



**UNIVERSITATEA DIN
BUCUREȘTI**



**FACULTATEA DE MATEMATICĂ ȘI
INFORMATICĂ**

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

**APLICATIE WEB PENTRU GESTIUNEA UNUI
CABINET MEDICAL**

Absolvent

Chiricu Ana-Bianca

Coordonator științific

Prof. Dr. Mincu Radu-Ștefan

București, iunie 2022

Rezumat

În ultimii ani, procesul de digitalizare a informației a căpătat amploare făcându-și simțită prezența în aproape toate domeniile. Printre aceste domenii, în mod inevitabil, se numără și cel medical, unde, în majoritatea cazurilor, datele sunt păstrate în format fizic, în dosare sau arhive, fără a exista o modalitate mai simplă prin care o persoană să aibă acces la informațiile necesare.

Scopul acestei lucrări este de a dezvolta o aplicație web care să permită gestiunea unui lanț de cabinete medicale într-un mod prietenos atât pentru doctori, cât și pentru utilizatori.

Aplicația a fost realizată în *C#* cu ajutorul framework-urilor *ASP.NET MVC* și *Bootstrap5*. Primul este un framework open-source aparținând de Microsoft care permite dezvoltarea de aplicații web folosind structura model-view-controller.

Bootstrap5 reprezintă cea mai nouă versiune a celui mai popular framework pentru *HTML*, *CSS* și *JavaScript*. Acesta este folosit pentru design-ul paginilor web și conține o varietate de componente predefinite pentru a crea pagini web dinamice.

Autentificarea și autorizarea sunt asigurate de *Identity Framework*, iar persistența datelor de către *Entity Framework*. Un cont poate fi creat fie prin metoda clasică, folosind un cont de email, fie prin utilizarea unui cont de social media, mai exact de *Facebook*. În momentul înregistrării, fiecare utilizator primește rol de utilizator, care poate fi schimbat de către administrator în Doctor sau în Administrator.

Un utilizator neautentificat poate vedea pagina de prezentare, lista de locații, de departamente și de investigații posibile, dar nu își poate urmări evoluția și nu poate solicita o programare.

Fiecare doctor are propriul cont unde poate vedea istoricul programărilor și programările viitoare, pe care le poate anula. De asemenea, el are un profil vizibil pentru orice utilizator, pe care îl poate edita.

Utilizatorul cu rol de administrator este cel care gestionează tot lanțul de cabinete și toți doctorii. Acesta se ocupa de adăugarea de locații noi, de editarea celor existente, și are posibilitatea de a șterge locațiile în cazul în care nu mai sunt valabile. De asemenea, poate crea, adăuga și edita departamente, pe care le poate asocia cu locațiile existente. În momentul adăugării unui nou doctor, administratorul trebuie să modifice rolul contului

creat, apoi să îi asocieze o locație și un departament.

Utilizatorul autentificat poate programa o vizita la un anumit doctor, își poate vedea istoricul programărilor și poate ține un jurnal propriu legat de evoluția sa.

Aplicația dezvoltată poate oferi un real ajutor, constituind o sursă de informații pentru clienți sau viitori clienți, pentru doctori care pot ține mai ușor evidența pacienților, dar și pentru administratori care pot ține evidența pentru fiecare cabinet.

Abstract

Lately, the digitalisation process of the information has boosted, being present in almost all the domains and fields of activity. Obviously among these, there is the medical one, where most of the data are still kept in physical format, paper based, in files and archives, without existing a simpler way through which a person can access the necessary information.

This paper aims to develop a web application that allows the management of a medical offices' chain in a friendly way, not only for the doctors but also for their customers. The application was made in *C#*, with the help of *ASP.NET MVC* framework and *Bootstrap5*. The first one is an open-source framework, belonging to Microsoft, that allows the development of web applications using the structure model-view-controller. *Bootstrap5* represents the latest version of the most popular framework for *HTML*, *CSS* and *JavaScript*. This is used for web pages' design and it contains a variety of predefined components which are used to create dynamic web pages.

The authentication and authorisation are assured by *Identity Framework*, and the data persistence by the *Entity Framework*. An account may be created either by the classical method, using an email account, or using a social media account, more precisely a Facebook one. When firstly enrolled, each customer gets a user role, that can be changed by the administrator in Doctor or in Administrator.

An unjustified user may visualise the presentation/home page, the lists of connected locations, departments and possible investigations, but he is not allowed to follow his own evolution or to settle an appointment. Each doctor has his own account where he can see the history of his appointments and the following ones that can be cancelled. He has also got a profile that he can edit and which is visible for any user.

The user that has the Administrator role is the one that administer/ manages the whole chain of medical offices and all the doctors. He is the one who adds new locations, edits the

existing ones and is able to delete locations that are no longer valid. He can also add and edit departments, that he can associate with the existing locations. When a new doctor is added, the administrator can modify the role of the created account, and then he can associate a new location and department.

The authenticated user can settle an appointment or a visit to a certain doctor, he is able to view his schedules' history and he can keep his own diary regarding his evolution.

The developed application may be of a real help, being a source of information for both present and future customers, for the doctors who can easily keep the record of their customers, but also for the administrators who can keep the record of each medical office.

Cuprins

Capitolul 1. Introducere	7
1.1 Încadrarea în context	7
1.2. Scopul și motivarea alegerii temei	8
1.3. Aplicații similare	8
1.4. Diferențe și elemente de noutate	8
Capitolul 2. Preliminarii	10
Capitolul 3. Tehnologii folosite	11
3.1 ASP.NET	11
3.1.1 Preliminarii.....	11
3.1.2 Arhitectura.....	11
3.1.3 Structura fișierelor.....	12
3.1.4 Sistemul de rutare.....	13
3.1.5 Modelul	13
3.1.6 Vizualizarea.....	13
3.1.7 Controller-ul	14
3.2 Identity Framework	16
3.2.1 Preliminarii.....	16
3.2.2. Autentificare.....	16
3.2.3. Autorizare.....	16
3.2.4. Autentificare folosind platforme sociale	16
3.3 Entity Framework.....	17
3.3.1 Preliminarii.....	17
3.3.2 Crearea tabelelor	17
3.3.3 Stocarea datelor	17
3.3.4 LINQ Query	17
3.4 Bootstrap 5	18
3.4.1 Preliminarii.....	18
3.4.2 Meniul de navigare.....	18
3.4.3 Card	18
3.4.4 Formularul.....	18
3.4.5 Jumbotron.....	18
3.4.6 Tabelul.....	19
3.4.7 Butoanele.....	19
3.4.8 Listele	19
3.4.9 Bara de căutare	19

3.4.10 Câmpurile pentru dată și ora	20
Capitolul 4. Descrierea aplicației	21
4.1 Preliminarii	21
4.2 Stocarea datelor	21
4.3 Autentificarea	23
4.4. Rolurile utilizatorilor	24
4.5 Administratorul	25
4.5.1 Editarea utilizatorilor	26
4.5.2 Adăugarea unui medic.....	28
4.5.3 Asociere Locație-Departament.....	29
4.5.4 Locații	29
4.5.5 Departamente	31
4.5.6 Investigații.....	32
4.6. Utilizatorul.....	33
4.6.1. Locații	33
4.6.2 Departamente	34
4.6.3 Doctori.....	34
4.6.4 Programări.....	35
4.6.5 Jurnalul.....	36
4.7 Doctorul.....	36
4.7.1 Profilul.....	37
4.7.2 Cabinetul	37
4.7.3 Programările.....	37
Capitolul 5. Concluzie	38
Bibliografie.....	40

Capitolul 1. Introducere

1.1 Încadrarea în context

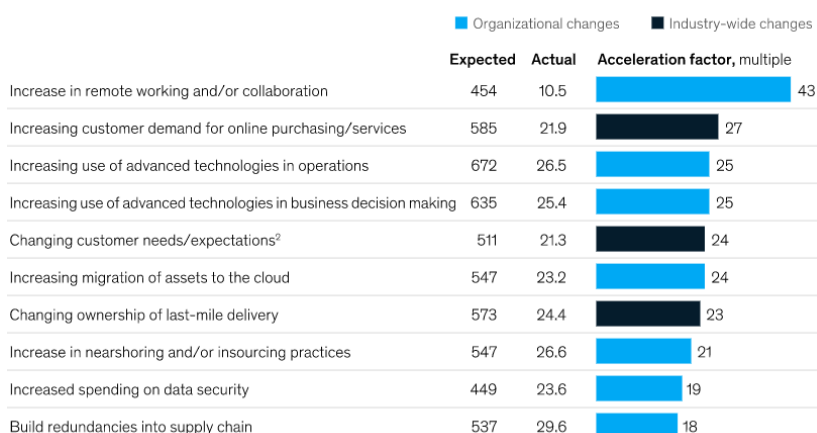
Ultima perioada a venit cu o varietate de schimbări în toate planurile și, inevitabil, cu nevoia de adaptare la nou. Astfel, a apărut necesitatea digitalizării și a stocării datelor într-un mod mai sigur și mai ușor de utilizat.

Odată cu izbucnirea pandemiei și necesitatea distanțării sociale, fiecare persoană a căutat o modalitate prin care să își depășească limitele și să își continue viața de zi cu zi, într-un mod cât mai normal. În acest sens, industria digitală a avut parte de o adevărată creștere și dezvoltare, companiile fiind nevoite să investească în platforme pentru lucrat de acasă prin telemuncă, școlile adaptându-se la modul de predare prin intermediul platformelor de conferințe și magazinele mutându-se în mediul online. În fiecare caz, s-a căutat cea mai potrivită metodă, sau chiar a fost dezvoltată una care să se muleze pe nevoile companiei și ale societății deopotrivă.

Deși părea greu de crezut, oamenii s-au adaptat mult mai rapid decât era de așteptat înainte de criză. În figura 1 este prezentată o comparație între perioada de adaptare la care se așteptau angajatorii înainte de pandemie în cazul unor schimbări drastice și perioada actuală în care persoanele s-au adaptat în momentul în care au fost puși față în față cu acestea.

Executives say their companies responded to a range of COVID-19-related changes much more quickly than they thought possible before the crisis.

Time required to respond to or implement changes,¹ expected vs actual, number of days



1.1. Diferența (nr de zile) între perioada preconizată în cazul adaptării și perioada reală

1.2. Scopul și motivarea alegerii temei

Industria medicală a fost asaltată și chiar depășită de situație, oamenii fiind speriați și dorind să primească ajutor și informații cât mai exacte într-un timp cât mai scurt. Acest fapt a determinat nevoia de platforme online care să scurteze timpul de așteptare, care să ofere informații mai clare, mai exacte și să permită urmărirea evoluției și istoricului unei persoane, fără a fi necesară prezența fizică de fiecare dată.

Astfel, s-au dezvoltat numeroase soluții pentru a rezolva aceste probleme, companiile medicale venind în ajutor cu consultații telefonice, cu liste de simptome și tratamente, majoritatea axându-se pe problema evidentă a momentului: pandemia de COVID-19.

Aplicația web prezentată în această lucrare a fost dezvoltată pentru a veni în ajutorul oricărei persoane care are nevoie de o consultație, care dorește să se informeze, să își păstreze evidența tuturor programărilor și a consultațiilor medicale efectuate și, opțional, să țină un jurnal propriu bazat pe starea din ziua respectivă. Astfel, o persoană poate avea în același loc informații, poate vedea cabinetele și departamentele disponibile, își poate rezerva o programare și poate ține evidența lor.

1.3. Aplicații similare

Odată cu revenirea la normalitate, multe dintre acestea și-au pierdut scopul (un exemplu fiind aplicația de informare cu privire la centrele de vaccinare). Pe de altă parte, există și unele care încă sunt de foarte mare ajutor, cum ar fi *Ada*, o aplicație dezvoltată de medici care, bazat pe simptomele introduse, cu ajutorul inteligenței artificiale, oferă un diagnostic orientativ și sfaturi pentru ce este de făcut în continuare. Pe lângă aceasta, există și aplicații similare cu cea dezvoltată și prezentată în această lucrare, care permit clienților să vadă și să solicite programări în funcție de specializarea căutată, de locația care se potrivește cel mai bine sau de medicul dorit, printre care se numără cea de la *MedLife* sau cea de la *Regina Maria*.

1.4. Diferențe și elemente de noutate

La fel ca și aplicațiile prezentate mai sus, aceasta își propune să ofere o varietate de domenii, de locații, de investigații și de medici la care un utilizator să poată solicita o programare. Fiecare doctor are un profil care poate fi văzut de oricine intră în aplicație, iar

un utilizator are un profil privat care este vizibil doar pentru el și pentru administrator. Prețurile sunt vizibile pentru oricine, la fel și lista de cabinete și departamente.

Diferența ar fi posibilitatea adăugării unui jurnal în care pacientul să menționeze orice consideră important cu privire la starea lui de sănătate. Acest jurnal este privat și are scopul de a ajuta pacientul să țină evidența simptomelor, a stării de spirit și a evoluției sale într-un mod simplu, util și personalizat. Astfel, sunt respectate și principiile de confidențialitate, dar sunt reținute și datele importate ale utilizatorului pe care are posibilitatea să i le arate doctorului la următorul control, fără a îi fi încălcată intimitatea.

Capitolul 2. Preliminarii

Lucrarea prezentata se încadrează în domeniul *Dezvoltarea Aplicațiilor Web* și a fost realizată folosind *C#* alături de framework-urile *ASP.NET MVC* și *Bootstrap5*. Aceasta are ca temă dezvoltarea unei aplicații web pentru gestiunea unui cabinet medical.

O aplicație web este o aplicație instalată doar pe un server web care poate fi accesată utilizând un browser, de exemplu *Google Chrome* sau *Microsoft Edge*. Acest tip de aplicație oferă multe avantaje utilizatorilor, printre care se află următoarele: nu necesită actualizări realizate de utilizatori, adresa pe care o accesează fiind mereu aceeași. Chiar dacă dezvoltatorul produce o schimbare asupra platformei, aceasta se modifică prin simpla accesare, fără nevoia unei actualizări propriu zise așa cum este cazul aplicațiilor pentru desktop sau pentru mobil.

Aplicația se rulează într-un mod simplu și se adaptează la browser și la dimensiunile ecranului. Simpla accesare a link-ului specific oferă unei persoane posibilitatea de a utiliza platforma, fără necesitatea descărcării acesteia din magazine precum *GooglePlay* sau *MicrosoftStore*.

Cu toate acestea, aplicațiile web prezintă și câteva dezavantaje, iar cel mai mare dintre acestea este nevoia permanentă a unei conexiuni de internet. Deși în timpurile noastre internetul a devenit ceva obișnuit și indispensabil, încă nu ne putem baza înntotdeauna pe acesta. Există momente când conexiunea este pierdută, sau când legătura dintre browser și server nu se poate realiza, momente în care aplicațiile web nu pot fi folosite.

De asemenea, viteza de rulare a acestor aplicații depinde de conexiunea și de gazda pe care aplicația este accesată. Comparativ cu accesarea sa de pe un server local, viteza este mult mai scăzută în cazul accesării unei aplicații hostate.

Deși numărul de avantaje pare apropiat de cel al dezavantajelor, o aplicație de acest tip este în cele mai multe cazuri soluția ideală datorită scopului ei, datorită mentenanței ușoare și, mai ales, datorită accesibilității ei pentru oricine dispune de o conexiune la internet.

Capitolul 3. Tehnologii folosite

3.1 ASP.NET

3.1.1 Preliminarii

ASP.NET este un framework open-source creat de către *Microsoft* pentru a facilita dezvoltarea de aplicații web moderne și dinamice utilizând platforma *.NET*. Prima versiune de *.NET* a fost publicată în 2002, iar până la aceasta era folosit *ASP-ul clasic*. Spre deosebire de *.NET* care se poate folosi pentru o varietate de aplicații și servicii web, Windows sau de tip consolă, *ASP.net* este limitat la servicii și aplicații web.

3.1.2 Arhitectura

Arhitectura *MVC (Model-View-Controller)* este un tipar arhitectural prin care aplicația este separată în trei componente principale: *modelul*, *vizualizarea* și *controller-ul*. Fiecare componentă este construită pentru a se ocupa de un anumit aspect al dezvoltării. Pașii care sunt executați într-un proces sunt următorii:

- Clientul (browser-ul) trimite o cerere către aplicație;
- Cererea este primită de *Global.ascx* care decide ruta potrivită dintre cele configurate, apoi este chemat controller-ul responsabil de ruta cerută;
- Controller-ul procesează cererea primită folosind modelul, apoi apelează metoda necesară;
- Modelul obținut este parsat la vizualizare, care produce rezultatul final.

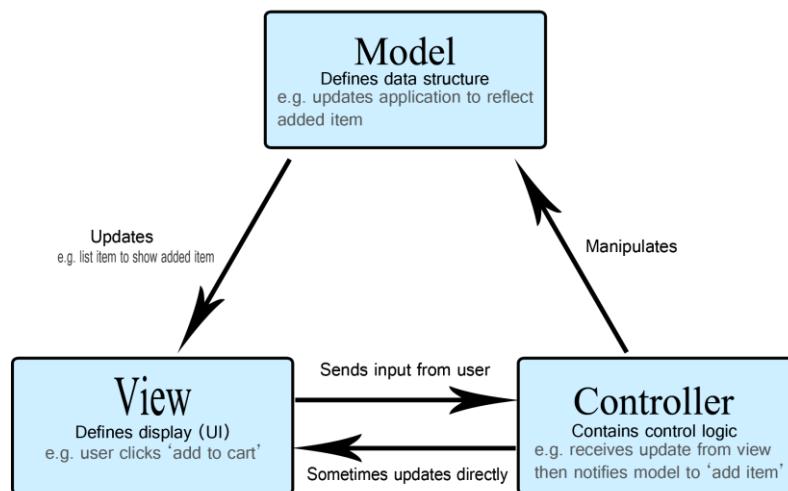


Figura 3.1 Arhitectura MVC

3.1.3 Structura fișierelor

Atunci când se creează un proiect nou, structura sa este implicit împărțită în mai multe fișiere, în funcție de întrebuințarea lor.

3.1.3.1. *App_Data*

În acest director sunt stocate fișierele relevante pentru datele aplicației, cum ar fi baza de date, fișierele de tip .mdf sau de tip .xml.

3.1.3.2. *App_Start*

Aici sunt prezente fișierele care se rulează în momentul în care se pornește aplicația și reprezintă fișierele necesare pentru configurarea inițială.

3.1.3.3. *Content*

În directorul Content se păstrează fișierele de tip css, imaginile și simbolurile aplicației.

3.1.3.4. *Controllers*

Dosarul *Controllers* stochează toate controllerele aplicației, iar numele fișierelor din acest dosar trebuie să aibă terminația *Controller.cs* pentru a fi utilizate.

3.1.3.5. *Fonts*

Aici se pot pune fonturi personalizate care se pot utiliza în scripturi css.

3.1.3.6. *Models*

Directorul *Models* conține toate fișierele cu clase de tip model, care sunt folosite pentru

a stoca și manipula datele aplicației.

3.1.3.7. Scripts

În *Scripts* se găsesc librării și fișiere pentru *JavaScript* și *Bootstrap*, care în *MVC5* sunt incluse în mod implicit.

3.1.3.8. Views

Dosarul *Views* este la rândul său împărțit în mai multe sub-dosare, grupate în funcție de numele controller-ului cu care fac legătura. Fiecare vizualizare reprezintă interfața cu utilizatorul, dar și răspunsul unei acțiuni din controller.

3.1.3.9. Packages.config

Fișierul *Packages.config* este gestionat de către *NuGet* și ține evidența pachetelor și versiunilor instalate.

3.1.3.10. Web.config

Fișierul *Web.config* conține configurările specifice aplicației.

3.1.4 Sistemul de rutare

Rutarea a fost introdusă în *ASP.NET MVC* pentru a elimina nevoia de a conecta fiecare *url* cu un fișier de tip *.aspx* specific, așa cum este în *ASP.NET Web Forms*. În acest sens, se pot defini rute necesare în fișierul *RouteConfig*, și poate fi creat un șablon care să fie folosit în mod implicit. Se pot configura mai multe rute cu nume diferite, dar trebuie adăugate înainte de cea implicită.

3.1.5 Modelul

Într-o aplicație *MVC*, *modelul* este responsabil de manipularea datelor. Acesta răspunde vizualizării dar și controller-ului, iar pe baza instrucțiunilor primite de la acesta se actualizează. Modelele pot fi create manual sau generate din entitățile bazei de date. De asemenea, aici este realizată și legătura cu baza de date, stocând datele într-un mod persistent. Acestea pot fi accesate cu ajutorul atributelor publice ale clasei.

3.1.6 Vizualizarea

Vizualizarea este componenta cu care utilizatorul interacționează în interfață. Vizualizările pot fi de mai multe tipuri: rezultate ale metodelor din controller, împărțite (care fac parte din toate paginile, cum ar fi *Layout*) sau parțiale (care pot fi reutilizate în mai multe

pagini).

3.1.6.1. Razor Engine

Utilizatorul poate vedea datele din aplicație și le poate modifica cu ajutorul unor motoare de vizualizare. Motorul folosit de *ASP.NET MVC5* este *Razor*. Acesta oferă posibilitatea integrării sintaxei de *C#* cu elementele de *HTML*.

3.1.6.2. Ajutoare HTML

Ajutoarele HTML unesc obiectul din model cu vizualizarea în sintaxa *Razor*. Pentru a trimite date din baza de date este inclus modelul dorit, apoi folosită sintaxa *@model*. Ajutoarele *HTML* sunt folosite pentru a genera cod *HTML*, pentru a afișa valorile modelului sau pentru a le modifica în cazul trimerii unui formular.

3.1.6.2.1 ViewBag: Un obiect de tip *ViewBag* este folosit pentru transmiterea datelor temporare(cele care nu se regăsesc în model) de la controller la vizualizare.

3.1.6.2.2. ViewData: Asemănător cu *Viewbag*, ajutorul *ViewData* trimite date de la controller la vizualizare, dar spre deosebire de primul unde era transmis un obiect, în acest caz se trimite un dicționar.

3.1.6.2.3. TempData: Acest ajutor poate transmite o valoare între 2 cereri consecutive. În momentul în care valoarea trimisă este primită de următoarea cerere, ea este ștearsă.

3.1.6.3. Vizualizarea Layout

Vizualizarea de tip layout conține elementele care se regăsesc pe fiecare pagină din aplicație (de exemplu meniul, antetul, subsolul). Astfel utilizatorul poate defini un șablon care să fie moștenit pe mai multe pagini, fără a repeta codul. De asemenea, este ușor de menținut, fiind necesare modificări doar într-un singur fișier.

3.1.6.4. Vizualizarea Parțială

O vizualizare parțială este o vizualizare reutilizabilă a unor pagini web care poate fi folosită în una sau mai multe vizualizări sau chiar în cea de tip Layout, pentru a elimina codul redundant.

3.1.7 Controller-ul

Controller-ul este clasa responsabilă de execuția aplicației, de returnarea răspunsului pentru cererea făcută de utilizator. El poate realiza mai multe acțiuni și poate returna tipuri diferite de rezultate pentru o cerere specifică. Acesta controlează logica aplicației și face

legătura între model și vizualizare (primește o cerere prin intermediul vizualizării, procesează datele cu ajutorul modelului și trimite rezultatele înapoi la vizualizare).

3.1.7.1. Acțiuni

Acțiunile sunt metodele publice ale unui *controller*. Ele sunt publice și nu pot fi supraîncărcate sau statice. Fiecare controller vine cu metoda *Index*, care returnează o vizualizare.

3.1.7.2. Rezultatele Acțiunilor

Framework-ul *ASP.NET MVC* conține numeroase clase care pot fi returnate ca rezultate ale acțiunilor. Ele pot avea tipuri diferite, de exemplu *string-uri*, fișiere *HTML*, fișiere *json* etc. Clasa *ActionResult* este folosită pentru a putea returna orice tip de rezultat inclus, fără a modifica antetul funcției.

3.1.7.3. Parametrii unei Acțiuni

Parametrii unei acțiuni pot fi de orice tip și reprezintă ruta pentru acțiune. Numele lor trebuie să fie identic cu cel definit în ruta configurată în fișierul *RouteConfig.cs*.

3.1.7.4. Selectorii

Selectorii sunt atribute care se aplică metodelor pentru a determina sistemul de rutare să folosească metoda corectă pentru o anumită cerere. Există 3 tipuri de selectorii:

3.1.7.4.1. *ActionName*: Acest atribut lasă utilizatorul să ofere acțiunii un nume diferit față de metodă. În acest caz, pentru a determina ruta va fi folosit numele dat în *ActionName*, nu cel al metodei.

3.1.7.4.2. *NonAction*: Atributul *NonAction* este folosit când se dorește ca o metodă să existe în controller, dar să nu poată fi accesată cu ajutorul unei rute.

3.1.7.4.3. *ActionVerbs*: Acest atribut determină accesarea unei rute în funcție de verbul HTTP folosit.

3.1.7.5. Verbe HTTP

Verbele HTTP sunt prezente în selectorii de tip *ActionVerbs* și definesc ruta care urmează să fie accesată. Pot fi mai multe verbe pentru o acțiune, iar în cazul în care nu este niciun verb specificat este considerat în mod implicit *get*.

3.1.7.5.1. *GET*: Este folosit pentru a cere informații de la server

3.1.7.5.2. *POST*: Creează o resursă nouă sau trimite date către server prin intermediul unui formular.

3.1.7.5.3. PUT: Este folosit pentru a modifica o resursă deja existentă

3.1.7.5.4. DELETE: Este folosit pentru ștergerea unei resurse existente.

3.1.7.5.5. HEAD: La fel ca și get, acesta cere informații de la server, dar, spre deosebire de acesta, nu returnează conținutul răspunsului, ci doar antetul.

3.1.7.5.6. OPTIONS: Reprezintă o cerere pentru informație, returnează variantele de răspuns posibile bazate pe o adresă specificată.

3.1.7.5.7. PATCH: Este asemănător cu PUT, modifică parțial o resursă deja existentă.

3.2 Identity Framework

3.2.1 Preliminarii

Framework-ul *ASP.NET MVC5* vine cu posibilitatea adăugării unui sistem de autentificare cu ajutorul framework-ului *Identity*. Acesta poate fi adăugat în momentul creării proiectului prin schimbarea modului de autentificare, dar se poate și instala ulterior cu ajutorul managerului de pachete *NuGet*.

El a fost creat pentru a înlocui sistemele de apartenență folosite înainte și include suport pentru profil, integrare cu *OAuth (open authorization)* și oferă posibilitatea autentificării folosind un cont de social-media.

3.2.2. Autentificare

În mod implicit, *Identity* oferă posibilitatea adăugării unui cont folosind o adresa de mail validă, o parolă complexă și confirmarea ei. Pe lângă aceste câmpuri, contul se poate personaliza prin adăugarea altor proprietăți, în modelul *IdentityModels*. De asemenea, framework-ul oferă și posibilitatea de a adăuga și personaliza roluri pentru fiecare utilizator, utilizând clasa *RoleManager*.

3.2.3. Autorizare

Odată cu adăugarea de roluri, utilizatorii pot primi permisiuni în funcție de acestea. Pentru a face diferența, se folosește ajutorul *Authorize*, la nivel de controller sau de metode. Astfel, dacă un utilizator dorește să acceseze o pagină pentru care nu are permisiuni, va fi redirecționat către pagina de Înregistrare.

3.2.4. Autentificare folosind platforme sociale

Cum rețele sociale au căpătat o foarte mare popularitate, majoritatea utilizatorilor preferă să se conecteze folosind un cont deja existent pe social media, decât să creeze încă unul pentru o altă aplicație. Odată cu evoluția acestor platforme, framework-ul *Identity* s-a adaptat și a oferit o modalitate de a permite autentificarea cu conturi de pe alte rețele.

Autentificarea de acest tip este posibilă prin adăugarea unei aplicații noi pe platforma dorită, care să fie utilizată în procesul de open *authentication*.

3.3 Entity Framework

3.3.1 Preliminarii

Entity Framework este o tehnologie creată de Microsoft pentru a automatiza toate activitățile aplicației care au legătură cu baza de date. Acesta susține tehnica *First-Code*, care permite utilizatorului să dezvolte modelele necesare bazei de date, apoi să le adauge folosind *Entity framework*. Astfel, nu există discrepanțe între dezvoltarea modelelor și baza de date creată.

3.3.2 Crearea tabelelor

Crearea tabelelor este bazată pe modelele existente și pe contextul bazei de date. Fiecare Model reprezintă un tabel, iar fiecare proprietate a modelului reprezintă o coloană a tabelului. În cazul relațiilor de tip mai mulți la mai mulți, se poate crea un model pentru tabelul asociativ atunci când se dorește adăugarea altor proprietăți pe lângă relația implicită, sau, în cazul în care acesta nu este creat, se generează în mod automat.

3.3.3 Stocarea datelor

Baza de date este actualizată la fiecare rulare de aplicație, dar și de fiecare dată când se produce o modificare asupra datelor existente. Pentru a nu exista diferențe, au fost create migrațiile. Ele se ocupă de persistența datelor și de potrivirea bazei cu modelele. Pentru a menține sincronizarea, atunci când migrațiile sunt activate, tehnica *Code-First Migrations* actualizează doar schema bazei de date, fără a șterge și reface baza de date de fiecare dată când se produce o modificare.

3.3.4 LINQ Query

LINQ (Language Integrated Query) este o sintaxă uniformă folosită în C# pentru a obține date din surse diferite și din diferite formate. A fost integrat în C# cu scopul de a face

legătura între limbajele de programare și bazele de date, dar și de a oferi o singură sintaxă indiferent de tipul sursei de date. Acest tip de interogare returnează un rezultat de tip obiect.

3.4 Bootstrap 5

3.4.1 Preliminarii

Bootstrap 5 este cea mai nouă versiune a celui mai popular framework pentru *frontend*. Acesta include șabloane de cod HTML și CSS pentru multiple componente, cum ar fi butoanele, meniul de navigare, listele derulante, și multe altele. Printre avantajele folosirii acestui framework se numără ușurința utilizării, design-ul consistent, compatibilitatea cu multiple *browsers* și adaptarea mai ușoară în funcție de dimensiunea ferestrei.

3.4.2 Meniul de navigare

Framework-ul *Bootstrap* include clasa *.navbar* care permite realizarea unui meniu de navigare. Acest meniu este strâns sub un buton în cazul în care fereastra este de dimensiuni mici, și aliniat în cazul ferestrelor mai mari. La clasa de bază se pot adăuga variații de alte clase pentru a modifica elementele, culorile sau pentru a adăuga altele, de exemplu poză de profil sau bară pentru căutare.

3.4.3 Card

În *Bootstrap*, un card este un container care este flexibil și extensibil, în funcție de conținut. Un element de acest tip poate conține subsol, antet, și o varietate largă de componente (butoane, poze, text, liste, link-uri, etc.). Acestea pot fi adăugate cu ajutorul claselor predefinite *.card* și *.card-body*.

3.4.4 Formularul

Deși formularele sunt una din cele mai importante componente ale unei aplicații web, crearea lor de mână poate fi plictisitoare și repetitivă. Pentru a rezolva această problemă și a ușura munca dezvoltatorilor, *Bootstrap* conține o varietate de clase predefinite, atât pentru tipul de formular, cât și pentru tipul de date de intrare și etichete. Există trei tipuri de formulare predefinite: cel vertical, cel orizontal și cel pe aceeași linie. Datorită împărțirii pe coloane specifică framework-ului, formularul se adaptează la dimensiunile ecranului pe care este afișat.

3.4.5 Jumbotron

Jumbotronul reprezintă o modalitate atractivă de a scoate în evidență conținutul important al paginii. Acesta își modifică dimensiunile în funcție de componentele pe care le conține și poate avea o varietate de tipuri de date (text, imagini, butoane, liste, etc.). Deși clasa predefinită a fost eliminată începând cu *Bootstrap5*, acesta se poate realiza cu ușurință prin combinarea a trei clase pentru un element de tip *div*.

3.4.6 Tabelul

La fel ca în viața reală, tabelele în *Bootstrap* sunt folosite pentru a prezenta informații prin împărțirea lor în linii și coloane. Folosirea clasei *.table* oferă un mod mai elegant de a îi îmbunătăți apariția. La această clasă se pot adăuga stiluri diferite pentru a face tabelul să se potrivească cu restul paginii, cum ar fi *.table-dark* pentru fundal negru și scris alb, *.table-success* pentru tabel verde, *.table-info* pentru tabel albastru, și multe altele.

3.4.7 Butoanele

Una din cele mai importante părți ale aplicației este reprezentată de butoane, fără de care multe acțiuni nu ar fi posibile. Ele sunt folosite cu mai multe scopuri, cum ar fi redirecționarea către anumite pagini, trimiterea unui formular, afișarea unor informații și lista poate continua. Framework-ul *Bootstrap* vine în ajutor cu numeroase clase de butoane, fie ele colorate, transparente, mari, mici sau animate. Printre cele mai utilizate clase, se numără *.btn-outline-primary* și *.btn-primary*, care creează clasicele butoane albastre.

3.4.8 Listele

În *Bootstrap 5*, listele sunt create prin adăugarea claselor *.list-group* și *.list-group-item* la elementele pe care le conține lista respectivă. Acestea au ca scop gruparea unor elemente de același tip și afișarea lor într-un mod asemănător. Folosind și ajutoare pentru *html*, pot fi create liste dinamice prin popularea acestora folosind elemente parcurse cu *@foreach*.

3.4.9 Bara de căutare

Bara de căutare este un element special realizat pentru motoare de căutare. Ea poate fi alcătuită din mai multe elemente, dar cel mai important este câmpul de introducere a datelor. Utilizatorul poate introduce text sau poate selecta elemente dintr-o listă derulantă, iar, pe baza acestei introduceri, sunt generate rezultate.

3.4.10 Câmpurile pentru dată și ora

Pentru a selecta data si ora pot fi folosite instrumente speciale. *Bootstrap 5* vine cu anumite clase care să permită acest lucru, pe lângă cele existente deja in *HTML*. Acești selectori pot fi configurați pentru a permite utilizatorului să selecteze doar anumite date sau ore, sau doar dintr-un anumit interval.

Capitolul 4. Descrierea aplicației

4.1 Preliminarii

Aplicația dezvoltată în această lucrare își propune să ofere o soluție utilizabilă pentru gestiunea unui lanț de cabinete medicale. Printre avantajele sale se numără varietatea de funcționalități aduse atât utilizatorilor cât și doctorilor sau administratorilor. Este o platformă personalizabilă care se adaptează în funcție de nevoile celor care o folosesc.

Fiind creată în *ASP.NET MVC* cu ajutorul framework-urilor *Identity*, *Entity* și *Bootstrap* 5, aceasta este compatibilă cu majoritatea *browserselor* și se adaptează la dimensiunea ecranului pe care este accesată.

4.2 Stocarea datelor

Conexiunea la baza de date este realizată folosind *Entity Framework* și *tehnica Code-First*. Astfel, tabelele din baza de date au fost create pe baza modelelor. Datorită migrațiilor, fiecare modificare din model este actualizată și în baza de date, fără ca datele introduse să se piardă.

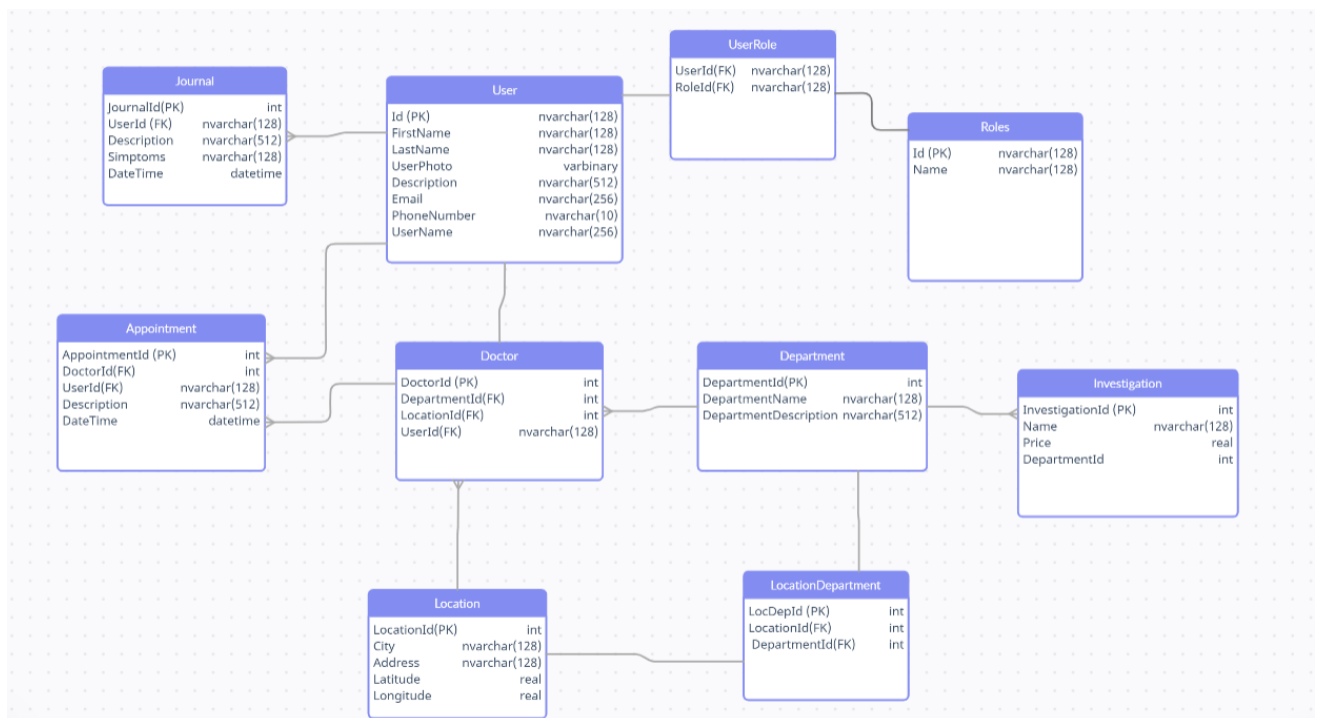


Figura 4.1 Diagrama conceptuală a bazei de date

Baza de date conține 10 tabele, dintre care 2 sunt asociative, create pentru a rezolva

relațiile de tip mai mulți la mai mulți. Tabelul de utilizatori este modificat prin adăugarea unor noi atribute față de cele generate de *Identity framework*. Un utilizator trebuie să aibă email și nume de utilizator, iar opțional poate adăuga numele, prenumele, numărul de telefon, descriere și poză de profil. Atunci când se conectează pentru prima dată în aplicație, el primește rol de utilizator, rol care poate fi modificat de către administrator în doctor sau administrator.

Tabela de Rol conține *id-ul* și numele pentru fiecare rol, și se leagă cu tabela de utilizatori printr-un tabel asociativ de *RolUtilizator*. În acest tabel este reținut *id-ul* utilizatorului și *id-ul* rolului pe care acesta îl are.

În legătura cu tabela Utilizator este și cea pentru Jurnal. Între ele este o relație de tip unul la mai mulți, un utilizator având posibilitatea de a ține mai multe jurnale, dar un jurnal aparținând doar unui utilizator. Un jurnal are următoarele atribute: *id-ul* care este generat automat pentru fiecare element introdus, *id-ul* utilizatorului, descrierea intrării în jurnal, simptomele și data la care a fost adăugat.

De asemenea, o relație de același tip se stabilește și între tabela de Utilizatori și cea de Programări. Un utilizator poate solicita mai multe programări, dar o programare este înregistrată pentru un singur utilizator, prin intermediul *id-ului* sau prezent drept cheie străină. O programare conține o descriere, nu este obligatorie dar este o modalitate ca doctorul să își facă o idee despre problema întâmpinată de pacient, data și ora la care va avea loc programarea și *id-ul* doctorului la care este aceasta. *Id-ul* doctorului prezent în tabla de programări face legătura cu tabela de Doctori, legătură de tip unul la mai mulți. La fel ca în cazul utilizatorilor, un doctor poate avea mai multe programări, dar o programare poate fi făcută la un singur doctor.

Doctorul este, la rândul său, un utilizator care are rol de doctor, deci se leagă de tabela de Utilizatori printr-o relație de tip unu-la-unu. Un doctor poate avea un singur cont de utilizator, iar, pe lângă proprietățile deja existente, se adaugă *id-ul* departamentului de care aparține și *id-ul* locației unde lucrează. În momentul în care un utilizator primește rolul de doctor, *id-ul* acestuia este actualizat automat în tabelă.

Pentru a putea fi adăugat în baza de date, doctorului trebuie să i se asocieze un departament și o locație. În ambele cazuri, este formată o relație de tipul unul la mai mulți. Un departament poate avea mai mulți doctori și mai mulți doctori pot lucra într-o locație, dar un doctor poate aparține doar de o specializare și de un cabinet.

Pentru a adăuga o locație sunt necesare următoarele atribute: orașul, adresa, latitudinea și longitudinea, iar pentru un departament trebuie adăugate numele și descrierea. Ambele tabele conțin un *id* care se actualizează în mod automat pentru fiecare entitate.

Între locații și departamente există o relație de tip mai mulți la mai mulți. Pentru a rezolva aceasta relație a fost creat un tabel asociativ care conține *id*-ul pentru locația și departamentul pentru care se realizează asocierea. Această asociere poate fi făcută de către administrator de fiecare dată când este necesar.

Pe lângă acestea, tabela departamentelor se unește cu cea a investigațiilor cu o relație de tip unul la mai mulți. Mai exact, o investigație se încadrează într-un singur departament, dar un departament este responsabil de mai multe investigații. Atributele unei investigații sunt numele și prețul, *id*-ul care se incrementează automat și *id*-ul departamentului de care aparține.

4.3 Autentificarea

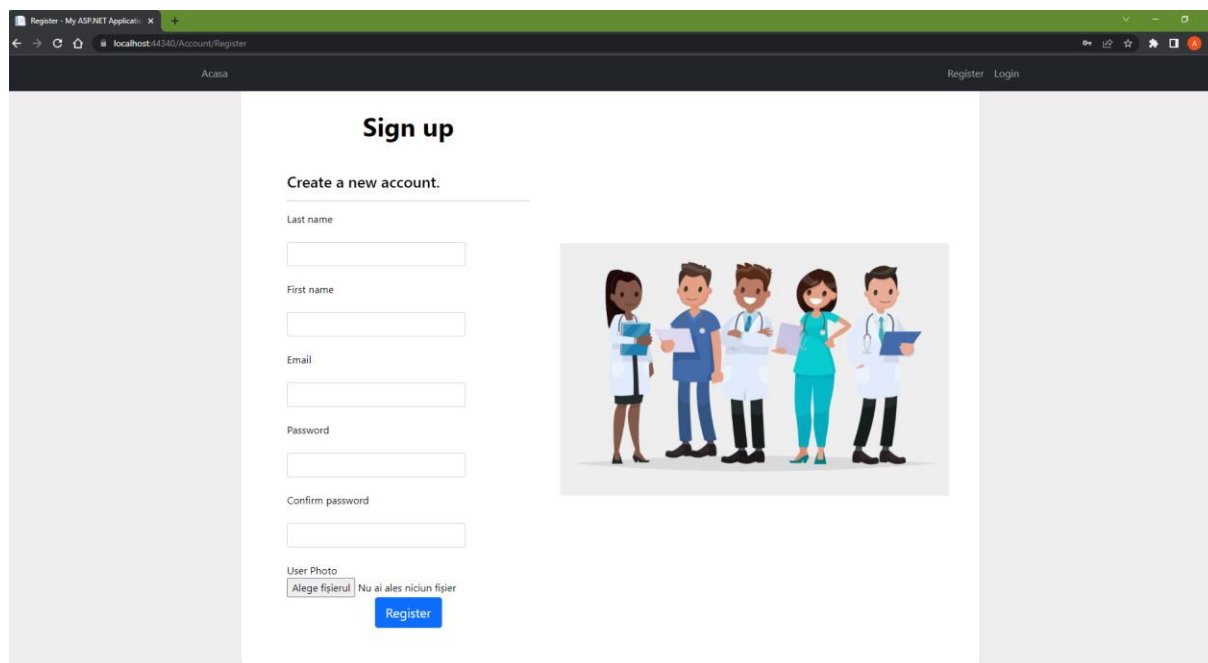
Framework-ul *ASP.NET MVC5* oferă posibilitatea autentificării folosind *Identity* Framework. În cazul de bază, se creează un profil bazat pe o adresă de email și parolă. Pentru aplicația din această lucrare, modelul a fost modificat pentru a conține un profil complet al utilizatorului. Astfel, pe lângă email și parolă, el poate adăuga numele complet (nume și prenume), descriere și poză de profil. Fiecare utilizator se poate autentifica prin crearea unui cont nou sau prin folosirea unei platforme sociale, în acest caz Facebook. În momentul creării contului, utilizatorul primește în mod automat un *id* și rol de utilizator.

În cazul conectării cu un cont de pe altă aplicație, după ce contul este creat, utilizatorul este redirecționat către un alt panou unde își poate completa profilul.

Structura profilului este reținută în directorul *Models*, modelul *IdentityModels*. Metodele pentru gestionarea profilului se găsesc în directorul *Controllers*, în *UsersController*. Orice utilizator își poate vedea și edita propriul profil, fără a își putea modifica rolul. Un utilizator cu rol de administrator poate vedea toate profilurile și le poate edita, inclusiv rolul.

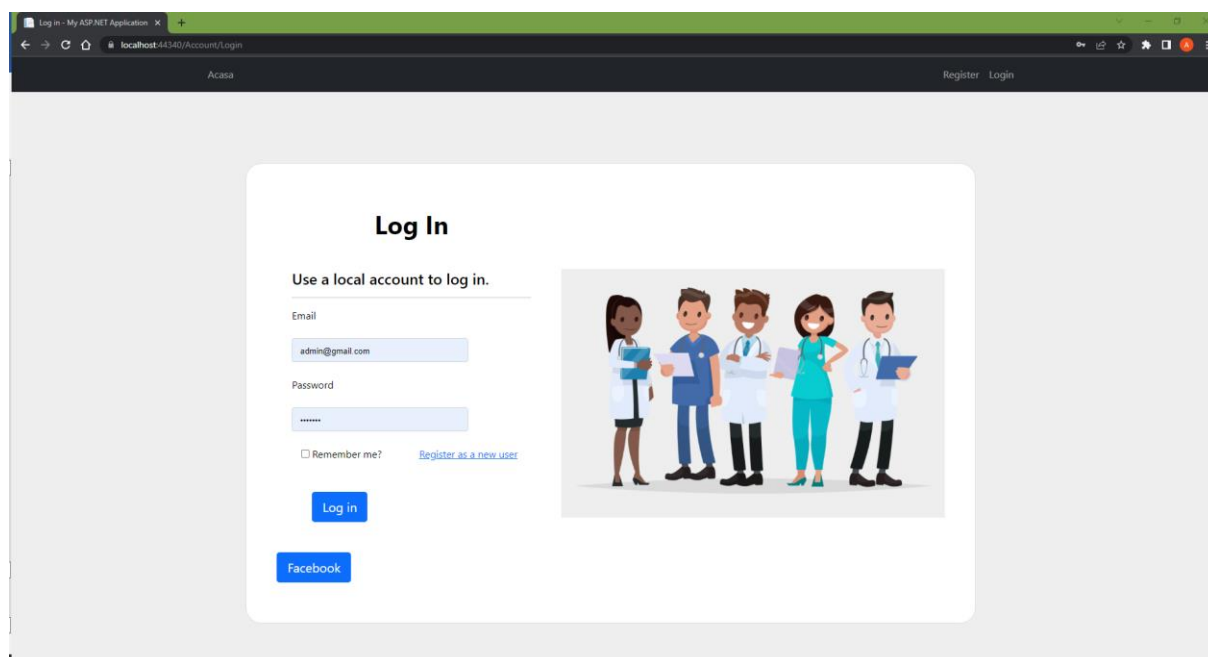
Interfața cu utilizatorul se afla în dosarul *Views*, în sub-dosarele *Users* și *Account*. Paginile pentru autentificare și adăugare cont se află în categoria *Account* și au fost create urmărind același șablon: culoarea de fundal a întregului corp a fost setată gri, apoi a fost adăugat un container și un card cu margini rotunjite. În corpul cardului au fost adăugate două coloane, una pentru formular și una pentru imagine. Pentru formular au fost folosite mai multe ajutoare pentru HTML cu sintaxă *Razor*. De exemplu, pentru etichetele câmpurilor a fost folosit `@Html.LabelFor` și pentru câmpurile de completat `@Html.TextBoxFor`. Datele au fost transmise de la controller la vizualizare folosind `@model`. Pentru ambele pagini a fost folosită aceeași imagine, pentru a oferi consistență în privința design-ului aplicației. De asemenea, datele introduse sunt validate folosind `@Html.ValidationSummary`, iar în cazul în care acestea

nu corespund cerințelor, apare un mesaj de eroare iar utilizatorul nu poate continua.



The screenshot shows a web browser window with the URL `localhost:44340/Account/Register`. The page has a dark header with "Acasa" on the left and "Register Login" on the right. The main content area is titled "Sign up" and contains the text "Create a new account." Below this are input fields for "Last name", "First name", "Email", "Password", and "Confirm password". There is a "User Photo" section with a button "Alege fișierul" and a text "Nu ai ales niciun fișier". A blue "Register" button is at the bottom. To the right of the form is an illustration of five healthcare professionals (three men and two women) in white coats and scrubs, holding clipboards.

4.2. Pagina de înregistrare



The screenshot shows a web browser window with the URL `localhost:44340/Account/Login`. The page has a dark header with "Acasa" on the left and "Register Login" on the right. The main content area is titled "Log In" and contains the text "Use a local account to log in." Below this are input fields for "Email" (containing "admin@gmail.com") and "Password" (masked with "*****"). There is a "Remember me?" checkbox and a link "Register as a new user". There are two buttons: a blue "Log in" button and a blue "Facebook" button. To the right of the form is an illustration of five healthcare professionals (three men and two women) in white coats and scrubs, holding clipboards.

4.3 Pagina de autentificare

4.4. Rolurile utilizatorilor

După cum a fost menționat anterior, atunci când un cont este creat el primește implicit rolul de utilizator. Acest lucru este datorat modificării făcute în fișierul *AccountController*, în

metoda *Register*, unde fiecărui utilizator adăugat i se atribuie rolul de utilizator.

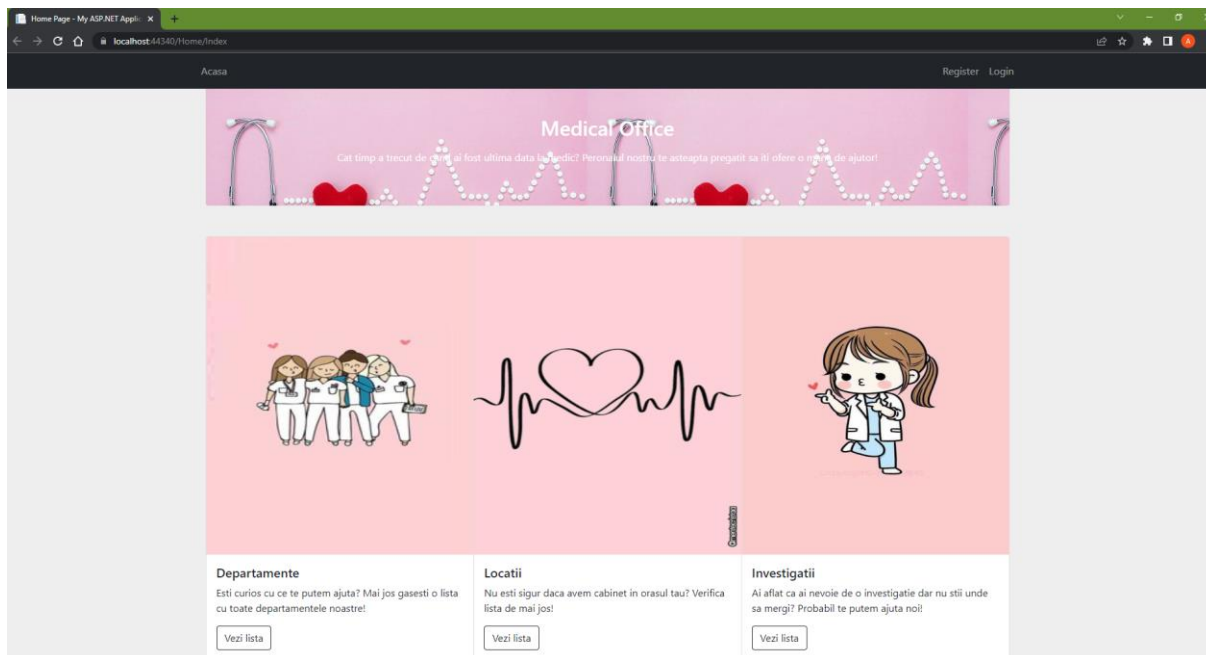
În funcție de rolul său, un utilizator poate face anumite acțiuni sau poate vedea anumite pagini, care pentru alt tip de utilizator nu sunt posibile. Utilizatorii neautentificați pot vedea pagina de pornire, și listele de departamente, de locații și investigații posibile.

Pagina de start se găsește în dosarul *View*, categoria *Home*, fișierul *Index* și este alcătuită din meniu, un element de tip *jumbotron* și trei elemente de tip card aliniate, toate pe un fundal gri, aceeași nuanță ca în paginile pentru înregistrare și autentificare.

Elementul de tip *jumbotron* este folosit în general pentru a capta atenția utilizatorilor cu informații importante, menite să iasă în evidență. Acesta este alcătuit dintr-un element de tip *div*, care are o imagine de fundal, apoi un titlu și un subtitlu.

Cele trei carduri sunt grupate folosind clasa *card-group*, toate având aceeași structură: o imagine sub care se află corpul cardului, alcătuit la rândul său din titlu, text și buton.

Meniul este configurat în fișierul *Layout* din directorul *Shared* și adaptează în funcție de tipul de utilizator. Utilizatorii neautentificați pot vedea doar opțiunea de Acasă și butoanele pentru Înregistrare și Autentificare, pe când cei conectați sunt întâmpinați cu un mesaj personalizat în funcție de numele de utilizator, alături de poza de profil. În cazul în care utilizatorul nu a adăugat o poză de profil, va fi afișată o poză clasică înlocuitoare.



4.4. Pagina de start

4.5 Administratorul

Utilizatorul cu rol de administrator are cele mai multe permisiuni si responsabilități în ceea ce privește gestiunea platformei. Permisuniile sunt stabilite la nivelul acțiunilor definite în controller cu ajutorul verbului *Authorize*. Acest verb decide dacă un anumit utilizator are permisiuni pentru comanda respectivă la nivel de *backend*.

Pentru a adapta interfața în funcție de utilizatorul curent, este necesară adăugarea unei verificări în codul pentru vizualizare. Pentru aceasta verificare este folosită funcția *User.IsInRole(..)*, care verifică rolul utilizatorului curent și determină dacă are rolul căutat sau nu. Ea se folosește într-o sintaxă de *if*, unde codul aflat înăuntru este apelat doar în cazul în care utilizatorul are rolul necesar.

Administratorul singurul utilizator care își primește drepturile direct din configurările aplicației, în momentul în care sunt create si rolurile. În cazul în care nu se dorește adăugarea rolului direct din configurări, acesta se poate adăuga si prin asocierea *id*-ului *userului* dorit cu *id*-ul rolului în tabela *UserRoles*. Odată ce există un utilizator care are rol de administrator, acesta poate modifica drepturile celorlalți utilizatori în doctor sau în administrator.

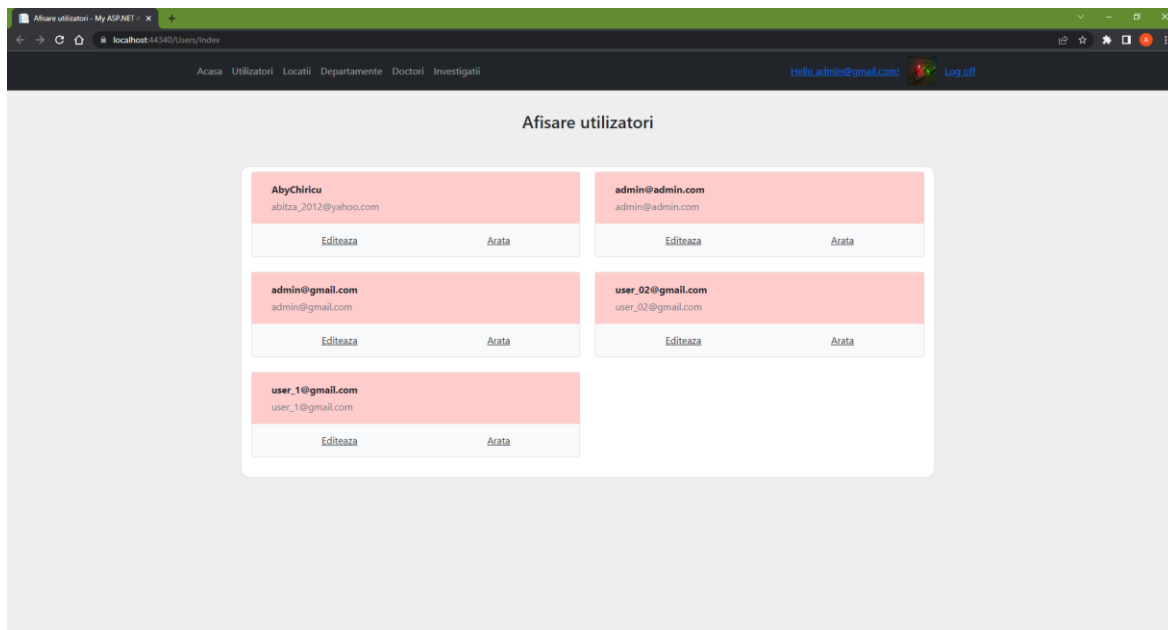
În momentul conectării cu un utilizator de tip administrator, în meniu apar butoane care nu erau vizibile înainte de conectare: Utilizatori, Locații, Departamente, Doctori și Investigații.

4.5.1 Editarea utilizatorilor

Una dintre cele mai importante funcționalități dintre cele ale administratorului este cea de care am vorbit și înainte, de a schimba rolul unui anumit utilizator. Pentru aceasta, administratorul trebuie să meargă pe pagina *Utilizatori*, unde este afșată lista tuturor celor existenți.

Similar cu paginile de autentificare și înregistrare, panoul utilizatorilor folosește drept fundal aceeași variație de gri, peste care se regăsește un element de tip card care își actualizează dimensiunile în funcție de numărul elementelor interioare. În acest card se află lista dinamică de utilizatori, preluată din controller folosind *Viewbag*, fiecare fiind afișat într-un card separat.

Fiecare element din lista are numele de utilizator și adresa de email afișate în corpul cardului, corp colorat cu roz, asemănător paginii de start, iar în subsolul cardului există două butoane: Editează și Arată, subsolul fiind aceeași culoare cu fundalul paginii.



4.5. Lista de utilizatori

În momentul în care administratorul dorește să editeze un utilizator, el apasă pe butonul Editează și este redirecționat către pagina de editare a utilizatorului. Acesta poate să modifice numele de utilizator, email-ul, numărul de telefon și rolul utilizatorului. Formularul respectă culorile și stilul formularelor pentru autentificare, dar este de menționat câmpul pentru rol, care este de tip listă derulantă. Astfel, utilizatorul nu are altă opțiune decât cele existente deja, deci nu poate alege o valoare greșită. Câmpul este populat prin transmiterea tuturor rolurilor posibile din controller către vizualizare folosind ajutorul *ViewBag*.

În model a fost adăugată o proprietate de tip *IEnumerable<SelectListItem>* care permite selectarea tuturor obiectelor de un anumit fel și utilizarea lor în cazul unei liste derulante. Pentru a putea fi transmisă, a fost nevoie de adăugarea în controller unei metode de tip *NonAction* prin care să se populeze lista dorită. În acest caz, pentru fiecare rol au fost selectate *id*-ul drept valoare și numele drept text și a fost returnată lista formată din aceste perechi de valori, care au fost trimise către vizualizare folosind *ViewBag*.

4.6 Formularul de editare utilizator de către administrator

4.5.2 Adăugarea unui medic

În cazul în care se dorește adăugarea unui doctor, acesta trebuie să aibă un cont pe care administratorul să îl poată transforma în cont de doctor. Astfel, în momentul în care administratorul selectează rolul de doctor și salvează schimbările, el este redirecționat către pagina de editare doctor pentru a completa restul de informații necesare.

În pagina de adăugare doctor, administratorul trebuie să asocieze doctorul creat cu departamentul și locația de care aparține. Acesta nu poate salva formularul dacă nu completează ambele câmpuri. În cazul în care nu selectează nimic și încearcă să iasă din pagină, el primește un mesaj care îl atenționează să completeze câmpurile respective.

Formularul are același stil cu cel precedent, singura diferență fiind reprezentată de câmpurile care necesită completare, anume cele două liste derulante pentru locație și departament. După selectarea acestora, utilizatorul este redirecționat către lista de doctori existenți. Asemenea listei pentru roluri, listele derulante pentru departamente și locații sunt create și populate în controller-ul pentru doctori, și transmise către vizualizarea de editare a doctorului prin intermediul *ViewBag*.

În panoul cu lista de doctori sunt afișați toți doctorii existenți alături de locația și departamentul de care aparțin. Această listă este pur informativă, fără posibilitatea efectuării altor acțiuni asupra lor de aici. Lista conține două butoane, Afișare lista locații și departamente și Asociază Locație cu Departament. În cazul primului buton, utilizatorul trimis către lista de

locații și departamente cu care sunt asociate.

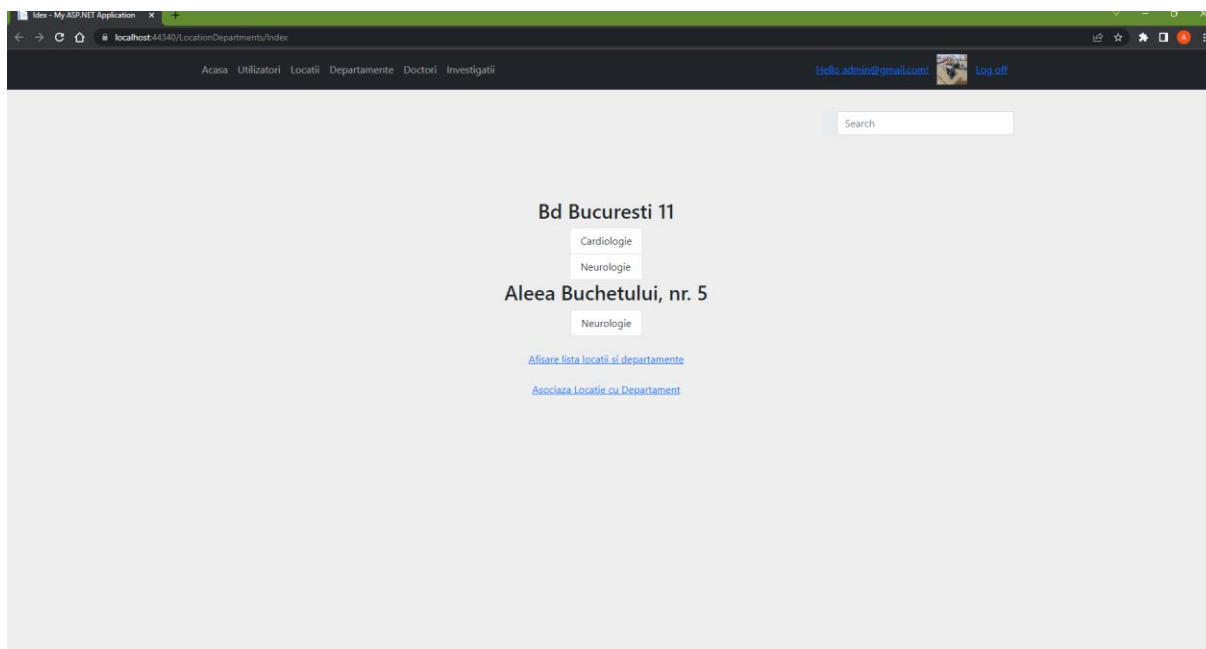
4.5.3 Asociere Locație-Departament

Aici departamentele sunt grupate în funcție de adresa de care aparțin și se poate observa ca titlu adresa, urmată de lista specializărilor prezente. La fel ca pe pagina precedentă, și aici este prezent un buton care permite administratorului să realizeze o nouă asociere între cele două clase. În momentul apăsării acestui buton, apare un panou asemănător cu cele de până acum care permite selectarea locației și departamentului dorite dintr-o listă derulantă.

Odată selectată, asocierea poate fi vizibilă în lista precedentă, cea la care este trimis înapoi utilizatorul. De asemenea, în aceeași pagină este prezent și un câmp de căutare, pentru a facilita găsirea asocierii dorite într-un timp cât mai scurt.

Departamentele sunt grupate în funcție de locația din care fac parte și sunt afișate sub formă de listă. Pentru ca elementele să fie grupate, a fost folosită clasa *.list-group* coținută de *Bootstrap5*.

Acest design renunță la elementul de tip card cu care utilizatorul a fost obișnuit din motive de aspect, fiind mai multe elemente cu același stil care ar fi generat senzația de supraîncărcare a paginii. Astfel, a fost ales un model mai simplu, care păstrează paleta de culori, dar oferă și un aspect elegant.



4.7 Lista de locații și departamentele asociate

4.5.4 Locații

Mergând în pagina de locații, administratorul poate observa un tabel cu toate locațiile existente, alcătuit din 4 coloane: oraș, adresă, un buton pentru detalii și unul pentru editare.

Butonul de editare este disponibil doar pentru administrator, fiind singurul care are voie să adauge sau să modifice locații. Formularul de editare este identic cu cel de adăugare, singura diferență fiind câmpurile pre-completate și este construit pe același principiu cu cele de până acum, singura diferență fiind reprezentată fotografia care ilustrează un simbol pe hartă pentru un cabinet medical. În momentul adăugării sau editării unei locații este recomandat să se completeze și coordonatele geografice, pentru ca locația exactă să fie mai ușor de găsit.

Butonul de afișare transferă utilizatorul către o pagină unde sunt afișate detaliile locației, dar și un buton de editare și unul de ștergere. Astfel, un utilizator cu rol de administrator are posibilitatea de a șterge locația în cazul în care aceasta nu mai este valabilă.

Tabelul este, de asemenea, completat în mod dinamic cu ajutorul locațiilor preluate din baza de date în controller și transmise către vizualizare folosind *ViewBag*.

Oras	Adresa	Detalii	Editeaza
Ploiesti	Bd Bucuresti 11	Arata	Editeaza
Bucuresti	Aleea Buchetului, nr. 5	Arata	Editeaza

[Afișare lista departamente](#)
[Adauga Locatie](#)

4.8 Tabel cu locațiile existente

Acasa Utilizatori Locatii Departamente Doctori Investigatii

Hello admin@gmail.com Log off

Editare Locatie


Oras

Adresa

Latitudine

Longitudine

Modifica Locatia



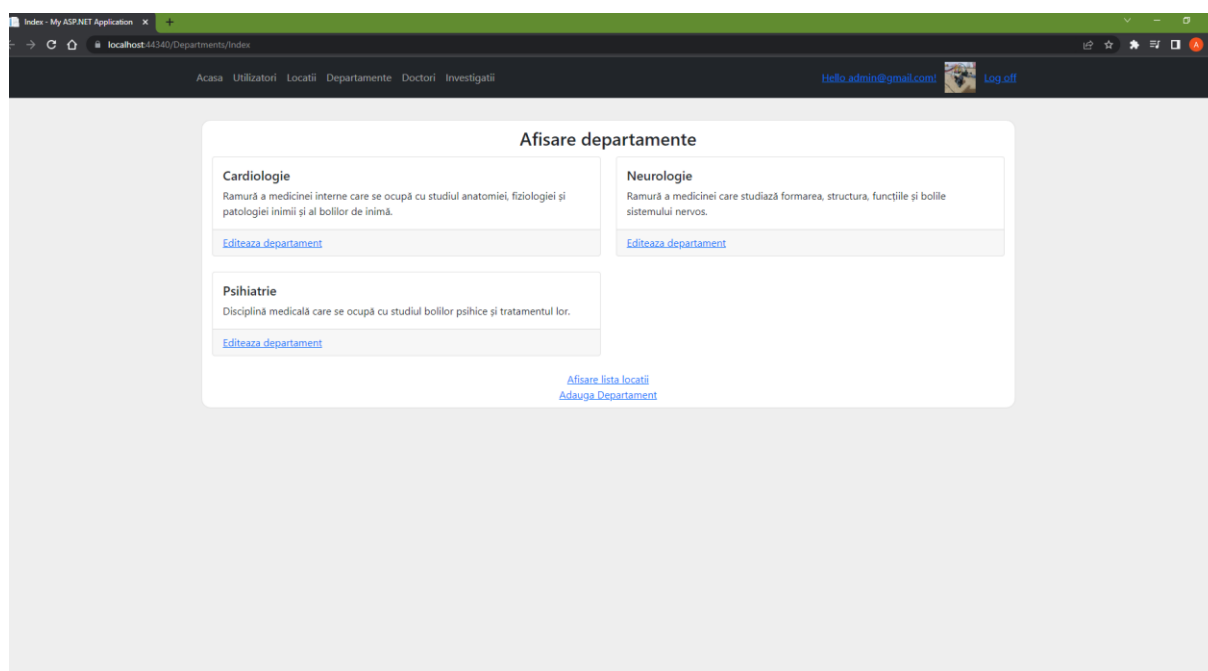
4.9 formularul pentru editarea locației

4.5.5 Departamente

În meniu este prezent și un buton numit Departamente, care afișează lista tuturor departamentelor existente împreună cu descrierea lor. Fiecare departament este însoțit de un buton de editare care permite administratorului să modifice numele sau descrierea acestuia. De asemenea, de pe această pagină putem folosi butoanele existente pentru a merge către lista de locații sau pentru a adăuga un departament nou.

Administratorul poate adăuga un departament care apoi să fie asociat cu locațiile existente folosind un formular asemănător cu cele de până acum. Pentru adăugarea unei specializări sunt necesare doar numele și descrierea acesteia.

Asemenea listei de utilizator, aceasta prezintă specializările sub forma unei liste de card-uri, unde se găsește câte unul pentru fiecare departament. Administratorul poate vedea butonul de editare care se află în subsolul cardului departamentului respectiv. Elementele au o structură simplă: numele departamentului drept titlu și descrierea sa în corp.



4.10 Lista de departamente văzută de administrator

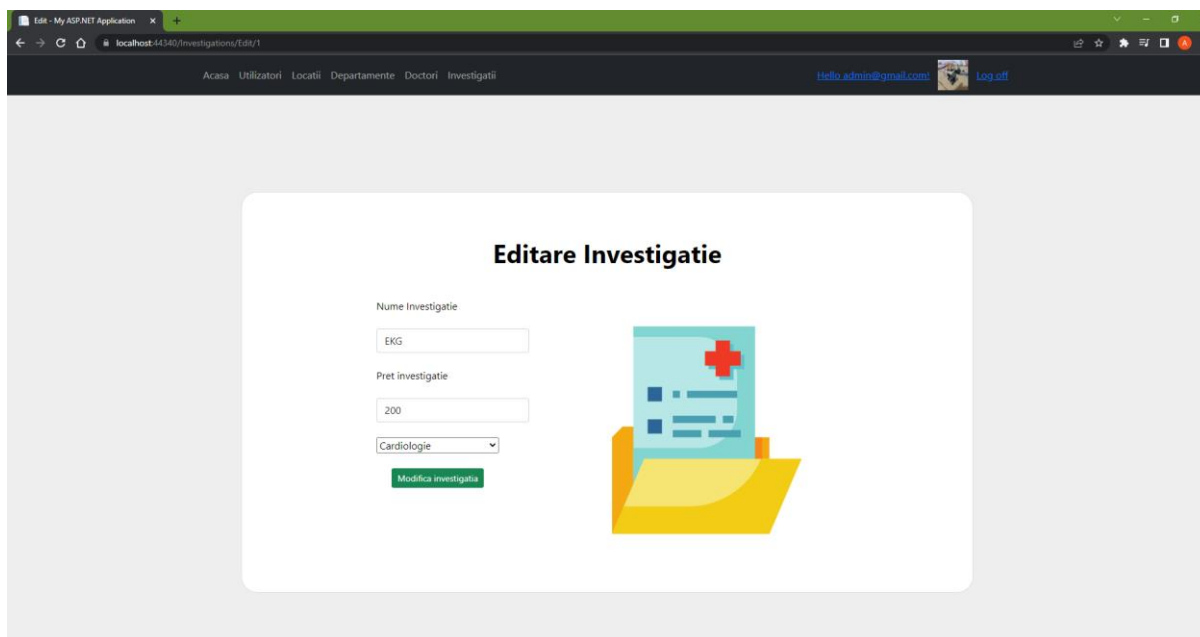
4.5.6 Investigații

Din pagina de investigații administratorul poate vedea un tabel cu toate investigațiile existente, departamentul lor, numele și prețul. De asemenea, pe fiecare rând al tabelului este prezent un buton de editare, buton disponibil doar administratorului. Tabelul are același stil clasic ca cel de la locații, cu numele coloanelor și al departamentelor scris îngroșat.

Butonul de editare deschide un formular asemenea celorlalte, în care câmpurile sunt pre-populate cu valorile precedente, iar departamentul este selectat folosind o listă derulantă. Același formular este folosit și pentru adăugarea unui nou departament, diferența între cele două fiind casetele care nu sunt completate în cazul unei investigații noi.

Departament	Investigatie	Pret	Editeaza
Cardiologie	EKG	200	Editeaza
Neurologie	CT	100	Editeaza

4.11 Investigațiile din lista administratorului



4.12 Editare investigație existentă

4.6. Utilizatorul

Utilizatorul are o varietate mai mică de acțiuni posibile comparativ cu administratorul, dar suficiente pentru a îi oferi o experiență plăcută și personalizabilă. Ca elemente deosebite, el poate solicita o programare la un anumit doctor și poate ține un jurnal în care să descrie starea sa de spirit din ziua respectivă, ce simptome a avut, cum s-a simțit, sau orice consideră relevant pentru a își urmări evoluția.

4.6.1. Locații

Un utilizator autentificat poate vedea lista de locații și detaliile fiecăreia, dar nu are posibilitatea de a adăuga sau de a modifica locația. În cazul în care, din întâmplare, acesta ajunge pe pagina de editare sau de adăugare, el este redirecționat către pagina de conectare pentru a se conecta cu un utilizator cu drepturi de administrator.

Lista de locații este afișată sub formă de tabel, de această dată cu 3 coloane în loc de 4, lipsind butonul de editare. În listă sunt afișate orașul și adresa, iar dacă utilizatorul dorește mai multe detalii poate merge pe pagina de afișare.

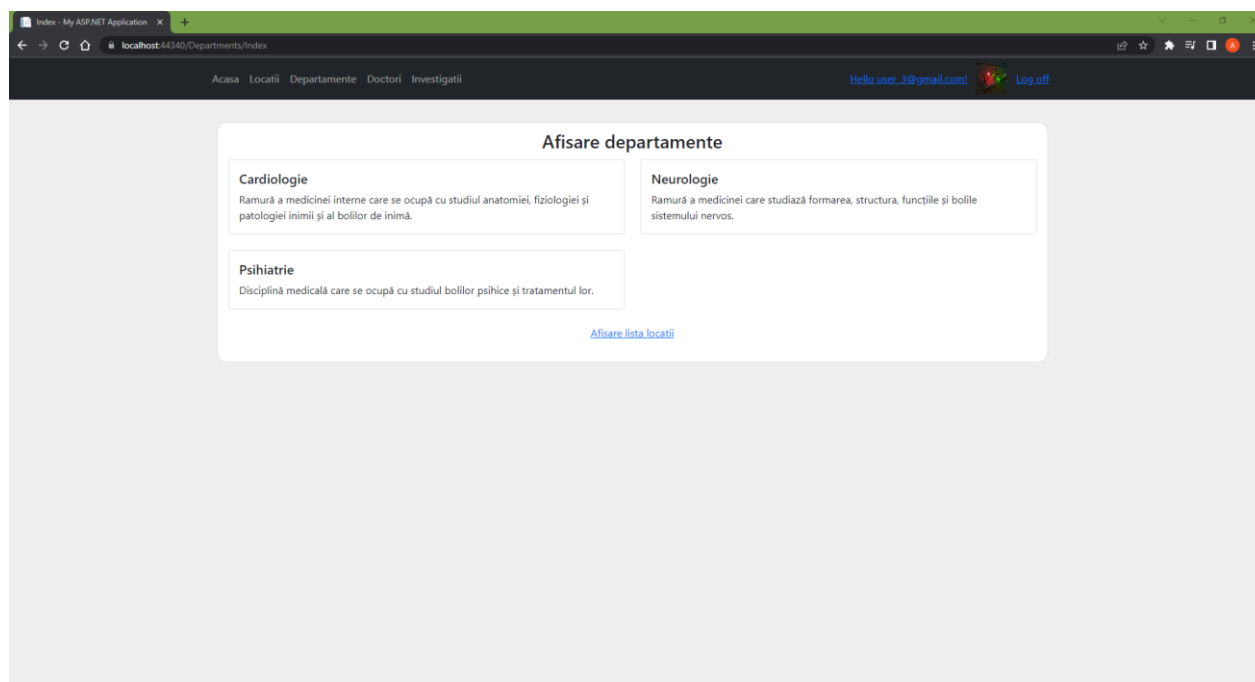
Pe pagina de afișare, pe lângă adresa și orașul sediului, sunt afișate și coordonatele geografice ale acestuia, mai exact latitudinea și longitudinea. Dacă administratorul avea posibilitatea să modifice sau să șteargă locația din acest panou, în cazul utilizatorului normal aceste lucruri nu sunt posibile.

Oras	Adresa	Detalii
Ploiesti	Bd Bucuresti 11	Arata
Bucuresti	Aleea Buchetului, nr. 5	Arata

4.12. Tabelul de locații vizibil pentru utilizator

4.6.2 Departamente

În cazul departamentelor, ele sunt afișate cu toate detaliile necesare, deci nu exista panou de afișare ca în cazul locațiilor. Fiecare element din listă conține numele departamentului și descrierea acestuia, fiind afișate sub formă de card fiecare. Deși administratorul are posibilitatea să adauge și să editeze departamente, singura acțiune disponibilă pentru utilizator este vizualizarea listei.



4.13 pagina de departamente văzută de utilizator

4.6.3 Doctori

Pagina de doctori reprezintă o pagină importantă pentru utilizatori, oferind informații referitoare la persoanele de care urmează să fie consultate. Pentru a găsi specialistul dorit mai ușor, această pagină pune la dispoziție trei tipuri de liste. În prima variantă, doctorii sunt afișati pe rând, așa cum sunt și în baza de date, alături de locația din care fac parte și de departamentul aparținător. În a doua variantă, doctorii sunt grupați în funcție de departamentul de care aparțin,

fiind mai ușor de găsit un specialist într-un anumit domeniu. Pentru al treilea caz, medicii sunt afișați pe baza locației de care aparțin, pentru persoanele care doresc să aleagă în funcție de cea mai apropiată locație.

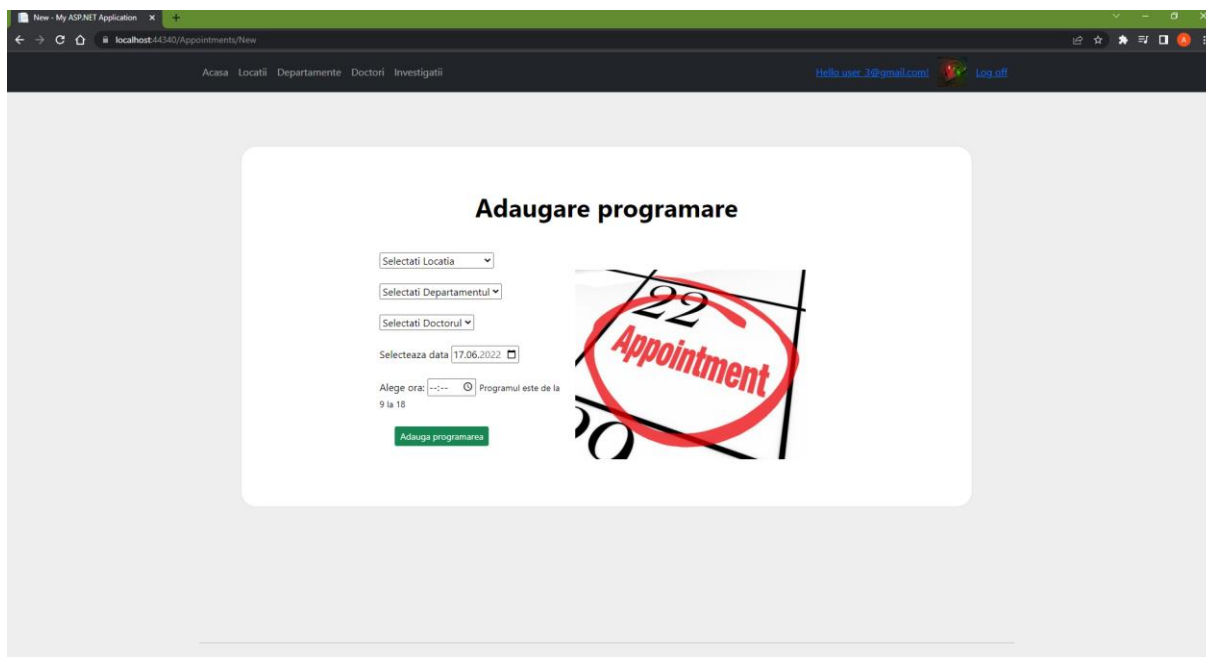
Utilizatorii au posibilitatea de a căuta un doctor folosind bara de căutare aflată pe prima pagină. De asemenea, ei au posibilitatea de a vedea profilul unui doctor, pentru a se convinge dacă el este soluția problemelor lor.

În momentul afișării profilului, utilizatorii vor vedea numele complet, numele de utilizator, email-ul, numărul de telefon (dacă există), descrierea și poza sa de profil. De asemenea vor vedea ce tip de doctor este și cabinetul unde efectuează programări. Din această pagină utilizatorii au posibilitatea de a solicita o programare, apăsând pe butonul Solicită programare, care îi redirecționează către pagina de Programare nouă.

4.6.4 Programări

Un utilizator poate solicita o programare la un anumit doctor prin completarea unui formular asemănător celor de autentificare și înregistrare. Acesta este nevoit să selecteze locația, departamentul și doctorul din liste derulante, apoi să completeze descrierea programării, care este opțională dar poate fi de ajutor, atât pentru doctor să se pregătească, cât și pentru pacient pentru a își evalua evoluția. Nu în ultimul rând, utilizatorul este nevoit să aleagă o dată și o oră dintre cele disponibile pentru a finaliza programarea.

În momentul în care programarea este realizată, aceasta apare în lista de programări ale utilizatorului. Fiecare programare este însoțită de status, care este calculat pe baza datei curente. Dacă data curentă este după cea a programării, programarea are statusul finalizata, iar în cazul în care este în viitor, ea are statusul viitoare.



4.14 Ecranul de adăugare a unei programări văzut de utilizator

4.6.5 Jurnalul

Un utilizator autentificat are posibilitatea de a își ține evidența simptomelor și a stării de sănătate prin adăugarea unui jurnal. Lista de elemente adăugate este privată, utilizatorul fiind singurul care are acces la ea.

Astfel, jurnalul oferă atât siguranță, cât și confidențialitate. Pentru a adăuga o nouă intrare, utilizatorul poate accesa panoul de jurnal din meniul principal, apoi poate adăuga folosind butonul Adaugă o nouă intrare. Fiecare intrare este afișată sub formă de card, iar lista este ordonată cronologic, de la cea mai recentă până la cea mai veche.

De asemenea, diferit față de restul elementelor accesibile utilizatorului, în acest caz există posibilitatea ștergerii și modificării unei pagini de jurnal. În momentul afișării unei instanțe a jurnalului, pe pagina apărută se regăsesc și butoanele pentru ștergere și modificare.

4.7 Doctorul

Comparativ cu administratorul și utilizatorul, medicul are o gamă mai mică de acțiuni pe care le poate face. Acesta își poate edita și vedea profilul, poate vedea programările sale și departamentul și locația din care face parte.

Meniul este puțin diferit față de restul utilizatorilor, conține pagina de editare profil, pagina de afișare profil, lista programărilor și proprietățile cabinetului și locației din care face parte. Spre deosebire de restul, el nu poate vedea listele cu departamente sau locații.

4.7.1 Profilul

Un medic își poate vedea și edita oricând profilul, cu mențiunea că nu poate modifica departamentul, cabinetul sau rolul care îi sunt asociate. De asemenea, el nu își poate șterge profilul.

Pentru a edita profilul este folosită o vizualizare asemănătoare cu cele de până acum, dar adaptată pentru a integra câmpurile potrivite.

4.7.2 Cabinetul

În momentul în care un doctor este adăugat, acesta este asociat la o anumită adresă și specializare. Utilizatorul cu rol de medic are posibilitatea de a vedea propriul departament, investigațiile care aparțin de acesta și locația la care lucrează, dar nu le poate vedea pe restul și nu se poate asocia la alt departament sau alt cabinet.

Investigațiile sunt afișate sub formă de tabel, dar fără posibilitatea de a adăuga sau a le modifica. Medicul le poate doar vedea.

4.7.3 Programările

Un medic poate vedea lista cu toate programările pe care urmează să le efectueze sau pe care le-a efectuat deja și are posibilitatea să editeze descrierea lor pentru a fi mai ușor de urmărit. Programările sunt sortate în funcție de data curentă în programări trecute și programări viitoare.

În cazul în care programarea este în viitor, doctorul o poate anula. În momentul anulării, programarea dispare atât din lista utilizatorului, cât și din cea a medicului, iar utilizatorul este notificat printr-un email trimis automat.

Capitolul 5. Concluzie

Luând în considerare cele prezentate de-a lungul acestei lucrări, aplicația oferită constituie o modalitate inovativă de a rezolva problemele clasice ale sistemului medical, anume dificultatea obținerii informațiilor necesare la momentul potrivit, imposibilitatea păstrării evidenței programărilor și simptomelor într-un mod digital și accesarea lor oricând și, mai ales, centralizarea tuturor serviciilor oferite de un anumit cabinet medical.

Având un nivel de personalizare foarte ridicat, aceasta se mulează pe nevoile oricărui lanț de servicii medicale. Listele pentru specializări, pentru sedii și pentru investigații sunt cu totul editabile și pot fi gestionate de către utilizatorii cu rol de administrator. Nu există un număr limită de utilizatori de un anumit tip, dar, cu toate acestea, la prima rulare a aplicației este creat un utilizator de bază, cu rol de administrator. După această accesare, utilizatorul creat poate fi modificat în același mod ca restul, chiar dacă a fost generat automat.

În momentul adăugării unui cont nou, acesta este creat cu rolul de utilizator, dar el poate fi modificat de către administrator în doctor sau, la rândul său, în administrator.

Pe lângă editarea de roluri, administratorul poate edita locații existente, departamente și investigații, dar poate și crea unele noi. Astfel, poate adapta datele din aplicație în funcție de cerințele utilizatorilor.

Pe partea de pacient, ea își propune să ofere o experiență cât de plăcută posibil. În acest sens, ei au acces la toate informațiile necesare unei persoane în căutare de ajutor medical, oferind lista completă de specializări, de analize și de cabinete existente. De asemenea, ei pot vedea lista de doctori dar și profilul acestora, pentru a se convinge că au ajuns unde trebuie.

Un mare plus pe care aplicația îl oferă este posibilitatea ținerii unui jurnal, în care o persoană poate ține evidența evoluției sale în mod privat, fiind singura care are acces la această informație.

În categoria funcționalităților importante se regăsește și posibilitatea solicitării unei programări fără a fi nevoie de un apel telefonic. Un utilizator poate cere o vizită la un anumit doctor, moment în care ea va apărea atât în lista pacientului cât și în cea a doctorului.

Utilizatorul de tip doctor poate ține evidența pacienților proprii prin intermediul vizitelor salvate în lista de programări. De asemenea, el poate anula o solicitare, iar în acest caz pacientul primește un mail prin care este anunțat că programarea nu mai este disponibilă.

În concluzie, date fiind nivelul de personalizare al aplicației, accesibilitatea ei și funcționalitățile oferite, aceasta poate fi considerată o soluție ideală pentru gestiunea oricărui

tip de cabinet, fie că face parte dintr-un lanț sau nu, fie ca oferă mai multe specializări sau este un sediu individual. Alegerea creării unei platforme de tip web este una potrivită datorită destinației acesteia, având un public țintă numeros și dorind să ajungă la o varietate cât mai mare de persoane, fără a întâmpina probleme legate de actualizări și fără a necesita descărcarea acesteia.

Bibliografie

- [1]. R.R.Conteras, Covid-19 and digitalisation, <https://www.eurofound.europa.eu/data/digitalisation/research-digests/covid-19-and-digitalisation> . Data accesării: 11 iunie 2022
- [2]. Soluții digitale în timpul pandemiei, https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/digital-solutions-during-pandemic_ro . Data accesării: 11 iunie 2022
- [3]. How COVID-19 has pushed companies over the technology tipping point- and transformed business forever, <https://www.mckinsey.com/business-functions/strategy-and-corporate-finance/our-insights/how-covid-19-has-pushed-companies-over-the-technology-tipping-point-and-transformed-business-forever> Data accesării: 11 iunie 2022
- [4]. Ada app description, <https://ada.com/app/>, Data accesării: 11 iunie 2022
- [5]. Medlife platform, <https://www.medlife.ro/>, Data accesării: 11 iunie 2022
- [6]. Regina Maria aplicație web, <https://www.reginamaria.ro/>, Data accesării: 11 iunie 2022
- [7]. A brief history of web app, <https://oleg-uryutin.medium.com/a-brief-history-of-web-app-50d188f30d>, Data accesării 11 iunie 2022
- [8]. Advantages and disadvantages of web app development, <https://en.yeeply.com/blog/advantages-and-disadvantages-of-web-app-development/>, Data accesării 11 iunie 2022
- [9]. TutorialsTeacher, <https://www.tutorialsteacher.com/mvc/mvc-architecture>, Data accesării: 12 iunie 2022
- [10]. Asp.net Documentatie oficială, <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview>, Data accesării 12 iunie 2022
- [11]. W3Schools, https://www.quanzhanketang.com/aspnet/mvc_folders.html, Data accesării 12 iunie 2022
- [12]. Asp.net Entity Framework, documentația oficială, <https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application>, Data accesării 12 iunie 2022
- [13]. C# Corner, <https://www.c-sharpcorner.com/article/asp-net-mvc5-entity->

- [framework-database-first-approach/](#), Data accesării 12 iunie 2022
- [14]. Stackoverflow, <https://stackoverflow.com/questions/35527170/how-do-i-save-related-data-using-entity-framework-asp-net-mvc5>, Data accesării 12 iunie 2022
- [15]. Pluralsight, <https://www.pluralsight.com/guides/asp-net-mvc-getting-default-data-binding-right-for-hierarchical-views>, Data accesării 12 iunie 2022
- [16]. Asp.net Identity framework documentație oficială, <https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity>, Data accesării 13 iunie 2022
- [17]. Asp.net Identity Framework documentație oficială tutorial, <https://docs.microsoft.com/en-us/aspnet/mvc/overview/security/create-an-aspnet-mvc-5-web-app-with-email-confirmation-and-password-reset>, Data accesării 13 iunie 2022
- [18]. Bootstrap documentație oficială, <https://getbootstrap.com/>, Data accesării 13 iunie 2022
- [19]. Material design for Bootstrap, <https://mdbootstrap.com/>, Data accesării 13 iunie 2022
- [20]. Tutorial Republic, <https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/>, Data accesării 13 iunie 2022