruia am facut echipă cu alte 5 persoane cu scopul comun de a dezolta o nouă aplicație web, cu o temă aleasă de noi: Course Manager(aplicație care să facă un “management” mai bun, concepută initial pentru cursul de Introducere in .NET, dar cu posibilitatea dezvoltării pe viitor pentru majoritatea cursurilor). Dar de data aceasta nu am folosit aceleași tehnologii ca și anul trecut(PHP, HTML, CSS și JavaScript), ci am trecut la următorul nivel: C# (folosind frameworkul .Net). Acest proiect mi-a captat cu adevărat atenția, considerându-l unul dintre cele mai interesante și mai complexe proiecte dezvoltate în decursul facultății. Astfel, știind că îmi va face plăcere să lucrez la un asemenea proiect, am decis să folosesc pentru licență această tehnologie (ASP.Net Core).

## Context

Aplicația „***MediArch”*** (Arhiva Medicală) este o aplicație web menită să servească numeroși clienți care doresc să aibă la dispoziție atât rezultatele la toate consultațiile anterioare, cât și prescripțiile și fișierele adiționale acordate de medicii la care au facut consultațiile respective.

După cum am precizat la început, am decis să introduc și 4 roluri, denumite destul de sugestiv, pentru a facilita utilizatorii în mod diferit. Cele 4 roluir sunt: Owner, Moderator, Medic(Doctor) și Pacient. Primele 2 roluri precizate vor avea acces la operații speciale pentru a face management aplicației, iar ultimele 2 roluri, reprezentând utilizatorii normali, vor fi de Clienții platformei.

Acestea sunt câteva din platforme care oferă funcționalități asemănătoare cu cele ale aplicației mele:

* Arcadia;
* MedLife;
* Regina Maria;
* Clinica Sante;

Dar pentru a avea acces la informațiile de pe aceste platforme vei fi nevoit să îți faci câte un cont pe fiecare platformă în parte, acest lucru fiind un consumator de timp, memorie(deoarece este posibil să îți alegi câte o parolă diferitî pentru fiecare cont, din motive de securitate) și este destul de incomod.

Aplicația **MediaArch** promite să rezolve această problemă, strângând toate informațiile clienților la un loc.Altfel spus, va fi nevoie de un singur cont, controlul datelor făcându-se de pe o singură platformă.

Un plus va fi faptul că aici se vor găsi Doctori de la diverse instituții și clinici, astfel ușurând munca pacienților și scurtând timpul de căutare.

## Cerințe funcționale

* ***Logare și înregistrare:*** În momentul în care un utilizator va dori să acceseze această aplicație, acesta va trebui să se logheze pe baza unui email și a unei parole (date stabilite de acesta în procesul de înregistrare). Dacă utilizatorul este pentru prima data pe site, acesta iși va putea face destul de repede și usor un cont, la secțiunea de înregistrare.
* ***Lista de medicamente:*** Pusă la dispoziție pe platformă pentru a scuti utilizatorii să intre pe diverse site-uri pentru a face rost de prospectele acestor medicamente.
* ***Liste de utilizatori:*** Aici vor fi: lista de Doctori(accesată cel mai frecvent de Pacienți), lista de Pacienți (cu impact mai mare pentru Doctori, deoarece le va fi mai ușor să adauge un Consult nou din această secțiune), și lista cu Specializările, unde va fi o extensie cu doctorii pentru fiecare specializare în parte.
* ***Informații despre utilizatori:*** Detaliile publice ale utilizatorilor, puse atât pe paginile de profil, dar care apar și când se caută un utilizator.
* ***Consultațiile:*** Lista cu toate consultațiile, și detaliile care vor fi puse de Doctor pentru fiecare utilizator.
* ***Secțiunea de Assistance:*** Rubrica de înrebări și răspunsuri.
* ***Managementul datelor:*** Secțiunea pentru utilizatorii speciali, cu numele de “Data Records” unde se va găsi tot ce s-a întâmplat pe site în ultimele 2 săptămâni.

## Gradul de noutate

Deși există și alte aplicații asemănătoare (de exemplu: Arcadia, MedLife, Clinica Sante), proiectul pe care l-am dezvoltat nu va restricționa câmpul de întindere al acestuia. Spun acest lucru pentru că în cadrul aplicațiilor deja existente, toate informațiile salvate sunt doar de la medicii/doctorii din interiorul mediului medical(clinicii) pentru care au fost dezvoltate aplicațiile respective (fiecare clinică va afișa doar rezultatele medicilor/doctorilor lor). Astfel, ca și utilizator normal, eu voi fi nevoit să îmi fac multiple conturi pe diverse plaforme pentru a avea acces la toate informațiile de care am nevoie. **MediArch** va strange toate aceste informații într-un singur loc, pentru a fi mult mai ușor unui utilizator să își găsească orice informație utilă.

## Uzabilitate

Principala uzabilitate a site-ului va fi în domeniul medical, deoarece servește drept aplicație medicală, dar și în domeniul informaticii, deoarece poate fi văzută drept material didactic.

# Contribuții

**MediArch** este o aplicație web, client - server, construită prin intermediul ASP.NET Core, orientată spre domeniul medical, care adună informațiile cu privire la consultațiile Pacienților de la fiecare Doctor în parte și le pune la îndemâna clienților, oferind o experiență plăcută în momentul accesării site-ului.

După ce am stabilit tema și tehnologia pe care le voie aborda, unul dintre cei mai importanți pași a fost definirea unei Baze de Date corespunzătoare cu cerințele aplicației. Desigur, am gândit-o de mai multe ori până să ajung la o formă finală, dar în cele din urmă am ajuns la aceasta. O voi prezenta în următorul capitol.

Odată stabilită baza, m-am putut apuca și de logica aplicației, urmând o dezvoltare continua a acesteia, pe parcurusl căreia am adăugat diverse funcționalități neprevăzute.

Ușor ușor, serviciile au început să prindă contur, iar aplicația își îndeplinea scopul ei inițial.

Văzând platforma într-o stare finală, cred ca unele dintre cele mai utile aspecte care aduc o contribuție crucială sunt:

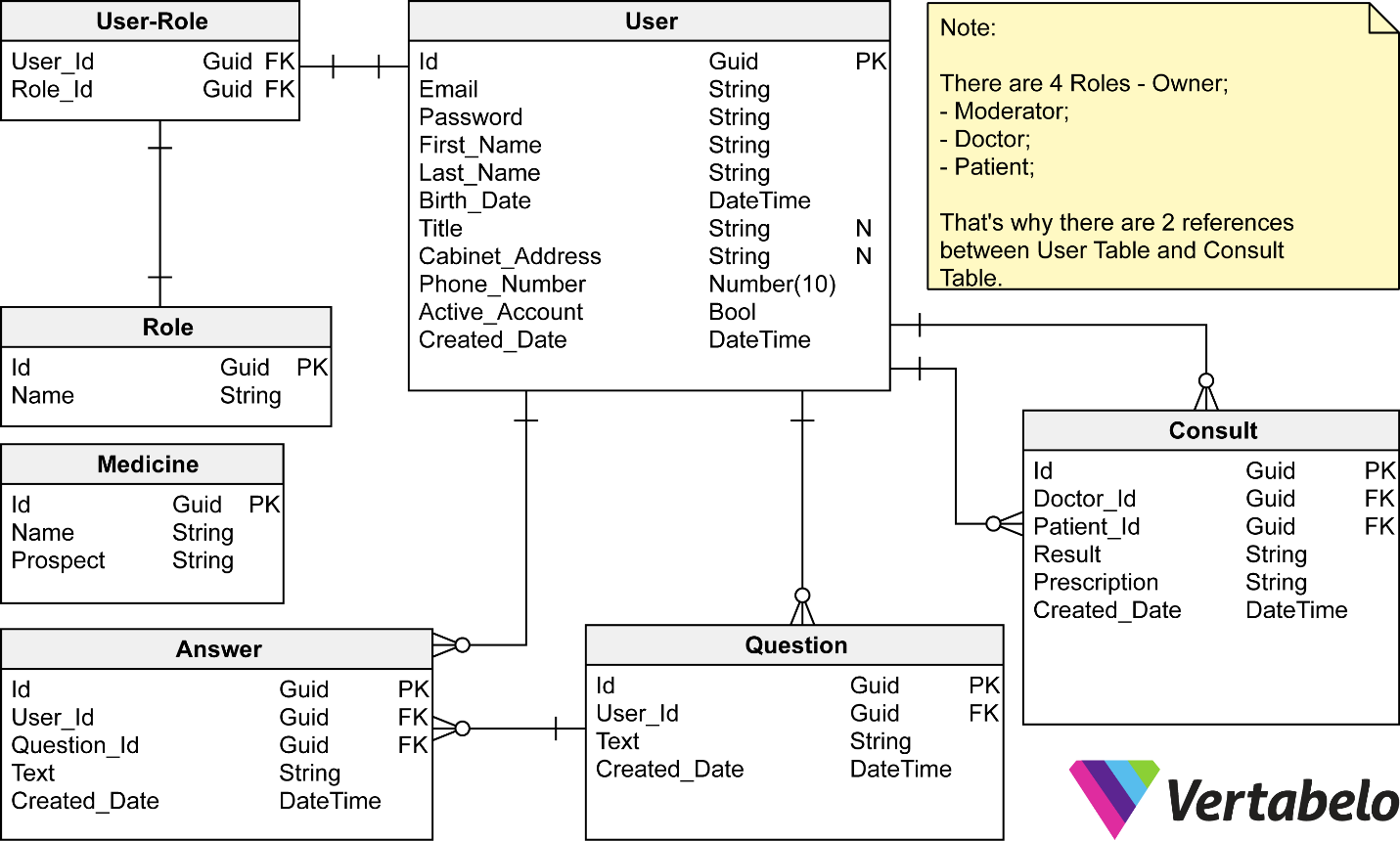
* **Accesarea usoară și intuitivă a consultațiilor** unui utilizator;
* **Navigabilitatea fluentă** prin aplicație, putând ajunge din orice colț în altul;
* **Accesibilitatea la informațiile medicamentelor**, lucru care salvează timp;
* **Secțiunea de Asistență**, unde te poți informa(căutând subiectul de interes) sau îți poți satisface curiozitățile;

# Proiectare și Implementare

După ce am stabilit tema finală a acestui proiect, am analizat ceea ce aveam de gând să facă aplicația, și am decis ca entitățile care mă vor ajuta în dezvolarea acesteia să fie următoarele:

* User
* Consult
* Medicine
* Question
* Answer
* Role
* User-Role (tabelă intermediară)

Astfel, (cu ajutorul aplicației Vertabelo,) Baza de date, împreună cu relația dintre fiecare tabelă a acesteia (unde este cazul), va arăta în modul următor:



**Fig. 1** Structura Bazei de Date și Relația dintre Tabele

Am încercat să aleg nume cât mai sugestive pentru fiecare entitate, inclusiv pentru fiecare câmp în parte, și consider că am și reușit acest lucru. În rândurile ce urmeaza voi descrie fiecare tabelă, relațiile dintre tabelele existente, și voi încerca să argumentez de ce am ales câmpurile respective, unde va fi cazul.

Toate tabelele au un câmp denumit ***„Id”***, acesta fiind Indentificatorul ales în mod unic pentur fiecare element în parte. Pe lângă acest prim comun câmp, va apărea la majoritatea tabelelor și câmpul ***“Created\_Date”***, reprezentând data la care a fost create entitatea respectivă; Acesta din urmă va ajuta la monitorizarea activităților ce au loc pe platformă, aplicația oferind *utilizatorilor speciali* o secțiune de „Data Records”, unde se vor afla 4 tabele, reprezentând fiecare în parte sistemul de gestionare al celor 4 tabele care conțin ultimul câmp specificat mai sus.

Tabela **Role** este tabela de roluri, iar cele 4 roluri alese de mine, pentru a dezvolta aplicația într-un mod mai fluent, au următoarele nume:

* ***Owner*** (*Utilizator special*);
* ***Moderator*** (*Utilizator special*);
* ***Medic*** (Doctorii – *Utilizator normal*);
* ***Pacient*** (Pacienții – *Utilizator normal*);

Tabela **User-Role** este o tabelă intermediară, aceasta asignând fiecărui utilizator câte un rol.

Tabela **User** ar fi cea mai importantă. Aceasta are câmpurile:

* *Email* - care va juca rol și de UserName, acesta fiind unic;
* *Password* – parola cu care se va loga utilizatorul;
* *First\_Name*, *Last\_Name*, *Birth\_Date*, *Phone\_Number* – date personale, cerute la înregistrare;
* *Title, Cabinet\_Address* – Date cerute Doctorilor la înregistrare;
* *Active\_Account* – câmp suplimentar (de tip Bool) care va împiedica utilizatorul să se logheze pe platformă dacă este setat pe *False*. Acesta va fi setat pe *False* doar dacă este suspectat că este ceva în neregulă cu contul utilizatorului: altfel va fi setat pe *True*;

Desigur, Relația dintre tabelele *User* și *Role* este una de ***1 la 1****(voi nota cu* ***1:1*** *de acum înainte)* deoarece oricărui utilizator îi este asociat un singur rol. Acest fapt explică și existența relației de *1:1* între tabela intermediară și cele 2 tabele pe care le unește.

Tabela **Question** este reprezentată de *Textul* întrebării, și de *User\_Id*, câmp care asociază înrebarea utilizatorului care a postat-o. Relația dintre tabela *User* și tabela *Question* este una de ***0 la mulți***(noi nota cu ***0..\**** de acum), deoarece un utilizator poate pune 0 sau mai multe înrebări, fiecarei întrebare fiindu-i asociat un singur utilizator.

Tablea **Answer** are un *User\_Id* care îi asociază utilizatorul care a postat răspunsul, un *Question\_Id* care leagă răspunsul de întrebarea la care face referență, și un *Text* care reprezintă efectiv răspunsul oferit de utilizator. Relația dintre tabelele *Answer* și *Question* este una de ***0..\****, deoarece O întrebare poate avea 0 sau mai multe răspunsuri, iar un răspuns este asociat unei singure întrebări; iar relația dintre *Answer* și *User* este, de asemenea, una de ***0..\****, deoarece un utilizator poate raspunde de mai multe ori, însă un răspuns îi aparține unei singure persoane.

Tabla **Consult** este puțin mai complicată decât celelalte, dar nu cu mult, deoarece în Fig. 1 apare o dublă relație cu tabela User, dar aceasta apare deoarece o consultație, când este creată, va fi între un Doctor și un Pacient. Astfel, In tabela Consult vor apărea 2 chei străine: Doctor\_Id și Pacient\_Id, acestea fiind denumite “Numele Rolului\_Id” și reprezentând Id-urile persoanelor implicate. Pe lângă aceste Id-uri, se va găsi și câmpurile *Result* si *Prescription*, acestea reprezentând Rezultatul aferent Consultației și Prescripția pusă la dispoziție de către medic. Adițional, pe platformă vor putea fi și fișiere adăugate de doctor, cum ar fi rezultatul unei scanări X-Ray medicale. Astfel, relația dintre tabelele Cosnult și User va fi una dubla, de ***0..\****, deoarece atât un Pacient, cât și un Doctor, pot avea 0 sau mai multe Consultații, însă o Consultație trebuie să fie obligatăriu între un Doctor si un Pacient.

Tabela **Medicine** este cea reprezentativă pentru fiecare medicament, acesta având un *Nume* și un *Prospect*, dar în interiorul aplicației vor putea apărea și fișiere adiționale, ca de exemplu Poze cu modul în care arata cutia medicamentului respectiv.

Câmpul *Prescription* al unei Consultații nu este o listă de Medicamente (List<Medicine>) deoarece pot exista medicamente care nu apar în baza noastră de date, și din acest motiv am decis ca, pentru moment, acest câmp să fie de tip String, motivând astfel lipsa oricărei relații dintre tablea *Medicine* și oricare altă tabelă.

Odată stabilită structura bazei de date, m-am putut apuca de implementarea aplicației în sine. Astfel, proiectul are următoarea structuă:

|  |  |
| --- | --- |
| **Fig. 2** Structura Aplicației | Am decis să folosesc o **Arhitectură** **MultiTier**, mulată dupa arhitectura de tip **Onion**, structurată pe 3 Layere: Business\_Layer, Data\_Layer și Presentation\_Layer. Această alegere a fost făcută pentru a avea un proiect ordonat și aerisit.  Pentru a dezvolta ideea de mai sus, voi explica fiecare Layer în parte, pe scurt:   * **Data\_Layer** este locul unde voi avea Entitățile și operațiile (grupate în Repository-uri și Servicii) cu care voi lucre pe parcursul consturirii aplicației. Am decis să impart acest layer în 2 părți: *Data.Persistence* (locul unde voi comunica cu baza de date – aici intervine ORM-ul folosit: Entity Framework, pe care îl voi aborda imediat) și *Data.Domain* (unde voi găsi Entitățile și Interfețele pentru operațiile pe care le folosesc); * **Business\_Layer** este locul unde am implementat logica din operațiile prezente în partea de *Data.Domain*, mai specific din *Interfaces*. Aici m-am folosit de datele din *Data\_Layer*; * **Presentation\_Layer** reprezintă aplicația web, strucurată pe o **arhitectură** de tipul **MVC** (Model-View-Controller). Acest layer folosește datele din *Business\_Layer*; |

În aplicație există 2 baze de date separate: una în care am Utilizatorii, Rolurile și relația dintre aceste 2 tabele, denumită **Mediarch.DevelopmentUsers**, și alta în care se găsesc înmtrebarile, răspunsurile, medicamentele și consultațiile, cu numele **Mediarch.Development** (Fig. 3).

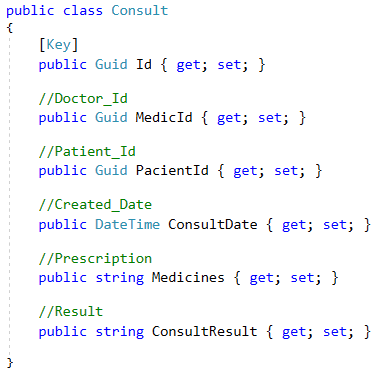
****

**Fig. 3** Baza de Date – SQL Server

După cum am spus, *Data\_Layer* este constituit din Entitățile aplicației și legătura cu Baza de Date. Entitățile aplicației sunt:

|  |  |
| --- | --- |
| **Fig. 4** Entitățile din aplicație | Clasele specifice tabelelor bazei de date ***Mediarch.DevelopmentUsers*** sunt cele din Data.Domain, în subfolderul Entities, iar Clasa denumită „User” în ***Mediarch.Development*** din Fig. 1 (Baza de date contruită cu ajutorul aplicației Vertabelo) este reprezentată de clasa *ApplicationUser*, aflată în *Presentation\_Layer*, deoarece acestă clasă va moșteni clasa *Identity* pusă la dispoziție la crearea aplicației Web, deci nu a trebuit să o fac de la 0, ci să îi adaug câmpurile necesare pentru a ajunge la rezultatul dorit. |

Am specificat mai sus de Entity Framework. Acesta este un ORM (Object-Relational Mapping), adică o punte de comunicare între aplicație și baza de date care asignează fiecărei Clase create de noi o Tabelă din baza de date. Clasa C# va trebui să aibă câmpuri cu aceleași nume ca și cele din tabela din baza de date pentru a le asigna cum trebuie. Exemplu vedem în ***Fig. 5***



**Fig. 5** Entitatea Consult

Un exemplu de folosire a ORM-ului se poate găsi în acest Layer, în Data.Persistance, la comunicarea cu baza de date:



**Fig. 6** Implementarea DatabaseContext-ului utilizând Entity Framework

Este de specificat faptul că Listele de Entități din C# sunt chiar Seturi de Elemente asignate Tabelelor din Baza de Date.

În Business\_Layer, dupa cum am precizat, va fi implementată logica din spatele Repository-urilor și al Serviciilor (care vor folosi operații din Repository-uri).

Repository-urile vor reprezenta clasele cu operațiile standard de tip CRUD, adică cele de genul Create, Read, Update, Delete, aici găsindu-se doar conexiunea directă cu baza de date, ca în *Fig. 6*



**Fig. 7** Relația dintre Repository și Baza de Date

În schimb, serviciile vor folosi atât operațiile definite în mai Repository-uri, cât și obiecte ajutătorare (cum ar fi Environment Data); iar aici se vor defini funcții mai complexe, ca în *Fig. 7*



**Fig. 8** Business Layer – Implementarea unui service

În *Presentation\_Layer* se află efectiv Aplicația Web, construită după arhitecura de tip MVC(Model-View-Controller). Desigur, în Controllere, logica este separată, folosindu-se funcțiile din Servicii (aflate în *Business\_Layer*), deoarece este bine ca acestea să fie aerisite, în special pentru ca ele (Controller-ele) sunt un fel de centru de comandă pentru aplicație.

Aici sunt câteva exemple de elemente ce apar în interiorul acestui Layer:

|  |  |
| --- | --- |
| **Fig. 9** Exemplu Model | **Fig. 10** Mică parte din ConsultController |



**Fig. 11** O parte dintr-un View existent în aplicație

Acum că am terminat de descris în mare Layerle, aș vrea să motivez afirmația făcută mai sus (că această aplicație este una de tip *Onion*) prin faptul că fiecare Layer, cu cât crește nivelul Tier-ului său, acesta se folosește de nivelurile inferioare.

Ca o mică analogie a Arhitecturii Onion: Totul este văzut ca o „Ceapă”. Miezul[Tier 1] reprezintă baza (Entitățile), prin intermediul căreia se dezvoltă următoarele învelișuri[Tier 2] (Repository-urile, din care se dezvoltă și Serviciile), și în final învelișul extern[Tier 3] (Aplicația în sine).

Mai explicit, proiectul are următoarea modelare: *Data\_Layer*(Tier 1) va avea toate Entitățile. Acesta va fi folosit de *Business\_Layer*(Tier 2) în definirea Repository-urilor și a Serviciilor, care la rândul lor, vor fi consumate de Aplicația Web de tip MVC, aflată cel mai sus, în *Presentation-Layer*(Tier 3).

În momentul în care am creat Aplicația Web în MVC am avut opțiunea de a adăuga o modalitate deja existentă de Autentificare, și desigur, am optat pentru această opțiune (***Fig. 12***). Astfel, în momentul Înregistrării, nu se va salva direct parola, ci Hash-ul aplicat ei.

|  |  |  |  |
| --- | --- | --- | --- |
| **Fig. 12** Opțiune de Autentificare | |  |  | | --- | --- | | **Fig. 13** Parola savată | Așa va arăta câmpul care salvează parola în Baza de Date (***Fig. 13***).  Astfel, La Logare se va verifica dacă Hash-ul parolei introduse va coincide cu cel salvat în Baza de Date. |   Desgiur, alegând opțiunea specificată în ***Fig. 12***, aplicația va folosi clasa *Identity* pentru a defini Utilizatorii. Din acest motiv a trebuit să construiesc o nouă clasă care să moștenească clasa *Identity*, unde să mai adaug câmpuri, reprezentând date necesare pentru accesarea Platformei. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Fig. 14** Clasa User moștenind Identity | Astfel, în final, Clasa ApplicationUser va avea următoarele câmpuri folosite, pe lângă cele din **Fig. 14**:   |  |  | | --- | --- | | * Id; * PasswordHash; | * Email; * UserName; |   Dar în momentul în care se înregistrează un nou cont, *Username*-ul va avea aceeași valoare ca și câmpul *Email.* |

Datorită faptului că am adăugat și tabela de Roluri, am fost nevoit să schimb structura funcției de Înregistrare deja existentă, și să construiesc 2 astfel de funcții (***Fig. 15***).



**Fig. 15** Task-urile de Înregistrare din AccountController

Am integrat doar aceste 2 metode de înregistrare deoarece contruile pentru celelalte 2 clase de utilizatori speciali, Owner și Moderatori, vor fi create la rularea aplicației (***Fig. 16*** și ***Fig. 17***):



**Fig. 16** Apelarea (în metodate de inițiere a Aplicațiie) funcției care definește Rolurile și introduce Utilizatorii predefiniți



**Fig. 17** Metoda care crează cele 4 roluri, urmată de crearea Utilizatorilor predefiniți, inclusive Ownerul și Moderatorii

Având în vedere GDPR(General Data Protection Regulation), care a intrat în vigoare pe 25 mai 2018, am considerat că este bine ca datele să fie salvate într-un mod mai sigur, așa că am dezvoltat o extensie a variabilelor de tip String, cu numele de *StringCryptoHelper*.

Aceasa am adăugat-o în interiorul aplicației în partea finală a implementări, pentru a mă asigura că pe platformă există o funcționalitate corectă.

Astfel, am aplicat o criptare de tip 3DES atât peste *majoritatea* datelor senzitive ale Utilizatorilor, cât și peste datele celorlalte entități salvate în baza de date. Iar în momentul când le-am luat din baza de date, am aplicat o funcție de decriptare pentru a obține valoare inițială a datelor respective.



**Fig. 18** Exemplu de salvare criptată a datelor Consultațiilor în baza de date

Am vrut ca aplicația să poată trimite și mesaje în unele situații: la înregistrare, când ți-a fost setat contul ca fiind activ sau inactiv, și (ca și pacient) în momentul în care ți s-a adăugat un nou consult.

O mică parte din cod o puteți vedea în ***Fig. 19*** și ***Fig. 20***.

|  |  |
| --- | --- |
| **Fig. 19** MailSender | Aici am fost nevoit să introduc Usename-ul și parola contului de pe care să se trimită e-mail-urile.  Parola am modificat-o pentru poză, dar în aplicație, ea trebuie să se găsească în formă normlă.  Acestea ar fi setările care au trebuit adăugate pentru a face posibilă trimiterea de e-mail-uri. |



**Fig. 20** Funcție care va trimite un e-mail către pacientul căruia tocmai i s-a introdus o nouă consultație

Un aspect pe care vreau să îl ating este salvarea locală a fișierelor adiționale, care apare pentru unele Entități: Consult, Medicamente și Utilizator. Pentru acest lucru, în fișierul wwwroot al proiectului se va crea (dacă este cazul) un folder cu numele clasei entității (de exemplu Consult/User), iar în interiorul lui se va crea un sub-folder nu Id-ul Entității pentur care se salvează fișierele. În acest ultim sub-folder vor fi salvate fișierele adiționale, introduse prin intermediul platformei.



**Fig. 21** Salvarea pozei de profil (fucnție din interiorul Serviciului pentur Utilizatori)

|  |  |
| --- | --- |
| **Fig. 22** Exemplu de fișiere salvate local în interiorul aplicației | Aici a fost introdus un nou Consult, care a avut ca și fișiere adiționale pe cele din ***Fig. 23***.  Totul s-a salvat local, în folderul *Consults* (deoarece acesta a fost o consultație. Dacă ar fi fost poza unui utilizator, aceasta s-ar fi aflat în folderol *Users*), în sub-folderul cu Id-ul Consultației create (e13c80f1-f112-4836-59f4-08d5d79af…) |

Printre dificultățile întâlnite pe parcursul dezvoltării acestui proiect se numără:

* Documentarea privind JavaScript (Ajax), limbaj folosit în mare parte la secțiunea de Căutare a entităților din aplicație și la paginarea medicamentelor, lucru aplicat în interiorul View-urilor

|  |  |
| --- | --- |
| **Fig. 23** Apelarea prin Ajax a unei funcții dintr-un controller | Aici am apelat prin Ajax funcția SearchMedics(searchedText) din AccountController, pentru a obține Doctorii care care au în interiorul numelui, sau al e-mail-ului, inclusă variabila searchedText, urmând să afișez datele într-un table, în interiorul View-ului. |

* Aplicarea criptării asupra UserName-ului (respectiv a Email-ului) unui utilizator, deoarece când porneam aplicația și se apela funcția SeedUsers (***Fig. 17***) pentru a crea conturile presetate, nu se adăugau în baza de date toate conturile, printre care și contul Onwerului și alte 2 conturi de Moderator. Nu mi-am putat da seama ce a condus la un asemenea scenariu. Din acest motiv, pentru a evita pe viitor situația în care nu se va putea crea un cont, am decis să las aceste date în forma lor inițială.

# Manualul utilizatorului

În această parte voi naviga prin aplicație și voi explica toate opțiunile puse la dispoziție de aplicație.

Odată intrat, vei fi întâmpinat de câteva informații utile, menite să facă o mică introducere persoanelor nou-venite pe această platformă (***Fig. 24***).



**Fig. 24** Primul contact cu aplicația

Aici vei găsi un carusel, care are 3 bannere, acesta fiind adăugat cu scopul de a atrage utilizatorii să folosească aplicația, la fel ca Și acel mesaj plasat imediat sub carusel. Celelalte 2 mesaje sunt:

|  |  |
| --- | --- |
| **Fig. 25** Banner 2 | **Fig. 26** Banner 3 |

În footer se vor afla 2 linkuri (***Fig. 24***) care vor duce la 2 pagini:

* *About*, unde se va găsi Logo-ul (***Fig. 27***) aplicației, urmat de informații generale;
* *Contact* (***Fig. 28***);

|  |  |
| --- | --- |
| **Fig. 27** Logoul | **Fig. 28** Contact |

În acest motiv vei putea alege una din opțiunile următoare: ***Înregistrare*** (fie ca Pacient, fi ca Doctor) sau ***Logare***;

|  |  |  |
| --- | --- | --- |
| **Fig. 29** Înregistrare | Diferențele care apar între înregistrarea drept Doctor și înregistrarea drept Pacient sunt:   * ca Doctor(**Fig. 29**) vor fi necesare 2 seturi de date adiționale: Specializarea pe care o are doctorul și Adresa Cabinetului unde poate fi găsit acesta. * la fiecare înregistrează se introduce în baza de date rolul adițional tipului înregistrării. Dacă te-ai înregistrat ca și Pacient, atunci în baza de date vei avea rol de Pacient. Dac[ te-ai înregistrat ca și Doctor, atunci in baza de date vei avea rol de Doctor | **Fig. 30** Login |

Odată logat,

(Viitor)

/\* Aplicația va permite clienților să se logheze pe baza adresei de e-mail și a unei parole.

Odată logat, un utilizator normal va fi fie un Pacient, fie un Doctor.

Ca și Pacient, poți (în mare parte): să vezi informațiile despre profilul unui doctor(pentru a afla informații utile) sau al altor pacienți, să accesezi informațiile și fișierele consultațiilor tale, și să pui întrebări sau să răspunzi(de exemplu sa lași un review la o intrebare despre un anumit doctor), lucru care se va întampla în secțiunea de Asistență (pe site se va nume „Assistance”);

Ca și Doctor, poți(în mare parte) vedea informațiile despre pacienți și alți doctori, poți adăuga rezultate și prescripții pentru diverși pacienți cu care ai avut de-a face, poți accesa toate consultațiile pe care le-ai creat , și, desigur, ca și în cazul Pacienților, vei avea acces la secțiunea de Asistență, unde vei putea pune întrebari sau raspunde.

\*Desigur, contul doctorului poate fi accesat de asistentul/asistenta acestuia, deoarece medicul în cauză poate fi ocupat cu alți pacienți în viața reală.

Restul de 2 categorii speciale de utilizatori, Owenerul și Moderatorii, vor putea modifica informațiile deja existente pe site, vor face management la lista cu medicamente pusă la dispoziție de către site, și vor fi cei care vor avea grijă de problemele care pot interveni(de exemplu: dacă s-a primit vreun mail de la un doctor care a constatat că cineva neautorizat a intrat pe contul său și a adaugat o consultație falsă, moderatorii vor fi cei care vor rezolva asemenea situații). Acestei clase de utilizatori le va fi destul de ușor să observe ce s-a întâmplat pe platformă în ultima vreme deoarece exisă o rubrică specială pentru ei, cu numele “Data Records”, unde vor găsi toate activitățile desfășurate în ultima vreme, de la înregistrări de utilizatori și crearea consultațiilor, până la adăugarea de întrebări și răspunsuri. \*/

# Concluzii

Această aplicație a reprezentat o provocare destul de mare încă de la început, deoarece inițial a fost doar o idee(ca orice alt proiect), care, după multă muncă, a ajuns să fie un produs finisat.

Pe partea de Back-end a reprezentat o provocare deoarece a trebuit să mă gândesc bine cum va arăta structura bazei de date, fiecare tabelă în parte și în mare cam ce operații va trebui să fac peste aceste tabele. Ca și model arhitectural am ales MVC-ul(Model-View-Controller), lucru care m-a ajutat destul de mult în dezvolarea aplicației, mai ales a logicii din spatele acesteia.

Pe partea de Front-end a reprezentat o provocare și mai mare deoarece eu nu m-a atras foarte tare, dar am ajuns să ma împac destul de bine și cu această parte, mai ales că am avut un ajutor destul de mare: Bootstrap-ul, care, În proiectul final, este de nerecunoscut, deoarece sunt foarte multe componente modificate și suficient de multe componente adăugate.

Marea majoritate a functionalității a fost cea gândită la început, dar, desigur, adăugări au mai apărut și pe parcursul finisării aplicației, idei apărând în timp ce navigam prin aceasta pentru a testa dacă totul este cum ar treubi să fie.

+Îmbunătățiri

# Bibliografie

Sfatulmedicului

Vertabelo

Stackoverflow(criptare)

Youtube? +si/sau site-uri oficial (mail sender)

<https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework>

<http://www.privacyone.ro/dpo/gdpr.html>

<https://www.microsoft.com/net/learn/what-is-dotnet>

<https://www.jetbrains.com/resharper/>

<https://searchwindevelopment.techtarget.com/definition/C>

<https://www.adobe.com/ro/products/illustrator/free-trial-download.html>

<https://en.wikipedia.org/wiki/Multitier_architecture>

<https://www.c-sharpcorner.com/article/onion-architecture-in-asp-net-core-mvc/>

<https://www.dpap.ro/blog/ce-se-schimba-gestionarea-datelor-personale-incepand-cu-luna-mai-2018/>

<https://www.medlife.ro/rezultate-analize> (descoperit recent)

<http://www.clinica-sante.ro/> (descoperit recent)

<https://www.reginamaria.ro/rezultate-analize> (descoperit recent)

<https://www.synevo.ro/pacienti/rezultate-analize/> (descoperit recent)

<https://www.arcadiamedical.ro/>

# Tehnologii Folosite



## C# (.Net, MVC, ORM-EntityFramework, ASP.NET, Identity)

C# este un limbaj de programare orientat-obiect de la Microsoft care are ca scop combinarea puterii de calcul a C++ cu ușurința de programare a Visual Basic. C# se bazează pe C++ și conține caracteristici similare cu cele din Java.

Info

## SQL (SQL Server Express)

Info

## Source Controlling -> GitHub (GitKraker)

Info+poza cu gitkraken

## Bootstrap

Info (poza din 2.1 -> lib-ul + “Am modificat chestii in bootstrap”)

## HTML, CSS & JavaScript

Info

## Adobe Illustrator

Info