

**UNIVERSITATEA „ALEXANDRU IOAN CUZA” DIN IAȘI**  
**FACULTATEA DE INFORMATICĂ**



**LUCRARE DE LICENȚĂ**

**MediArch**  
**Aplicație Web**

Propusă de

**Moroșanu C.D. George-Cosmin**

**Sesiunea: Iulie 2018**

Coordonator științific

**Drd. Colab. Florin Olariu**

**UNIVERSITATEA „ALEXANDRU IOAN CUZA” DIN IAȘI**  
**FACULTATEA DE INFORMATICĂ**

# **MediArch**

## **Aplicație Web**

**Moroșanu C.D. George-Cosmin**

**Sesiunea: Iulie 2018**

**Coordonator științific**

**Drd. Colab. Florin Olariu**

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele \_\_\_\_\_

Data \_\_\_\_\_ Semnătura \_\_\_\_\_

**DECLARAȚIE privind originalitatea conținutului lucrării de licență**

Subsemnatul(a) \_\_\_\_\_,

cu domiciliul în \_\_\_\_\_,

născut(ă) la data de \_\_\_\_\_, identificat prin CNP \_\_\_\_\_,

absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de  
\_\_\_\_\_ specializarea \_\_\_\_\_, promoția

\_\_\_\_\_, declar pe propria răspundere, cunoscând consecințele falsului în  
declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr.  
1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul:

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

\_\_\_\_\_elaborată sub îndrumarea dlui / dnei  
\_\_\_\_\_, pe care urmează să o susțin în fața  
comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin  
orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea  
conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări  
științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei  
lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere  
că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data azi, \_\_\_\_\_

Semnătură student \_\_\_\_\_

## DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „**MediArch**”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași,

Absolvent *Moroșanu George-Cosmin*

---

# Cuprins

Introducere .....	6
Motivație .....	6
Scopul .....	6
Alegerea temei .....	6
Alegerea tehnologiei.....	7
Context .....	7
Cerințe funcționale .....	8
Gradul de noutate .....	9
Uzabilitate .....	9
1. Contribuții.....	10
2. Proiectare și Implementare .....	11
3. Manualul utilizatorului .....	25
4. Concluzii.....	36
Bibliografie .....	37
Anexa 1 (Tehnologii Utilizate) .....	38
ASP.NET.....	38
SQL (SQL Server Express).....	38
Version Control - GitHub (GitKraken) .....	39
HTML, CSS, JavaScript și Bootstrap .....	39
Adobe Illustrator .....	40

# 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, oriunde s-ar afla. 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 ușor și rapid să aibă acces la informațiile consultațiilor pe care le-au avut până acum. Și va fi și foarte ușor de gestionat, deoarece utilizatorii speciali, adică Ownerul și Moderatorii, vor avea acces la o listă cu tot ce s-a întâmplat pe platformă în ultimele 2 săptămâni, și vor putea imediat să intervină, luând măsurile necesare dacă observă 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 pe o singură platformă, precum și accesarea rapidă a acestora. Astfel, dacă cineva va avea nevoie la un moment dat 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 respectivă 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, de la care am primit și prescripții medicale. Bine înțeles că s-a întâmplat 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 foarte 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 destul de bine, și a început să îmi suradă ideea dezvoltării unei aplicații web.

În primul semestru din anul III am participat la materia „Introducere în .NET”, în cadrul căruia am făcut echipă cu alte 5 persoane cu scopul comun de a dezvolta o nouă aplicație web, pe o temă aleasă de noi. Aceasta a fost „Course Manager” (aplicație care să facă un management mai bun, concepută inițial pentru cursul de „Introducere în .NET”, și 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).

## Context

Aplicația „**MediArch**” (prescurtare de la „*Medical Archive*”) 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 care au participat la 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 roluri sunt: Owner, Moderator, Medic(Doctor) și Pacient. Primele 2 roluri precizate vor avea acces la funcționalități mai complexe 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 diferită pentru fiecare cont, din motive de securitate), astfel devenind 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 utilizator în procesul de înregistrare). Dacă utilizatorul este pentru prima dată pe platformă, acesta își va putea face destul de repede și ușor 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 o consultație nouă din această secțiune), și lista cu Specializări, 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 Asistență:** Rubrica de întrebări și răspunsuri;
- **Managementul datelor:** Secțiunea pentru utilizatorii speciali, denumită „Data Records” unde se va găsi tot ce s-a întâmplat pe aplicație 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 ale Medicilor/Doctorilor 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 platforme pentru a obține acces la toate informațiile de care am nevoie.

**MediArch** va strânge toate aceste informații într-un singur loc. Astfel orice persoană își va putea accesa întregul istoric medical propriu și privat. În plus, aici va exista și o listă de medicamente care va ajuta utilizatorii să se documenteze despre medicamentele căutate.

## **Uzabilitate**

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

# 1. Contribuții

**MediArch** este o aplicație web, de tip client - server, proiectată 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 platformei.

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 continuă a acesteia, pe parcursul 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 ușoară și intuitivă a tuturor secțiunilor;**
- **Navigabilitatea fluentă** prin aplicație, putând ajunge din orice pagină în oricare alta foarte rapid;
- **Accesibilitatea la informațiile medicamentelor**, lucru care salvează timp;
- **Secțiunea de Asistență**, unde te poți informa (căutând subiectul de interes) sau poți găsi răspunsuri la diverse curiozități medicale;
- **Lista cu toți Doctorii înregistrați**, aceștia aparținând de diverse clinici, sau fiind private. Astfel se pot obține date utile, și se poate cere și părerea altor utilizatori (despre Doctorul căutat) în secțiunea de Asistență;
- **Afișarea informațiilor în mod privat pentru fiecare utilizator în parte;**
- **Mod eficient de mentenanță**, datorat în mare parte opțiunii de „Data Records”, disponibilă pentru utilizatorii speciali;
- **Tratarea cazurilor de eroare** cauzate de baza de date;

## 2. 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 dezvoltarea 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:

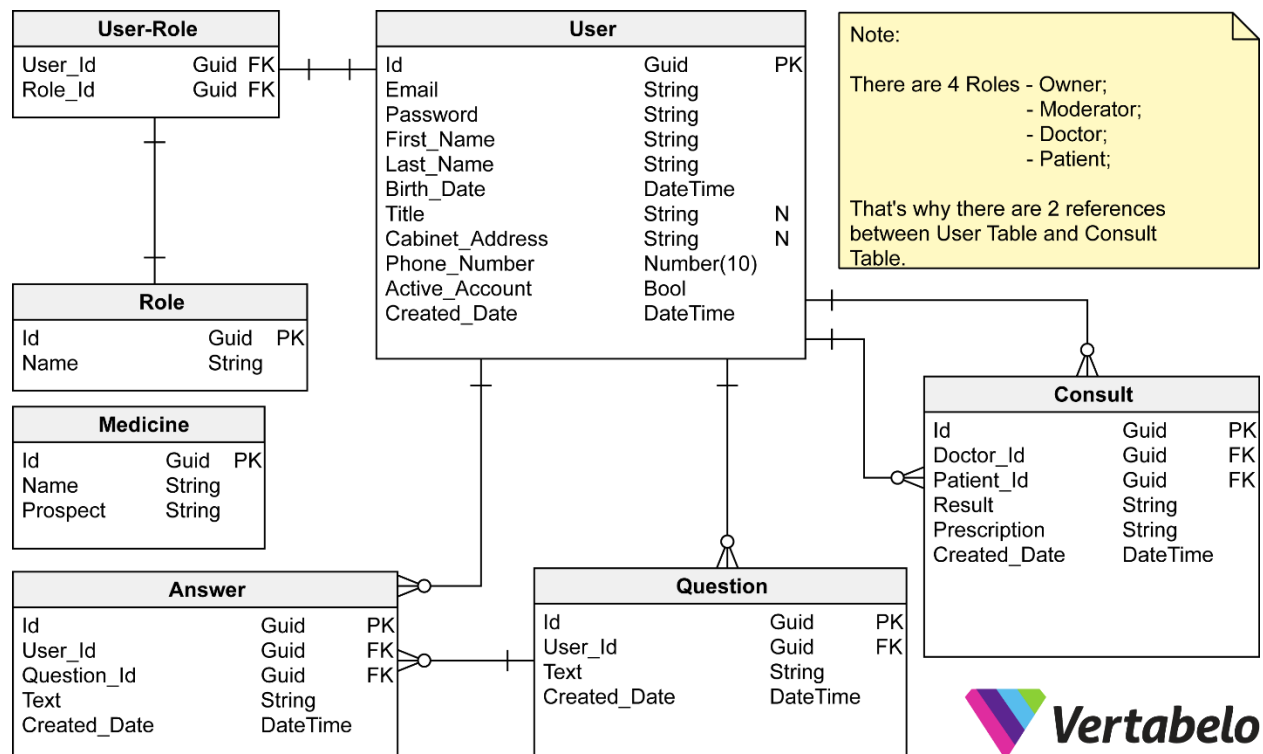


Fig. 1 Structura bazei de date și relația dintre tabele

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 urmează 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 Identificatorul ales în mod unic pentru fiecare element în parte. Pe lângă acest prim câmp comun, va apărea la majoritatea tabelelor și câmpul „***Created\_Date***”, reprezentând data la care a fost creată 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, fiecare în parte reprezentând 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 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** este cea mai importantă. Aceasta are câmpurile:

- *Email* - care va juca rol și de *UserName*, 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 formată din *Textul* întrebării, și din *User\_Id*, câmp care asociază întrebarea cu utilizatorul care a postat-o. Relația dintre tabela *User* și tabela *Question* este una de **0 la mulți** (voi nota cu **0..\*** de acum), deoarece un utilizator poate pune 0 sau mai multe întrebări, fiecărei întrebări 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 referință, 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 răspunde 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*. Această dublă relație apare deoarece o consultație, când este creată, va fi între un Doctor și un Pacient. Astfel, în tabela *Consult* vor apărea 2 chei străine: *Doctor\_Id* și *Patient\_Id*, acestea având structura „*Rol\_Id*” și reprezentând Id-urile persoanelor implicate. Pe lângă aceste Id-uri se va găsi și câmpurile *Result* și *Prescription*, acestea reprezentând Rezultatul aferent Consultației, și Prescripția pusă la dispoziție de către Medic. Adicional, pe platformă vor putea fi și fișiere adăugate de Doctor, cum ar fi rezultatul unei scanări X-Ray medicale. Astfel, relația dintre tabelele *Consult* și *User* va fi una dublă, de **0..\***, deoarece atât un Pacient, cât și un Doctor, pot avea 0 sau mai multe Consultații, însă o Consultație trebuie să fie obligatoriu între un Doctor și 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 de date, și din acest motiv am decis ca,

pentru moment, acest câmp să fie de tip String, motivând astfel lipsa oricărei relații dintre tabela *Medicine* și oricare altă tabelă.

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

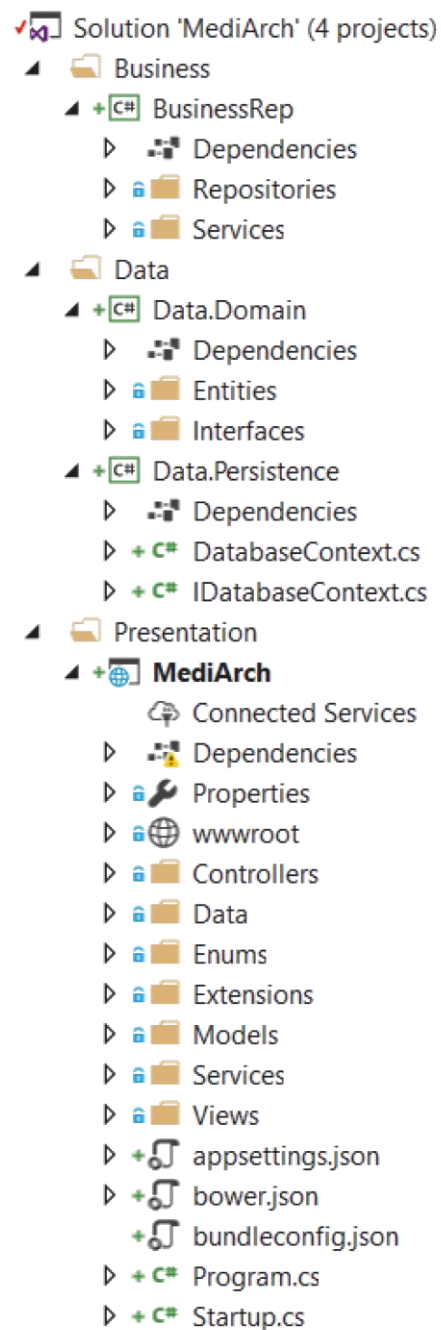


Fig. 2 Structura aplicației

Am decis să folosesc o **Arhitectură Onion**, mulată după arhitectura de tip **Multitier**, 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 lucra pe parcursul proiectării aplicației. Am decis să împart 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*, folosind un **pattern** de tip **Repository**;
- **Presentation\_Layer** reprezintă aplicația web, structurată 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 întrebările, răspunsurile, medicamentele și consultațiile, cu numele **Mediarch.Development** (Fig. 3).

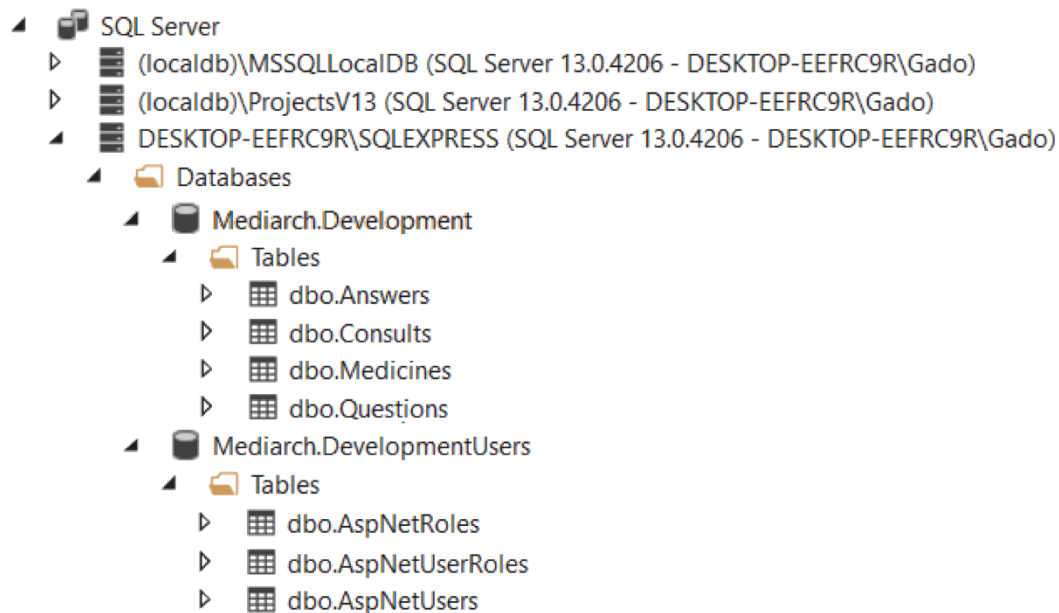


Fig. 3 Baza de date – SQL Server

După cum am spus, *Data\_Layer* este constituit din Entitățile aplicației și legătura cu Baza de Date. Entitățile aplicației sunt cele din Fig. 4.

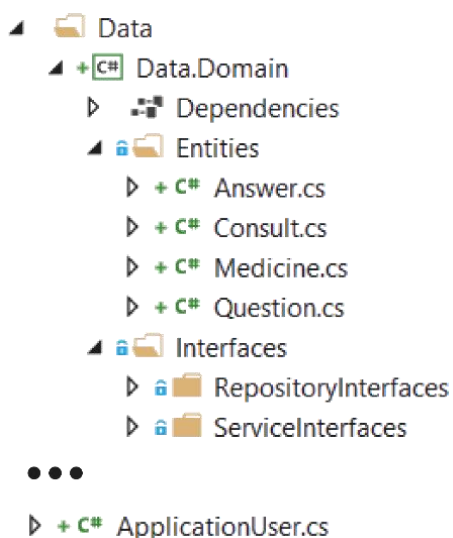


Fig. 4 Entitățile din aplicație

Clasele specifice tabelor bazei de date **Mediarch.DevelopmentUsers** sunt cele din *Data.Domain*, în subfolderul *Entities*.

Referitor la baza de date **Mediarch.Development**, Clasa denumită „User” în Fig. 1 este reprezentată de clasa *ApplicationUser*, aflată în *Presentation\_Layer*. Aceasta se află aici deoarece va moșteni clasa *Identity* pusă la dispoziție la crearea aplicației Web. Despre *Identity* voi vorbi în paginile următoare.

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. Un exemplu îl vedem în *Fig. 5*.

```
namespace Data.Domain.Entities
{
    public class Consult
    {
        [Key]
        public Guid Id { get; set; }

        public Guid Doctor_Id { get; set; }

        public Guid Patient_Id { get; set; }

        public DateTime Created_Date { get; set; }

        public string Prescription { get; set; }

        public string Result { get; set; }
    }
}
```

**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**).

```
using Microsoft.EntityFrameworkCore;

namespace Data.Persistance
{
    public class DatabaseContext : DbContext, IDatabaseContext
    {
        public DatabaseContext(DbContextOptions<DatabaseContext> options) : base(options)
        {
            Database.EnsureCreated();
        }

        public DbSet<Data.Domain.Entities.Consult> Consults { get; set; }

        public DbSet<Data.Domain.Entities.Medicine> Medicines { get; set; }

        public DbSet<Data.Domain.Entities.Answer> Answers { get; set; }

        public DbSet<Data.Domain.Entities.Question> Questions { get; set; }
    }
}
```

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

Este de specificat faptul că listele de entități din C# sunt chiar seturi de elemente asignate tabelor din baza de date.



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

Aici am ales un *pattern* de tip **Repository** deoarece am vrut să nu existe o dependență între Layerele superioare și baza de date. Din acest motiv am optat pentru abstractizarea tuturor Serviciilor și a Repository-urilor, ducând la prezența Interfețelor.

Repository-urile vor reprezenta clasele cu operațiile standard de tip CRUD, adică cele de genul Create, Read, Update, Delete, aici găsiindu-se doar conexiunea directă cu baza de date, ca în Fig. 7.

```
public class MedicineRepository : IMedicineRepository
{
    private readonly DatabaseContext _databaseContext;

    public MedicineRepository(DatabaseContext databaseContext)
    {
        _databaseContext = databaseContext;
    }
}
```

Fig. 7 Relația dintre Repository și Baza de Date

În schimb, serviciile vor folosi atât funcțiile unor Repository-uri, cât și obiecte ajutătoare (cum ar fi Environment Data). Aici se vor defini funcții mai complexe, ca în Fig. 8.

```
public class ConsultService : IConsultService
{
    private readonly IConsultRepository _repository;
    private readonly IHostingEnvironment _env;

    public ConsultService(IConsultRepository repository, IHostingEnvironment env)
    {
        _repository = repository;
        _env = env;
    }

    public async Task Create(ConsultCreateModel consultCreateModel)
    {
        Consult consult = new Consult();
        _repository.Create(consult);
        if (consultCreateModel.File != null)
        {
            foreach (var file in consultCreateModel.File)
            {
                if (file.Length > 0)
                {
                    string path = Path.Combine(_env.WebRootPath, "Consults\\" + consult.Id);
                    if (!Directory.Exists(path))
                    {
                        Directory.CreateDirectory(path);
                    }
                    using (var fileStream = new FileStream(Path.Combine(path, file.FileName), FileMode.Create))
                    {
                        await file.CopyToAsync(fileStream);
                    }
                }
            }
        }
    }
}
```

Fig. 8 Business\_Layer – Implementarea unui service

În *Presentation\_Layer* se află efectiv Aplicația Web, conturată după arhitectura 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 că ele (Controller-ele) sunt un fel de centru de comandă pentru aplicație.

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

```
public class ApplicationUserViewModel
{
    public string Id { get; set; }

    public string PhoneNumber { get; set; }

    public string FirstName { get; set; }

    public string LastName { get; set; }

    public DateTime BirthDate { get; set; }

    public string Title { get; set; }

    public string CabinetAdress { get; set; }
}
```

Fig. 9 Exemplu de Model

```
[Authorize(Roles = "Medic")]
public IActionResult MyConsultsPaginated(string id, int NoPage)
{
    ViewData["Id"] = id;
    Guid medicId = new Guid(id);
    return View(_service.Get5ConsultsForDoctorByIndex(medicId, NoPage));
}

[Authorize(Roles = "Pacient")]
public IActionResult MyResultsPaginated(string id, int NoPage)
{
    ViewData["Id"] = id;
    Guid pacientId = new Guid(id);
    return View(_service.Get5ConsultsForPacientByIndex(pacientId, NoPage));
}
```

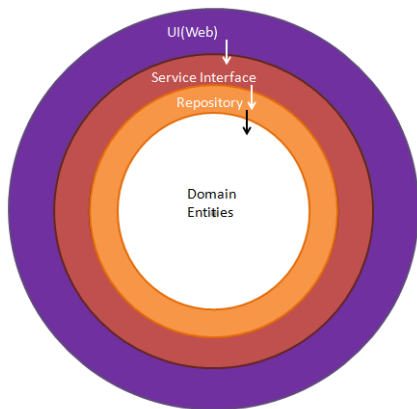
Fig. 10 Mică parte din ConsultController

```
<tbody>
    @if(int NoElement == 0; )
    @foreach (var item in Model)
    {
        <tr id="Element_@NoElement++">
            <td>
                <div class="btn-group dropright">
                    <a class="btn btn-secondary dropdown-toggle dropdown-toggle-split flt_right" data-toggle="dropdown"><b>@item.Name </b></a>
                    <ul class="dropdown-menu special_dropright">
                        @if (User.IsInRole("Owner") || User.IsInRole("Moderator"))
                        {
                            <li><a asp-action="Edit" asp-route-id="@item.Id">Edit</a></li>
                            <li><a asp-action="Details" asp-route-id="@item.Id">Details</a></li>
                            @if (User.IsInRole("Owner") || User.IsInRole("Moderator"))
                            {
                                <li><a asp-action="Delete" asp-route-id="@item.Id">Delete</a></li>
                            }
                        }
                    </ul>
                </div>
            </td>
            <td>...</td>
            <td>...</td>
        </tr>
    }
</tbody>
```

Fig. 11 O parte dintr-un View, din aplicație

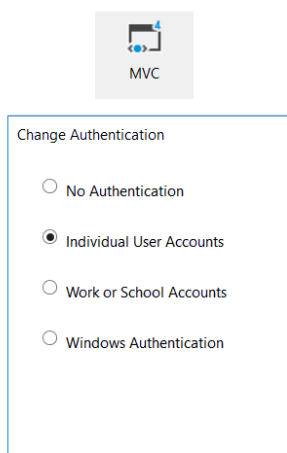
Acum că am terminat de descris în mare Layerele, aş vrea să motivez afirmația făcută mai sus (că această aplicație este una de tip *Onion*) prin faptul că fiecare Layer, cu cât crește nivelul Tier-ului său, acesta se folosește de nivelurile inferioare.

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).



**Fig. 12** Arhitectura de tip Onion (Bibliografie - „Arhitectura Onion”)

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. 13). Astfel, în momentul Înregistrării, nu se va salva direct parola, ci Hash-ul aplicat ei.



**Fig. 13** Opțiune de Autentificare

PasswordHash
AQAAAAEAACc...
AQAAAAEAACc...
AQAAAAEAACc...
AQAAAAEAACc...

**Fig. 14** Parola salvată

Așa va arăta câmpul care salvează parola în baza de date (Fig. 14).

Astfel, la Logare se va verifica dacă Hash-ul parolei introduse va coincide cu cel salvat în baza de date.

Desigur, alegând opțiunea specificată în Fig. 13, aplicația va folosi clasa *Identity* pentru a defini Utilizatorii. Din acest motiv a trebuit să concep o nouă clasă care să moștenească clasa *Identity*, unde să mai adaug câmpuri, reprezentând date necesare pentru accesarea Platformei.

```
using System;
using Microsoft.AspNetCore.Identity;

namespace MediArch.Models
{
    public class ApplicationUser : IdentityUser
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public DateTime BirthDate { get; set; }
        public string Title { get; set; }
        public string CabinetAddress { get; set; }
        public bool ActiveAccount { get; set; }
        public DateTime CreatedDate { get; set; }
    }
}
```

**Fig. 15** Clasa „User” moștenind Identity

Astfel, în final, Clasa *ApplicationUser* va avea următoarele câmpuri folosite, pe lângă cele din Fig. 15:

- Id;
- Email;
- PasswordHash;
- 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 proiectez 2 astfel de funcții (Fig. 16).

```
[HttpGet]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public IActionResult RegisterMedic(string returnUrl = null)...

[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<IActionResult> RegisterMedic(RegisterMedicViewModel model, string returnUrl = null)...

[HttpGet]
[AllowAnonymous]
public IActionResult RegisterPacient(string returnUrl = null)...

[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<IActionResult> RegisterPacient(RegisterPacientViewModel model, string returnUrl = null)...
```

Fig. 16 Metodele de Înregistrare din AccountController

Am integrat doar aceste 2 metode de înregistrare deoarece conturile pentru celelalte 2 clase de utilizatori speciali, Owner și Moderatori, vor fi create la rularea aplicației (Fig. 17 și Fig. 18).

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env, UserManager<ApplicationUser> userManager, RoleManager<IdentityRole> roleManager)
{
    MyIdentityDataInitializer.SeedData(roleManager, userManager);
}
```

Fig. 17 Apelarea (în metodate de inițiere a aplicației) funcției unde se definesc rolurile și se introduc utilizatorii predefiniți

```
public static class MyIdentityDataInitializer
{
    public static void SeedData(RoleManager<IdentityRole> roleManager,
                                UserManager<ApplicationUser> userManager)
    {
        SeedRoles(roleManager);
        SeedUsers(userManager);
    }
}
```

Fig. 18 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*.

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

Astfel, am aplicat o criptare de tip 3DES atât peste *majoritatea* datelor sensitive 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.

Un exemplu de salvare criptată a datelor în baza de date se observă în **Fig. 19**.

Id	ConsultDate	ConsultResult	Me...	Medicines	Paci...
787...	6/21/2018 6:16:45 PM	6UElbn++p84QYXi6SF4Q==	c609...	secNeVwk8sqeYzWtcCv+DeRSYdI4STHW3fJOldJFu/tfAzIXWgzpIkZSwPLeZy+RhXlrv1NJ9yMo=	b2c...
ad8...	6/21/2018 6:20:01 PM	M407IGtxnjA=	c609...	9RfwkGJzelacwLwHhKneuDaA6N3YD/7y	Sc99...
▶*	NULL	NULL	NULL	NULL	NULL

**Fig. 19** Date din tabela „Consult”, stocate în baza de date.

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

O mică parte din cod o puteți vedea în **Fig. 20** și **Fig. 21**.

```
public class EmailSender : IEmailSender
{
    private const string Username = "mediarch.noreply@gmail.com";
    private const string Password = "*****";
    private const string Server = "smtp.gmail.com";

    public Task SendEmailAsync(string email, string subject, string message)
    {
        var mail = new MailMessage(Username, email, subject, message);
        var client = new SmtpClient(Server)
        {
            Port = 587, // Gmail port
            Credentials = new NetworkCredential(Username, Password),
            EnableSsl = true,
        };
        return client.SendMailAsync(mail);
    }
}
```

**Fig. 20** MailSender

Aici am fost nevoit să introduc Username-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 normală.

Acestea ar fi setările care au trebuit adăugate pentru a face posibilă trimiterea de e-mail-uri.

```
public static Task SendEmailNewConsultAsync(this IEmailSender emailSender, string patientEmail, string patientName, string doctorName)
{
    string MessageSubject = "You got a new Consult!";
    string MessageBody = "Hey, " + patientName + "\r\n\r\n" +
        "You have a new Consult added by" + doctorName + "!\r\n" +
        "You can go and check it out!" + "\r\n" +
        "\r\n" +
        "Best Regards, " + "\r\n" +
        "MeriArch Staff" + "\r\n";
    return emailSender.SendEmailAsync(patientEmail, MessageSubject, MessageBody);
}
```

**Fig. 21** 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: Consultații, Medicamente și Utilizatori. 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 subfolder cu Id-ul Entității pentru care se salvează fișierele. În acest ultim subfolder vor fi salvate fișierele adiționale, introduse prin intermediul platformei.

```
public async Task UploadProfilePicture(string id, IEnumerable<IFormFile> UploadedFile)
{
    //Lista cu extensii pentru Imagini
    var extensions = new List<string>{...};
    if (UploadedFile != null && UploadedFile.Count() == 1)
    {
        bool ok = false;
        //Verific dacă este imagine. Dacă da, atunci ok va fi True
        foreach (var file in UploadedFile){...}
        if (ok == true)
        {
            var path = Path.Combine(_env.WebRootPath, "Users/" + id);
            if (!Directory.Exists(path))
            {
                Directory.CreateDirectory(path);

                //Sterg imaginea care era înainte și o pun pe cea nouă
                Guid UIId = new Guid(id);
                string currentFileName = GetNameOfProfilePictureById(UIId);
                DeleteCertainFile(UIId, currentFileName);

                foreach (var file in UploadedFile)
                {
                    if (file.Length > 0)
                    {
                        using (var fileStream = new FileStream(Path.Combine(path, file.FileName), FileMode.Create))
                        {
                            await file.CopyToAsync(fileStream);
                        }
                    }
                }
            }
        }
    }
}
```

Fig. 22 Salvarea pozei de profil (funcție din interiorul serviciului pentru Utilizatori)

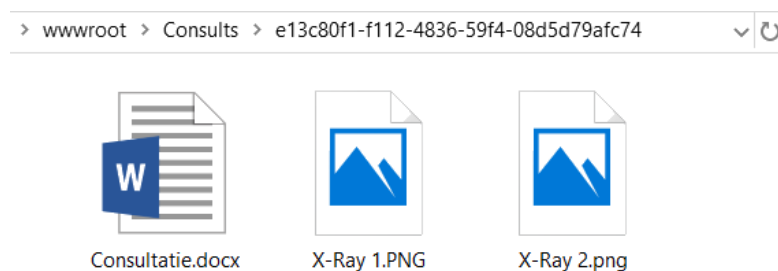


Fig. 23 Exemplu de fișiere salvate local în interiorul aplicației

Aici a fost introdusă o nouă Consultație, care a avut ca și fișiere adiționale pe cele din Fig. 23.

Totul s-a salvat local, în folderul *Consults*, în subfolderul cu Id-ul Consultației create (e13c80f1-f112-...). Dacă ar fi fost poza unui utilizator, aceasta s-ar fi aflat în folderul *Users*, subfolderul cu Id-ul utilizatorului.

Pentru validarea datelor m-am folosit de FluentValidation (Fig. 24) și DataAnnotations (Fig. 25).

```
using FluentValidation;

namespace MediArch.Models.Validation
{
    public class UserValidation : AbstractValidator<RegisterMedicViewModel>
    {
        private readonly ApplicationDbContext _databaseService;

        public UserValidation(ApplicationDbContext databaseService)
        {
            _databaseService = databaseService;

            RuleFor(x => x.Email).Must(BeUniqueEmail).WithMessage("This mail address was already used!");
        }
    }
}
```

Fig. 24 Exemplu de FluentValidation

```
[Required(ErrorMessage = "Phone number is required!")]
[Display(Name = "Phone Number")]
[DataType("PhoneNumber", ErrorMessage = "Format not allowed.")]
[RegularExpression(@"([0-9]){10}", ErrorMessage = "Format not respected.")]
public string PhoneNumber { get; set; }
```

Fig. 25 Exemplu de DataAnnotation

Pe partea de front-end am abordat și *tratarea de erori* cauzate de baza de date, cum ar fi ștergerea accidental (sau din anumite motive) a unui utilizator. Întru-un asemenea scenariu, de exemplu la pagina de consultații, în locul numelui Doctorului/Pacientului care lipsește din baza de date, va apărea mesajul „Doctor/Patient not found”, depinzând de rolul utilizatorului lipsă. Un astfel de exemplu se găsește în Fig. 26.

Consult's Date	Medic	Patient
6/24/2018 11:28:25 AM ▶	Doctor not found	 Phillips Seth
6/24/2018 11:23:27 AM ▶	 Simpson Lara	Patient not found

Fig. 26 Abordarea erorilor pe pagina de Consultații

Mesaje asemănătoare vor apărea și pe pagina de Asistență, și pe cea de Data Records. Pentru Întrebări și/sau Răspunsuri, în loc de utilizatorul care va lipsi din baza de date va apărea mesajul „User not found”.

Printre dificultățile întâlnite pe parcursul dezvoltării acestui proiect se numără:

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

```
$(document).ready(function () {  
    SearchByName.addListener("input", function (e) {  
        runSearchByCertainString();  
    });  
    function runSearchByCertainString() {  
        var searchedText = $('#SearchByName').val();  
        if (!searchedText) {  
            searchedText = "null";  
        }  
        $.ajax({  
            type: "GET",  
            url: '@Url.Action("SearchMedics", "Account")',  
            contentType: "application/json; charset=utf-8",  
            data: { text: searchedText },  
            dataType: "json",  
            success: function (rez) {
```

Aici am apelat prin Ajax funcția SearchMedics(searchedText) din AccountController, pentru a obține doctorii care au în interiorul numelui, sau al e-mail-ului, inclusiv variabila searchedText, urmând să afișez datele într-un tabel, în interiorul View-ului.

Fig. 27 Apelarea prin Ajax a unei funcții dintr-un controller

- Aplicarea criptării asupra Username-ului (reprezentat de Email) unui utilizator, deoarece când porneam aplicația și se apela funcția SeedUsers (Fig. 18) pentru a crea conturile presetate, nu se adăugau în baza de date toate conturile, printre care contul Ownerului și alte 2 conturi de Moderator. Nu mi-am putut da seama ce a cauzat 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.



### 3. 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. 28).

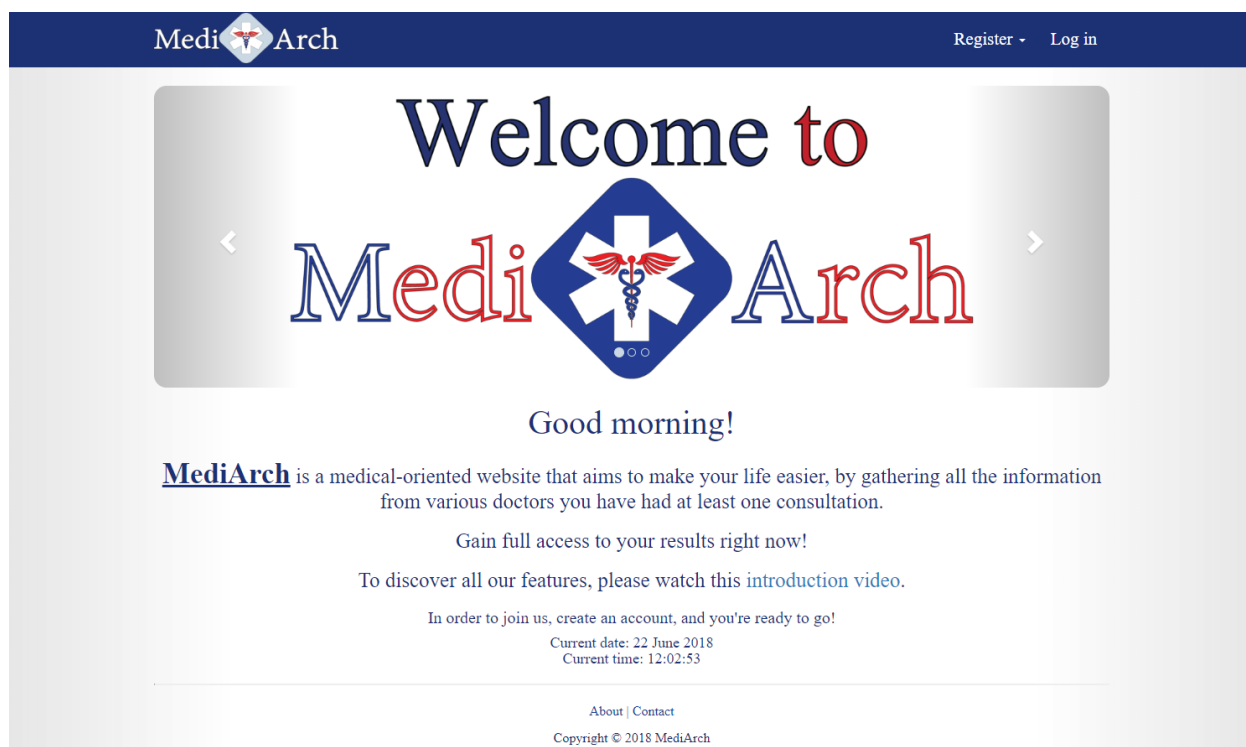


Fig. 28 Primul contact cu aplicația

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:

**Make your life easier!**  
Your medical profile is **NOW**  
at your fingertips!

Fig. 29 Banner 2

**For any problems,**  
**don't hesitate to contact us!**

For more information about us, [press here](#).

Fig. 30 Banner 3

În footer se vor afla 2 linkuri (Fig. 28) care vor duce la 2 pagini:

- *About*, unde se va găsi Logo-ul (Fig. 31) aplicației, urmat de informații generale;
- *Contact* (Fig. 32);



Fig. 31 Logo-ul aplicației



Contact:

Address: Headquarter's Adress  
 Fax: ...  
 Phone: 07518710\*\*  
 Official e-mail: [mediarch.noreply@gmail.com](mailto:mediarch.noreply@gmail.com)  
 Support(General): [MediArchSupport@mediarch.com](mailto:MediArchSupport@mediarch.com)  
 Moderators: [Moderator1@mediarch.com](mailto:Moderator1@mediarch.com)  
[Moderator2@mediarch.com](mailto:Moderator2@mediarch.com)  
[Moderator3@mediarch.com](mailto:Moderator3@mediarch.com)  
[Moderator4@mediarch.com](mailto:Moderator4@mediarch.com)  
[Moderator5@mediarch.com](mailto:Moderator5@mediarch.com)

Fig. 32 Contact

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

Register as a Doctor!

First Name <input type="text"/>	Birth Date <input type="text"/>
Last Name <input type="text"/>	Phone Number <input type="text"/>
Your Email(Username) <input type="text"/>	Specialization <input type="text"/>
Password <input type="password"/>	Cabinet Address <input type="text"/>
Confirm Password <input type="password"/>	

Fig. 33 Înregistrare

Diferențele care apar între înregistrarea drept Doctor și înregistrarea drept Pacient sunt:

- ca Doctor (Fig. 33) 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. Dacă te-ai înregistrat ca și Doctor, atunci în baza de date vei avea rol de Doctor;

Log in

Use your account to log in.

Email <input type="text"/>
Password <input type="password"/>
<input type="checkbox"/> Remember me?
<input type="button" value="Log in"/>
<a href="#">Forgot your password?</a>

Fig. 34 Login

Imediat după înregistrat, vei primi un Email care va confirma validitatea înregistrării. Astfel de notificări vor apărea atât în momentul înregistrării, cât și atunci când se va adăuga o consultație nouă, sau când contul tău a fost setat drept Activ/Inactiv (**Fig. 35**).

<input type="checkbox"/> ☆	mediarch.noreply@gmail.	Active Account - Hey, George-Cosmin Your account was just set on active! Best Regards, MediArch Staff
<input type="checkbox"/> ☆	mediarch.noreply@gmail.	Inactive Account - Hey, George-Cosmin Your account was set on inactive, due to some issues! You can get in contact with our staff to solve this situation. Best Regards, MediArch Staff
<input type="checkbox"/> ☆	mediarch.noreply@gmail.	You got a new Consult! - Hey, George-Cosmin You have a new Consult added by (Medic1@gmail.com)! You can go and check it out! Best Regards, MediArch Staff
<input type="checkbox"/> ☆	mediarch.noreply@gmail.	Welcome to MediArch! - Hey, George-Cosmin Morosanu Thank you for registering on our website! Hope you will have a pleasant experience here! For further information, contact us!

**Fig. 35** Mailurile primite de la aplicație

## InactiveAccount

Your account is set to Inactive for now!

We think that there are some suspicious activities on your account.

Until we figure out what caused this, you won't be able to access your account.

You will be notified via E-mail when you will be able to Log In again.

We apologize for any inconvenience we have created.

**Fig. 36** Cont inactiv

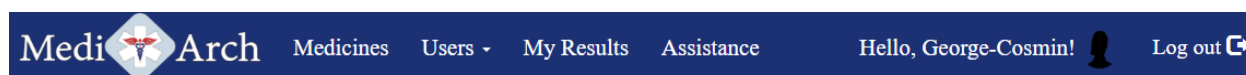
Mesajul din **Fig. 36** apare în momentul în care se va încerca Logarea pe un cont care a fost setat drept Inactiv de către un Moderator, sau chiar de Owner.

Acesta va dispărea când contul va fi setat din nou ca fiind Activ.

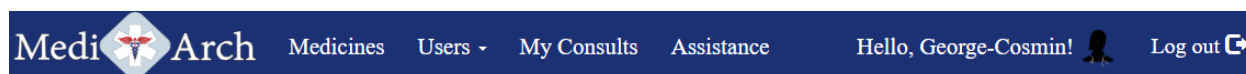
Odată creat contul, se poate naviga prin aplicație. Meniul de navigare va arăta astfel:



**Fig. 37** Meniul pentru utilizatorii speciali (Owner și Moderatori)



**Fig. 38** Meniul pentru Pacienți



**Fig. 39** Meniul pentru Doctori

După cum se observă și în imaginile de mai sus (**Fig. 37**, **Fig. 38** și **Fig. 39**), meniul diferă foarte puțin, deoarece am încercat să menținem un echilibru între aceste roluri. De exemplu, a 3-a opțiune (Lista de Consultații) va fi foarte puțin diferită în sensul operațiilor accesibile și a informațiilor afișate pe paginile respective. De asemenea, ultima opțiune accesibilă ca și utilizator special,

denumită *Maintenance*, are inclusă și opțiunea de Asistență (*Assistance*). Voi explica fiecare opțiune în parte când voi ajunge la acestea.

Prima opțiune pusă la dispoziție de aplicație, este vizualizarea unei liste de medicamente, disponibilă la secțiunea „*Medicines*”. Aici apare o paginare deoarece este posibil să fie destul de multe astfel de entități, și pentru a păstra un aspect aerisit, am decis să le grupez câte 5 pe o pagină. Desigur, dacă știi ce vrei să cauți, poți folosi opțiunea de căutare, pe această pagină fiind disponibilă (Fig. 40).



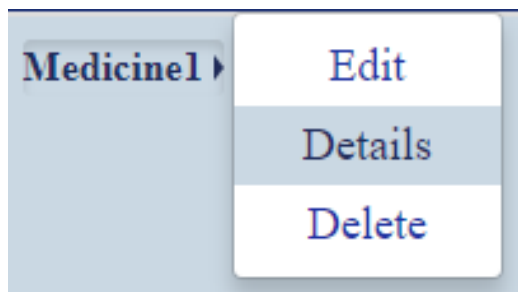
Name	Prospect	Additional Files:
Acc-200 - comprimate efervescente ▶	Prospect Acc-200 - comprimate efervescente Ce este ACC-200 comprimate efervescente si pentru ce se utilizeaza Indicatii ...	
Acetazalamida ▶	Prospect Acetazalamida, comprimate Ce este Acetazalamida si pentru ce se utilizeaza Indicatii -glaucom; -epilepsie (ca ...	
Acid acetilsalicilic, comprimate ▶	Prospect Acid acetilsalicilic, comprimate Ce este Acid acetilsalicilic si pentru ce se utilizeaza Indicatii Acest medicam ...	
Actovegin, fiole ▶	Prospect Actovegin, fiole Ce este Actovegin si pentru ce se utilizeaza Indicatii  Profilaxia si tratamentul tulburarilor ...	
Aerius ▶	Prospect Aerius Ce este Aerius si pentru ce se utilizeaza Indicatii Substanta activa din Aerius, desloratadina, este un a ...	

<< < 1 2 3 ... > >>

Name	Prospect
Acc-200 - comprimate efervescente	Prospect Acc-200 - comprimate efervescente Ce este ACC-200 comprimate efervescente si pentru ce se utilizeaza Indicatii In toate afectiunile respiratorii insotite de secretii abundente ale mucoaselor. In pneumologie: In forme acute si cronice al ...

Fig. 40 Lista de medicamente, împreună cu rezultatul unei căutări.

În imaginea **Fig. 40**, tabelul cu primul rând albastru este cel care conține medicamentele, iar cel cu primul rând roșu reprezintă rezultatele căutării. Dacă se va apăsa pe numele oricărui medicament căutat, va avea loc o redirecționare către pagina cu detaliile despre medicamentul ales.



**Fig. 41** Opțiuni disponibile

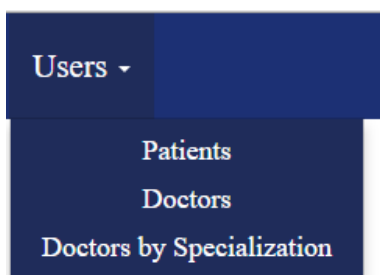
În schimb, dacă se va apăsa pe numele medicamentului din tabela cu primul rând albastru, vor apărea opțiunile disponibile. Pentru utilizatorii speciali, sunt cele din **Fig. 41**, iar pentru Pacienți și Doctori, va fi disponibilă doar opțiunea de detalii.

De pe această pagină, ca Ownerul sau Moderator, dacă se apasă pe iconița de sus din **Fig. 40**, va avea loc o redirecționare pe o pagină de unde se pot adăuga medicamente noi. Un astfel de comportament se găsește la majoritatea iconițelor care apar în aplicație. Tot de aici se pot downloada și fișierele adiționale, apăsând pe acestea.

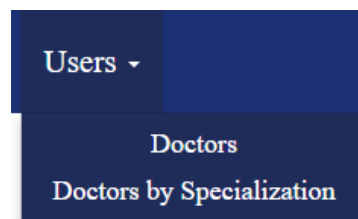
Următoarea secțiune este „*Users*”, unde vor apărea diverse opțiuni, ordonate convenabil după zonele de interes, pentru fiecare clasă de utilizator în parte (**Fig. 42**, **Fig. 43** și **Fig. 44**).



**Fig. 42** Utilizatori speciali



**Fig. 43** Doctori



**Fig. 44** Pacienți

În mare:

- „*All Users*” va returna o listă cu toți utilizatorii, aici fiind incluși Ownerul, Moderatorii, Doctorii și Pacienții;
- „*Doctors*” va returna lista cu toți Doctorii înregistrați;
- „*Patients*” va returna lista cu toți Pacienții care folosesc aplicația;

În paginile ce urmează voi detalia toate aceste opțiuni.

În secțiunea „All Users”, utilizatorii speciali vor putea să vadă lista cu toți utilizatorii, rolul lor fiind vizibil, alături de alte date (Fig. 45).







Role	Status	Email	Name	Age	Phone Number	Title	Cabinet Address
OWNER	Active	Owner@gmail.com ▶	 George-Cosmin Morosanu	21	0750000000		
MODERATOR	Active	Moderator1@gmail.com ▶	 Ben Stark	21	0750000001		
MODERATOR	Active	Moderator2@gmail.com ▶	 Barny Cliff	21	0750000002		
MODERATOR	Active	Moderator3@gmail.com ▶	 Joe Bill	21	0750000003		
MEDIC	Active	Medic1@gmail.com ▶	 Steve Hans	37	0751000000	Cardiologist	str. Decebal, Bl 374, Iasi

< 1 2 3 4 >

Fig. 45 Toți utilizatorii

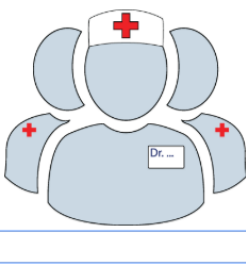
Opțiunea „Patients” va pune la dispoziție lista cu toți pacienții aplicației (Fig. 46). Aceasta va fi vizibilă doar Doctorilor și utilizatorilor speciali, nefiind valabilă Pacienților din cauza confidențialității datelor.






Email	Name	Age	Phone Number
georgocosminmorosanu@... ▶	 Morosanu George-Cosmin	21	0751871000
Pacient1@gmail.com ▶	 Phillips Seth	21	0752000000
Pacient10@gmail.com ▶	 O'Neil Steve	22	0752000009
Pacient2@gmail.com ▶	 Clark Ryan	22	0752000001
Pacient3@gmail.com ▶	 Croft Emma	22	0752000002

< 1 2 3 >

Fig. 46 Users

Opțiunea de „Doctors” va fi asemănătoare cu „All Users”, prezentată mai sus, dar voi afișa doar datele importante pentru Pacienți (Fig. 47). Cât despre opțiunea „Patients”, este la fel ca la „Doctors”, dar fără cele 2 câmpuri specific doctorilor, aceasta fiind vizibilă doar doctorilor.



Email	Name	Age	Phone Number	Title	Cabinet Address
Medic1@gmail.com ▶	 Hans Steve	37	Born in 12/18/1980 000	Cardiologist	str. Decebal, B1 374, Iasi
Medic2@gmail.com ▶	 Clark Larry	58	0751000001	Cardiologist	str. Decebal, B1 373, Iasi
Medic3@gmail.com ▶	 Brown Fred	31	0751000002	Anesthesiologist	str. Decebal, B1 372, Vaslui
Medic4@gmail.com ▶	 O'Brien Lara	28	0751000003	Dermatologist	str. Decebal, B1 371, Piatra Neamt
Medic5@gmail.com ▶	 O'Brien James	32	0751000004	Internist	str. Decebal, B1 370, Vaslui

Name	Phone Number	Title	Cabinet Address
Lara O'Brien	0751000003	Dermatologist	str. Decebal, B1 371, Piatra Neamt
James O'Brien	0751000004	Internist	str. Decebal, B1 370, Vaslui

Fig. 47 Toți doctorii

Diferența dintre aceste 3 opțiuni, pe lângă clasele de utilizatori afișate, va fi vizibilă la funcția de căutare, deoarece la „Doctors” se vor căuta doar doctorii corespunzători, la „Patients” vor fi afișați Pacienții, iar la „All Users” vor fi afișați atât Doctorii, cât și Pacienții. În ultima căutare nu am afișat Ownerul sau Moderatorii deoarece aceștia apar la începutul tabelului din „All Users”.

Cât despre opțiunea de „Doctors by Specialization”, aici va avea un aspect diferit față de celelalte 3 opțiuni care implică utilizatorii. Astfel, vor apărea pe 2 coloane toate specializările, urmat de numărul de Doctori existenți pe fiecare specializare, iar dacă se va accesa o specializare, va apărea un Pop-up (Modal) cu specializarea aleasă și cu doctorii care se include aici (Fig. 48 și Fig. 49).



Fig. 48 Doctorii după specializare - normal

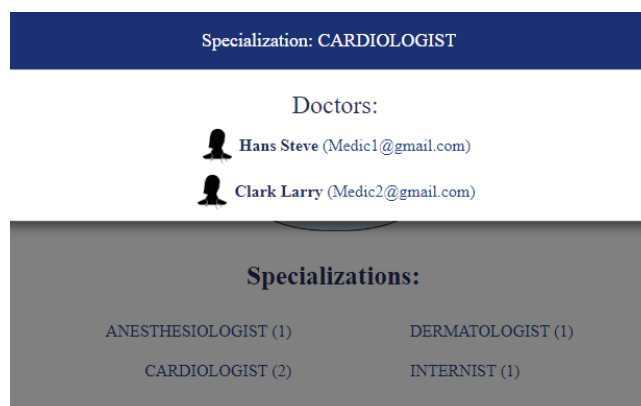



Fig. 49 Doctorii după specializare – selectată o specializare

Doctorii pot găsi opțiunea de a adăuga o consultație fie direct din tabela „Patients”, fie intrând pe pagina de detalii a oricărui Pacient. Această opțiune este vizibilă doar doctorilor.

Secțiunea de **Consultații** va avea 3 nume, diferite pentru fiecare clasă de utilizator: *Consults* (pentru utilizatorii speciali), *My Results* (pentru Pacienți) și *My Consults* (pentru Doctori).



Consult's Date	Medic	Patient	Result	Prescription	Files
6/22/2018 7:25:06 PM ▶	O'Brien James	Morosanu George-Cosmin	Heart Problems	Medicine2(2/day)	
6/22/2018 7:24:30 PM ▶	Brown Fred	Clooney George	Common Cold	Medicine3(3/day) Medicine7(3/day)	
6/22/2018 2:46:07 PM ▶	Clark Larry	Onion Brianne	Ear Infection	Medicine22(2/day) Medicine1(3/day)	
6/21/2018 8:19:09 PM ▶	Brown Fred	Morosanu George-Cosmin	Spine Problems	Medicine5(3/day) Medicine9(1/day)	<a href="#">Consultatie.docx</a> <a href="#">X-Ray 1.PNG</a> <a href="#">X-Ray 2.png</a>
6/21/2018 8:18:20 PM ▶	O'Brien Lara	Clooney George	Fractured Hand	Medicine21(1/day)	<a href="#">X-Ray 1.PNG</a> <a href="#">X-Ray 2.png</a>

< 1 2 3 >

Fig. 50 Toate Consultațiile

Fig. 50 reprezintă tabelul pus la dispoziție Ownerului și Moderatorilor. Tabelele care vor fi afișate Pacienților (în *My Results*) și Doctorilor (în *My Consults*) sunt asemănătoare, doar că acolo se vor găsi doar consultațiile care îi privesc pe ei. Și desigur, coloana cu numele lor va lipsi.



De pe pagina de Consultații se pot downloada fișierele adiționale și se poate naviga spre celelalte pagini prin intermediul link-urilor adăugate. De exemplu, spre medicamentele de la coloana *Prescription*, sau către profilul unui utilizator implicat.

La secțiunea de *Asistență* (*Assistance* - Fig. 51) vor apărea întrebări puse de orice utilizator, la care poate răspunde oricine are un cont creat. Principiul este același ca și la lista cu *doctors după Specializări* ( Fig. 48 și Fig. 49), adică dacă se va apăsa pe întrebare, va apărea o fereastră cu toate răspunsurile postate la întrebarea respectivă până în momentul respectiv.

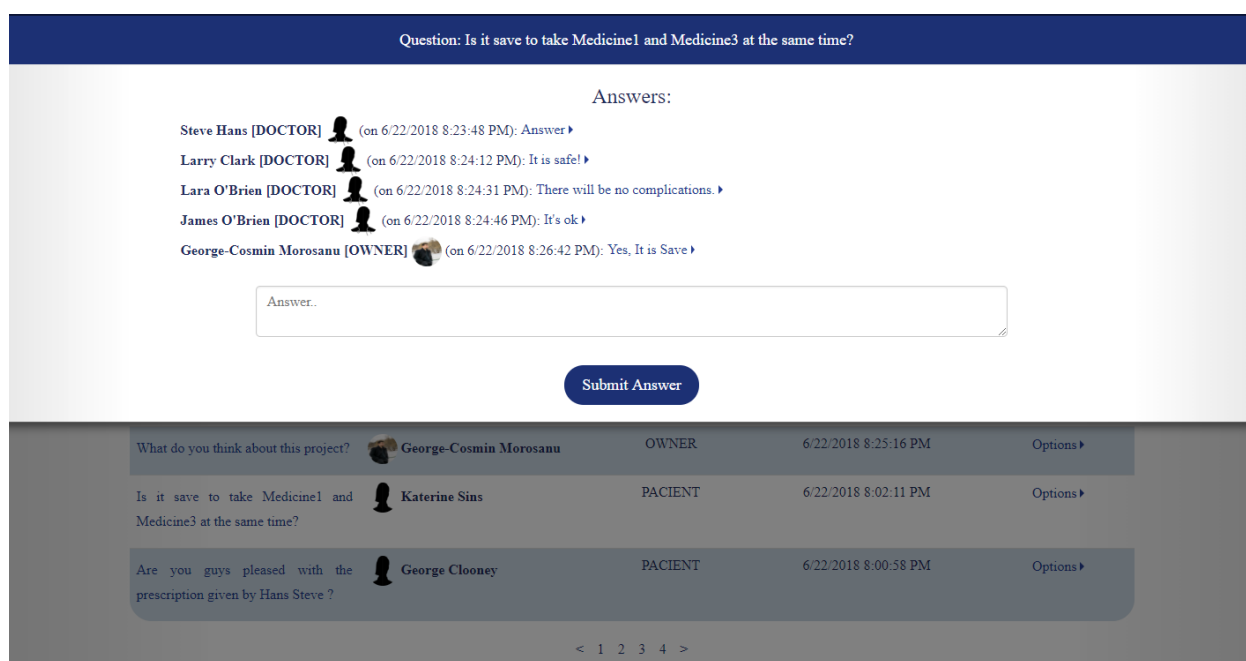


Fig. 51 Secțiunea de Asistență

La utilizatorii speciali, această opțiune se găsește în secțiunea „*Maintenance*”, împreună cu opțiunea „*Data Records*”, deoarece această clasă de utilizatori vor fi cei care se vor ocupa de întreținerea platformei, având grijă să răspundă la întrebările la care are sens să răspundă, să aibă grijă de lista de Medicamente pusă la dispoziție de aplicație, și să urmărească tot ce se întâmplă.

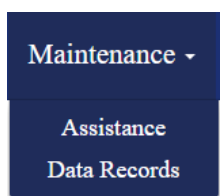



Fig. 52 Maintenance

*Dara Records* (Fig. 52) reprezintă zona unde se va ține evidența înregistrărilor entităților relevante. Astfel, aici se vor afla tabele cu:

- Utilizatorii înregistrați;
- Întrebări postate;
- Consultații adăugate;
- Răspunsuri primite;

Pentru a păstra aerisită și această secțiune, aici vor fi afișate doar datele care sunt noi, de până la 2 săptămâni, presupunând că cele mai vechi, care nu mai apar aici, au fost deja validate.




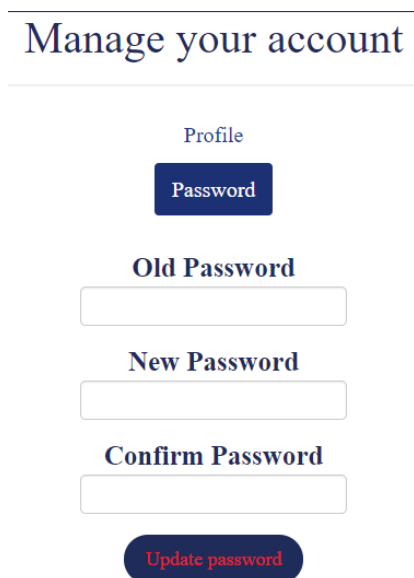
Role	Name	Age	Phone Number	Title	Cabinet Address	Details
PATIENT	 George-Cosmin Morosanu	21	0751871000			Details
PATIENT	 George Clooney	21	0752000000			Details
DOCTOR	 Larry Clark	58	0751000001	Cardiologist	str. Decebal, Bl 373, Iasi	Details
DOCTOR	 Steve Hans	37	0751000000	Cardiologist	str. Decebal, Bl 374, Iasi	Details

Fig. 53 Userii aflați în Data Records

Fiecare opțiune prezentă în partea de sus din Fig. 53 va duce la afișarea tabelor cu entitățile respective și câmpurile specifice fiecăreia.

Ultima secțiune este reprezentată de pagina de **Profil** a unui utilizator.



Manage your account

Profile

Password

Old Password

New Password

Confirm Password

Update password

Fig. 54 Update la parolă

Aici utilizatorul își poate schimba

- fie Parola (Fig. 54) – secțiune separată,
- fie poza de profil, numărul de telefon sau adresa cabinetului (dacă este Doctor) – direct din pagina sa de profil, modificând datele dorite și apăsând pe butonul *Save Changes* din josul paginii (Fig. 55).

**Role:** OWNER

**Current Profile Picture:**



**Choose a new Profile Picture**

**Username (Email):**  
georgocosminmorosanu@gmail.com

**Full Name:**  
George-Cosmin Morosanu

**Phone Number:**  
0750000000

**Birth Date:**  
09/17/1996 12:00 AM

[Save changes](#)

## Account - Details



**Role:** Owner

**Full name:** George-Cosmin Morosanu

**Email:** georgocosminmorosanu@gmail.com

**Age:** 21

**BirthDate:** 9/17/1996 12:00:00 AM

**PhoneNumber:** 0750000000

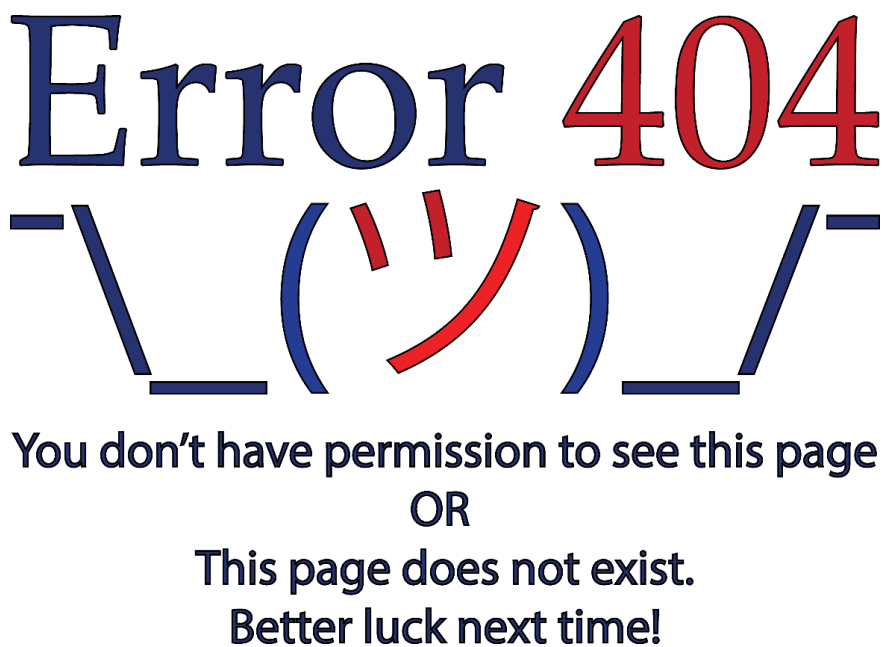
**Status:** Active

[Edit](#) | [Set Active](#) | [Set Inactive](#) | [Delete](#) | [Go back](#)

**Fig. 56** Detaliile vizibile, împreună cu opțiunile utilizatorilor speciali

**Fig. 55** Pagina de profil a utilizatorului

Dacă la un moment dat, pe parcursul navigării prin aplicație, se va accesa o pagină inexistentă, atunci pe ecran va apărea un mesaj de eroare (Fig. 57).



**Fig. 57** Pagina de eroare

## 4. Concluzii

MediArch este o aplicație web conturată pe domeniul medical care va ține la un loc sigur toate datele din istoricul medical al unei persoane.

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 dezvoltarea 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 la început, dar am ajuns să mă 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 funcționalității a fost cea gândită la început, dar, desigur, adăugări s-au mai ivit și pe parcursul finisării aplicației, idei apărând în timp ce navigam prin aceasta pentru a testa dacă totul este cum ar trebui să fie.

Chiar am învățat lucruri noi pe parcursul dezvoltării acestui proiect, și mi-am întărit cunoștințele asupra acestei tehnologii. Pe lângă lucruri tehnice am învățat să notez toate ideile care apar cu privire la proiectul pe care îl dezvolt, pentru că în orice idee poate genera o funcționalitate neașteptat de folositoare.

Îmbunătățiri și idei pentru viitor:

- Logare cu ajutorul Facebook/Google/altor platforme media. Dar această funcționalitate trebuie să fie valabilă doar pentru Pacienți, deoarece ca și Doctor se cer acele 2 câmpuri adiționale (*Titlu* și *Adresa\_Cabinetului*) care nu se găsesc pe asemenea platforme;
- Folosirea Google Maps pentru a afla locația unui utilizator și pentru a-i sugera acestuia o listă cu doctori aflați în apropiere, într-o rază de  $n$  kilometri;
- Mod de a verifica corectitudinea prospectelor medicamentelor aflate în baza de date;
- Folosirea unui criptosistem mult mai bine optimizat – pentru securizarea tuturor datelor senzitive;
- Mod de verificare a autenticității unui Doctor;

# Bibliografie

1. Structura Bazei de Date – Vertabelo: <https://www.vertabelo.com/>
2. Criptosistem: <https://www.c-sharpcorner.com/UploadFile/f8fa6c/data-encryption-and-decryption-in-C-Sharp/>
3. Tipuri de Doctori: <https://www.webmd.com/health-insurance/insurance-doctor-types#1>
4. GDPR: <http://www.privacyone.ro/dpo/gdpr.html>
5. Informații despre tehnologiile folosite:
  - C# - <https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework>
  - .NET - <https://www.microsoft.com/net/learn/what-is-dotnet>
  - ASP.NET - <https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx>
  - MVC - <https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-2.1>
  - Arhitectura Onion - <https://www.c-sharpcorner.com/article/onion-architecture-in-asp-net-core-mvc/>
  - SQL - [https://www.w3schools.com/sql/sql\\_intro.asp](https://www.w3schools.com/sql/sql_intro.asp)
  - Version Control - <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>
  - Github - <https://github.com/>
  - GitKraken - <https://www.gitkraken.com/>
  - HTML - [https://www.w3schools.com/html/html\\_intro.asp](https://www.w3schools.com/html/html_intro.asp)
  - CSS - [https://www.w3schools.com/css/css\\_intro.asp](https://www.w3schools.com/css/css_intro.asp)
  - Bootstrap - <https://getbootstrap.com/>
  - Adobe Illustrator - <https://www.adobe.com/products/illustrator.html>

## **Anexa 1 (Tehnologii Utilizate)**

### **ASP.NET**

**C#** este un limbaj de programare orientat-obiect, proiectat de către Microsoft care are ca scop combinarea puterii de calcul a limbajului C++ cu ușurința de programare a Visual Basic. C# se bazează pe C++ și conține caracteristici similare cu cele din Java. Acesta permite proiectarea de aplicații complexe și sigure, care vor rula pe Framework-ul .NET.

**.NET** este o multi-platformă, concepută cu scopul de a construi diverse tipuri de aplicații, cum ar fi Desktop, Web sau Mobile. Aici se vor putea rula în mod nativ limbajele C#, F# și Visual Basic.

**ASP.NET** (ASP este prescurtarea de la „Active Server Pages”) este partea din Framework-ul .NET care se ocupă cu dezvoltarea aplicațiilor Web

**MVC** (*Model-View-Controller*) este un model arhitectural care împarte aplicația în 3 componente: Modele, Viewuri și Controllere. Astfel, când un utilizator intră pe o aplicație prin intermediul unui browser, acesta va trimite către Controller o cerere HTTP (HTTP Request); Controllerul va lua datele necesare din Model, va construi View-ul necesar, și îl va trimite înapoi pe browser sub formă de răspuns (HTTP Reponse).

### **SQL (SQL Server Express)**

**SQL** este prescurtarea de la „Structured Query Language”. Reprezintă o punte de comunicare cu datele aflate în baza de date.

Am ales varianta de SQL implementată de cei de la Microsoft (SQL Server Express) pentru a îmi fi mai ușor, eu fiind deja familiarizat cu această versiune deoarece am folosit-o în primul semestru al anului 3, la proiectul pentru materia „Introducere în .Net”. Dar pe viitor se poate trece foarte ușor la o altă versiune de bază de date.

## Version Control - GitHub (GitKraken)

**Version Control** este un sistem care face managementul fișierelor salvate. Prin intermediul acestuia se poate ajunge la orice stagiu anterior al fișierelor.

Datele acestui proiect au fost salvate pe GitHub, prin intermediul aplicației *GitKraken* (Fig. 58).

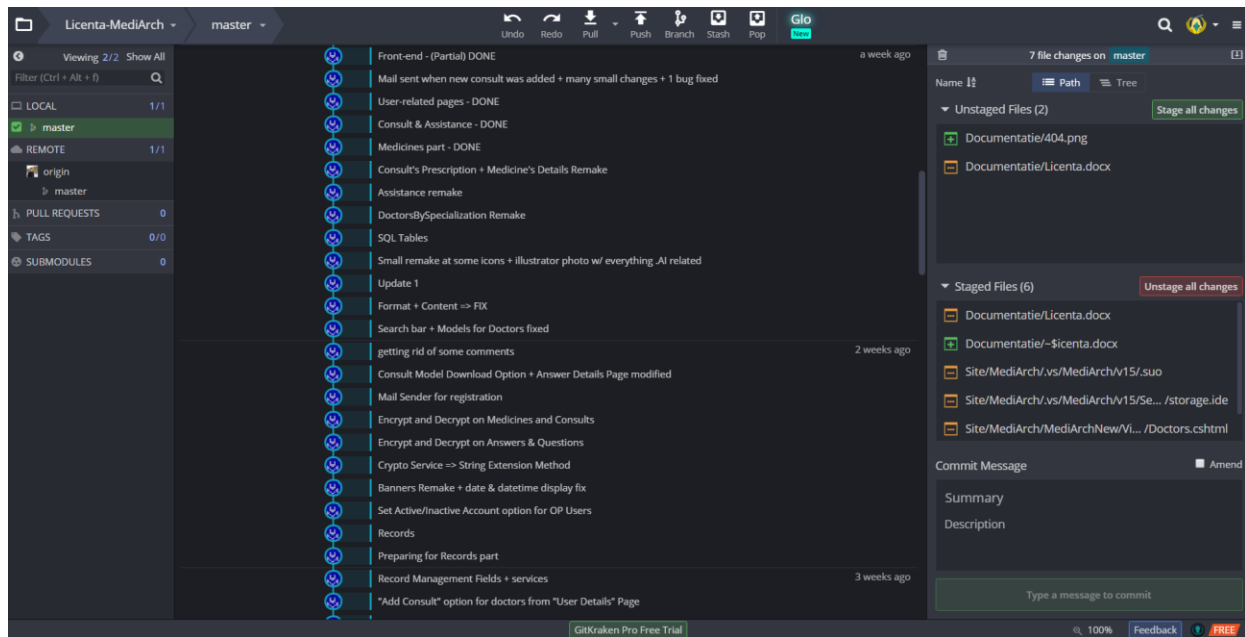


Fig. 58 GitKraken

## HTML, CSS, JavaScript și Bootstrap

**HTML** vine de la „*Hyper Text Markup Language*”, și este limbajul care definește structura unei pagini web, fiecare element reprezentând un tag cu anumite proprietăți.

**CSS** vine de la „*Cascading Style Sheets*”, și este un limbaj ce descrie stilul elementelor din HTML.

**JavaScript** este limbajul ce oferă dinamicitate elementelor HTML.

**Bootstrap** este un *framework* (preinstalat în aplicațiile web de tip MVC, găsit în partea de librării) care are elemente HTML, CSS și JavaScript prestabilite.

În varianta finală a proiectului vor exista multe clase (din Bootstrap) modificate și multe adăugate.

## Adobe Illustrator

**Adobe Illustrator** este un program de editare, proiectat de Adobe Systems. Spre deosebire de *Adobe Photoshop* (care este mai cunoscut), Illustrator lucrează în format vectorial, calitatea rezultată fiind superioară, mai ales la detalii.

Astfel, dacă vreodată se va dori obținerea unui detaliu, dar în format mult mai mare, redimensionarea nu va afecta calitatea detaliului ales. În Photoshop (care lucrează pe pixeli), dacă se dorea acest lucru, calitatea rezultată ar fi fost foarte slabă.

Majoritatea elementelor folosite se găsesc în pozele următoare:

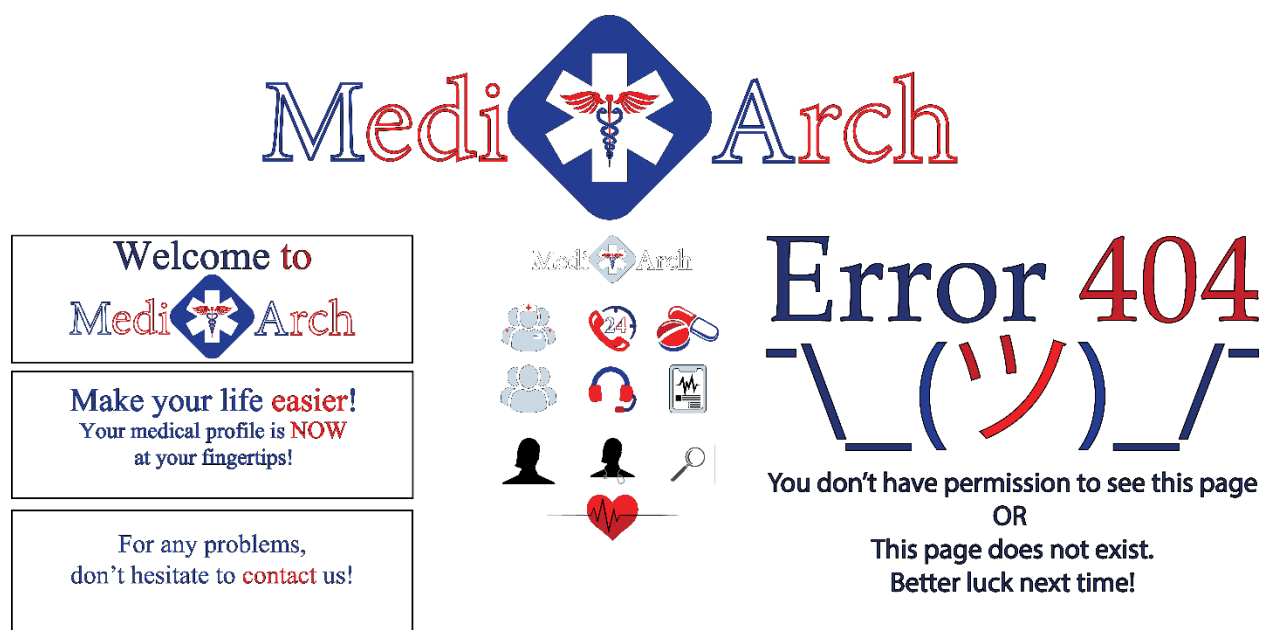


Fig. 59 Elemente realizate cu Adobe Illustrator