MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PLATFORMĂ PENTRU MANAGMENTUL ACTIVITĂȚILOR MEDICALE DIN CADRUL UNEI POLICLINICI

LUCRARE DE LICENŢĂ

Absolvent: **Casiana Ștefana CÂMPEAN**

Coordonator ştiinţific: **Asis. Ing. Cosmina IVAN**

2020

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Casiana Ștefana CÂMPEAN**

PLATFORMĂ PENTRU MANAGMENTUL ACTIVITĂȚILOR MEDICALE DIN CADRUL UNEI POLICLINICI

1. **Enunţul temei:** *Proiectul își propune dezvoltarea unei platforme online medicale pentru managementul unei policlinici. Aplicația va îmbunătăți comunicarea între medic și pacient, modul de găsire a medicilor si a serviciilor medicale, sistemul de programări, modul în care se păstrează evidența istoricului unui pacient și gestiunea datelor despre medici, specializări și servicii medicale.*

2. **Conţinutul lucrării:** *Pagina de prezentare, Introducere, Obiectivele proiectului, Studiu bibliografic, Analiză și fundamentare teoretică, proiectare de detaliu și implementare, Testare și validare, Manual de instalare și utilizare, Concluzii, Bibliografie.*

3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Tehnologia Informației

4. **Consultanţi**: Asis. Ing. Cosmina Ivan

5. **Data emiterii temei:** 1 februarie 2020

6. **Data predării:** 7 septembrie 2020

Absolvent: Casiana Ștefana Câmpean

Coordonator ştiinţific: Asis. Ing. Cosmina IVAN

Cuprins

Capitolul 1. Introducere – Contextul proiectului 1
1.1. Contextul proiectului 1
1.2. Motivația 1
1.3. Conținutul lucrării 2
Capitolul 2. Obiectivele Proiectului 4
2.1. Obiectivul principal 4
2.2. Obiective secundare 4
Capitolul 3. Studiu Bibliografic 6
3.1. Telemedicina 6
3.2. E-Health 6
3.3. Sistemele de online booking 7
3.4. Sisteme similare 9
3.4.1. DocBook 9
3.4.2. Veribook 11
3.4.3. Simple Practice 12
3.4.4. MyHealth1st 13
3.4.5. HealthEngine 13
3.4.6. Comparație sisteme 14
Capitolul 4. Analiză şi Fundamentare Teoretică 16
4.1. Cerințe funcționale 16
4.2. Cerințe non-funcționale 17
4.2.1. Securitatea 17
4.2.2. Performanța 17
4.2.3. Disponibilitatea 18
4.2.4. Utilizabilitatea 18
4.2.5. Extensibilitatea 18
4.2.6. Accesibilitatea 18
4.3. Cazuri de utilizare 18
4.3.1. Actorii sistemului 18
4.3.2. Descrierea detaliată a cazurilor de utilizare 20
4.4. Perspectiva tehnologică 27
4.4.1. Tehnologii back-end 27

4.4.2. Tehnologii front-end 28
Capitolul 5. Proiectare de Detaliu si Implementare 31
5.1. Arhitectura sistemului 31
5.1.1. Arhitectura conceptuală 32
5.1.2. Arhitectura componentei backend 33
5.1.3. Arhitectura componentei frontend 41
5.1.4. Diagrama de deployment 48
5.2. Structura bazei de date 48
Capitolul 6. Testare şi Validare 52
6.1.1. Testarea unitară 52
6.1.2. Testarea de integrare 53
6.1.3. Testarea de acceptanță 54
6.1.4. Testare de performanță 56
Capitolul 7. Manual de Instalare si Utilizare 58
7.1. Manual de instalare 58
7.2. Manual de utilizare 59
7.2.1. Funcționalități utilizatori 59
7.2.2. Funcționalități doctor și pacient 60
7.2.3. Funcționalități administrator 67
Capitolul 8. Concluzii 69
8.1. Contribuții personale și rezultate obținute 69
8.2. Dezvoltări ulterioare 69
Bibliografie 71

Capitolul 1. Introducere – Contextul proiectului

1.1. Contextul proiectului

Sănătatea reprezintă pentru fiecare om cel mai mare avut al său, aceasta fiind definită de Organizația Mondială a Sănătații în felul următor[1]: „Sănătatea reprezintă o stare de bine completă, fizică, psihică și socială, și nu reprezintă doar absența unei boli sau a unei infirmități". Fiecare dintre noi vrem să fim mereu sănătoși, iar în momentele în care ajungem să fim bolnavi ne dorim să fim tratați de cei mai buni medici. Procesul de găsire și alegere a unui medic nu este ușor, mulți dintre oameni se bazează de cele mai multe ori pe recomandări din partea apropiaților, dar acestea nu sunt mereu disponibile.

Aplicația își propune să ofere unui un utilizator o platformă prin care poate găsi cu ușurință orice medic în funcție de locație și specializare, recomandări și serviciul medical dorit. Odată ales medicul, utilizatorul poate vizualiza serviciile medicale oferite, prețurile, feedback-urile de la alți pacienți și poate să comunice direct cu medicul prin mesaje. Dacă va dori să realizeze o programare, poate urmări cu ușurință programul medicului și poate realiza o rezervare în funcție de disponibilitate. De asemenea, în aplicație poate să își gestioneze programul, mesagerie și poate să ofere la rândul lui feedback unui medic.

1.2. Motivația

Această aplicație a fost dezvoltată pentru a îmbunătății modul în care anumite procese din sistemul medical sunt realizate în prezent, precum programările medicale, comunicarea între medic și pacient, punerea la dispoziție a documentelor și a rezultatelor medicale, oferirea de informații legate de serviciile medicale, de costul acestora, de părerile altor pacienți cu privire la doctori și despre vizualizarea disponibilității unui doctor. În prezent aceste acțiuni pot implica mai multe dezavantaje, precum:

- **Realizarea unei rezervări poate costa mult timp**

Mulți dintre pacienți sunt nevoiți să sune pentru a realiza o rezervare, sau chiar să meargă până la cabinet. De multe ori se întâmplă să nu răspundă cineva la telefon, linia telefonică să fie ocupată. În unele cazuri, recepționerii nu vor putea răspunde sau vor fi plecați, astfel se vor pierde pacienți. Având o aplicație de rezervări online, fiecare pacient va putea intra în aplicație și va putea să își rezerve vizita la cabinet în funcție de programul sau și al medicului.

- **Realizarea unei rezervări nu este mereu disponibilă**

Datorită faptului că o programare poate fi făcută doar telefonic sau direct la clinică, pacientul are un program restrâns pentru realizarea unei programări. Astfel, aplicația fiind non stop disponibilă, orice pacient poate să se programeze oricând.

- **Neprezentarea pacienților la programări**

Unii dintre pacienți se întâmplă să uite de programarea făcută, astfel este necesară notificarea pacienților prin mail.

[1] https://www.who.int/about/who-we-are/constitution

- **Procesul de programări necesită angajați suplimentari**

Programările se fac de cele mai multe ori telefonic, iar pentru acest caz sunt necesari mai mulți angajați care să răspundă la telefon. Având un sistem de rezervari online, nu mai sunt necesare resurse umane care să se ocupe de clienți.

- **Identificarea dificilă a medicilor și a serviciilor oferite**

Se întâmplă ca o persoană să aibă o problemă medicală, dar nu cunoaște medicii disponibili, nu știe unde este mai bine să meargă și nu are cine să îi recomande un medic bun. Astfel, în sistemul propus există posibilitatea de a oferi feedback unui medic și a serviciului oferit. În ziua de azi, totul se bazează pe recomandări, iar un client va asculta de sfaturile și de experientă altor oameni care au mai fost în aceeași situaite că și ei.

- **Necunoașterea costurilor serviciilor medicale**

De multe ori un pacient își dorește să știe la ce sumă de bani să se aștepte să plătească când merge la medic, dar acesta află doar când se prezintă la clinică sau doar la finalul programării. Astfel, sistemul oferă transparență totală, toate prețurile serivicilor oferite există în aplicație.

- **Lipsa istoricului medical**

După o vizită la cabinet, în general primești un rezultat medical, de cele mai multe ori acesta fiind o foaie sau o radiografie. Dacă pacientul nu își organizează propriile documente, cel mai probabil ajunge să le rătăcească, astfel cu ajutorul aplicației există posibilitatea de a descarcă documentele dorite, pacientul mereu având acces la istoricul sau medical.

- **Comunicarea directă între pacient și doctor are lipsuri**

De multe ori ca pacient ai dori să afli informații directe de la un doctor, dar acest lucru poate dura mult timp. Astfel, sistemul oferă posibilitatea de trimitere de mesaje în aplicație, de a pune întrebări printr-un sistem de mesagerie, de a îmbunătăți comunicarea între pacient și doctor prin existența acestei funcționalități.

1.3. Conținutul lucrării

Primul capitol prezintă contextul lucrării de licență, conturarea domeniului exact al temei și motivația acesteia.

Al 2-lea capitol prezintă tema proriu-zisă în care sunt prezentate obiectivul principal, precum și obiectivele secundare.

Al 3-lea capitol conține informații despre studiul bibliografic în care este descris stadiul actual al domeniului în care este situată tema. De asemenea sunt prezentate sistemele similare, avantajele și dezavantajele acestora și o comparație între ele și propriul sistem.

În al 4-lea capitol sunt explicate principiile funcționale ale aplicației implementate, sunt prezentate cerințele funcționale ale aplicației, actorii sistemului și tehnologiile folosite.

Capitolul 5, cel de proiectare de detaliu și implementare indentifică funcțiile principale și conține arhitectura sistemului, arhitectura componenței back-end și front-end, a bazei de date.

Capitolul 6 cuprinde detalii despre testarea și validarea proiectului, precum și detalii despre tehnologiile de testare alese.

În capitolul 7 este prezentat manualul de instalare și utilizare în care sunt detaliate resursele necesare pentru instalarea și rularea aplicației, precum și despre descrierea pas cu pas a procesului de instalare.

Capitolul 8 prezintă concluziile sistemului care conține rezumatul contribuțiilor proprii, analiza critică a rezultatelor obținute și o descriere a posibilelor dezvoltări și îmbunătățiri ulterioare ale proiectului.

Capitolul 2. Obiectivele Proiectului

2.1. Obiectivul principal

Proiectul are ca scop dezvoltarea unei platforme medicale destinate unei policlinici care are mai multe subdiviuni în mai multe orașe. Această platformă va fi folosită de către medici și pacienți și are ca obiectiv principal îmbunătățirea modului în care se realizează comunicarea între aceștia, procesul prin care se desfășoară sistemul de programări și de asemenea modul în care se păstrează evidența istoricului unui pacient. De asemenea, această aplicație va fi disponibilă unor administratori care au ca și scop gestiunea datelor despre medici, specializări și servicii medicale.

2.2. Obiective secundare

Ca și obiective secundare, sistemul dorește să ofere soluții pentru neajunsurile din sistemul medical. Aceste soluții sunt următoarele:

- **Programarea medicală se va realiza mult mai eficient**

Pentru a realiza o programare, de cele mai multe ori trebuie să suni sau să te prezinți la locul dorit și să te programezi. Acest mod poate costă mult timp, în unele cazuri trebuie făcute mai multe apeluri, trebuie întrebat doctorul când este disponibil, astfel tot acest proces ajunge să fie foarte ineficient. Modul de programări al sistemului propus este online, orice client poate să vizualizeze programul oricărui medic și se poate programa la ora dorită și disponibilă fără a mai fi nevoie de alte detalii.

- **Realizarea unei rezervări va fi mereu disponibilă**

Daca o programare se realizează telefonic sau direct la o clinică, fiecare dintre aceste variante dispune de un program limitat. Astfel, sistemul fiind mereu disponibil, orice utilizator va putea naviga prin aplicație și va putea să realizeze o rezervare oricând, fără a depinde de programul clinicii.

- **Gestionarea neprezentării pacienților la programări se va îmbunătăți**

Se întâmplă în unele momente ca pacienții să nu se prezinte la programarea făcută din diverse motive, astfel sistemul implementează funcționalitatea de notificare a clienților prin email cu o perioada de timp înainte de rezervarea lor.

- **Realizarea unei rezervări nu va necesita angajați suplimentari**

Pentru a te programa la un servicu medical, este necesar să efectuezi un apel telefonic, iar persoanele care răspund la telefon sunt angajate special pentru acest serviciu, astfel fiind necesare resurse umane suplimentare. Acest lucru nu va mai fi necesar deoarece proiectul propus dispune de un sistem online de rezervări pentru care nu este necesară angajarea altor persoane.

- **Cunoașterea medicilor și a serviciilor acestora va fi mult mai facilă**

Nu orice persoană care are o problemă medicală cunoaște informații despre doctorii existenți și nu are persoane apropiate care să îl ajute cu o recomandare. Astfel, în aplicație există posibilitatea de a vizualiza și a oferi feedback doctorilor. Prin acest proces, viitorii pacienți vor putea alege un doctor mult mai ușor prin intermediul acestor recomandări direct din aplicație.

- **Orice pacient va avea informații despre costurile serviciilor medicale**

De multe ori se întâmplă să nu știi concret prețul serviciului medical dorit, nu te poți informa întotdeauna asupra acestui detaliu, pacientul aflând doar mergând la o clinică sau află la finalul programării costul serviciului. Astfel, sistemul oferă transparență totală, toate prețurile serviciilor oferite există în aplicație.

- **Pacientul va avea mereu acces la istoricul medical**

În urmă unei programări medicale, rezultate primite, fie ele analize, radiografii sau alte diverse documente medicale, sunt de cele mai multe ori documente fizice, pe foaie sau CD. Fiecare dintre aceste documentele medicale trebuie păstrate cu grijă de pacienți într-un singur loc, dar acest mod nu este mereu sigur, unele documente ajungând să fie peirdute. Din această cauză, sistemul oferă un istoric medical digital unde toate rezultatele medicale vor fi centralizate și securizate.

- **Comunicarea dintre pacient și doctor va fi îmbunătățită**

În unele momente îți dorești să afli informații suplimentare de la medici, dar acest lucru se poate întâmplă doar în momentul în care ești față în față cu aceștia, iar pentru o a afla răspunsul la o singură întrebare trebuie să aștepți mult timp. Din acest motiv, în aplicație poți comunica direct cu medicul prin intermediul unei mesagerii, unde poți afla răspunsuri mult mai rapid la orice întrebare medicală.

Capitolul 3. Studiu Bibliografic

În acest capitol se vor detalia următoarele concepte care contribuie la înțelegerea mecanismelor implementate la nivelul platformei propuse: telemedicină, sănătatea digitală și sistemele de online booking. Pe lângă acestea, se vor prezența mai multe sisteme similare și se va realiza o comparație între acestea și sistemul propus.

3.1. Telemedicina

Telemedicina reprezintă o distribuire a mai multor servicii de medicină care sunt realizate prin intermediul tehnologiei si a telecomunicațiilor. Aceasta permite conexiunea la distanță dintre pacient si doctor, în special pentru servicii de informare, diagnosticare și monitorizare.

Domeniul telemedicinei a apărut odată cu apariția telecomunicației, în special pentru a soluționa problema pacienților care se aflau în zone izolate. Fiecare specialist în parte poate oferi servicii medicale fără contact direct cu pacientul, poate trimite recomandări de diagnostic şi tratament înapoi la cel care l-a solicitat. În felul acesta pot avea acces la specialişti inclusiv persoane care nu se pot deplasa, dar şi cei care se află în zone izolate.

În prezent, există cel puțin trei modele operaționale unde telemedicina poate fi implementată. Conform articolului spitalului "Dr. Carol Davila"[2], unul dintre aceastea reprezintă stocarea si trimiterea a tot ceea ce înseamnă documente medicale, dosare, fotografii, imagistică, analize și tot ceea ce poate fi rezultat medical palpabil. Toate aceste documente pot fi trimise de la pacient la medic, de la medic la un pacient sau de la medic către un alt medic. Această comunicare este una asincronă, exepeditorul și destinatarul nu trebuie să fie prezenți în același timp. Totul se poate derula pe platforme de comunicare sigure, unde confidențialitatea datelor este strict pastrată.

Un alt model semnificativ reprezintă monitorizarea la distanță. Aceasta este foarte practică, economică și mai eficientă decât vizita la cabinet și pentru medic si pentru pacient. De asemenea, o altă variantă reprezintă telemedicina interactivă, acesta implicând o interacțiune sincron, în timp real între pacient și medic prin tehnologiile audio și video.

3.2. E-Health

E-Health [1] sau sănătatea digitală referă la toate formele de asistență medicală electronice, transmise prin intermediul internetului, variind de la servicii medicale precum monitorizarea la distanță, transmiterea de informații medicale sau de documente medicale, senzori de monitorizare și multe altele. E-Health cuprinde o mare varietate de activități clinice care au caracteriat telesanatatea, dar care sunt transmise prin intermediul internetului. Prin acest mod, E-Health realizează asistența medicală mult mai eficientă.

Sănătatea digitală este o formă relativ nouă a medicinei, suportată de procesele si comunicarea electronică. Pe parcursul istoriei, informatica a fost o parte importantă a medicinei, făcând posibilă stocarea și accesarea a multor date și informații medicale. Astfel s-a ajuns la aparația sănătății digitale, a Electronic Health-ului. Din cauza internet-

[2] http://www.scumc.ro/telemedicina-medicina-viitorului/

ului, s-au creat noi oportunități și provocări de dezvlotare a medicinei și a interacțiunii dintre medici și pacienți. Principalele provocari ale industriei medicale conform articolului [1] au fost:

1. Capabilitatea consumatorului să interacționeze direct cu sistemele online (B2C – Bussiness to Consumer)
2. Posibilități avansate de transmitere a datelor dintre instituții (B2B – Bussiness to bussiness)
3. Noi posibilități de comincare între consumatori (C2C – Consumer to Consumer)

Printre formele de e-sănătate se numără de exemplu furnizarea de informații către pacienți sau medici, schimbul de informații sau date între părți, cu o reacție directă a partenerului de comunicare. Totodată, un alt serviciu reprezintă înregistrarea pe toată durata vieții a tuturor datelor deținute de un pacient cu privire la starea lui de sănătate. Astfel, prin fuzionarea tuturor datelor medicale și parmedicale și completarea informațiilor de sănătate ale pacientului se realizeaza o carte de sănătate în format electronic.

De asemenea, conform articolului [1] printre principalele avantaje ale sănătății digitale se numără:

- Eficiența – datorită comunicării electronice, crește eficiciența, se reduce timpul și costurile, mai ales prin lipsa nevoii de întalnire directă înre medic și pacient.
- Calitatea serviciilor medicale – datorită creșterii eficienței, crește astfel și calitatea serviciilor medicale. Spre exemplu, sănătatea digitală oferă posibilitatea comparării serviciilor medicale și îmbunătățirea acestora cu ajutorul pacienților, aceștia fiind o sursă în plus de a asigura calitatea.
- Educația – prin intermediul E-Health-ului, se dezvoltă sursele pentru educarea pacienților si pentru informarea acestora cu privire la tot ceea ce ține de domeniul medicinei.

3.3. Sistemele de online booking

Un sistem de online booking reprezintă un sistem software care îi permite unui client să rezerve cu ușurintă servicii care necesită programări. Sistemul de programări este un lucru important în orice tip de organizație sau companie care necesită program cu clienții. Acesta a evoluat pe parcursul anilor, de la a realiza programări prin telefon, după care sa fie notate pe hartie, până la a face programări folosind calendare eletronice, precum Google Calendar sau Microsoft Outlook. Aceeași acțiune continuă sa fie una care costă mult timp și resurse.

Pe parcursul timpului, odată cu aparația tehnologiei rezervărilor online, s-a simplificat acest proces pentru un număr mare de organizații. Astfel, aceste sisteme accesibile și ușor de folosit, au schimbat modul în care un client realiza o programare. Conform articolelor [2] și [3] nevoia acestora a aparut din cauza multor dezavantaje, precum:

- Nevoia de resurse umane pentru realizarea programărilor – Modul de a realiza o programare fără un sistem software necesită resurse umane, fie că este nevoie de un recepționist sau un administrator care gestionează programările pe hartie. Pentru acest tip de realizare a programărilor manual sunt pecesare costuri în plus pentru agajați.
- Timpul îndelungat pentru gestionarea programărilor – Datorită faptului că procesul de programări este realizat manual, acesta necesită timp. Chiar dacă în unele cazuri o programare poate dura mai puțin, în cele mai multe cazuri aceasta necesită timp deoarece trebuie verificat intervalul orar disponibil, în anumite situații trebuie apelat la alte persoane, de exemplu se apelează la un medic pentru confirmarea programării, iar un alt factor reprezintă indisponibilitatea persoanei care gestionează programările, dacă se realizează alte rezervări și toate liniile telefonice sunt ocupate, un client va petrece mult timp să finalizeze o programare.
- Orele limitate pentru realizarea programărilor - De cele mai multe ori, un client este limitat să realizeze o rezervare deoarece programul pentru rezervări se poate face doar într-un anumit interval orar. Astfel, un sistem online ar putea rezolva această problemă prin faptul ca ar fi disponibil non-stop.
- Nerespectarea programărilor – În multe dintre cazuri se întamplă ca și clienții care au realizat o programare să nu se mai prezinte la aceasta, unul dintre motive ar putea fi faptul că a uitat de aceasta sau că a uitat detaliile acesteia, astfel printr-un sistem de rezervări online, cu ajutorul notificărilor. Există cazuri în care pot exista notificări pentru programări, dar din nou, acestea ar necesita resurse umane.

Astfel, conform articolului [2], apariția acestor sisteme au adus mai multe beneficii, precum:

- Realizarea de programări la orice oră – Prin accesul clienților non-stop pentru a realiza progrări, se caștigă mult timp pentru angajații unei instituții, deoarece aceștia nu trebuie să se mai ocupe de acest proces.
- Notificări automate – Pentru a rezolva problema clienților care nu se prezintă la programări, a apărut ca și soluție notificările, iar daca în trecut acest proces trebuia realizat manual și necesita multe resurse, printr-un sistem sofware, acestea vor putea fi trimise automat.
- Posibilitatea de stocare a datelor și de realizare de rapoarte – Datorită faptului că înainte sau după o programare apar anumite informații sau detalii despre clienți, prin existența unui sistem online, toate aceste date pot fi stocate direct acolo. De asemenea, cu toate informațiile despre programări, se pot genera diferite rapoarte și statistici.
- Opțiuni de plată online – Deoarece majoritatea serviciilor pentru care se face programarea sunt pe bani, instituțiile au optat pentru posibilitatea de a realiza plata după ce realizezi programarea. Un mare avantaj ar fi scăderea cazurilor în care clienții nu s-ar prezenta la programare și de asemenea simplificarea procesului de plată la fața locului.

3.4. Sisteme similare

In acest subcapitol se vor prezenta mai multe sisteme similare, precum: DocBook, Veribook, Simple Practice, MyHealth1st și HealthEngine. De asemenea, se va realiza o comparație între acestea și sistemul propus.

3.4.1. DocBook

DocBook[3] este o platformă dezvoltată pentru mai multe cabinete din mai multe orașe din România. Această aplicație a fost lansată în anul 2017 și dorește să faciliteze si să simplifice situația programărilor din România. Prin acest sistem, orice persoană se poate programa la un medic în mai puțin de un minut. Conform reportajului, tocmai fondatorul aplicației DocBook a afirmat faptul că "Mi-am spus că sigur în situația mea mai sunt cel puțin câteva sute de oameni, care și-ar dori să se programeze la medic în maxim 1 minut, de pe telefonul mobil sau de pe web", și de aici a pornit această idee, dorința de a realiza o soluție pentru o problemă care apare la nivel de țară.

DocBook are următoarele funcționalități:

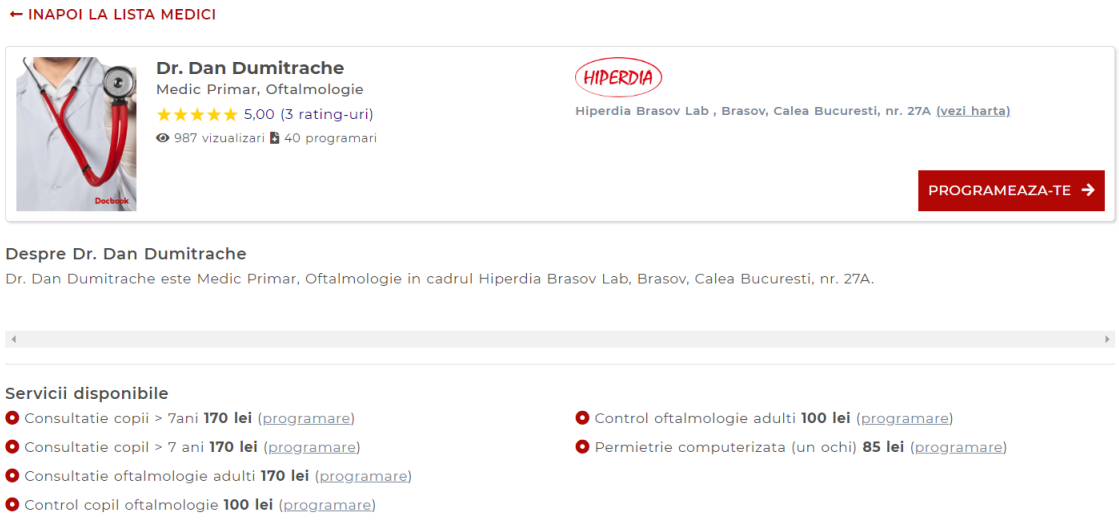
- utilizatorii au acces la informații despre medici, la competențele lor profesionale și să consulte servicile și despre costul acestora, optiuni ilustrate in figura 3.1
- utilizatorii pot realiza programări pentru serviciul unui anumit doctor și va fi afișată prima dată disponibilă și mai multe opțiuni pentru ore din data respectivă, functionalitatea fiind ilustrata in figura 3.2
- Sistemul este conectat non-stop la toate sistemele de programare ale clinicilor și cabinetelor medicale partenere și permite căutarea după nume, localitate, specialitate, dată și forma de plată acceptată.
- sistemul afișează medicii și clinicile ordonate după geolocație, și de asemenea poți vedea locația fiecărei clinici apăsând pe butonul „vezi harta"
- Orice programare poate fi anulată
- După orice programare ai posibilitatea de a oferi feedback doctorului, astfel îmbunătățind increderea pentru viitorii pacienți.
- Se pot realiza filtre pentru medici și clinici după locație, specialitate și se pot căuta rezultate dupa numele doctorului sau al clinicii
- Plată online
- Reduceri pentru servicii

Consider că această aplicație aduce un mare plus pentru procesul de realizare a programărilor medicale în România. Ca și avantaje se numără următoarele:

- Eficiența – datorită schimbarii procesului de rezervări manual, în unul automat, procesul de vine unul mult mai eficient și mai puțin costitisitor din punct de vedere al timpului pentru orice viitor pacient în căutare de un serviciu sau un medic
- Disponibilitatea – faptul că totul este online, pe web sau pe telefon, oferă acces nelimitat pentru a realiza orice tip de programare la orice oră și pentru a consulta informațiile dispobile în aplicație, cum ar fi date despre medici, clinici, servicii și locații

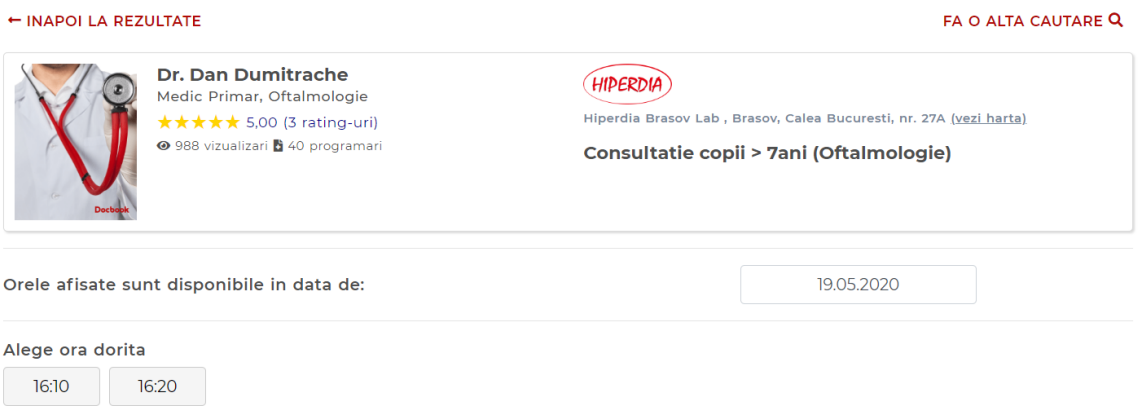
[3] https://www.docbook.ro/

- Ușurința procesului de rezervare – Ca și intefață grafică, aplicația este una user-friendly, ușor de folosit, procesul de găsire a unui medic sau al unui serviciu este foarte ușor, există multe filtrări după locație, specialitate, clinică și multe altele
- Transparența – Această aplicație oferă multe informații legate de medici, de compentențele lor profesionale, informații legate de clinici, de localizare și despre servicii și specialități, și de asemenea despre costurile ficarui serviciu și despre ofertele disponibile
- Opțiunea de localizare a clinicii – un avantaj foarte mare reprezintă posibilitatea căutării unei clinici sau a unui medic după locația actuală sau după un anumit oraș din România. Totodată, există și opțiunea de localizare exactă a locației dorite, care te redirecționează către google maps și ai posibilitatea de acolo să ajungi direct la clinica dorită.
- Securitatea datelor – conform sistemului docbook, la partea de confidențialitate[4], aceștia menționează faptul că datele colectate "vor fi accesate numai de catre salariatii, contractorii si partenerii Prestatorului, precum si ai Clinicilor/Policlinicilor partenere la care doriti sa efectuati o programare pentru un control medical, numai in scopul furnizarii Serviciilor." și asigură faptul că datele nu sunt folosite în alt scop și că acestea vor fi stocate atâta timp cât utilzatorul va avea un cont în aplicație



Figură 3.1 Screenshot pagină detalii doctor site Docbook[5]

[4] https://www.docbook.ro/home/confidentialitate
https://www.docbook.ro/Search/DoctorResult?doctorId=15699&serviceId=675&searchDate=19.05.2020&locationId=115



Figură 3.2 Screenshot detalii programare consultație oftalmologică[6]

3.4.2. Veribook

Veribook[7] este un sistem sofware realizat pentru programările online, destinat întreprinderilor mici, precum cabinete medicale.

Funcționalitățile pentru pacienți sunt următoarele:

- rezervări online în timp real
- posibilitatea de a vedea datele si orele disponibile și să realizeze rezervarea dorită
- utilizatorii pot să primească notificări pentru a i se aminti de o anumită programare.

Ca și funcționalități pentru clinici există:

- aplicația oferă posibilitatea de a publica toate orele de lucru existente intr-un calendar, disponibile pentru toți utilizatorii
- fiecare program al unui doctor si fiecare perioadă în care acesta este disponibil sau nu este accesibil
- se oferă un sistem flexibil care se adaptează pentru orice tip de cabinet și pentru orice tip de practici ale acestuia, de exemplu cât de mult în avans poate fi realizată o programare.
- Nu în ultimul rând, aplicația oferă un sistem de plată online pentru orice programare.

Acest sistem oferă următoarele avantaje:

- Orice pacient va fi notificat pentru programăriile sale, astfel nu vor mai fi cazuri în care pacienții nu se prezintă la programare
- Sistemul de plată se poate realiza online, astfel nu va mai trebui ca angajații să se ocupe de acect lucru. Pe lângă acest fapt, la fel ca și avantajul de mai sus, se va rezolva problema pacienților care nu se prezintă la programări, deoarece deja au plătit pentru serviciul medical
- Programările se realizează online, astfel orice client va salva mai mult timp, procesul fiind mult mai eficient

[6] https://www.docbook.ro/Search/DoctorResult?doctorId=15699&serviceId=675&searchDate=19.05.2020&locationId=115

[7] https://www.capterra.com/p/124063/Veribook/#features

3.4.3. Simple Practice

Simple practice[8] este un sistem sofware dezvoltat pentru cabinetele medicale private care se ocupă cu rezervările online.

Acest sistem oferă diverse funcționalități pentru orice instituție medicală privată precum:

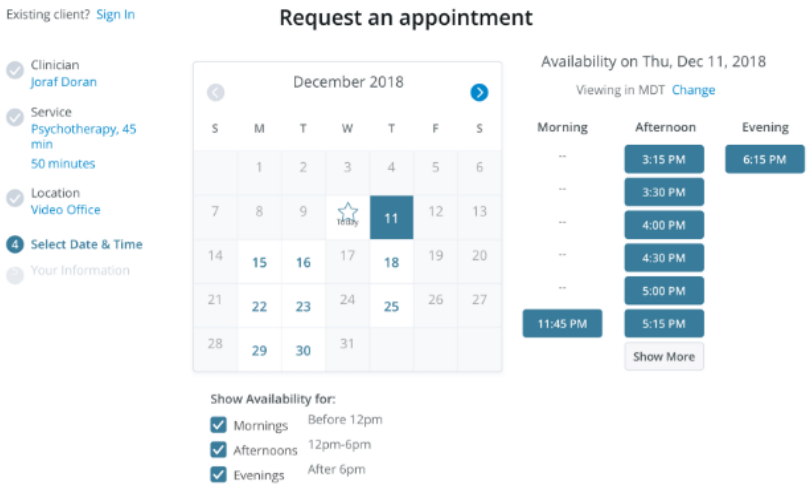
- acces la orice programare facută de utilizatori și la orice mesaj oferit de pacienți pentru programările lor
- sistemul elimină nevoia de a avea documente fizice, totul fiind online, toate aceste documente fiind securizate
- sistemul facilitează comunicarea dintre pacient si doctor printr-un portal online, mesajele sau întalnirile video fiind securizate
- Utilizatorii primesc mesaje și email-uri pentru programările sale și clientul poate alege modalitatea de plată, aceasta putând fi făcută direct în aplicație

De asemenea, câteva exemple de funcționalități pentru clienți sunt:

- Posibilitate de a realiza programări online în timp real, opțiune ilustrată în figura 3.4
- Accesul la documente medicale și încărcarea acestora în sistem
- Posibilitatea de comunicare directă cu medicul sau clinica

În opnia mea, câteva dintre avantajele oferite de acest sistemul sunt următoarele:

- Accesul la programările online este foarte ușor, astfel modul de realizare a unei programări va fi mult mai rapid și eficient pentru orice client
- Plata se poate realiza online, pacienții pot primi facturi direct în aplicație, astfel sistemul de platădevine mult mai eficient deoarece nu este necesar ca și un angajat să se ocupe de acest lucru
- Pacientul va putea contacta mult mai rapid medicul datorită existenței unui sistem de mesagerie securizat



Figură 3.3 Exemplu de realizare a unei programări pentru Simple Practice[9]

[8] https://www.capterra.com/p/130710/SimplePractice/

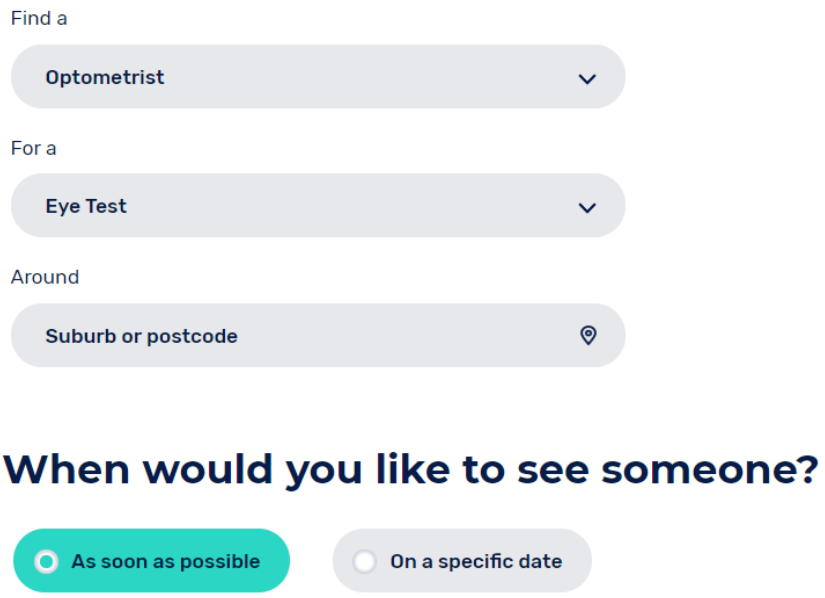
3.4.4. MyHealth1st

MyHealth1st [10] este o platformă online pentru programări de servicii medicale din Australia. Acest sistem facilitează și simplifică interacțiunea dintre pacienți și doctori. Acesta sistem de rezervări este un sistem automat, datele programărilor sunt în timp real, sistemul fiind disponibil non stop. Sistemul este accesibil și de pe mobil.

Aplicația oferă multe funcționalități utilizatorilor, precum:

- posibilitatea de a alege perioada programării în funcție de disponibilitatea clinicii și a medicilor
- utilizatorul poate să aleagă doctorul la care dorește să se programeze
- se poate alege orice locație, oferindu-se opțiuni cu cele mai apropriate locuri în funcție de locație, opțiune ilustrata in figura 3.5
- utilizatorul are posibilitatea de a anula orice rezervare sau de a o modifica,
- odată ce se realizează rezervarea, aceasta se salvează și se poate vizualiza în calendarul pacientului și al doctorului

Consider că acest sistem oferă multe avantaje, cel mai important fiind ușurința găsirii unui doctor, a unei clinici sau a unui serviciu medical prin toate filtrele existente si în special pe baza locației.



Figură 3.4 Exemplu de realizare programare în aplicația MyHealth1st[10]

3.4.5. HealthEngine

HealthEngine [11] este o largă platformă pentru rezervări în domeniul medical. Sistemul ajută utilizatorii să găsească și să realizeze programări medicale diverși medici, precum dentiști, psihoterapeuți, chirugi si orice alt fel de medici.

[9] https://www.simplepractice.com/patient-portal
[10] https://www.myhealth1st.com.au/

[11] https://healthengine.com.au/

Acestă aplicație dispune de următoarele funcționalități:

- Existența unei căutari rapide pentru servicii medicale, doctori și clinici, informații bazate pe locație
- Odată găsită o clinică, sistemul oferă direcții pentru locația nou găsită
- Dacă se alege un serviciu medical, se va afișa o listă a tutror clinicilor, împreună cu date despre preț și disponibilitate
- Se pot realiza progrmări online, sistemul oferind posibilitatea să alegi medicul, data și ora în funcție de disponibilitate
- Când un utilizator realizează o programare, sistemul oferă cea mai apropiată dată și oră liberă

Sistemul cuprinde următoarele avantaje:

- Modul de identificare a unei clinici, medic sau serviciu medical este foarte ușor, sistemul oferind multe opțiuni de servicii sau specializări încă din prima pagină
- Orice clinică poate fi localizată cu ususurință datorită sistemului de navigare oferit de aplicație
- Aplicația este disponibilă și pentru mobil

Un dezavantaj al acestei aplicații reprezintă faptul că nu oferă înregistrare pentru un utilizator, astfel acesta trebuie să își introducă datele de fiecare dată când trebuie să realizeze o programare.

3.4.6. Comparație sisteme

În acest subcapitol se vor compara mai multe dintre sistemele similare descrise mai sus cu sistemul propus în funcție de mai multe criterii, precum operații utilizator, administrare conturi, notificări, rezervări, feedback și fișiere pacient.

Sisteme similare:

1. Veribook
2. Simple Practice
3. DocBook
4. Sistemul propus

Tabel 3.1 Comparație între sistemul propus și sistemele similare

#	Funcționalitate	1	2	3	4
	Modul operatii utilizator				
1	Autentificare	da	da	da	da
2	Inregistrare folosind Facebook	nu	nu	nu	da
3	Inregistrare folosind Google	da	da	da	da
4	Delogare	da	da	da	da
	Modul administrare conturi				
5	Adaugare, Editare, Stergere si vizualizare doctori	da	da	da	da
6	Adaugare, Editare, Stergere si vizualizare admini	nu	nu	nu	da
7	Adaugare, Editare, Stergere si vizualizare pacienti	da	da	nu	da
8	Vizualizare cont personal	da	nu	da	da
	Modul administrare servicii si specializari				
9	Adaugare, Editare, Stergere si vizualizare specializari	da	nu	da	da
10	Adaugare, Editare, Stergere si vizualizare doctori servicii doctor	da	da	nu	da
	Modul rezervari				
11	Adaugare rezervare	da	da	da	da
12	Stergere rezervare	da	da	da	da
13	Editare rezervare	da	da	nu	da
14	Vizualizare rezervari personale	da	da	da	da
15	Primire notificari prin mail pentru rezervari	da	nu	nu	da
	Sistem de plată	da	da	da	nu
	Modul feedback				
16	Adaugare feedback pentru doctor	nu	nu	da	da
17	Vizualizare feedback-uri	nu	nu	da	da
	Modul fisiere pacient				
18	Adaugare fisiere pacient (plan medical, radiografie, rezultate)	da	da	nu	da
19	Stergere fisiere pacient (plan medical, radiografie, rezultate)	da	da	nu	da
20	Vizualizare fisiere pacient	da	da	nu	da

Capitolul 4. Analiză şi Fundamentare Teoretică

În acest capitol se vor prezenta cerințele funtionale și non-funcționale ale aplicației, cazurile de utilizare pentru cei trei actori ai aplicației, mai exact administrator, doctor și pacient, se vor descrie în detaliu câteva dintre aceste cazuri. Nu în ultimul rând, se va prezența perspectiva tehnologică pentru componenta backend și frontend.

4.1. Cerințe funcționale

În tabelul 4.1 sunt prezentate toate cerințele funcționale ale aplicației fiind specificat pentru fiecare cerință care sunt actorii care pot realiza acea funcționalitate. Acestea sunt grupate în mai multe categorii, precum cele de administrare, adăugare, ștergere sau editare.

Tabel 4.1 Cerintele functionale ale sistemului

	Cerință funcțională	Utilizator
CF 1	Autentificare	Admin, Pacient, Doctor
CF 2	Delogare	Admin, Pacient, Doctor
CF 3	**Adăugare cont utilizator**	
CF 3.1	Adăugare cont pacient	Pacient
CF 3.2	Adăugare cont administrator	Admin
CF 3.3	Adăugare cont doctor	Admin
CF 4	**Ștergere cont utilizator**	
CF 4.1	Ștergere profil administrator	Admin
CF 4.2	Ștergere profil doctor	Admin, Doctor
CF 4.3	Ștergere profil pacient	Admin, Pacient
CF 5	**Editare cont utilizator**	
CF 5.1	Editare profil administrator	Admin
CF 5.2	Editare profil doctor	Admin, Doctor
CF 5.3	Editare profil pacient	Pacient
CF 6	**Vizualizare conturi utilizatori**	
CF 6.1	Vizualizare conturi administratori	Admin
CF 6.2	Vizualizare conturi doctori	Admin, Pacient
CF 6.3	Vizualizare conturi pacienți	Admin
CF 7	**Vizualizare cont utilizator**	
CF 7.1	Vizualizare cont doctor	Doctor, Pacient, Admin
CF 7.2	Vizualizare cont pacient	Doctor, Pacient, Admin
CF 8	**Administrare specializări**	
CF 8.1	Adăugare specializări	Admin
CF 8.2	Editare specializări	Admin
CF 8.3	Ștergere specializări	Admin
CF 8.4	Vizualizare specializări	Admin, Pacient
CF 9	**Administrare servicii doctori**	
CF 9.1	Ștergere servicii doctor	Admin, Doctor
CF 9.2	Vizualizare servicii doctor	Admin, Pacient, Doctor

CF 9.3	Editare servicii doctor	Admin, Doctor
CF 9.4	Adăugare servicii doctor	Admin, Doctor
CF 10	**Administrare rezervari**	
CF 10.1	Adăugare rezervare	Pacient
CF 10.2	Ștergere rezervare	Pacient
CF 10.3	Editare rezervare	Pacient
CF 10.4	Vizualizare rezervări	Pacient, Doctor
CF 10.5	Primire notificări prin mail pentru rezervări	Pacient
CF 11	**Administrare feedback doctor**	
CF 11.1	Adăugare feedback pentru doctor	Pacient
CF 11.2	Vizualizare feedback-uri	Pacient, Doctor
CF 12	**Administrare fișiere pacient**	
CF 12.1	Adăugare fișiere pacient (plan medical, radiografie, rezultate)	Doctor
CF 12.2	Stergere fișiere pacient (plan medical, radiografie, rezultate)	Doctor
CF 12.3	Vizualizare fișiere pacient	Pacient, Doctor
CF 13	Trimitere de mesaje în aplicatie	Pacient, Doctor

4.2. Cerințe non-funcționale

4.2.1. Securitatea

Această proprietate este foarte importantă pentru un sistem, acesta trebuie protejat împotriva persoanelor neautorizate, datele și credențialele utilizatorilor trebuie să fie confidențiale. Pentru aceste lucruri, în aplicație s-au implementat:

S1: Parola unui utilizator trebuie să fie cât mai sigură, astfel ca o parolă să fie validă acesta trebuie să aibă minim 8 caractere, să conțină minim o literă mică, o literă mare și un număr.

S2: Parola unui utilizator este încripată în baza de date

S3: După ce utilizatorul se loghează în aplicație, se creează o sesiune în browser unde va fi stocat JWT Token-ul, iar pe baza acestui token se vor putea realiza request-urile în aplcație.

S4: Fiecare request la server va necesita un token, dacă acest token nu este valid, atunci utilizatorul va fi scos din aplicație și va fi redirecționat catre pagina de login

S5: În funcție de rolul fiecarui utilizator, accesul va fi restricționat în aplicație, de exemplu un utilizator cu rolul de admin nu va putea apela sau ajunge pe pagina unui utilizator cu rolul de doctor.

4.2.2. Performanța

Aceast indicator de calitate măsoară eficiența sistemului. Pentru aceasta, există următoarele caracteristici de performanta:

P1: Fiecare operație a sistemului implică un timp mic de răspuns pentru request-uri, fiecare raspuns fiind sub 5 secunde

P2: Sistemul suportă până la 50000 utilizatori concurenți, rezultat măsurat cu ajutorul tool-ului JMeter, unde s-a masurat accesul a 50000 până la utilizatori care au realizat mai multe tipuri de request-uri, precum înregistrarea în aplicație, realizarea unei rezervari, căutarea de doctori, servicii, etc.

4.2.3. Disponibilitatea

Această cerință non-funcțională reprezintă timpul în care aplicația va fi disponibilă. Aplicația va putea fi disponibilă oricând, clienții având acces mereu la aplicație, dacă există o conexiune la internet pentru a accesa aplicatia web.

4.2.4. Utilizabilitatea

Acest indicator de calitate reprezintă gradul de ușurință în utilizarea aplicației. Astfel, sistemul trebuie să fie ușor de folosit, interfața grafică să aibă un design ușor, plăcut petru orice utilizator, fiecare pagină să fie simplificată și clară și user-friendly.

Pentru acest indicator, s-au realizat următoarele:

U1: S-a creat o interfață ușor de folosit, pe o pagină nu există un număr mare de operații, acestea sunt împărțite în cât mai multe pagini pentru o utilizare mai simplă și mai clară pentru orice utilizator

U2: S-au implementat metode pentru gestionarea erorilor, astfel utilizatorul va primi un mesaj de eroare în cazul în care nu se poate realiza o operație, spre exemplu dacă acesta nu a introdus credențiale corecte, se va afișa un mesaj corespunzător.

4.2.5. Extensibilitatea

Această proprietate reprezintă ușurința dezvoltării sistemului, fiind posibilă adăugarea altor funcționalități în viitor. Pentru această prorietate, s-au realizat următoarele:

E1: S-a început cu implementarea sistemului dupa arhitectura multi-layered pe partea de back-end, existând de la bun inceput o organizare pe pachete riguroasă.

E2: S-au respectat principiile de clean code și SOLID

4.2.6. Accesibilitatea

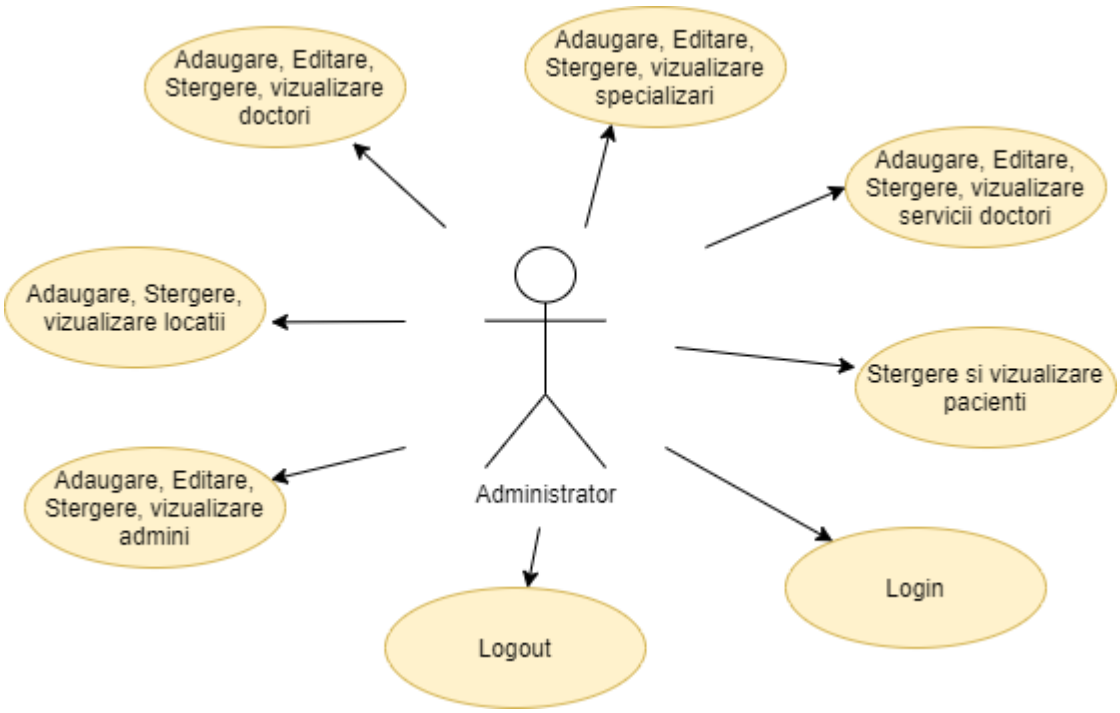
Acest indicator reprezinta modul in care se poate accesa sistemul. Aplicatia este una web, astfel se va putea accesa usor, prin intermediul unui browser web si conexiune la internet. Utilizatorul cand va intra in aplicatie, va trebuie sa se inregistreze si sa se logheze dupa care va putea intra in aplicatie.

4.3. Cazuri de utilizare

4.3.1. Actorii sistemului

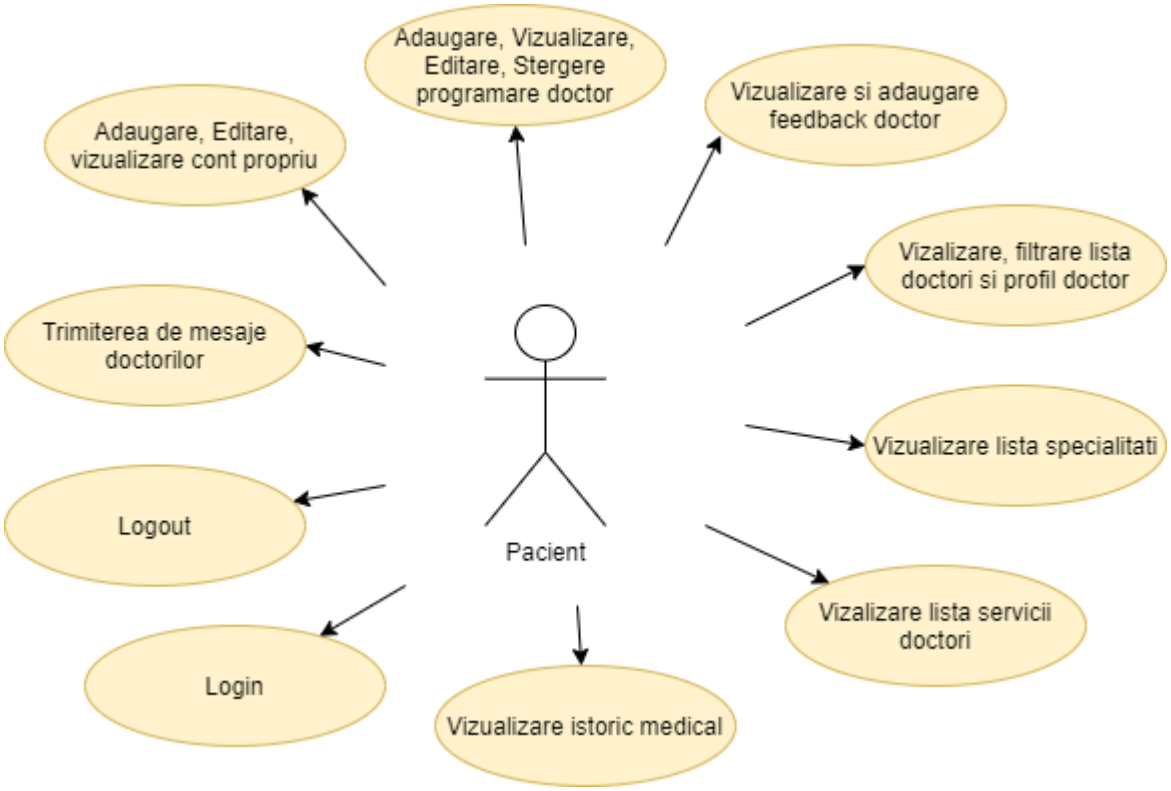
In acest subcapitol sunt prezentate cazurile de utilizare pentru fiecare tip de utlizator, mai exact administrator, pacient si doctor. Aceste diagrame de utilizare indetifica actorul implicat in actiuni si specifica numele interactiunii. Acestestea sunt foarte importante pentru dezvoltarea unui sistem deoarece prezinta toate actiunile posibile pe care le poate realiza un actor.

În figura de mai jos sunt prezentate toate cazurile de utilizare ale administratorului.



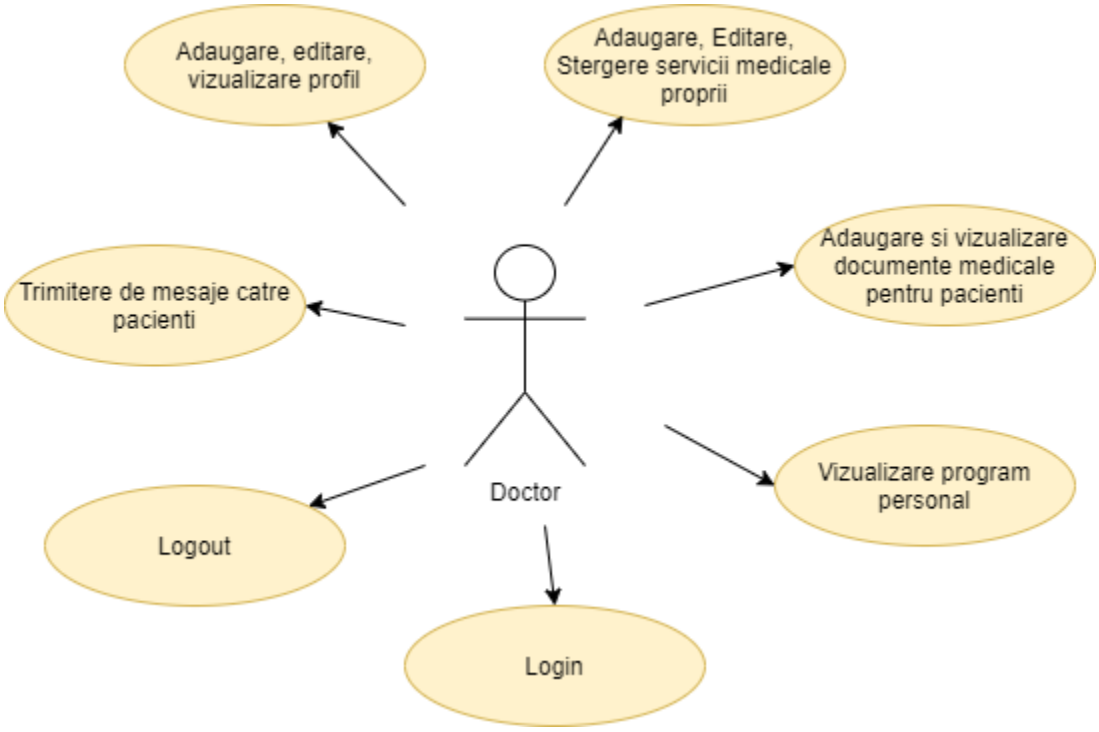
Figură 4.1 – Cazuri de utilizare pentru administrator

În figura de mai jos sunt prezentate toate cazurile de utilizare ale pacientului.



Figură 4.2 – Cazuri de utilizare pentru pacient

În figura de mai jos sunt prezentate toate cazurile de utilizare ale administratorului.



Figură 4.3 – Cazuri de utilizare pentru doctor

4.3.2. Descrierea detaliată a cazurilor de utilizare

În acest subcapitol, se va realiza descrierea detaliată a cazurilor de utilizare pentru fiecare tip de actor.

4.3.2.1. Descriere cazuri de utilizare administrator

CU1
Nume caz de utilizare: Adăugare specialitate
Actor principal: Administrator
Precondiții:

- Actorul trebuie sa fie autentificat in aplicație si trebuie sa fie un utilizator autorizat pentru aceasta operație, un utilizator cu rolul de administrator

Postcondiții:

- Noua specialitate a fost adaugată cu succes in baza de date
- Utilizatorul este redirecționat catre lista de specialități după ce apasă butonul de adaugare

Scenariu favorabil:

1. Utilizatorul se loghează în aplicație și este autentificat cu succes
2. Utilizatorul apasă pe butonul “Specialties” și este redirecționat către pagina cu lista tutor specialităților medicale existente în aplicație.

3. Administratorul apasă pe butonul “+” care îl redirecționează către pagina de adaugare a unei specialități
4. Administratorul completează câmpurile “Name” și “Description” cu date corecte și poate încărca o poză pentru noua specialitate
5. Utilizatorul apasă pe butonul “Add” și este redirecționat către pagina cu lista de specialități, unde poate vizualiza noua specialitate adăugată

Scenariu nefavorabil:

Dacă administratorul nu completează câmpurile pentru nume și descriere, atunci nu va putea adăuga o nouă specialitate.

CU2

Nume caz de utilizare: Ștergere doctor

Actor principal: Administrator

Precondiții:

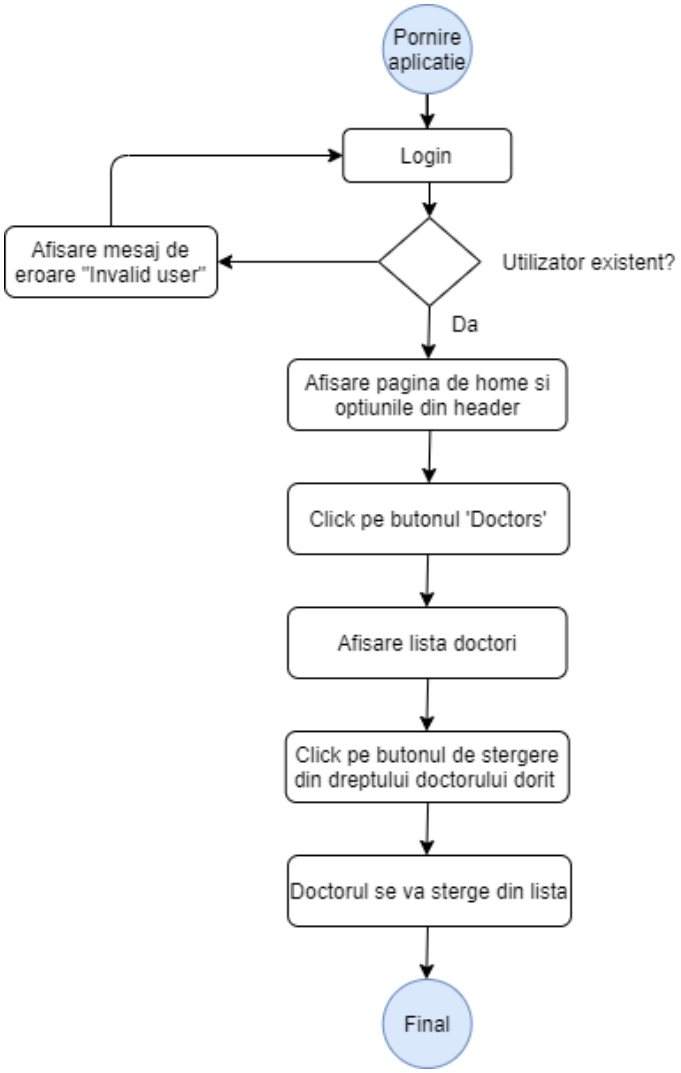
- Actorul trebuie sa fie autentificat in aplicație si trebuie sa fie un utilizator autorizat pentru aceasta operație, un utilizator cu rolul de administrator

Postcondiții:

- Doctorul a fost șters cu succes din baza de date
- Utilizatorul fiind în lista de doctori, după ce apasă butonul de ștergere poate vedea cum doctorul dispare din listă

Scenariu favorabil:

1. Utilizatorul se loghează în aplicație și este autentificat cu succes
2. Utilizatorul apasă pe butonul “Doctors” și este redirecționat către pagina cu lista tutor doctorilor existenți în aplicație.
3. Administratorul apasă pe butonul de ștergere din dreptul doctorului pe care dorește să îl șteargă și astfel doctorul respectiv dispare din lista de doctori



Figură 4.4 – Diagrama flowchart ștergere doctor al utilizatorului admin

4.3.2.2. Descriere cazuri de utilizare pacient

CU1
Nume caz de utilizare: Înregistrare în aplicație
Actor principal: Pacient
Precondiții: Nu există pentru acest caz de utilizare
Postcondiții:

- Noul cont va fi adăugat cu succes in baza de date
- Noul pacient se va putea loga cu succes în aplicație

Scenariu favorabil:

1. Utilizatorul apasă pe butonul de “Sign up” și este redirecționat către pagina de înregistrare
2. Utilizatorul completează pe rând toate datele necesare pentru a se înregistra în aplicație, mai exact va completa date valide pentru următoarele câmpuri: email, parolă, nume, prenume, telefon, data nașterii, CNP și sex.

Doar când toate datele vor fi completate si vor fi valide se va putea apăsa pe butonul “Next” iar la final pe butonul “Register”

3. Pacientul apasă pe butonul de înregistrare după care acesta este redirecționat către pagina de login, unde se va putea loga cu succes in aplicație pe baza email-ului și a parolei nou introduse

Scenariu nefavorabil:

Dacă pacientul nu completează toate campurile, atunci nu se va putea înregistra în aplicație, butonul de “Register” nefiind disponibil, pasul 3 din scenariul favorabil nu va putea fi îndeplinit.

CU2

Nume caz de utilizare: Realizare programare

Actor principal: Pacient

Precondiții:

- Actorul trebuie sa fie autentificat in aplicație si trebuie să fie un utilizator autorizat pentru aceasta operație, un utilizator cu rolul de pacient

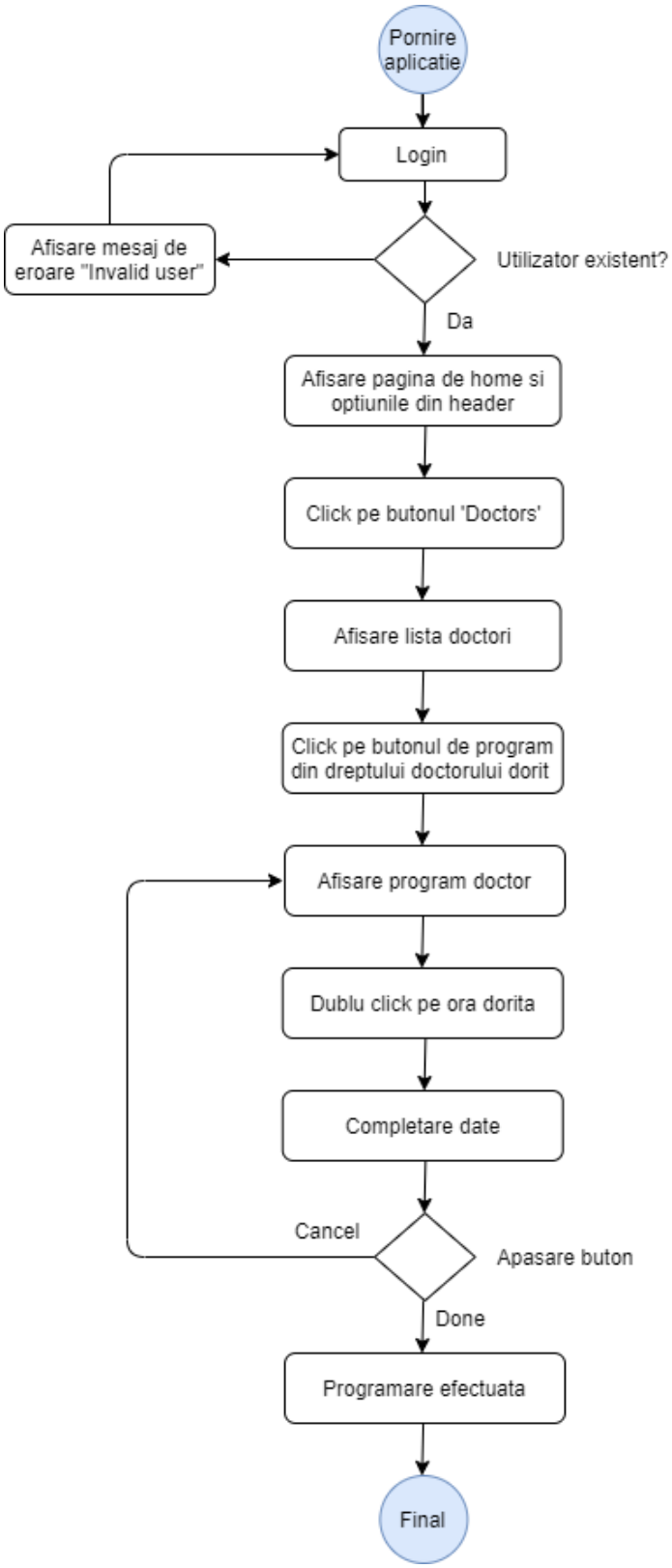
Postcondiții:

- Noua programare a fost adăugată cu succes în baza de date

Scenariu favorabil:

1. Utilizatorul se loghează în aplicație și este autentificat cu succes
2. Utilizatorul apasă pe butonul “Doctori” și este redirecționat către pagina cu lista tutor doctorilor existenți în aplicație.
3. Pacientul apasă pe butonul de program al unui doctor la care dorește să realizeze o rezervare și este redirecționat către pagina cu programul acestuia
4. Pacientul trebuie sa apese dublu click la ora dorită și liberă din calendar
5. Utilizatorul trebuie să selecteze serviciul dorit din cele existente și eventual va putea adauga o descriere pentru programarea sa, iar la final va apăsa pe butonul done
6. Utilizatorul revine la program unde isi va putea vizualiza noua rezervare

Scenariu favorabil:



Figură 4.5 Diagrama flowchart realizare programare al utilizatorului pacient

4.3.2.3. Descriere cazuri de utilizare doctor

CU1

Nume caz de utilizare: Vizualizare programări proprii

Actor principal: Doctor

Precondiții:

- Actorul trebuie sa fie autentificat in aplicație si trebuie să fie un utilizator autorizat pentru aceasta operație, un utilizator cu rolul de doctor

Postcondiții: Nu există pentru acest caz de utilizare

Scenariu favorabil:

1. Utilizatorul se loghează în aplicație și este autentificat cu succes
2. Utilizatorul apasă pe butonul "My Profile" și este redirecționat către pagina cu profilul personal
3. Doctorul apasă pe butonul "My Program" și este redirecționat către pagina cu programul personal
4. Doctorul este în pagina cu programul personal, unde poate sa vizualizeze cu success toate programarile existente

CU2

Nume caz de utilizare: Adăugare serviciu medical propriu

Actor principal: Doctor

Precondiții:

- Actorul trebuie sa fie autentificat in aplicație si trebuie să fie un utilizator autorizat pentru aceasta operație, un utilizator cu rolul de doctor

Postcondiții:

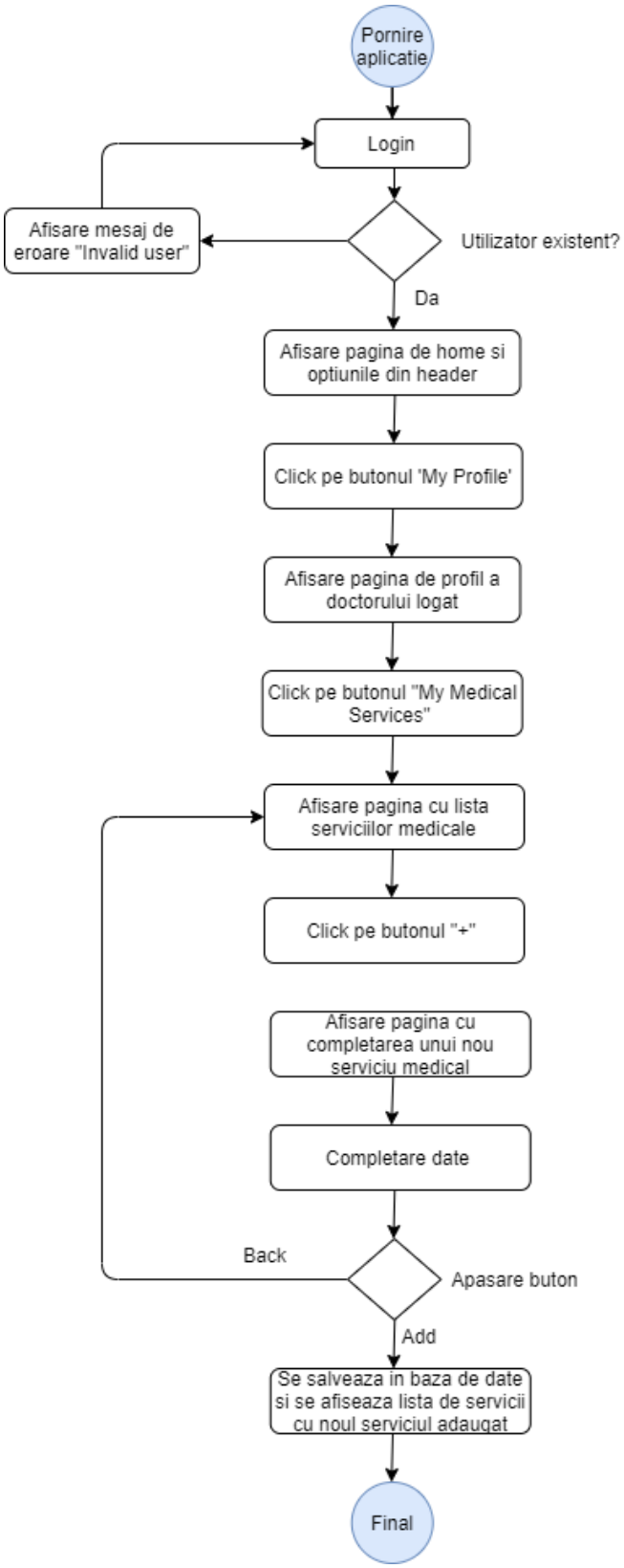
- Noul serviciu medical a fost adăugat cu succes în baza de date

Scenariu favorabil:

1. Utilizatorul se loghează în aplicație și este autentificat cu succes
2. Utilizatorul apasă pe butonul "My Profile" și este redirecționat către pagina cu profilul personal
3. Utilizatorul apasă pe butonul "My Medical Services" și este redirecționat către pagina cu lista tutor serviciilor sale existente în aplicație.
4. Doctorul apasă pe butonul "+" care îl redirecționează către pagina de adaugare a unui serviciu
5. Doctorul completează câmpurile "Name", "Price" și "Duration in minutes" cu date corecte
6. Utilizatorul apasă pe butonul "Add" și este redirecționat către pagina cu lista de servicii medicale, unde poate vizualiza noul serviciu adăugat

Scenariu nefavorabil:

Dacă doctorul nu completează toate câmpurile, atunci nu va putea adăuga un nou serviciu, astfel nu va putea apăsa pe butonul "Add", astfel pasul 6 din scenariul favorabil nu va fi incheiat cu succes.



Figură 4.6 Diagrama flowchart adaugare serviciul medical propriu

4.4. Perspectiva tehnologică

4.4.1. Tehnologii back-end

4.4.1.1. Spring Boot framework

Spring Boot[5] reprezintă un framework folosit pentru a crea aplicații stand-alone, acesta fiind o extensie a framework-ului Spring. Configurarea se realizează mult mai ușor, se reduce timpul de development și crește astfel produtivitatea. Pricipalele caracteristici sunt configurarea automata a Spirng-ului, existența unui fișier pom.xml pentru configurarea maven, integrarea cu Tomcat, Jetty, înlocuirea fișierelor XML cu anotari și multe altele.

Am ales acest framework pentru dezvoltarea aplicației back-end tocmai datorită avantajelor aduse, precum flexibilitatea configurarii aplicației, ușurința managmentului dependințelor și pentru dezvlotarea rapidă și eficientă a unei aplicații de sine stătătoare.

4.4.1.2. Apache Maven

Maven[12] este un sistem de build și administrare a proiectelor, scris în Java. Face parte din proiectele găzduite de Apache Software Foundation. Funcționalitățile sale principale sunt descrierea procesului de build al software-ului și descrierea dependențelor acestuia. Proiectele sunt descrise printr-unul sau mai multe fișier XML, denumite POM-uri (Project Object Model), dar au o structură implicită, ceea ce încurajează structurarea similară a proiectelor.

4.4.1.3. Hibernate

Hibernate[13] este un tool care mapează obiectele model orientate obiect, clasele java la tabelele din baza de date. Maparea se face prin configurarea fișierelor XML folosind adnotari java. Este un framework ușor și rapid de folosit pentru accesarea unei baze de date și pentru prelucrarea datelor din baza de date.

4.4.1.4. MySQL

MySQL[14] este un sistem de gestiune a bazelor de date relaționale, produs de compania suedeză MySQL AB și distribuit sub Licența Publică Generală GNU. Este cel mai popular SGBD open-source la ora actuală, fiind o componentă cheie a stivei LAMP (Linux, Apache, MySQL, PHP).

Am ales acest sistem de gestiune a bazelor de date datorită avantajelor pe care acesta le oferă, precum performanță ridicată, securitatea datelor, scalabilitate, flexibiltate și licență publică.

4.4.1.5. JWT

JSON Web Token[6] reprezintă un standard pentru a crea JSON access tokens care folosesc la partea de securitate a unei aplicatii, mai exact la autentificarea utilizatorilor într-o aplicație. Acest standard definește un mod companct și auto-conținut

[12] https://maven.apache.org/
[13] https://hibernate.org/orm/
[14] https://www.mysql.com/

pentru transmiterea sigură a inforamtiilor printr-un obiect JSON. De fiecare dată când se va apela un endpoint, se va verifică dacă există un token valid, dacă nu va fi valid atunci utilizatorul nu va fi autorizat în aplicație. JWT poate fi semnat folosind o cheie secretă, astfel fiind verificat și utilizat tokenul.

Am ales acest mod de a securiza aplicația deoarece este unul sigur și eficient, precum se menționează mai sus, la fiecare apel al unui request de orice tip (PUT, POST, DELETE, GET) se va verifică prezența acestui token, iar dacă acesta este invalid sau inexistent atunci nicio persoană nu va putea apela cu succes endpoint-ul și va primi un mesaj de eroare.

4.4.1.6. Amazon Web Services (AWS)

4.4.1.6.1. Simple Email Service (SES)

Simple Email Service [15] reprezintă un serviciu de email care permite programatorilor să implementeze trimiterea de mail-uri din orice aplicație. Acest serviciul este eficient, felixibil și scalabil și poate suportă mai multe tipuri de mail-uri, precum cele de tranzacții și de marketing. Cu ajutorul acestui serviciu se pot trimite mail-uri sigure, securizate peste tot în lume.

Am ales acest sistem deoarece este ușor de intregrat în aplicație și va putea fi folosit pentru a trimite notificări utilizatorilor care urmează să aibă o programare.

4.4.1.6.2. Simple Storage Service S3

Amazon Simple Storage Service [16] este un serviciu de stocare care oferă scalabilitate, disponibilitatea datelor, securitate și performanță. Acest serviciul este folosit pentru a stoca orice fel de tip de date, de la imagini, videoclip-uri, până la fișiere, aceste date fiind folosite pentru orice tipuri de cazuri de uilizare, precum aplicații web, mobile, servicii de back-up și arhive.

Am ales acest serviciu deoarece este ușor de folosit, este foarte organizat, datele sunt mereu disponibile și sunt securizate, nu oricine poate avea acces la ele. Acest serviciu de stocare este folosit în mai multe situații în propria aplicație, precum stocarea imaginilor doctorilor și al specializărilor și stocarea fișierelor medicale ale pacienților.

4.4.2. Tehnologii front-end

4.4.2.1. Angular

Angular[17] este o platformă de dezvoltare web cu sursă deschisă bazată pe limbajul TypeScript. Acesta utilizează o ierarhie de componente ca principală caracteristică arhitecturală, iar mare parte din funcționalitatea frameworkului a fost mutată în module.

Am ales să lucrez în Angular pentru partea de front-end a aplicației datorită avantajelor oferite, precum performanța și nu în ultimul rând datorită limbajului Typescript.

[15] https://aws.amazon.com/ses/
[16] https://aws.amazon.com/s3/
[17] https://angular.io/

4.4.2.2. Typescript

TypeScript[18] este un limbaj de programare open source dezvoltat și menținut de Microsoft. Este un superset sintactic al limbajului JavaScript și asigură un sistem de tipuri opțional. TypeScript este proiectat pentru dezvoltarea de aplicații de mari dimensiuni și se compilează în JavaScript. TypeScript poate fi utilizat atât pentru a dezvolta aplicații JavaScript pentru partea de client, cât și pentru partea de server (Node.js).

4.4.2.3. CSS

Cascading style sheets CSS[19] reprezintă un mecanism de adăugare a stilului unor documente web HTML. Prin acesta se pot defini culori, fonturi, spațiere, poziționarea în pagina a elementelor HTML, etc. Pentru a folosi CSS se pot crea fișiere externe .css sau se poate defini în cadrul documentului HTML cu ajutorul tag-ului <style>.

Am folosit acest mecanism pentru a realiza o aplicație web plăcută vizual și ușor de folosit.

4.4.2.4. Javascript

Javascript[20] este un limbaj de programare folosit pentru realizarea paginilor web interactive. Marea majoritate a aplicațiilor folosesc javascript, aflandu-se la baza tehnologiilor pentru aplicațiile web alaturi de HTML și CSS.

Acest limbaj a fost folosit în special la partea de mesagerie a aplicației pentru a realiza partea de server și conexiunea cu serviciul de mesagerie Stream.

4.4.2.5. Devextreme

Devextreme[21] reprezintă un framework pentru crearea aplicațiilor web realizat de DevExpress. Acesta se bazează pe HTML5 și Javascript și poate genera aplicații native pentru telefoane și tablete și pentru aplicații web.

Am ales să folosesc acest framework pentru a implementa partea de programări a aplicației cu ajutorul lui Devextreme Scheduler[22]. Aceasta componenta reprezintă un mod de a realiza și de a gestiona programări într-un mod facil. Aceasta oferă diferite tipuri de a vizualiza un calendar, precum vizualizarea în modul zi, săptămână și lună. De asemenea se pot crea, edita și șterge cu ușurință orice programare. Această componentă oferă multe avantaje, precum faptul că totul este configurabil, de la interfață, până la evenimentele de tip adăugare, editare, orice poate fi realizat și configurat pentru propria aplicație. Pentru propriul sistem, elementele configurate in aplicație sunt următoarele:

- Adăugarea unei programări în baza de date în momentul în care se apasă pe butonul save, funcția librăriei fiind *onAppointmentAdding*
- Adăugarea mesajelor de eroare în cazul în care programul doctorului sau programul pacientului care realizează programarea era deja ocupat la ora aleasa, verificare realizata prin backend

[18] https://www.typescriptlang.org/
[19] https://www.w3schools.com/css/css_intro.asp
[20] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction
[21] https://js.devexpress.com/Overview/
[22] https://js.devexpress.com/Overview/Scheduler/

- Editarea programărilor în baza de date în momentul în care se apela funcția librăriei *onAppointmentUpdated*
- Ștergerea programărilor prin execuția functiei *onAppointmentDeleted* din libraria DevExreme
- Realizarea de verificări înainte de a se apăsa pe celulele din program, precum: se verifică faptul că persoana autentificată nu poate modifica sau șterge alte programări prin apelul unei metode în Typescript în momentul executării funcțiilor *onAppointmentDblClick* si *onAppointmentDeleting*, se verifica ora și data unei celule înainte de a fi apăsată pentru a nu permite utilizatorului să efectueze programări în trecut prin executarea funcției *onCellClick*
- Începutul și sfârșitul programului uni doctor este configurat să fie intre orele 9-18, acest lucru fiind realizat prin atributele din HTML [startDayHour]="9" si [endDayHour]="18"

4.4.2.6. Stream

Stream[23] reprezintă un serviciu prin care se pot realiza aplicații de feed și de mesagerie. Acest serviciu este utilizat datorită scalabilității, performanței și a modului de personalizare a serviciilor oferite. Serviciul de mesagerie oferă multe funcționalități, precum: crearea de noi conversații de mai multe tipuri, precum pentru livestream, pentru două persoane, pentru grupuri sau pentru comerț, notificări pentru noile mesaje în timp real, reacții și reply-uri la mesaje, încărcare de imagini, eveniment pentru scrierea mesajelor și securitatea canalelor prin token.

Am ales să folosesc acest serviciu pentru a realiza partea de mesagerie a aplicației. Pentru a realiza mesageria a fost necesară implementarea părții de server cu ajutorul limbajului Javascript și prin realizarea interfeței cu ajutorul tehnologiilor HTML, CSS și Typescript. Prin limbajul Typescript se apelează partea de server și se realizează astfel conexiunea cu aplicația Stream. Funcționalitățile utilizate în propria aplicație sunt următoarele: adăugarea unui nou canal unic pentru pacient și doctor prin apelarea funcției *channel.create()* și stergerea canalelor prin apelarea funcției *channgel.delete()*. Pentru a crea un canal destinat pentru două persoane se pune ca și parametru “messaging” ţn momentul definirii canalului prin funcția *.channel*.

Partea de mesagerie are mai multe avantaje, precum:

- Ușurința integrării mesageriei în aplicație
- Scalabilitatea
- Performanța
- Organizarea simplă a conversațiilor

[23] https://getstream.io/

Capitolul 5. Proiectare de Detaliu si Implementare

În acest capitol se va descrie arhitectura generală și conceptuală a sistemului, urmând descrierea amănunțită a arhitecturii componentei backend și a componentei frontend, iar la final se va descrie diagrama bazei de date și tabele bazei de date.

5.1. Arhitectura sistemului

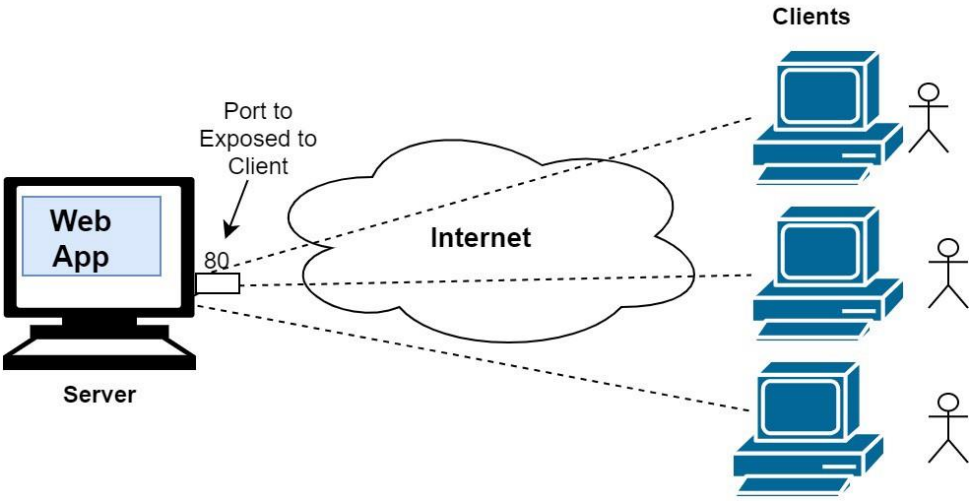
Arhitectura sistemului este cea de client-server[7]. Aceasta arhitectura se împarte în două mari componente în care partea de client apelează partea de server. În cele mai multe cazuri, partea de server reprezintă baza de date împreună cu logica aplicației, iar partea de client constă în interfața utilizatorului. Această arhitectură ajută la dezvoltarea sistemelor care implică cele două sisteme de client și de server și o parte de legătură, prin care cele două comunică în mod direct.

Această arhitectură este de asemenea denumită o structură de „network-computing" deoarece fiecare apel se realizează prin intermediul unei rețele. În cele mai multe cazuri, această legătură se realizează prin intermediul protocolului de comunicare HTTP[8] si HTTPS (Hyper Text Transfer Protocol Secure). Pentru a realiza comunicarea, este necesar un mecanism pentru a trimite sau a cere date de la server. Cele mai folosite astfel de mecanisme sunt XML și JSON. Spre exemplu, partea de client trimite un apel de tip GET prin care îi cere serverului anumite date pe baza unor parametrii, iar ca răspuns primește de la server informațiile dorite în unul dintre formatele menționate.

Aceasta arhitectura implica urmatoarele avantaje[24]:

- Îmbunătățește modul de transmitere a datelor – datele sunt transmise și manipulate de server, aceastea fiind disponibile pentru utilizatori printr-un acces autorizat. Aceste date sunt transmise intre mai mulți utilizatori.
- Integrarea serviciilor – elimină nevoia folosirii unui terminal pentru că acum clientul se poate folosi de o interfață
- Ușor de menținut – sistemele de server sunt ușor de înlocuit, de reparat, de îmbunătățit și de mutat fară a afecta într-un anumit mod partea de client
- Sunt securizate – serverul are abilitatea de a controla și de a autentifica utilizatorii autorizați pentru a asigura accesul sigur la date

[24] https://dwzimsuhaili.wordpress.com/2016/04/12/client-server-architecture-vs-p2p-architecture/



Figură 5.1 Arhitectura Client-Server[25]

De asemenea, sistemul s-a implementat folosind serviciile REST[9]. REST Representational State Transfer resprezintă un stil arhitectural folosit pentru a oferi standarde între sisteme și web, astfel încât comunicarea între acestea devine mult mai ușoară și sunt definite mai multe proprietăți precum performanța și scalabilitatea. În acest stil arhitectural, datele și functionalitatile sunt considerate ca fiind resurse care sunt accesate prin URI. Astfel, o aplicație REST cu o arhitectură client-server reprezintă separarea dintre client și server, mai exact partea de client și partea de server nu cunosc date una despre cealaltă, înafară de cele împărțite în mod direct.

Am ales să implementez acest stil arhitectural datorită următoarelor avantaje:

- Resursele sunt identificate prin URI – după ce sunt identificate aceastea oferă un spațiu global pentru adresare și pentru descoperirea serviciilor
- Uniformitatea interfeței – resursele sunt manipulate prin folosirea a celor patru tipuri de operații: creare, citire, modificare și ștergere
- Mesaje descriptive – resursele sunt decuplate de forma actuală astfel încat pot fi accesate în mai multe formate precum HTML, JSON, XML și altele

5.1.1. Arhitectura conceptuală

În figura de mai jos este ilustrată arhitectura conceptuală a sistemului. Sistemul este compus din următoarele componente: Front-end, back-end, baza de date, serviciul stream și serviciile amazon S3 și SES.

Partea de front-end s-a implementat folosind framework-ul Angular. Această componentă comunică direct cu partea de back-end pentru efecuarea operațiilor de citire, adăugare, editare și de ștergere a datelor. De asemenea această componentă comunică direct cu serviciul stream, pentru a implementa serviciul de mesagerie.

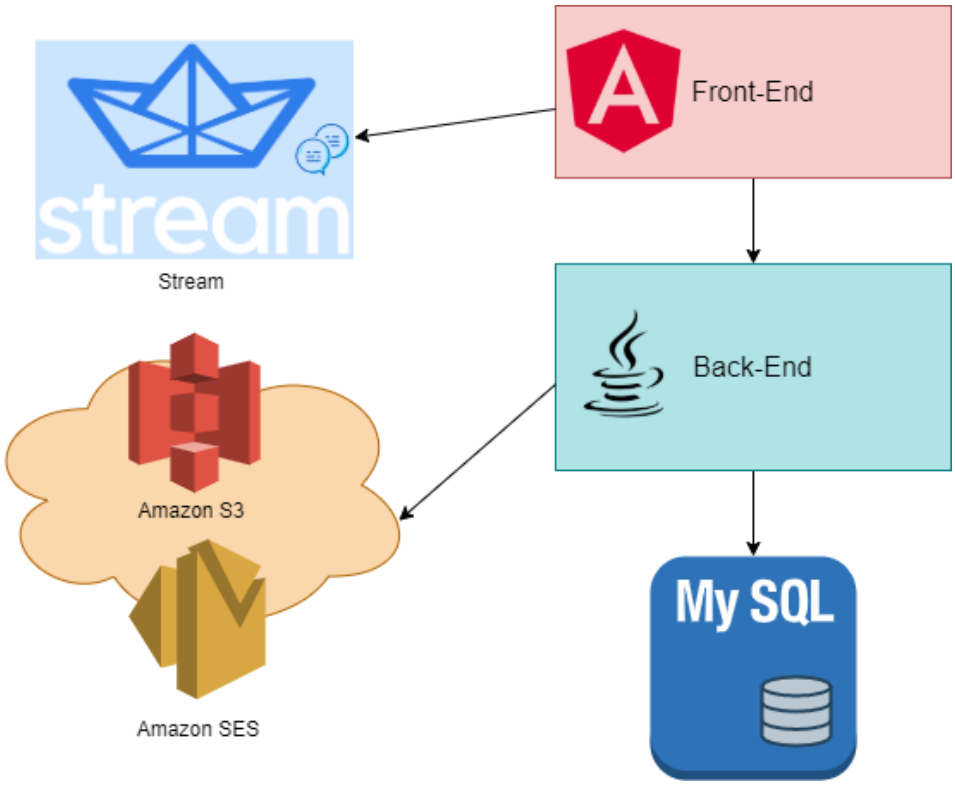
Componenta back-end comunică cu partea de front-end și cu baza de date. Serviciile din această componentă sunt apelate de componenta front-end, back-end-ul apelând mai departe baza de date pentru a realiza operațiile de citire, creare, editare și ștergere cerute de partea fron-end. Această parte se ocupă cu logica aplicației.

[25] https://medium.com/@nimeshadilini999/simple-introduction-to-client-server-architecture-concept-7d2979bed31d

Baza de date este reprezentată de baza de date MySQL în care sunt stocate toate datele aplicației. Baza de date comunică direct cu partea de back-end care realizează operațiile CRUD asupra datelor din MySQL.

Serviciile Amazon Simple Storage Service S3 și Simple Emailing Service SES sunt folosite pentru stocarea datelor și pentru trimitea mail-urilor în cadrul aplicației. Aceste servicii sunt apelate direct de componenta back-end.

Serviciul extern Stream este folosit pentru crearea sistemului de mesagerie din cadrul aplicației. Acest serviciu este apelat direct componenta front-end.



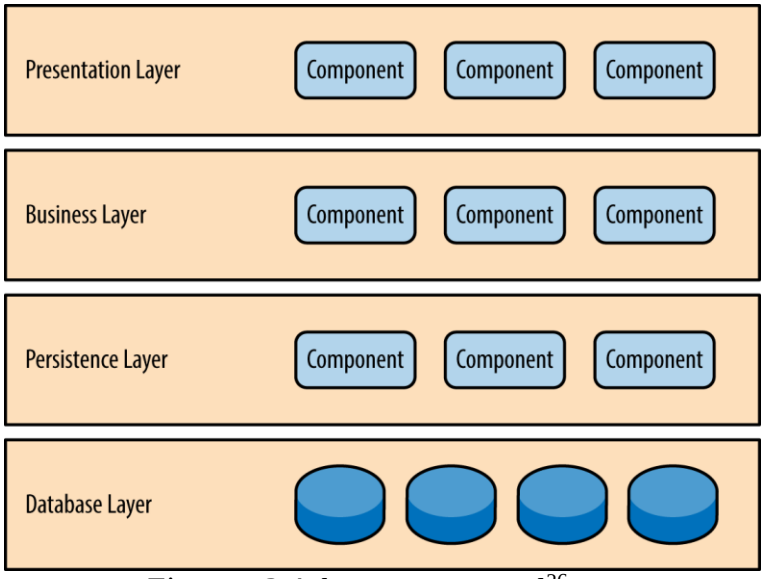
Figură 5.2 Arhitectura conceptuală a sistemului

5.1.2. Arhitectura componentei backend

Componenta arhitecturii backend a fost dezvoltată în limbajul Java folosind framework-ul Spring Boot. Arhitectura componentei backend este arhitectura layered.

Arhitectura layered[7] reprezintă o arhitectură organizată pe mai multe nivele orizontale, fiecare nivel având un rol specific în aplicație și fiecare nivel folosindu-se de cel de mai jos lui. De cele mai multe ori această arhitectură are următoarele layere, nivele ilustrate și în figura 5.3:

- Prezentare
- Bussiness
- Persistence
- Baza de date

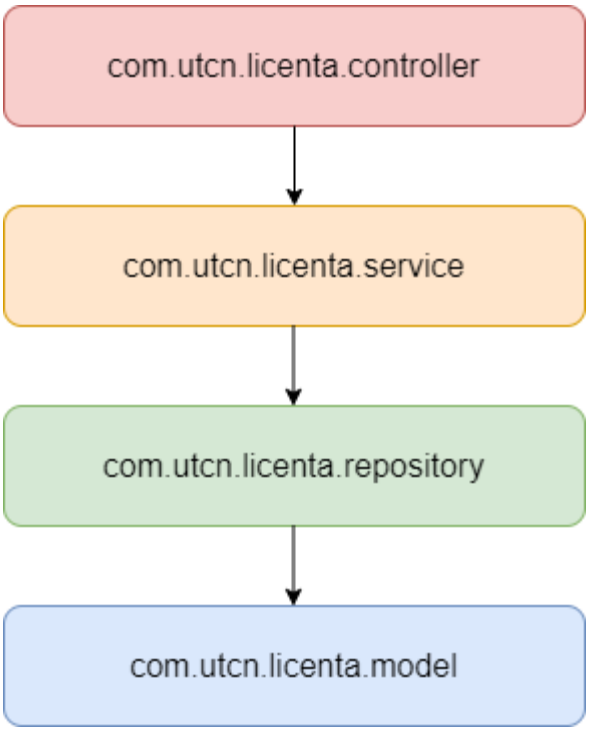


Figură 5.3 Arhitectura Layered[26]

Fiecare nivel are responsabilitatea sa pentru aplicație. În cazul sistemului propus, fiecare strat are denumirile prezentate în figura 5.3. Nivelul de prezentare are rolul de a gestiona interfața cu utilizatorul și comunicarea cu browser-ul, nivelul de bussiness are rolul de a executa partea de logica asociată cu diferitele apeluri de ale nivelului de prezentare, nivelul de persistence are rolul de a efectua operațiile CRUD[27], operațiile de creare, modificare, returnare și ștergere asupra datelor din baza de date, iar nu în ultimul rând, nivelul bazei de date este reprezentat sub denumirea model care conține clasele aferente entităților din baza de date.

Conceptul important al acestei arhitecturi reprezintă izolarea fiecărui nivel. Acesta reprezintă faptul că în general schimbările realizate într-un strat nu afectează celelalte straturi, astfel această schimbare este izolată de acel nivel. Această izolare are ca și avantaje modificarea ușoară și eficientă a diferitelor componente din aplicație, acestea nefiind strâns conectate. Dacă de exemplu stratul de prezentare ar accesa direct stratul de persistence, atunci schimbările realizate în SQL ar afecta și stratul de prezentare, dar și cel de bussiness, rezultând o aplicație dificil și costisitor de modificat. Totodată, prin acest concept de izolare rezultă faptul că straturile nu cunosc prea multe despre celelalte straturi, fiecare dintre ele fiind independent.

[26] https://learning.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html
[27] https://www.codecademy.com/articles/what-is-crud



Figură 5.4 Arhitectura Layered a sistemului propus

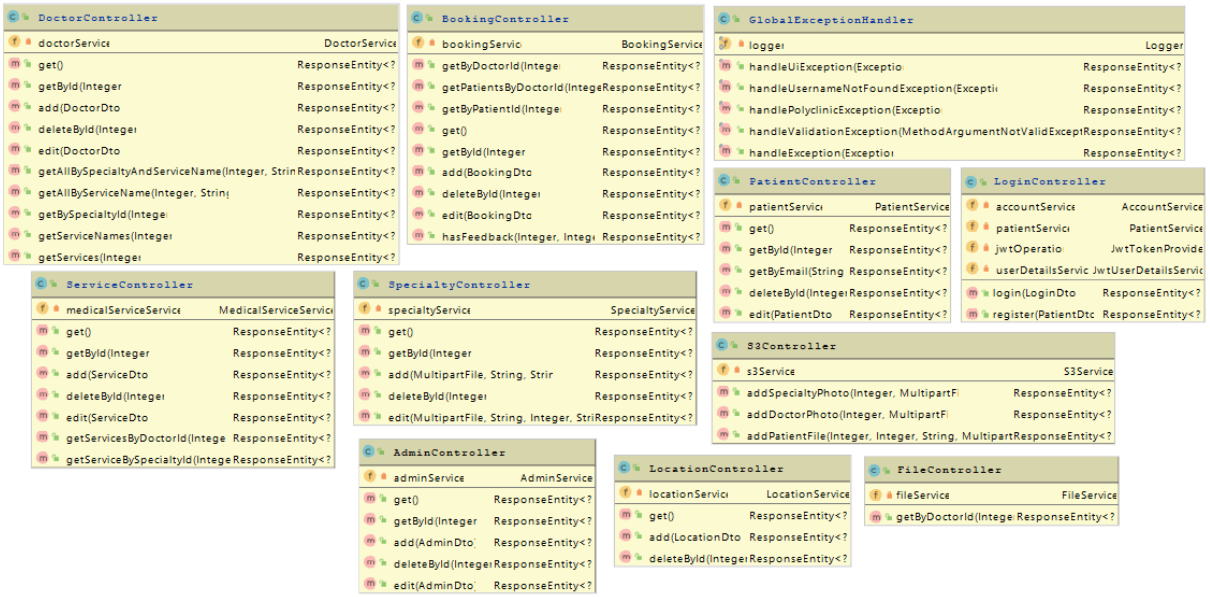
Acest concept de izolare a fost realizat în sistemul propus prin împărțirea acestuia în mai multe straturi, în mai multe pachete, enumerate mai jos:

Controller

Acest pachet reprezintă nivelul de prezentare din arhitectura layered, în fiecare clasă fiind definite endpoint-urile care se vor apela din partea de front-end a aplicației. În acest pachet sunt incluse mai multe clase, ilustrate în figura 5.5. Acestea sunt urmatoarele: *DoctorController, BookingControler, ServiceController, SpecialtyController, PatientController, LoginController, S3Controller, AdminController, LocationController, FileController, GlobalExceptionHandler*. Majoritatea dintre aceste clase au endpoint-uri de tip POST, PUT, GET, DELETE care realizează operațiile aferente de creare, editare, returnare sau ștergere a datelor. Fiecare dintre acestea au ca și atribut clasa service, care reprezintă stratul de bussiness din arhitectura layered. Spre exemplu, clasa *BookingController* are ca și metode mai multe metode de tip GET care vor fi folosite de către componenta frontend în mai multe situații, cum ar fi cea în care se va vizualiza programul unui docor prin metoda *getByDoctorId*, unde se vor returna toate programările unui anumit doctor.

O alta clasă importantă este *GlobalExceptionHandler*, aceasta având rolul de a gestiona toate erorile apărute în aplicație. Aceasta folosește adnotările @ControllerAdvice[28] si @ExceptionHandler care ne permite gestionam toate excepțiile într-un singur loc. Astfel putem avea control total asupra răspunsului și a statusului pe care dorim să le returnam în cazul unei erori a aplicației.

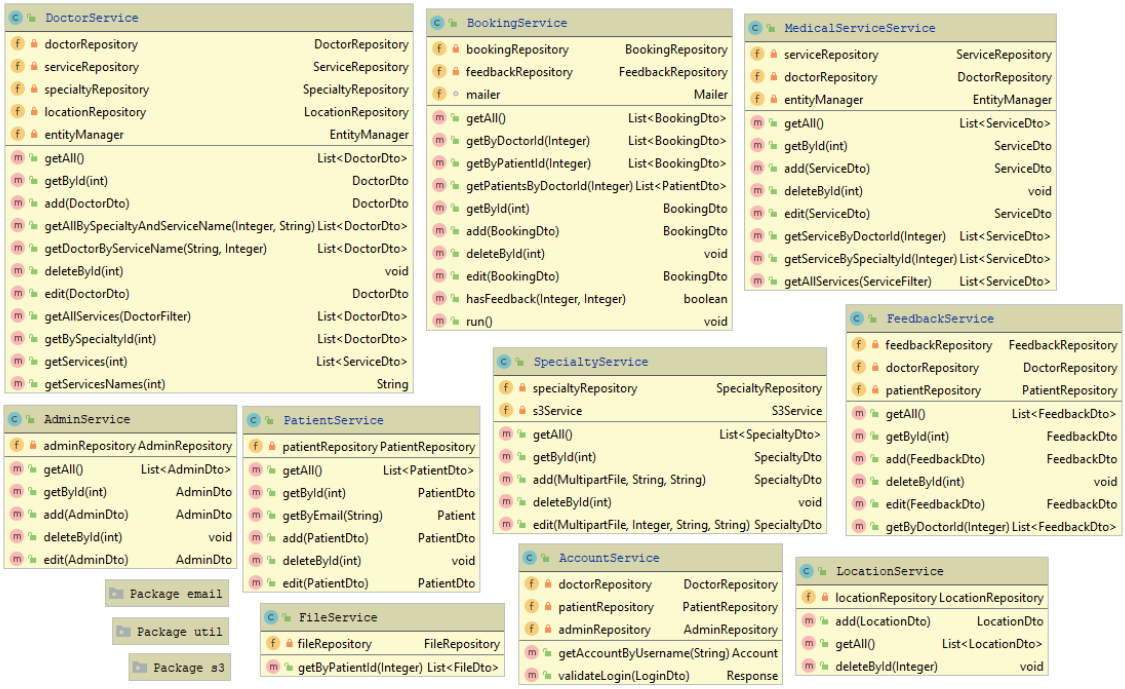
[28] https://www.baeldung.com/exception-handling-for-rest-with-spring



Figură 5.5 Diagrama de clase a pachetului controller

Service

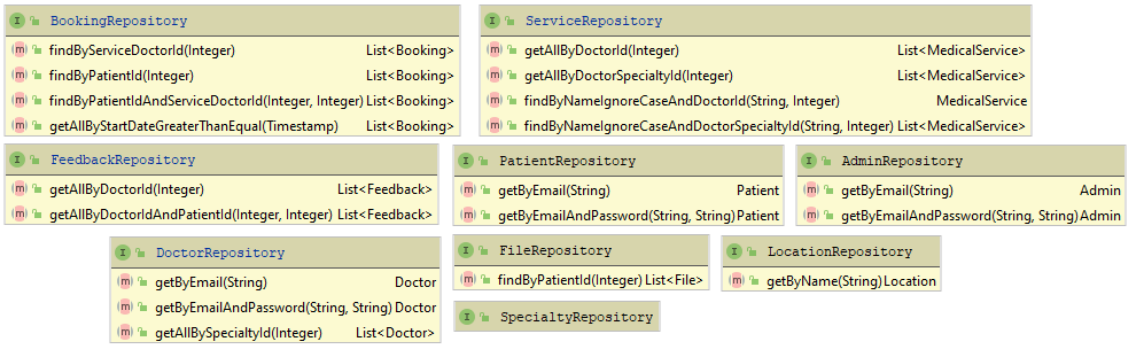
Acest pachet reprezintă stratul de bussiness al arhitecturii layered. Acesta se ocupă de partea de logică a aplicației și este apelat de stratul superior al arhitecturii, de nivelul de prezentare. Acest pachet conține următoarele clase: *DoctorService, BookingService, MedicalServiceService, AdminService, PatientService, SpecialtyService, FeedbackService, AccountService, LocationService si FileService*. Pe lângă aceste clase mai există și trei pachete care au în interiorul său alte servicii, mai exact cele de pentru email și pentru încărcarea imaginilor în S3. După cum se poate vedea în figură, fiecare clasă se folosește de nivelul persistence al arhitecturii layered, mai exact interfețele repository. Pe lângă logica aplicației, aceaste clase se ocupă cu maparea entităților din baza de date cu datele din obiectele de transfer (DTO – data transfer object). Dacă de exemplu o anumită metodă de tip GET ia din baza de date o listă de entități, în această clasă se convertește rapsunsul și se trimite mai departe o listă de obiecte de transfer spre nivelul de prezentare, date care vor fi preluate de componenta frontend și vor apărea în interfața utilizatorului.



Figură 5.6 Diagrama de clase a pachetului service

Repository

Pachetul repository reprezintă nivelul persistence al aplicației, nivelul care comunică direct cu baza de date. Fiecare dintre aceste interfețe extind interfața JpaRepository[29], prin care ne putem folosi de metodele generice de creare, citire, editare și ștergere a datelor din baza de date. Pe lângă aceste metode generice, am definit propriile interogări, de exemplu *getByPatientId*, metoda care va returna din baza de date pacientul cu id-ul dat.



Figură 5.7 Diagrama de clase a pachetului repository

Model

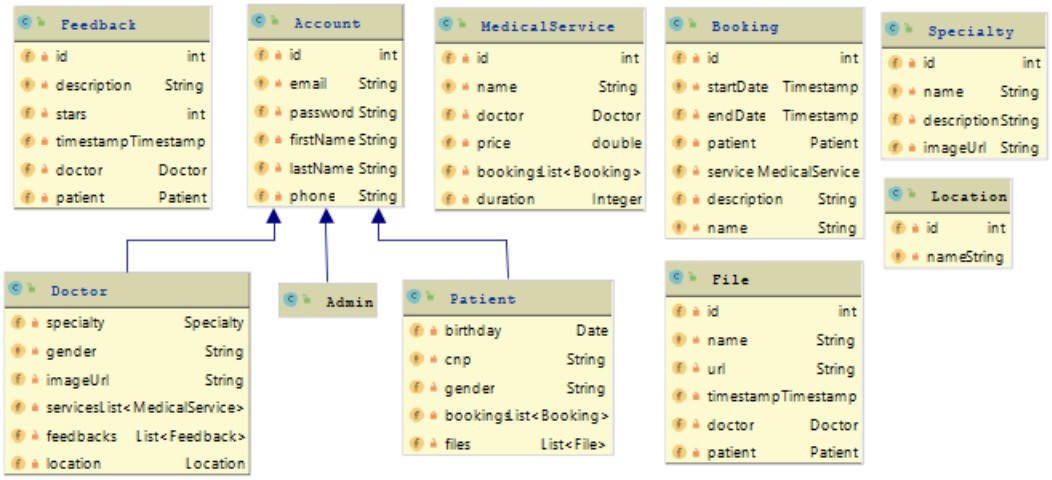
Acesta reprezintă stratul bazei de date din arhitectura sistemului, acest pachet fiind alcătuit din entitățile bazei de date. Clasele din acest pachet reprezintă tabele din baza de date, atributele din aceste clase fiind coloanele tabelelor. Pentru a face această legătură, se folosește adnotarea @Entity pentru a se realiza maparea cu entitatea din baza

[29] https://spring.io/blog/2011/02/10/getting-started-with-spring-data-jpa

de date. Pentru a defini cheia primară a tabelei se folosește adnotarea @Id, iar pentru a se defini auto-incremetarea cheii primare se folosește adnotarea @GeneratedValue(strategy=GenerationType.AUTO). Spre exemplu, în clasa Patient am definit cheia primara în acest fel:

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int id;
```

Astfel, atributul id va reprezenta cheia primară pentru tabelul pacient, iar la fiecare creare a unui nou pacient în baza de date valoarea câmpului id se va auto-incrementa, astfel având o valoarea unică.



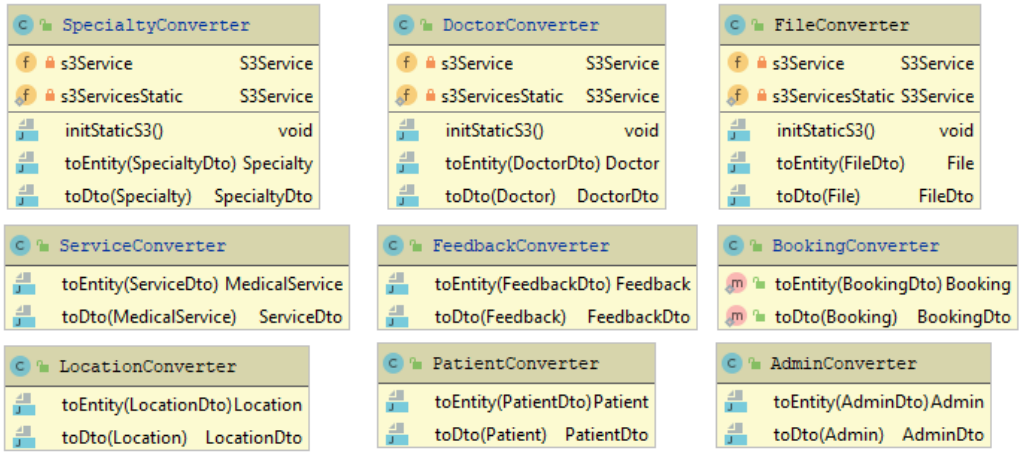
Figură 5.8 Diagrama de clase a pachetului model

Dto

Acest pachet se ocupă de obiectele de transfer. Acestea sunt obiecte speciale care este expus către componenta frontend și care mapeaza entitățile bazei de date. Acest pachet conține alte două pachete: converter și model.

Converter

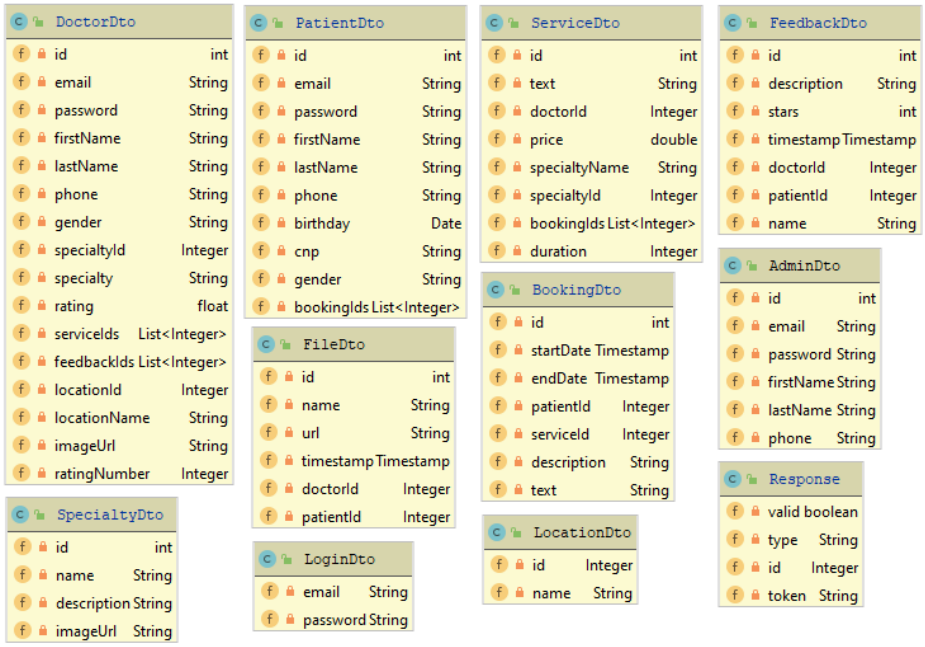
Pachetul converter conține toate clasele care se ocupă cu maparea obiectelor de transfer în entități și invers, după cum se poate observa în figura 5.9. Metoda toDto transformă obiectul entiate într-un obiect de transfer, iar metoda toEntity mapeaza obiectul de transfer într-un obiect entitate. Maprile s-au realizat cu ajutorul librăriei *ModelMapper*.



Figură 5.9 Diagrama de clase a pachetului converter

Model

Acest pachet contine toate obiectele de transfer folosite în aplicație. Aceste obiecte sunt folosite la nivelul de prezentare al aplicației, fiind trimise către componenta frontend. În general, obiectele de transfer conțin aceleași atribute ca și cele ale entității aferente, dar în unele cazuri pot conține mai multe atribute. Spre exemplu, entitatea *MedicalService* conține obiectul specialty, iar *ServiceDto* conține specialtyId si specialtyName, ultimul atribut fiind folosit pentru afișarea mai ușoară în interfața utilizatorului a serviciul medical și a numelui specialității serviciului.

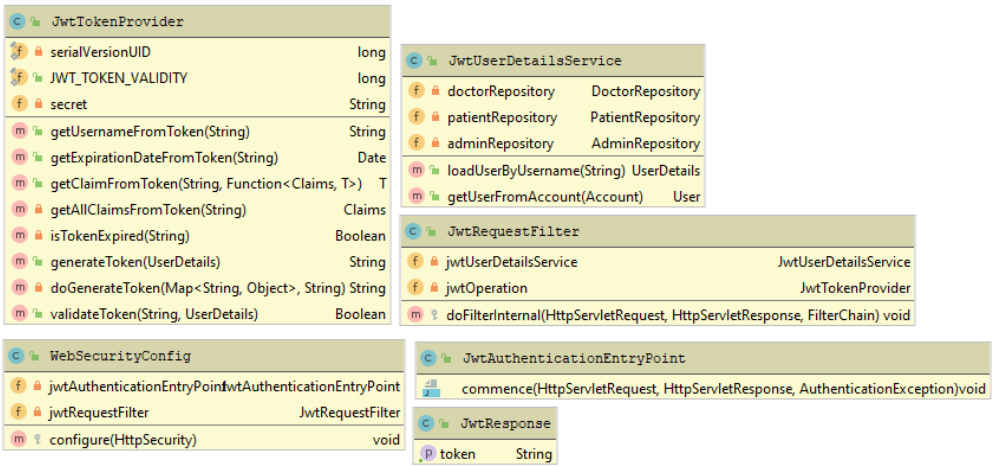


Figură 5.10 Diagrama de clase a pachetului model

Security

Acest pachet conține clasele care se ocupă de partea de securitate a aplicației. Acest pachet cuprinde clasele: JwtTokenProvider, JwtAuthenticationEntryPoint, JwtRequestFilter, JwtResponse, JwtUserDetailsService, WebSecurityConfig. Aceste clase au rolul de a realiza partea de securitate a endpoint-urilor printr-un JWT Token.

- JwtTokenProvider – această clasă are rolul de a genera și a valida tokenul trimis în partea de header a unui apel și de a returna utilizatorul pe baza token-ului
- JwtAuthenticationEntryPoint – clasa aceasta are rolul de a trimite o eroare în momentul în care tokenul nu este valid
- JwtRequestFilter – această clasă are ca și scop verificarea prezenței token-ului în apeluri și de a verifica credențialele dacă tokenul este existent
- JwtResponse – această clasă conține token-ul
- JwtUserDetailsService – clasa aceasta se ocupă de returnarea utilizatorului din baza de date pe baza token-ului existent
- WebSecurityConfig – în această clasă am configurat endpoint-urile care nu necesită prezența unui token pentru a se putea apela, aceastea fiind cel pentru înregistrare si cel pentru autentificare



Figură 5.11 Diagrama de clase a pachetului security

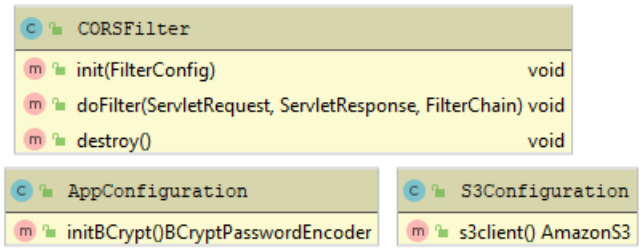
Configuration

Clasele din acest pachet se ocupă cu partea de configurație a aplicatiei.
Aceste clase sunt:

- *CorsFilter* – aceasta are rolul de a intercepta orice apel și de a include în partea de antet HTTP ca și răspuns. Cross-Origin Resource Sharing (CORS)[30] este un mecanism care se folosește de antetul HTTP să îi transmită browser-ului să permită unei aplicații web să aibă acces la resurse din altă origine.

[30] https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

- *AppConfiguration* – clasa are ca si rol inițializarea bean-ului *BCryptPasswordEncoder* pentru a fi folosit pentru criptarea parolei utilizatorilor
- *S3Configuration* – aceasta clasa scopul de a configura legatura cu serviciile Amazon pentru a putea folosi serviciul de stocare Simple Storage Service S3 și serviciul de email Simple Email Service SES.

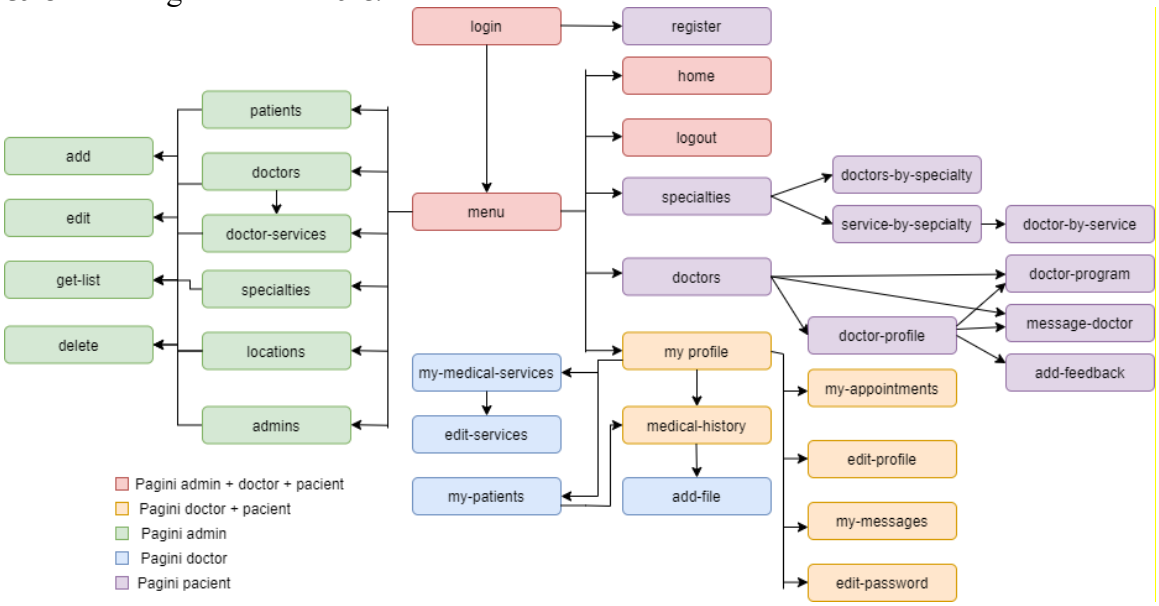


Figură 5.12 Diagrama de clase a pachetului configuration

5.1.3. Arhitectura componentei frontend

În acest subcapitol se vor prezenta principalele componente ale sistemului frontend și structura acestei componente. Pentru implementarea interfeței utilizatorului am folosit framework-ul Angular și limbajul Typescript.

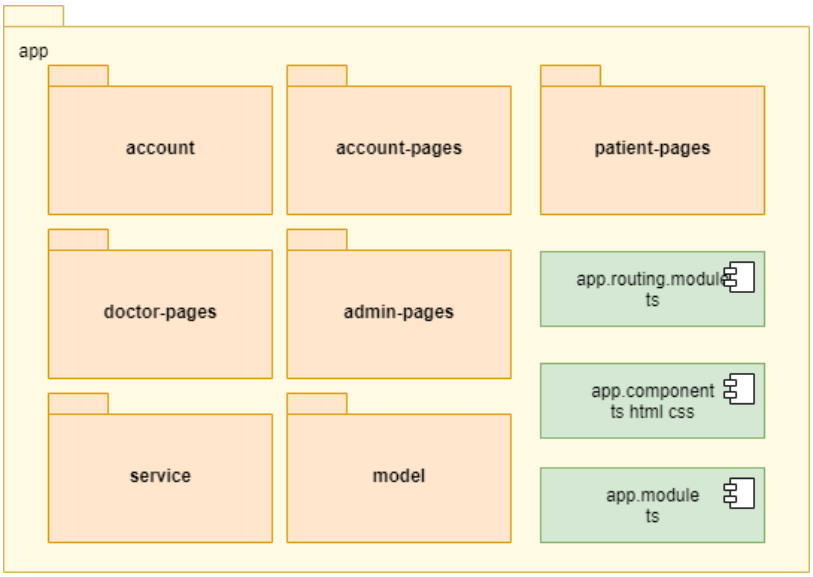
Imaginea următoare resprezinta structura logică a componentei frontend. În momentul în care se deschide aplicația, prima pagină este cea de login, iar după ce utilizatorul este autentificat urmează pagina de home și meniul în partea de sus. De acolo, fiecare tip de utilizator are mai multe funcționalități, aceastea fiind diferențiate după culori în imaginea următoare.



Figură 5.13 Structura logică a componentei frontend

Pentru a organiza codul interfeței grafice mai ușor, am structurat sistemul în mai multe pachete, după cum se poate observa în figura 5.13. Aceste pachete sunt: *account, account-pages, patient-pages, doctor-pages, admin-pages, service si model.* Pe langa

acestea, sunt componentele *app.component.ts, appcomponent.html, app.component.css, app.routing.module.ts si app.module.ts.*



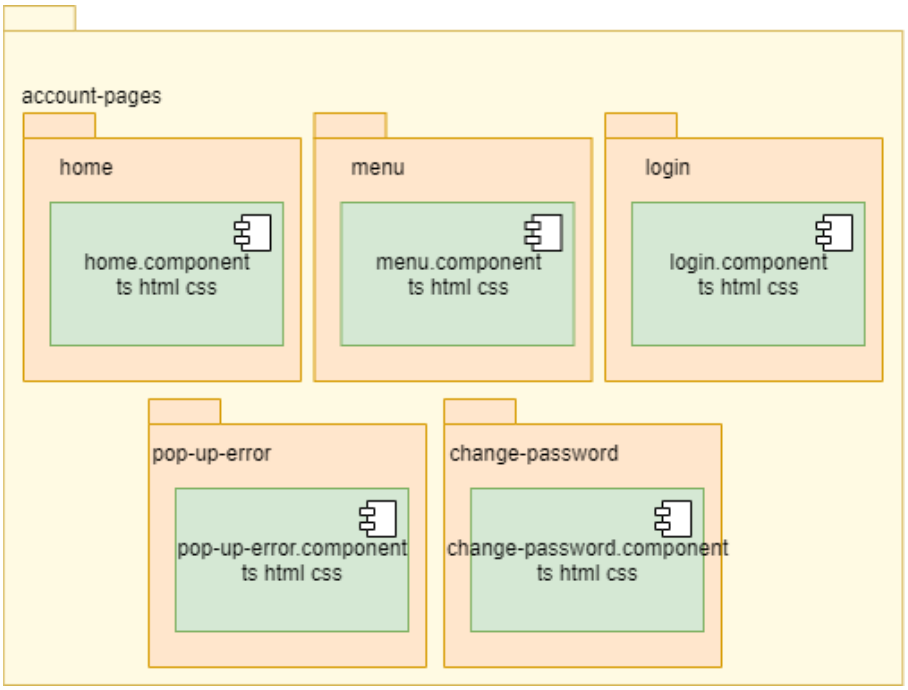
Figură 5.14 Diagrama de pachete a componentei frontend

În continuare se va descrie fiecare pachet din imagine de mai sus, rolul sau și componentele sale.

Account-pages

Acest pachet conține componentele comune tuturor utilizatorilor. Aceste componente sunt următoarele:

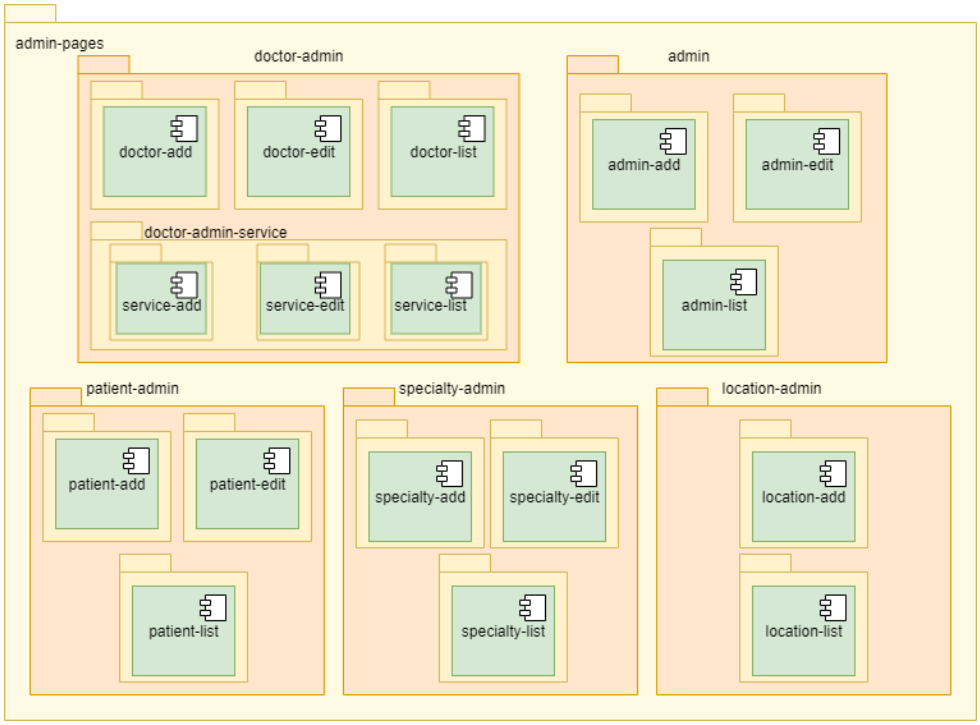
- *Home* – aceasta pagina reprezintă prima pagină pe care utilizatorul o va vedea după ce acesta se loghează cu succes în aplicație
- *Menu* – această componentă reprezintă partea de header a aplicației, fiecare utilizator având alte opțiuni în funcție de rolul lor, putând naviga cu ușurință în aplicație
- *Pop-up-error* – această componentă va fi folosită când se vor afișa pe ecran anumite erori, spre exemplu dacă un utilizator va introduce un email sau o parolă greșită, atunci îi va apărea pe ecran un mesaj de eroare
- *Change-password* – componenta aceasta este utilizată în momentul în care se dorește să se schimbe parola contului
- *Login* - reprezintă prima pagina care îi apare unui utilizator în momentul când pornește aplicația, prin aceasta orice utilizator care este înregistrat deja în aplicație va putea să se logheze. De asemenea, tot prin această pagină orice persoană își poate crea un cont daca va apăsa pe butonul de sign up care îl va conduce către pagina de înregistrare



Figură 5.15 Diagrama de pachete a paginilor *account-pages*

Admin-pages

Pachetul *admin-pages* conține toate componentele pe care utilizatorul adminitrator le va accesa. Acest utilizator va putea accesa paginile incluse în următoarele pachete: *doctor-admin, doctor-admin-service, admin, patient-admin, specialty-admin si location-admin*. Toate aceste componente incluse în aceste pachete au ca și rol crearea, editarea, ștergerea și vizualizarea doctorilor, pacienților, specializărilor, locațiilor, serviciilor medicale și a administratorilor.

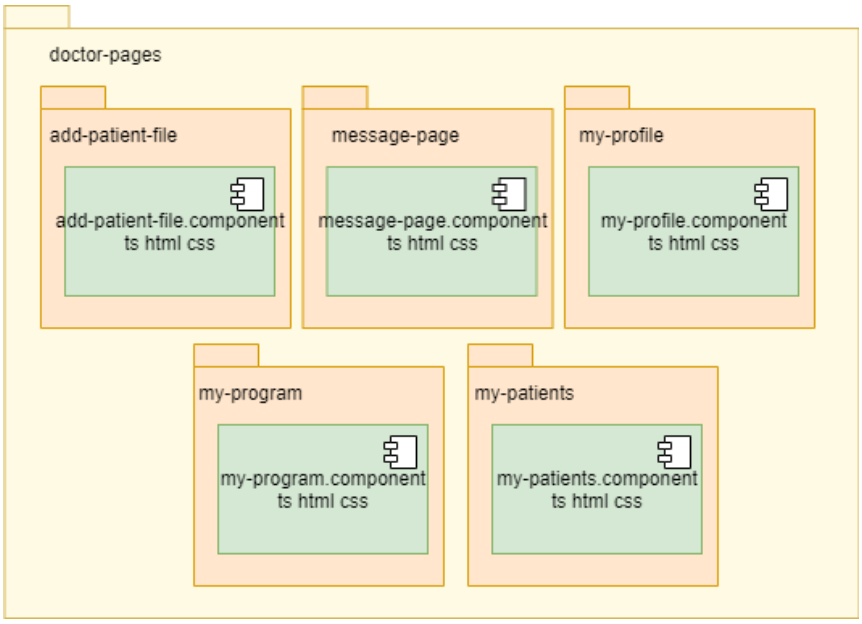


Figură 5.16 Diagrama de pachete a paginilor *admin-pages*

Doctor-pages

Acest pachet conține toate paginile pe care doar utilizatorul de tip doctor le va putea accesa. Aceste pagini sunt următoarele:

- *Add-patient-file* – această pagină se ocupă de adăugarea unui fișier medical unuia dintre pacienții doctorului logat în aplicație
- Message-page – această componentă are rolul de a afișa toate mesajele unui doctor, această pagină fiind accesată din pagina de profil a unui medic
- *My-profile* – această componentă reprezintă pagina de profil a unui doctor. Prin această pagină medicul poate să își editeze datele personale, parola și serviciile medcale. De asemenea, poate să își vizualizele mesajele, programul, pacienții și să le adauge fișierele medicale în urma unui serviciu oferit
- *My-program* – prin această pagină un doctor poate să își vizualizeze propriul program, să vadă toate rezervările din acea zi sau acea lună
- *My-patients* – pagina aceasta are rolul de a afișa o listă de pacienți care au fost programați la acest doctor, prin această pagină orice doctor poate să adauge pacientului fișiere medicale



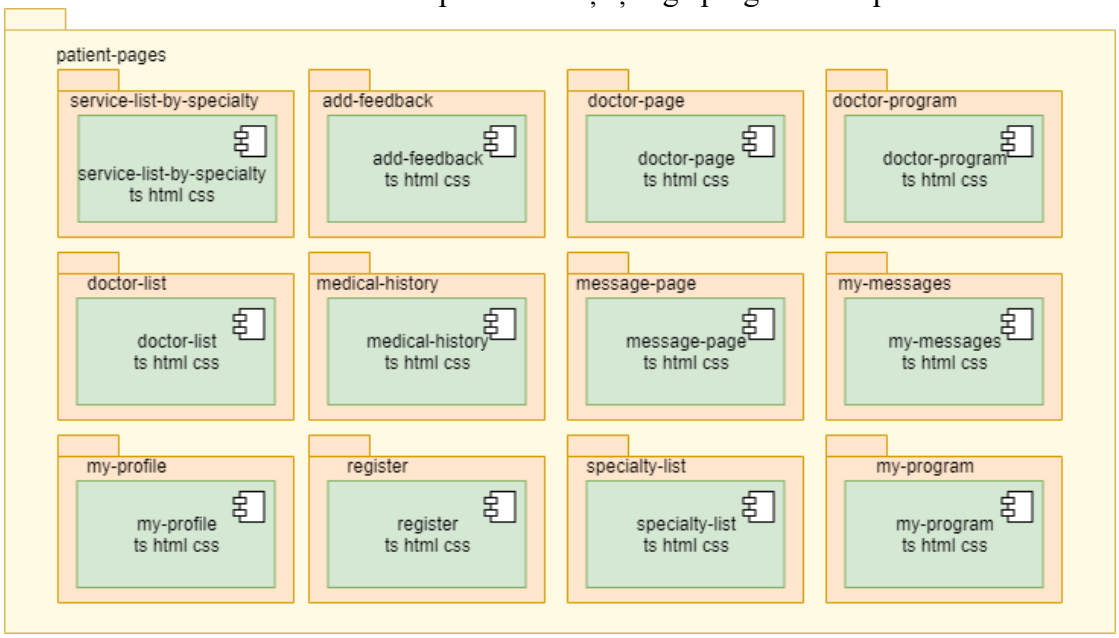
Figură 5.17 Diagrama de pachete a paginilor *doctor-pages*

Patient-pages

Pachetul conține toate componentele pe care utilizatorul pacient le va accesa. Aceastea sunt următoarele:

- *Service-list-by-specialty* – această componentă va afișa toate serviciile medicale din cadrul unei anumite specialități. Componenta poate fi selectată din pagina cu lista tuturor specializărilor, iar din această pagină se va putea ajunge la lista tuturor doctorilor care au serviciul selectat
- *Add-feedback* – prin această pagină un pacient va putea adăuga un feedback pentru un doctor. La această pagină va putea avea acces un pacient doar dacă a fost programat la acel doctor și nu a completat deja un feedback
- *Doctor-page* – această pagină va afișa profilul unui anumit doctor accesat din pagina cu lista întreagă de doctori. Ajungând aici, un pacient va avea mai multe funcționalități, precum: vizualizarea serviciilor medicale și a prețurilor, citirea feedback-urilor, accesul la programul unui medic, contactarea medicului în mod direct printr-un mesaj, iar dacă deja a avut o programare la acest doctor, va putea să îi lase un feedback
- *Doctor-program* – componenta doctor-program se va putea accesa din profilul unui medic sau din pagina care conține lista de doctori, afișând programul unui doctor. Tot în această pagină se vor putea adăuga, edita sau șterge programările personale ale pacientului înregistrat în aplicație
- *Doctor-list* – prin această pagină utilizatorul va putea să vizualizeze lista completă de doctori. Fiecare doctor va avea în dreptul său rating-ul bazat pe feedback-ul oferit de alți pacienți, va avea detalii despre nume, locație, specialitate și vor exista butoane prin care vei putea ajunge la paginile care conțin profilul personal, programul sau mesageria

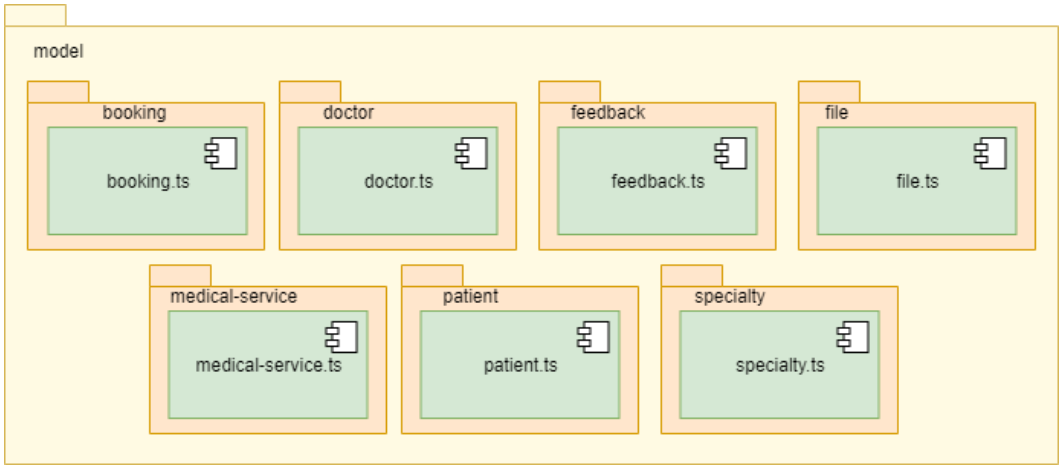
- *Medical-hisory* – componenta va conține istoricul medical al unui pacient, aici utilizatorul va avea acces la toate documentele sale medicale și le va putea descărca
- *Message-page* – accesand *message-page*, pacientul va intra în conversație cu un doctor, putând sa îi scrie mesaje
- *My-messages* – această pagină are rolul de a afișa toate mesajele și conversațiile unui anumit pacient cu unul sau mai mulți doctori
- *My-profile* – prin această pagină pacientul logat va putea să își accese profilul personal, va putea să își editeze datele personale, parola, va putea să își vadă istoricul medical și să își vadă programul
- *Register* – componenta este destinată înregistrării pacienților noi în aplicație. Pentru a se putea înregistra, pacientul va trebui să completeze emailul, parola, numele, numărul de telefon, cnp-ul, genul și ziua de naștere
- *Specialty-list* – prin această pagină un pacient va putea să vadă toate specialitățile existente în sistem. În dreptul fiecărei specialități se afișează numele și o descriere, iar pe lângă acestea vor exisa butoane prin care vei putea ajunge la pagina cu toți doctorii care au acea specializare selectată sau la toate serviciile care aparțin de acea specializare
- *My-program* – în această componentă se va afișa programul pacientului autentificat. Tot aici se va putea edita și șterge programările personale



Figură 5.18 Diagrama de pachete a paginilor patient-pages

Model

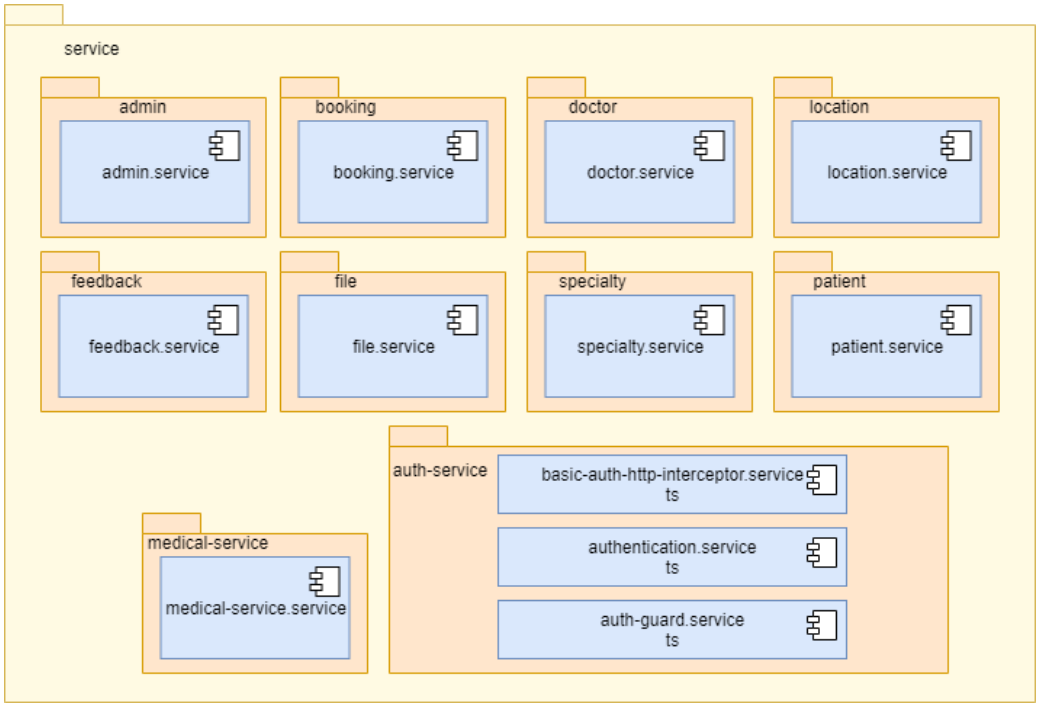
Acest pachet conține componentele model care corespund cu obiectele de transfer din partea de backend. Aceste componente sunt: *booking, doctor, feedback, file, medical-service, patient, specialty*. Toate aceste componente au fost utilizate pentru afișarea, citirea, editarea și ștergerea datelor prin intermediul comunicării cu partea de backend.



Figură 5.19 Diagrama de pachete a componentelor model

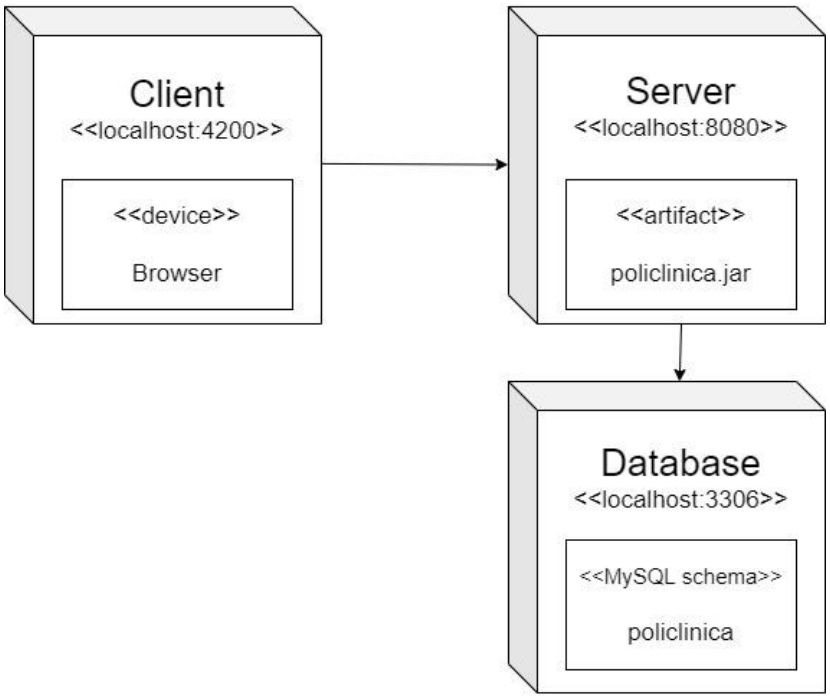
Service

Pachetul service conține toate componentele care au ca și scop apelarea endpoint-urilor din partea de backend. Majoritatea serviciilor au rolul de a apela endpoint-urile din componenta backend pentru citirea, adăugarea, editarea și ștergerea datelor. Pe lângă acestea, există și serviciile din pachetul auth-service. Acestea se ocupă de partea de autentificare și de securitate a interfeței. Serviciul authentication.service validează emailul și parola utilizatorului din pagina de login și dacă acestea sunt corecte atunci va crea o nouă sesiune unde se vor salva tokenul JWT pentru validarea apelurilor către back-end, emailul, id-ul utilziatorului auntetificat și rolul acestuia.



Figură 5.20 Diagrama de pachete a serviciilor

5.1.4. Diagrama de deployment



Figură 5.21 Diagrama de deployment a sistemului

După cum se poate observa in figura de mai sus, sistemul este compus din 3 părți:

- Partea de client – aceasta parte reprezintă componenta frontend dezvoltată în framework-ul Angular, portul pentru aceasta aplicație fiind 4200. Interfața va putea fi utilizată din orice browser
- Partea de server – acestă parte reprezintă partea de backend care a fost implementată în IntelliJ folosind SpringBoot, aceasta rulând pe portul 8080
- Partea de bază de date MySql care rulează pe portul 3306

5.2. Structura bazei de date

În acest subcapitol se vor prezenta tabelele din baza de date și coloanele din fiecare tabel și se va prezenta normalizarea bazei de date.

Normalizarea reprezintă descompunerea unui tabel relațional în mai multe tabele care satisfac anumite reguli și care stochează aceleași date ca și tabelul inițial. Baza de date se află în forma a treia de normalizare deoarece îndeplinește primele două forme, mai exact fiecare înregistrare reprezintă o valoare atomică care nu mai poate fi descompusă și oricare dintre atribute sunt dependente funcțional de totalitatea cheii primare. Pe lângă acestea, nu există dependințe funcționale tranzitive față de cheia

primară, toate atributele sunt mutual independente de cheia primară, sau altfel spus orice atribut dintr-o relație care nu este cheie depinde de cheie, de întreaga cheie și de nimic altceva decât de cheie.

În continuare se vor prezenta următoarele tabele: doctor, patient, admin, specialty, medical service, booking, location, file și feedback.

Doctor

Această tabelă conține detaliile despre un doctor. Tabela cuprinde atributele următoare:

- Id – aceasta este cheia primară a tabelului; tip de date: integer
- Email – adresa de email a doctorului; tip de date: varchar(100)
- First_name – prenumele doctorului; tip de date: varchar(50)
- Last_name – numele doctorului; tip de date: varchar(50)
- Password – parola doctorului încriptată; tip de date: varchar(255)
- Phone – numărul de telefon; tip de date: varchar(20)
- Specialty_id – specializarea doctorului; cheie străină care referențiază tabelul specialty; tip de date: integer
- Gender – genul doctorului; tip de date: varchar(10)
- Location_id – locația; cheie străină care referențiază tabelul location; tip de date: integer
- Image_url – link-ul către poza doctorului salvat în serviciul de stocare S3; tip de date: varchar(255)

Patient

Aceasta tabela definește toatele datele despre un pacient. Tabela cuprinde atributele următoare:

- Id – cheia primară; tip de date: integer
- Email – adresa de email; tip de date: varchar(100)
- First_name - prenumele; tip de date: varchar(50)
- Last_name - numele; tip de date: varchar(50)
- Password – parola incriptata; tip de date: varchar(255)
- Phone – numărul de telefon; tip de date: varchar(20)
- Cnp – codul numeric personal; tip de date: varchar(20)
- Gender – genul pacientului; tip de date: varchar(10)

Admin

Acest tabel cuprinde datele despre administratorii aplicației. Aceasta are următoarele atribute:

- Id – cheia primară a tabelului; tip de date: integer
- Email – adresa de email; tip de date: varchar(100)
- First_name - prenumele; tip de date: varchar(50)
- Last_name - numele; tip de date: varchar(50)
- Password – parola incriptata; tip de date: varchar(255)
- Phone – numărul de telefon; tip de date: varchar(20)

Specialty cuprinde informațiile despre specializările medicale ale sistemului. Tabelul conține atributele următoare:

- Id – cheia primară a tabelului; tip de date: integer
- Description – descrierea specialitatii; tip de date: varchar(500)

- Name – numele specialității; tip de date: varchar(50)
- Image_url – linkul către imaginea din servicul de stocare S3; tip de date: varchar(255)

Medical Service

Tabelul medical_service cuprinde detalii despre serviciile medicale. Acesta conține următoarele coloane:

- Id – cheia primară a tabelului; tip de date: integer
- Name – numele serviciului medical; tip de date: varchar(100)
- Price – prețul; tip de date: double
- Doctor_id – cheie străină care referențiază tabelul doctor; tip de date: integer
- Duration – durata serviciului medical; tip de date: integer

Booking

Această tabelă cuprinde lista tuturor programariilor realizate de pacienți. Tabelul conține următoarele coloane:

- Id – cheia primară a tabelului; tip de date: integer
- Description – descrierea programarii; tip de date: varchar(255)
- Name – numele programarii; tip de date: varchar(255)
- Start_date – data și ora de început; tip de date: datetime
- End_date – dat și ora de final; tip de date: datetime
- Patient_id – câmpul care definește ce pacient a facut rezervarea; cheie străină care referențiază tabela patient; tip de date: integer
- Doctor_id – câmpul care defineste doctorul la care s-a facut rezervarea; cheie straina care referențiază tabelul doctor; tip de date: integer

Location

Tabelul location conține informațiile despre locație. Acest tabel are următoarele date:

- Id - cheia primară a tabelului; tip de date: integer
- Name – numele locației; tip de date: varchar(50)

File

Acest tabel cuprinde date despre fișierele medicale ale pacienților. Tabelul are următoarele coloane:

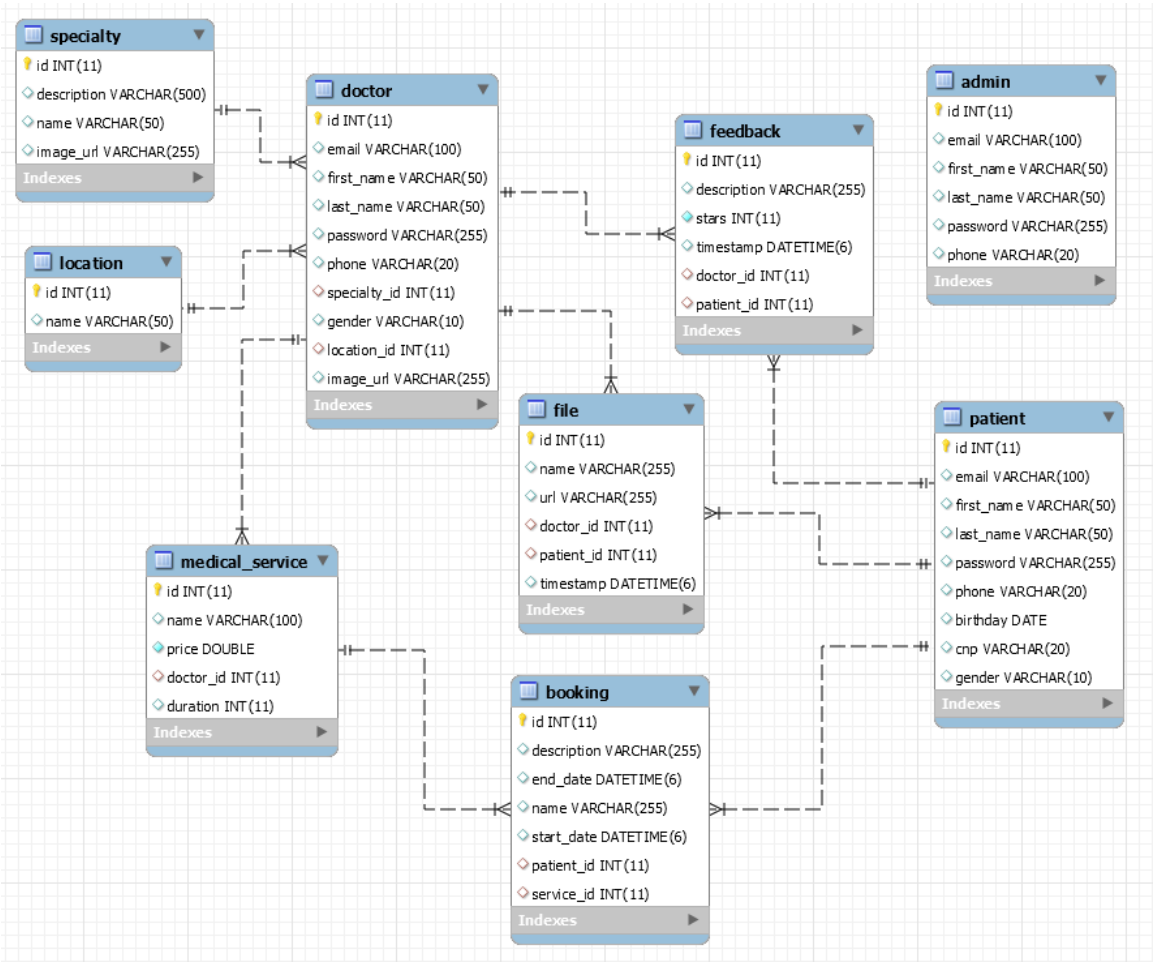
- Id – cheia primară a tabelului; tip de date: integer
- Name – numele fisierului; tip de date: varchar(255)
- url – linkul din serviciul de stocare S3; tip de date: varchar(255)
- doctor_id – câmpul care definește ce doctor a incarcat fisierul medial; cheie straina care referentiaza tabelul doctor; tip de date: integer
- patient_id – câmpul care definește pacientul care are fisierul medical; cheie străină care referențiază tabelul patient; tip de date: integer
- timestamp – data și ora la care a fost incarcat documentul medical; tip de date: datetime

Feedback

Tabelul feedback conține informațiile despre părerile pacienților despre doctori. Acesta conține datele următoare:

- id – cheia primară a tabelului; tip de date: integer

- description – descrierea feedback-ului; tip de date: varchar(255)
- stars – numărul de stele acordate de către un pacient; tip de date: integer
- timestamp – data și ora la care s-a scris comentariul; tip de date: datetime
- doctor_id – doctorul pentru care a fost scris feedback-ul; cheie străină care referențiază tabelul doctor; tip de date: integer
- patient_id – pacientul care a scris feedback-ul; cheie străină care referențiază tabelul patient; tip de date: integer



Figură 5.22 Diagrama bazei de date

Capitolul 6. Testare şi Validare

În acest capitol se vor prezenta tipurile de testare folosite pentru testarea componentei frontend și a componentei backend, se vor prezenta tehnologiile folosite și se vor da câteva exemple de teste. Testarea sistemului s-a realizat cu ajutorul testelor unitare, de integrare, de acceptanță și de performanță.

6.1.1. Testarea unitară

Testarea unitară [10] se focalizează pe testarea modulelor individuale față de specificațiile modulelor, generate ca parte a design-ului sistemului. Acesta permite scalarea, conduce la un design mai bun, ușurează efectuarea schimbărilor și previne apariția erorilor.

6.1.1.1. Tehnologii

Tehnologiile folosite pentru testarea unitară sunt Junit si Mockito.

Junit[31] reprezintă un framework pentru testare pentru limbajul Java. Junit a avut un rol important în implementarea testării test-driven development. Un unit test reprezintă un test care verifică funcționalitatea unei părți de cod, de exemplu metoda unei clase. Pentru a determina succesul metodei, în acest test vom compara rezultatul așteptat cu rezultatul dat printr-un assert.

Avantaje ale testării unitare: permite scalarea, conduce la un design mai bun, ușurează efectuarea schimbărilor, previne apariția unei erori, furnizează un ritm de lucru constant și specifică un comportament și furnizează o documentare a codului.

Mockito[32] reprezintă un framework open source pentru Java dezvoltat de MIT. Acesta permite crearea de teste care folosesc mock-uri, un fel de “dubluri”, fiind folosit la testarea TDD – Test Driven Development si la testarea BDD – Behaviour-Driven Development. Datorita faptului că obiectele sunt mock-uite, nu este nevoie de baza de date pentru a realiza testarea, deoarece funcționalitatea este testată în izolare. Doar dacă dorim să realizam teste integrate, atunci se va folosi o bază de date din memorie, nu efectiv cea reală.

6.1.1.2. Exemple

Exemplul de test unitar este pentru cazul în care se dorește returnarea tuturor doctorilor din aplicație care au o anumită specialitate. Pentru asta, pentru a folosi Mockito, pentru clasele care trebuie “dublate” s–au folosit adnotările @InjectMocks și @Mock, iar înainte de fiecare test acestea au fost initializate prin metoda statică din libraria mockito MockitoAnnotations.initMocks(this);. Exemplul de test este următorul:

```
@Test public void getBySpecialtyId_Success() {
Mockito.when(specialtyRepository.findById(any())).thenReturn(Optional.of(new
Specialty()));
Mockito.when(doctorRepository.getAllBySpecialtyId(any())).thenReturn(doctors)
try {
```

[31] https://junit.org/junit4/
[32] https://site.mockito.org/

```
            assertEquals(list, doctorService.getBySpecialtyId(1));
        } catch (UiException e) {
            fail();
        }
```

6.1.2. Testarea de integrare

Prin testarea de integrare [10] se testează un grup de unități cu scopul de a identifica defectele datorate interacțiunii dintre unitățile de integrare.

6.1.2.1. Tehnologii

Tehnologiile utilizate pentru testarea de integrare sunt MockMvc, Junit, Spring Boot si baza de date H2, acestea urmând să fie prezentate:

MockMvc[6] reprezintă o modalitate de a realiza testele integrate, modalitate prin care nu este necesară pornirea serverului. În această abordare, Spring se ocupă de call-ul HTTP și îl trimite către controller. Astfel, codul se apelează la fel cum s-ar apela în realitate, dar fară a se mai porni serverul. Pentru a folosi mockMvc trebuie să punem annotarea @AutoConfigureMockMvc.

Și în acest caz se va folosi framework Junit prezentat mai sus. De asemenea, se folosește funcționalitatea Spring Boot-ului de a realiza teste integrate. Spring Boot are un modul dedicat acestor teste, fiing cunoscut ca și spring-test. Pentru aceste teste s-a introdus prin maven librăria spring-boot-starter-test. Pentru a folosi testele integrate, s-a folosit annotarea @SpringBootTest.

Baza de date H2[33] reprezintă o bază de date Java open-source. Această bază de date s-a construit în memorie și a fost folosită pentru a înlocui baza de date reală a aplicației pentru testele integrate. Ca și avantaje, această bază de date este foarte rapidă, are dimensiuni mici, suportă SQL și este securizată.

6.1.2.2. Exemple

Pentru testarea de integrare se va ilustra exemplul în care un utilizator se înregistrează în aplicație. Pentru a realiza testarea de integrare, trebuie scrise următoarele adnotari la nivel de clasă de test: @RunWith(SpringRunner.class), @SpringBootTest, @ActiveProfiles(“test”), @AutoConfigureMockMvc, @WithMockUser. Pentru a realiza testele, se creează o metodă care se va rula înainte de fiecare test și se inițializează anumite obiecte. Astfel, în metoda setup vom adăuga un pacient în baza de date din memorie făcând un post call la endpoint-ul register. După fiecare test se vă executa metoda cleanUp care vă stergge orice pacient din baza de date H2.

```
@After public void cleanUp() throws Exception {
patientService.getAll().forEach(t -> patientService.deleteById(t.getId()));
}
```

Exemplu de test de integrare este următorul:

```
@Test public void successfullyRegister() throws Exception {
PatientDto patientDto = new
PatientDto(„email2@email.com”, ”password1”);
Patient patient = new Patient(„email2@email.com”, ”password1”);
```

[33] https://www.h2database.com/html/main.html

```
mvc.perform(post("/register").header("Origin","*")
.contentType(MediaType.APPLICATION_JSON)
.content(asJsonString(patientDto))).andExpect(status().isOk());

byte[] response = mvc.perform(get("/patient/email").header("Origin","*")
.param("email",patientDto.getEmail())).andExpect(status().isOk())
.andReturn() .getResponse().getContentAsByteArray();
Patient patientResponse = objectMapper.readValue(response, new
TypeReference<Patient>() { });
Assert.assertEquals(patient, patientResponse);
}
```

6.1.3. Testarea de acceptanță

Acceptance testing reprezintă activitatea efectuată de utilizatorii finali pentru a valida că software-ul respectă specificațiile și așteptările aplicației. Acest test este aplicat pe metoda din controller, pe endpoint-uri.

6.1.3.1. Tehnologii

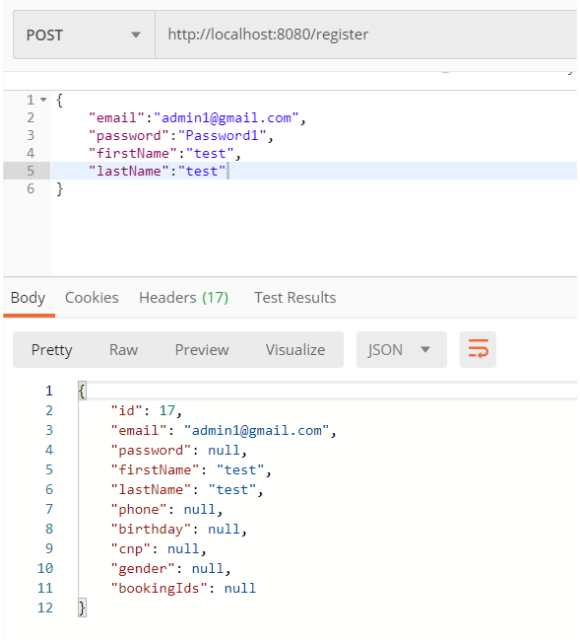
Postman[34] reprezintă o unealtă pentru testarea API-urilor unui sistem. Acesta are o interfață plăcută pentru construirea apelurilor și pentru citirea răspunsurilor și se utilizează pentru testarea părții backend pentru a verifica fapul că aplicația funcționează cum trebuie. Pentru a testa componenta backend am ales să utilizez acest sistem pentru ușurința folosirii acestei unelte. Cu ajutorul ei am testat endpoint-urile create pentru a verifica dacă funcționează corect și răspunsurile sunt cele așteptate. Acest mijloc de testare a fost foarte folositor deoarece a fost de ajutor în momentul în care implementam un apel și doream să îl testez. Pe lângă testarea endpoint-urilor după crearea acestora, am folosit Postman și în momentul în care am implementat partea de frontend a aplicației în momentul în care ceva nu merge corespunzător.

6.1.3.2. Exemple

Exemplul de testare de acceptanță este reprezentat de cazul în care utilizatorul se înregistrează în aplicație. Ca și criterii de acceptanță avem: utilizatorul trebuie să își creeze cont pe baza unui email și a unei parole. Pe lângă acestea, trebuie să mai completeze nume, prenume, cnp, adresă, telefon și gen. Ca și validări pentru acestea avem următoarele: adresa de mail trebuie să fie o adresă validă și nu trebuie să depășească 100 de caractere, parola trebui să fie una validă, să aibă minim 8 caractere și să conțină literă mică și mare, numele și prenumele trebuie să aibă maixm 50 de caractere, cnp-ul trebuie să aibă maxim 15 caractere.

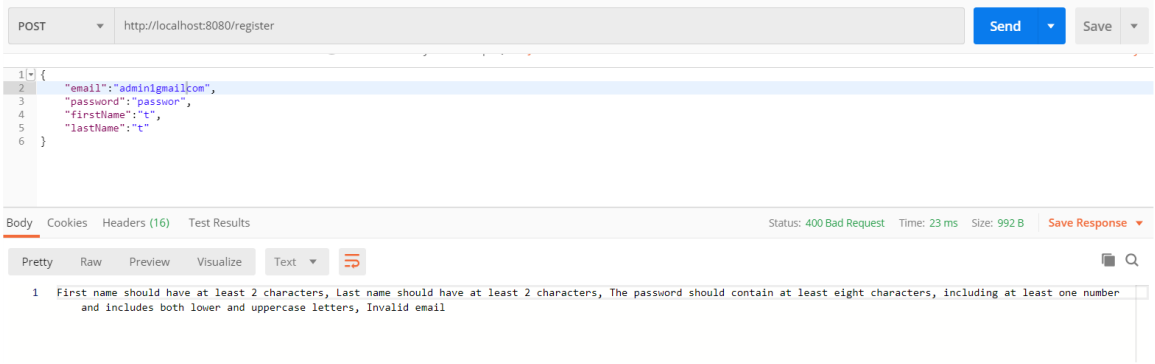
În figura următoare se prezintă cazul în care un utilizator s-a înregistrat cu succes.

[34] https://www.postman.com/



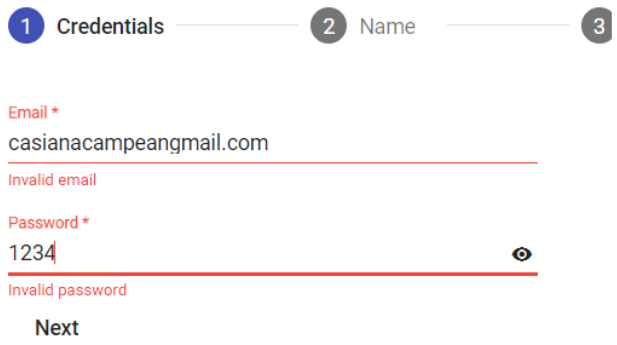
Figură 6.1 Exemplu de testare în Postman pentru înregistrarea cu succes

Un caz care ar provoca o eroare este cel în care utilizatorul dorește să introducă o adresă de email greșită sau o parolă, nume, prenume greșite, care nu respectă criteriile de acceptanță.



Figură 6.2 Exemplu caz de testare pentru exemplificarea mesajului de eroare

De asemenea aceleași validări sunt existente și pe partea de frontend a aplicației, validări ilustrate în figura următoare.



Figură 6.3 Exemplu de afișare a erorilor în aplicație

6.1.4. Testare de performanță

Apache JMeter[35] este un proiect Apache folosit pentru realizarea testelor de performanță pentru servicii, în special pentru aplicațiile web. Acesta este gratuit, implementat in limbajul Java, dezvoltat pentru a realiza teste de performanta. Printre functionalitati se numara urmaotarele:

- Abilitatea de a testa diferite aplicații, servere sau tipuri de protocoale precum: Web (Java, NodeJS, PHP, ASP.NET, etc.), SOAP/REST, FTP, SMTP, TCP și altele
- Posibilitatea de a extrage rezultatele și datele în diverse formate, precum: HTML, JSON, XML, CSV si multe altele
- Existența rulării testelor pe mai multe thread-uri concurente
- șIntegrarea continuă prin servicii externe precum Maven, Gradle ți Jenkins

Un exemplu de testare de performanță reprezintă cel pentru cazul în care se realizează vizualizarea programului unui doctor. Pentru a realiza acest test a trebuit creat un *Test Plan*, un *Thread Group*, un *HTTP Request* unde a trebuit să se completeze protocolul, ip-ul, portul, metoda și path-ul care trebuie testat, acesta fiind /booking/doctor/id=200 și nu în ultimul rând a trebuit creat un *HTTP Header Manager* unde a trebuit inclus tokenul potrivit și valabil. După care a trebuit modificate valorile din *Thread Group* pentru *Number of Threads(users), Ramp-up Period(in seconds)* și pentru *Loop Count*. Primul câmp reprezintă numărul de utilizatori care apelează endpoint-ul, următorul câmp reprezintă numărul de secunde până când JMeter va influența perioada de secunde între pornirea thread-urilor și ultimul reprezintă de câte ori se vor executa testele. Dacă de exemplu avem 30 de thread-uri și perioada de ramp-up este de 120 de secunde, atunci fiecare thread va fi întârziat cu 4 secunde (120/30).

Am realizat testele pentru 1000 si 10000 de utilizatori, primul având ramp-up period de 10 și al doilea de 100, iar rezultatele sunt următoarele:

[35] https://jmeter.apache.org/

Tabel 6.1 Tabel rezultate teste de performanță folosind JMeter

Samples	Average	Min	Max	Std. Dev	Error%	Troughput	Received KB/sec	Sent KB/Sec	Avg Bytes
3000	57	36	135	8.3	0	26.7	77.5	10.09	2971
10000	87	30	1440	103.7	0	98.8	268.6	37.34	291

Primul rând reprezintă valorile pentru 1000 de utilizatori, iar cel de-al doilea sunt valorile pentru 10000 de utilizatori. Samples reprezintă numărul de apeluri trimise, Average reprezintă media timpului de răspuns, Min și Max reprezintă timpul minim și maxim de răspuns, Std. Dev reprezintă deviația standard, diferența între valorile minime și maxime față de medie, Error este procentul de teste cu erori și Troughput reprezintă câte request-uri pe secundă suportă sistemul. Ultimele trei câmpuri fiind câte date se trimit și se primesc și media mărimii răspunsurilor primite. Un ultim scenariu de test a fost realizat pentru 100000 de utilzitori și o perioadă de ramp-up de 1000, dar aceste teste au durat foarte mult, dar după o perioada de timp maximul procenatj de erori a fost de doar 4%.

Capitolul 7. Manual de Instalare si Utilizare

În acest capitol se prezintă manualul de instalare al aplicației, resursele nececare pentru a instala aplicația și se prezintă manualul de utilizare, explicând pe rând funcționalitățile fiecărui utilizator.

7.1. Manual de instalare

Pentru a putea folosi aplicația sunt necesare următoarele resurse:

- Spatiul de disk necesar sistemului de 400MB liberă pentru a instala partea de backend si frontend si inca 250MB pentru a instala librariile necesare pentru interfață
- Laptop sau calculator sau mașină virtuală pe care să putem instala și folosi aplicația
- Memorie RAM de minim 4GB, de preferat 8GB
- Orice sistem de operare, dar de preferat Windows deoarece în acest sistem de operare a fost dezvoltată aplicația
- Conexiune la internet
- Orice browser undeva se va putea deschide aplicația web

Pentru a putea instala și rula aplicația sunt necesare următoarele instrumente software:

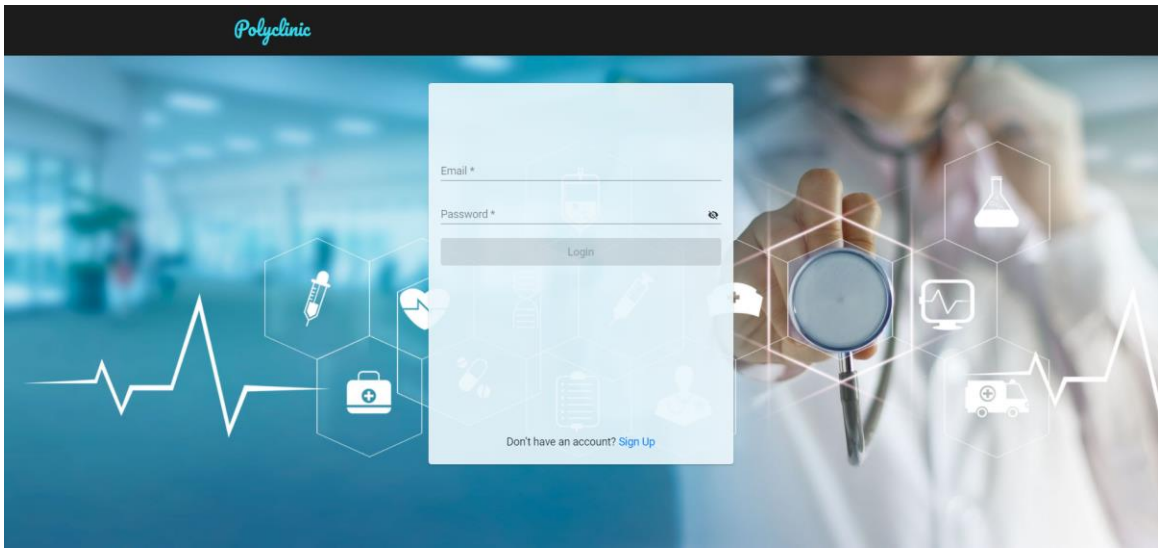
- **IntelliJ IDEA** – pentru a putea deschide aplicația, este necesar mediul de dezvoltare care poate fi descărcat de pe următorul site, iar versiunea necesară este cea Ultimate: https://www.jetbrains.com/idea/download/#section=windows. După descărcare, se va putea deschide cu ușurință proiectul aflat pe calculator.
- **MySQL Workbench** – programul se poate descărca de pe site-ul urmator: https://dev.mysql.com/downloads/workbench/. Mai apoi, odată descărcată aplicația se va creea o schemă numită "policlinica" și se vor rula script-urile care se află în pachetul database-script din aplicația backend.
- **Java JDK** – pentru a putea rula proiectul este necesar de kit-ul de dezvoltare Java găsit pe site-ul urmator: https://www.oracle.com/java/technologies/javase-jdk11-downloads.html
- **NodeJS** – pentru a putea rula aplicația frontend avem nevoie de NodeJs care poate fi descărcat de aici: https://nodejs.org/en/download/.
- **Angular** – pentru a instala angular, trebuie rulată intr-un terminal comanda urmatoare: „npm install -g @angular/cli".

După ce toate instrumentele au fost instalate, prima dată se va deschide proiectul din intelliJ, după care se vă rula partea de backend. Mai apoi, se va rula comanda "ng serge –open", care va porni aplicația în browser. Pe lângă această comandă, trebuie introdusă și comanda "node server.js" pentru a porni partea de mesagerie din aplicație. Având totul pornit, ne putem autentifica sau înregistra în aplicație.

7.2. Manual de utilizare

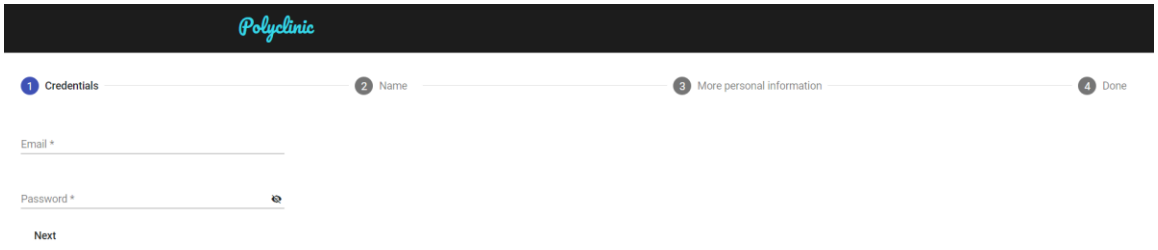
7.2.1. Funcționalități utilizatori

Aplicația dezvoltată poate fi utilizată de 3 tipuri de utilizatori, mai exact de administrator, pacient și doctor. Majoritatea functionalitatilor din cadrul aplicațiilor sunt diferite, dar unele sunt comune pentru toți. Aceste pagini sunt cea de înregistrare, autentificare și *Home*. Odată ce deschizi aplicația într-un browser, prima pagină care poate fi accesată este cea de login. Pentru a te autentifica în aplicație, trebuie să introduci emailul și parola și să apeși pe butonul de Login, funcționalități care apar în figura următoare.



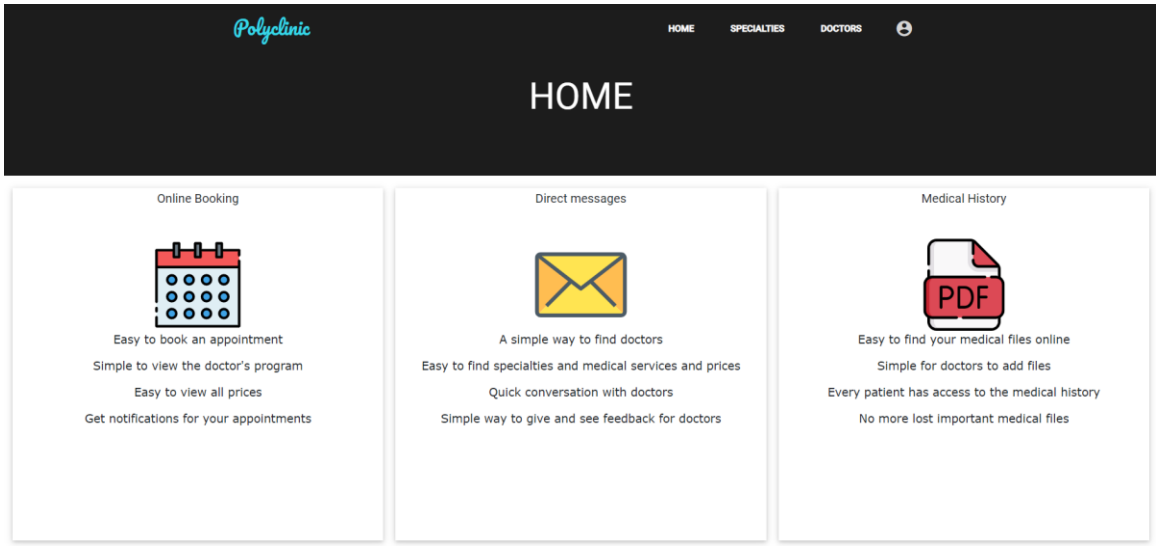
Figură 7.1 Pagina de login a aplicației

Pentru a te înregistra în aplicație, trebuie să apeși pe butonul de Sign up din pagina de login, după care vei fi redirecționat către pagina de înregistrare, ilustrată în figura de mai jos. Aici trebuie să introduci emailul, parola, numele, telefonul, data nașterii, genul și CNP-ul și să apeși pe butonul de Save. După ce se apasă pe butonul save, noul pacient va fi redirecționat către pagina de login.



Figură 7.2 Pagina de înregistrare

După ce utilizatorul este autentificat cu succes, prima pagină pe care o va putea vizualiza este pagina de Home, unde sunt descrise funcționalitățile aplicației.



Figură 7.3 Pagina *Home* a aplicației

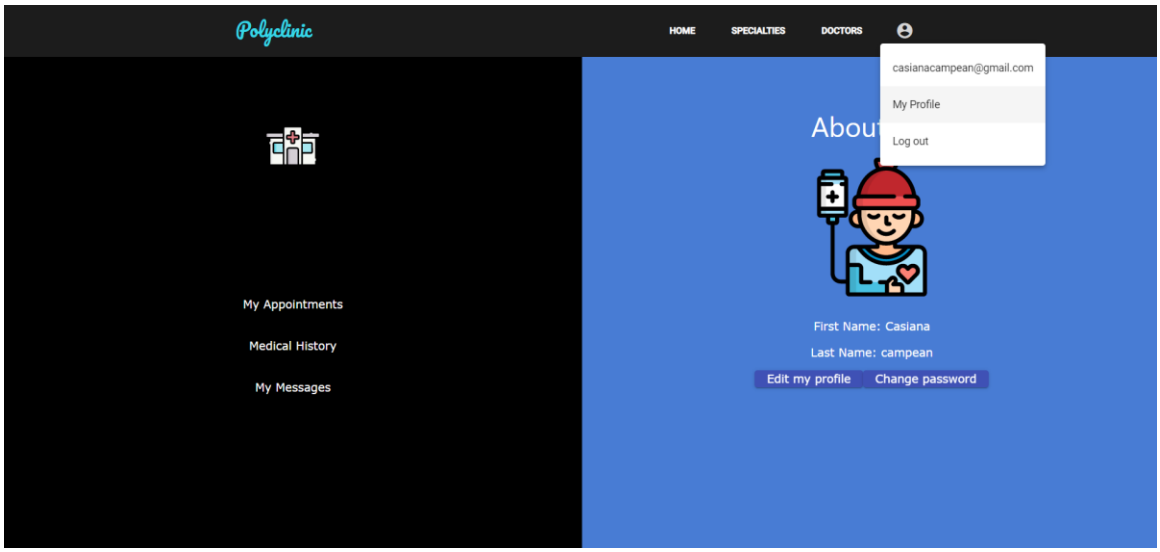
7.2.2. Funcționalități doctor și pacient

7.2.2.1. Funcționalități comune

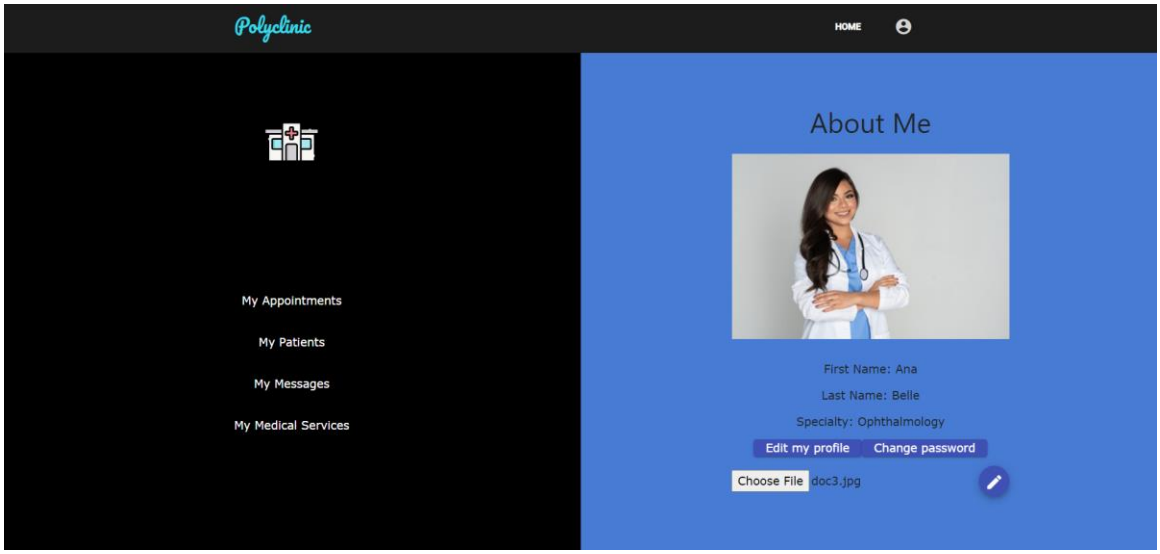
Pe lângă paginile utilizate de orice utilizatori, exista pagini comune doar pentru medic și pacient, aceastea fiind paginile de *My profile, Medical History, My appointments, Edit my profile, My messages* si *Change password.*

Pagina *My profile* poate fi accesată apăsând iconița din header și mai apoi apăsând butonul “My profile”. Aici există mai multe funcționalități pentru medic și pacient. Înafară de opțiunile comune menționate mai sus, funcționalitățile suplimentare pentru medic sunt: editarea pozei de profil, accesarea pacienților, a istoricului medical, accesarea serviciilor medicale și gestionarea acestora.

În cele doua figuri de mai jos sunt ilustrate paginile de profil ale pacienților și ale doctorilor.

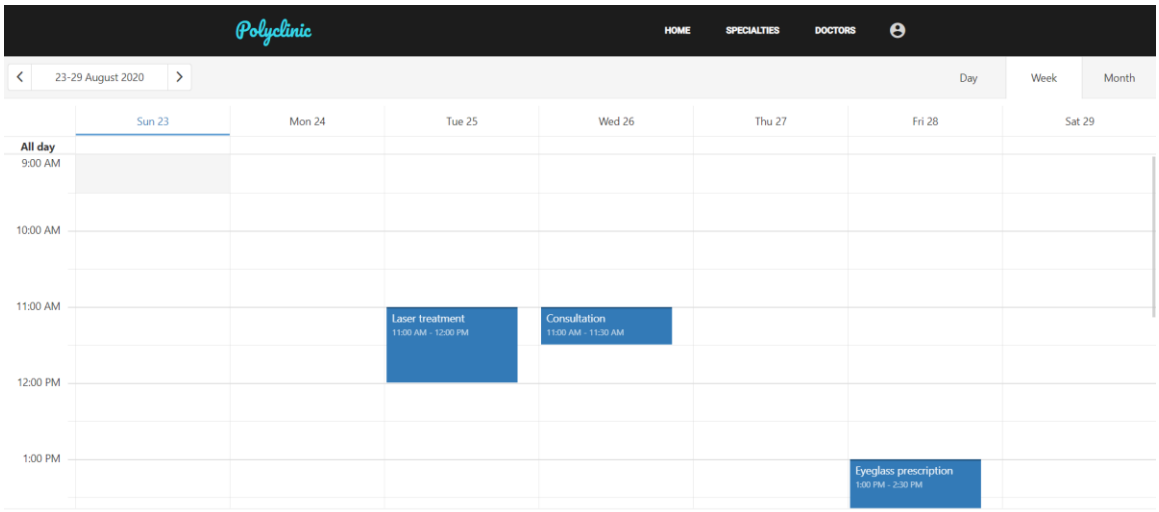


Figură 7.4 Pagina de My profile a unui pacient



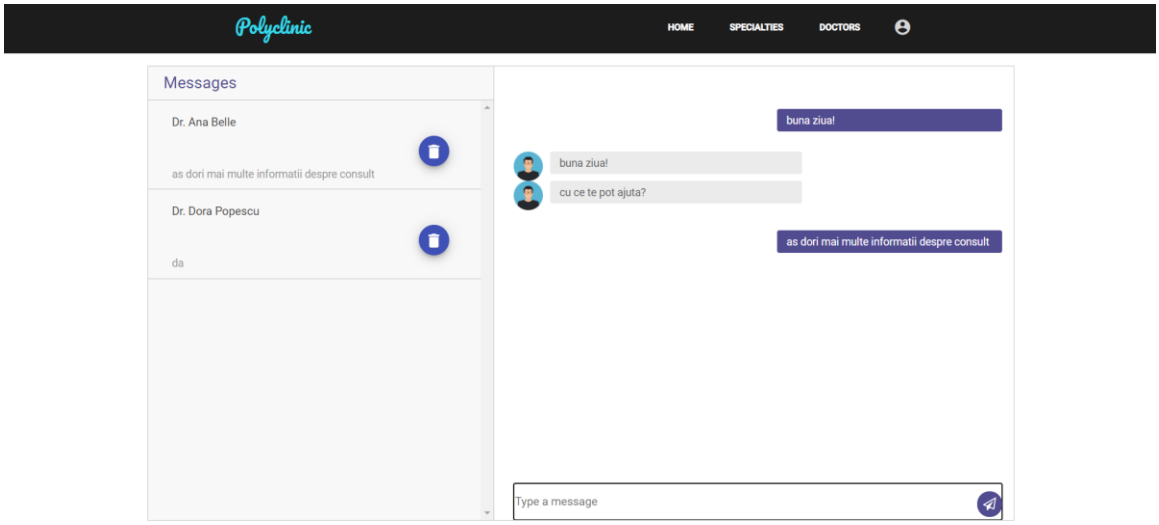
Figură 7.5 Pagina de My profile a unui doctor

Prima funcționalitate comună reprezintă vizualizarea propriului program, acesta putând fi vizualizat prin apăsarea butonului „My Appointments”. Apăsând acest buton, utilizatorul vă fi redirecționat către programul său, compus dintr-un calendar, unde vor apărea programările făcute. Pentru orice pacient, acesta are posibilitatea de a edita sau a șterge programarea făcută. Pe lângă asta, există opțiunea de a vizualiza programul pe zi, pe săptămână sau pe lună. Modul în care arată programul este ilustrat mai jos în figură.



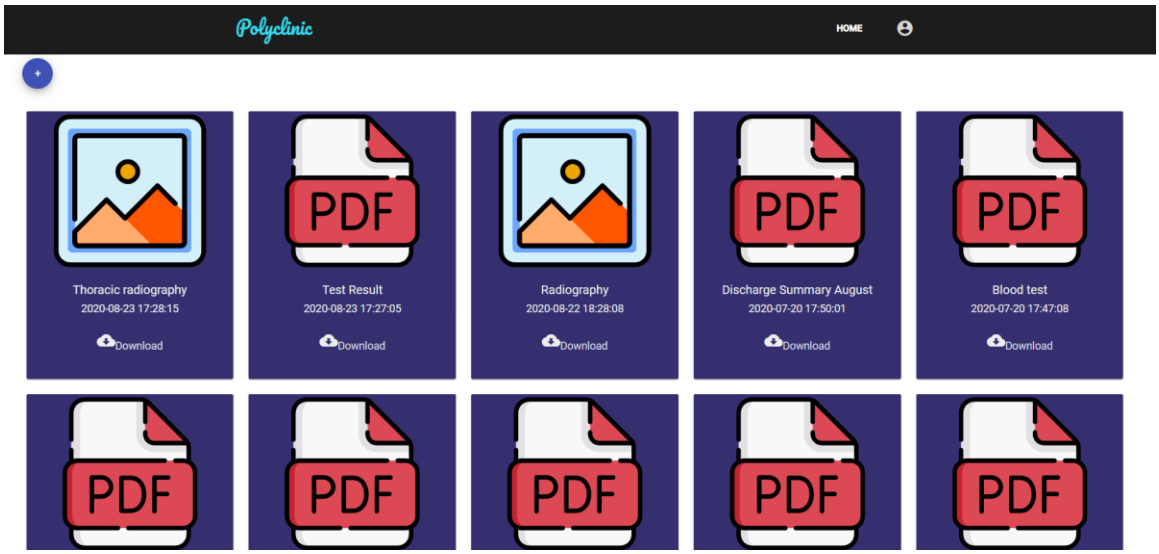
Figură 7.6 Pagina cu programul unui pacient sau al unui medic

Următoarea pagină comună reprezintă mesageria. Aceasta poate fi accesată apăsând butonul „My messages”. Aici vor fi apărea toate mesajele unui pacient sau ale unui medic. În următoarea figură se ilustrează mesageria unui pacient.



Figură 7.7 Pagina cu mesageria unui pacient

O altă pagină accesibilă din profil și comună reprezintă Medical History. Această pagină poate fi accesată direct din meniul unui pacient apăsand pe butonul "Medical History" sau poate fi accesată din meniul unui doctor apăsând pe butonul „My patients", după care apăsând pe iconița pentru fișierele unui pacient din listă. În figura următoare este ilustrat istoricul medical al unui pacient accesat de către un doctor, doar un doctor având opțiunea de adăugare a unui fișier medical.

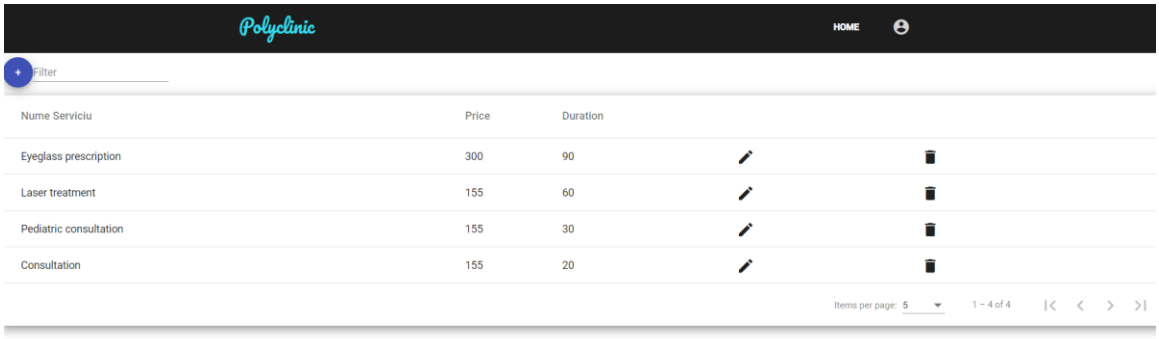


Figură 7.8 Pagina cu istoricul medical al unui pacient

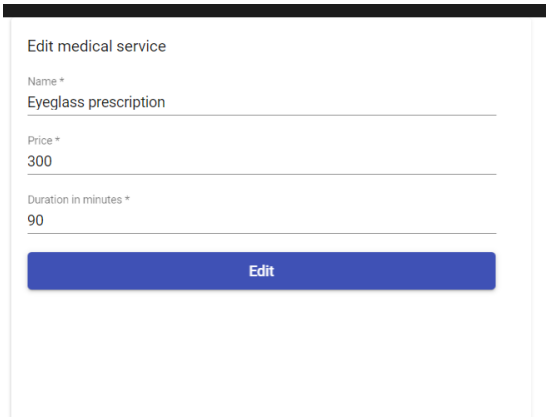
7.2.2.2. Funcționalități doctor

Ca și funcționalități suplimentare, un medic poate să își gestioneze serviciile medicale apăsând pe butonul „My medical services", astfel va fi redirecționat către pagina care conține lista tuturor serviciilor sale medicale, pe care le poate edita, șterge și poate adăuga alte servicii.

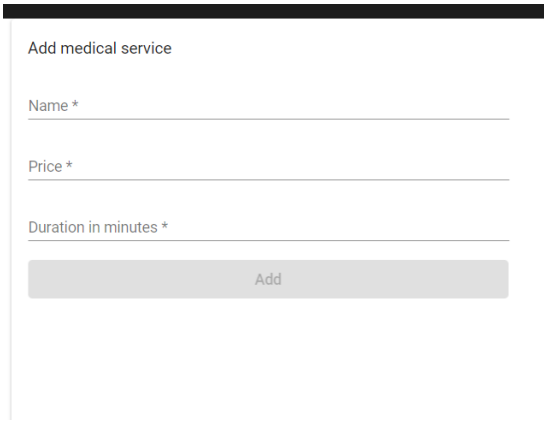
Figura următoare reprezintă lista cu serviciile medicale ale unui doctor, apăsând pe butonul „+” se poate adăuga un nou serviciu medical, buton care va duce către pagina reprezentată de figura 7.11. Apăsând pe iconița cu un creion, serviciile se pot edita, pagina ilustrată în figura 7.10, iar ultima iconiță va șterge serviciul medical din listă.



Figură 7.9 Pagina cu lista serviciilor medicale ale unui doctor



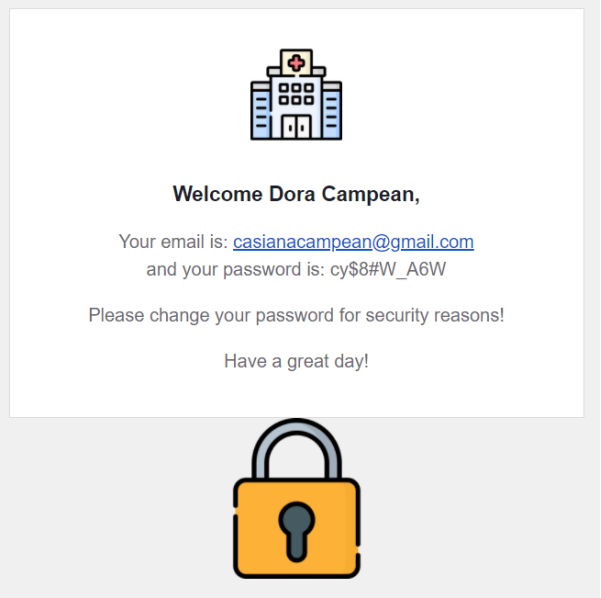
Figură 7.10 Editarea unui serviciu medical



Figură 7.11 Adăugarea unui serviciu medical

O ultimă opțiune pe care o are un medic este primirea unui email în momentul în care este adăugat în sistem. Acesta primește pe mail un mesaj de bun venit în care i se

transmite emailul și parola temporară și mesajul de a schimba parola din motive de securitate. Emailul este ilustrat în figura următoare:

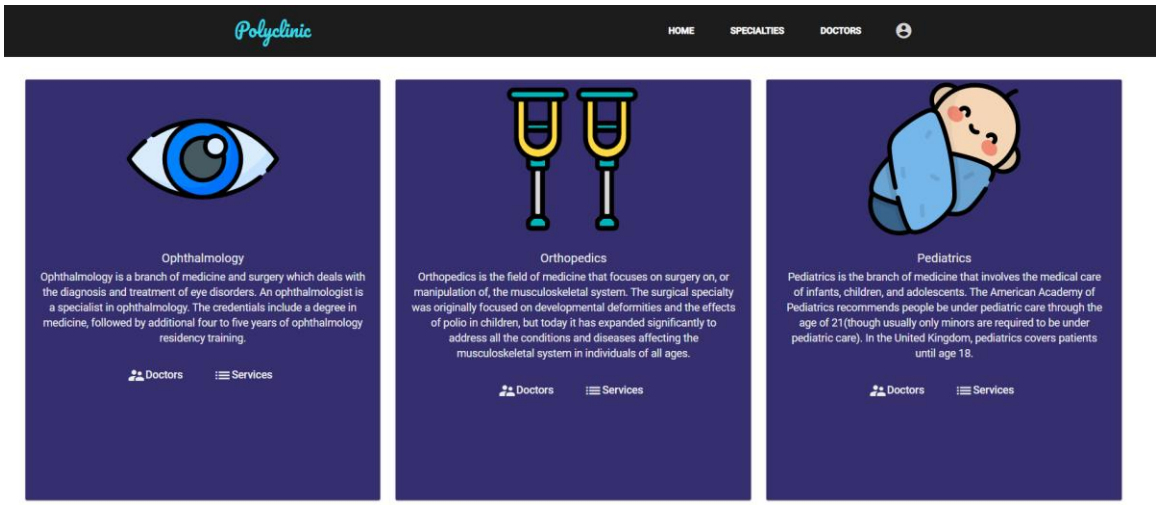


Figură 7.12 Mesajul din mail-ul unui medic în momentul în care a fost adăugat

7.2.2.3. Funcționalități pacient

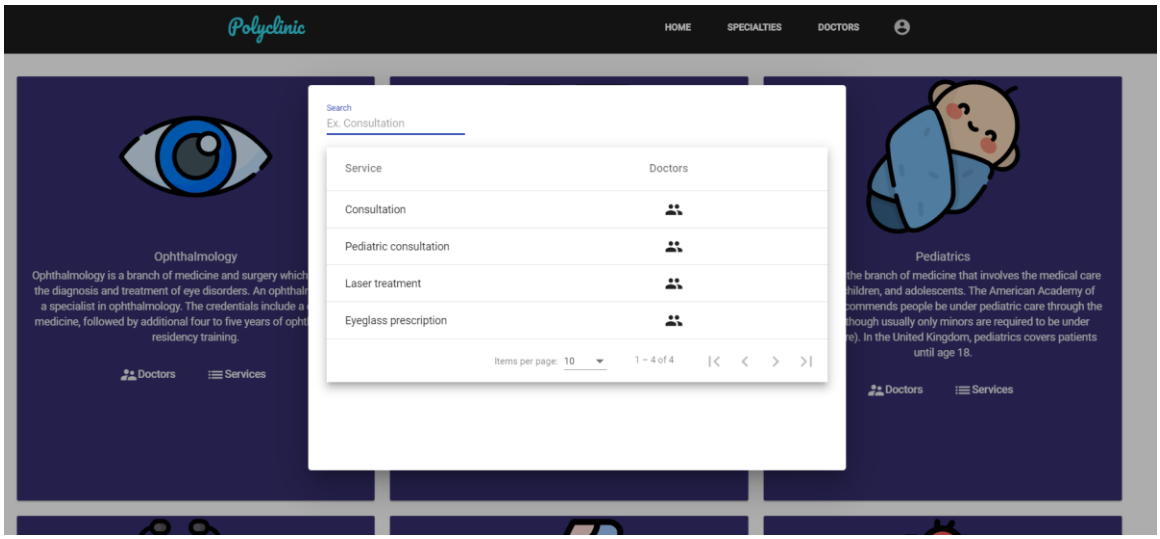
Pe lângă funcționalitățile comune menționate mai sus, pacientul dispune de următoarele opțiuni: vizualizare listă specialități, vizualizare listă medici, vizualizare doctori în funcție de specializare, locație și serviciu medical, vizualizare servicii medicale în funcție de specialitate, vizualizare profil medic, adăugare, editare și ștergere rezervare și primirea de notificări înainte de o programare.

Prima funcționalitate reprezintă cea de vizualizare specialități. Pentru a ajunge la această pagină trebuie să se apese pe butonul „Specialties” din partea de header. Figura următoare reprezintă această pagină.



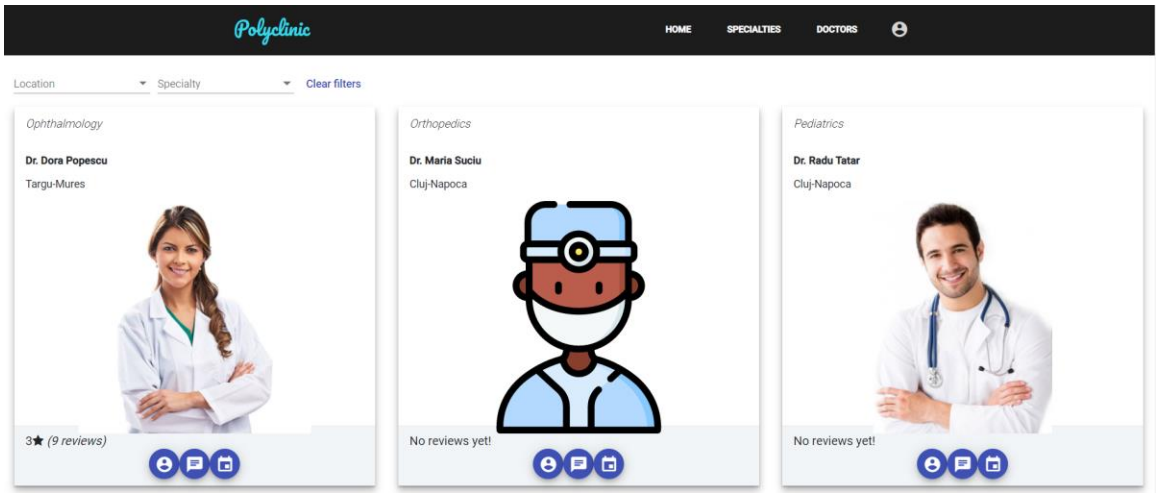
Figură 7.13 Pagina cu specializari

După cum se poate observa în figură, în dreptul fiecărei specialități se poate citi descrierea și se poate apăsa pe butonul doctors care te va redirecționa către pagina tuturor doctorilor care aparțin de această specialitate. Butonul „Services” va afișa lista tuturor serviciilor medicale care aparțin de acea specialitate. Daca apăsam pe butonul respectiv din dreptul specializării de oftalmologie, se va deschide pagina din figura următoare:



Figură 7.14 Lista serviciilor medicale ale specializarii oftalmologie

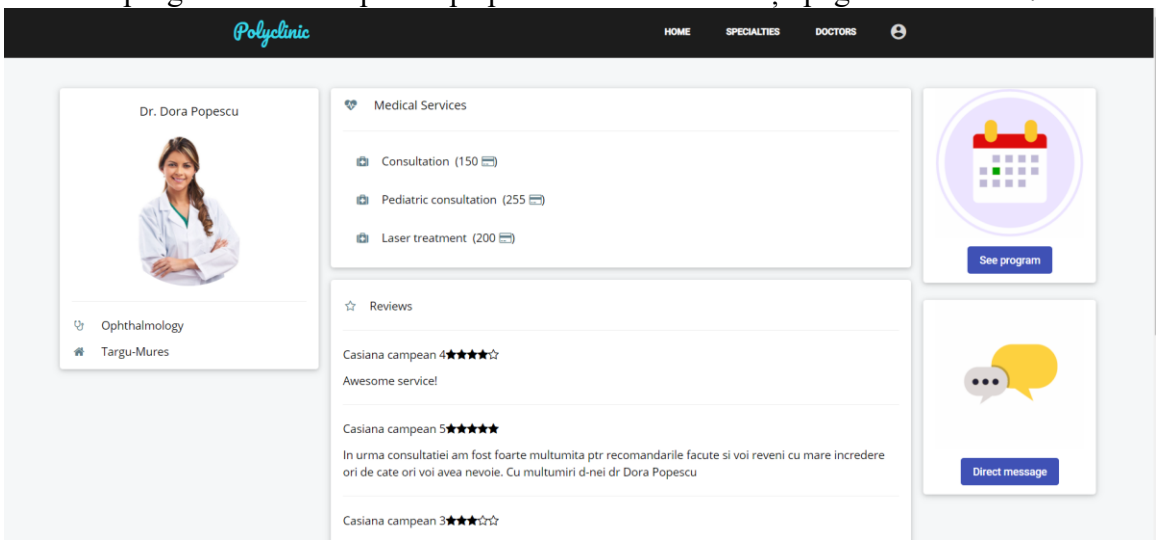
În dreptul fiecărui serviciu, putem apăsa butonul din dreptul coloanei „Doctors” care ne va redirecționa către pagina cu toți doctorii care oferă acel serviciu medical. Aceasta pagină cu doctori se poate accesa și din partea de antet, apăsând pe butonul „Doctors”, astfel pacientul va fi redirecționat spre pagina care conține toți medicii din sistem. Această pagină este ilustrată în figura urmatoare:



Figură 7.15 Lista medicilor

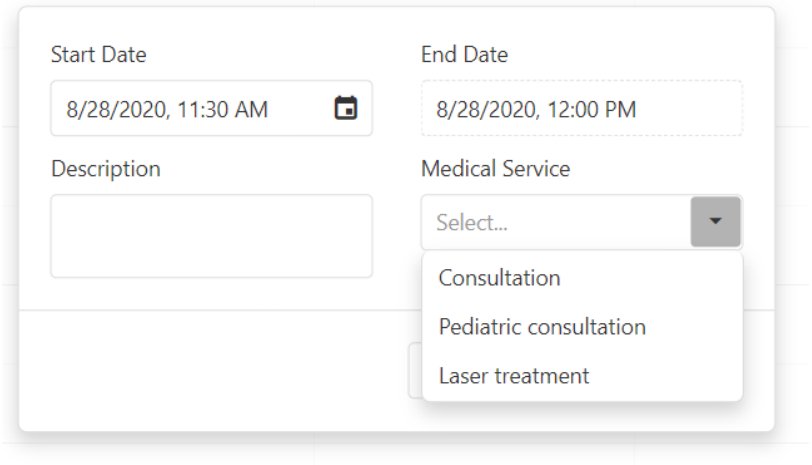
În această pagină, se pot filtra medicii în funcție de locație și specializare. În dreptul fiecărui medic sunt 3 butoane, primul vă deschide pagina de profil a unui medic,

al doilea va deschide o conversație directă cu acesta în pagina de mesagerie și ultimul deschide programul. Daca apăsam pe primul buton se va afișa pagina următoare:



Figură 7.16 Pagina de profil a unui medic

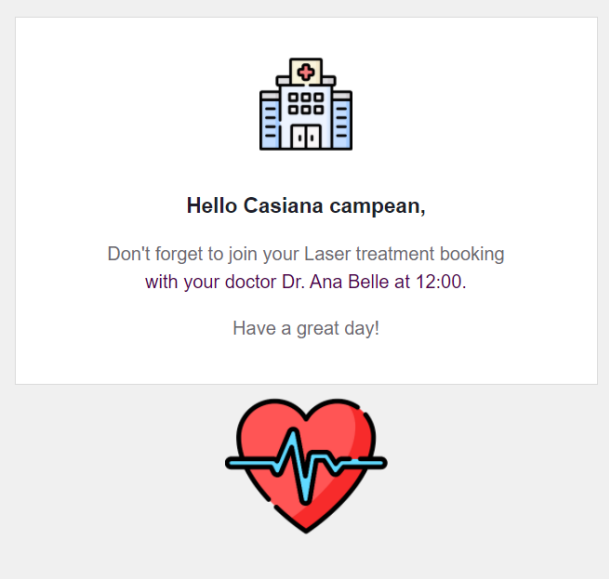
În această pagină de profil se pot vizualiza serviciile medicale oferite, prețul acestora, recenziile altor pacienți și se poate ajunge și din această pagină către pagina de mesagerie sau către programul doctorului. Pentru a realiza o programare, trebuie apăsat pe butonul de program și în pagina următoare trebuie făcut dublu click în dreptul orei dorite, după care selectat serviciul medical. Făcând dublu click, se vă deschide următorul dialog:



Figură 7.17 Realizarea unei programari

Din acest dialog se trebuie selectat serviciul medical dorit și se poate menționa o descriere. De asemenea se poate schimba și ora de început, ora de final se generează automat în funcție de durata serviciului medical. În cazul în care pacientul dorește să selecteze o oră ocupată, se va afișa mesajul următor într-un diaog: „The doctor already has an appointment în the requested period of time! Please select another time!”.

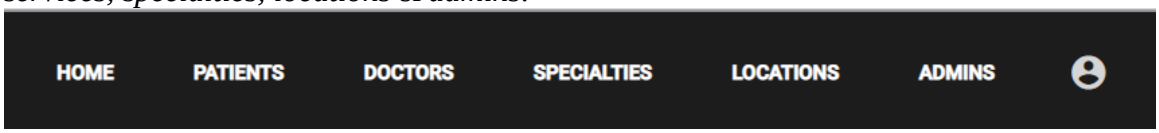
O ultimă funcționalitate a unui pacient este primirea de notificări pe mail înainte de programări. Această notificare va fi afișată pe email astfel:



Figură 7.18 Notificare din email pentru o programare a unui pacient

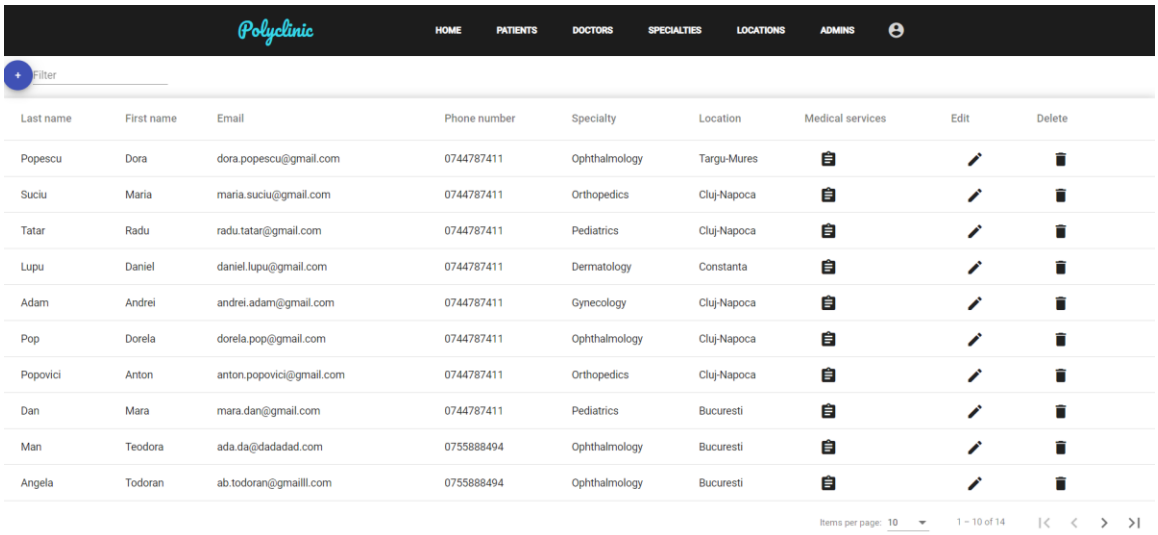
7.2.3. Funcționalități administrator

Ultimul actor al aplicației reprezintă administratorul care se ocupă cu gestionarea sistemului. Funcționalitățile sale sunt reprezentate de opțiunile din partea de header. Paginile disponibile pentru administrator sunt următoarele: *patients, doctors, doctor-services, specialties, locations* si *admins*.



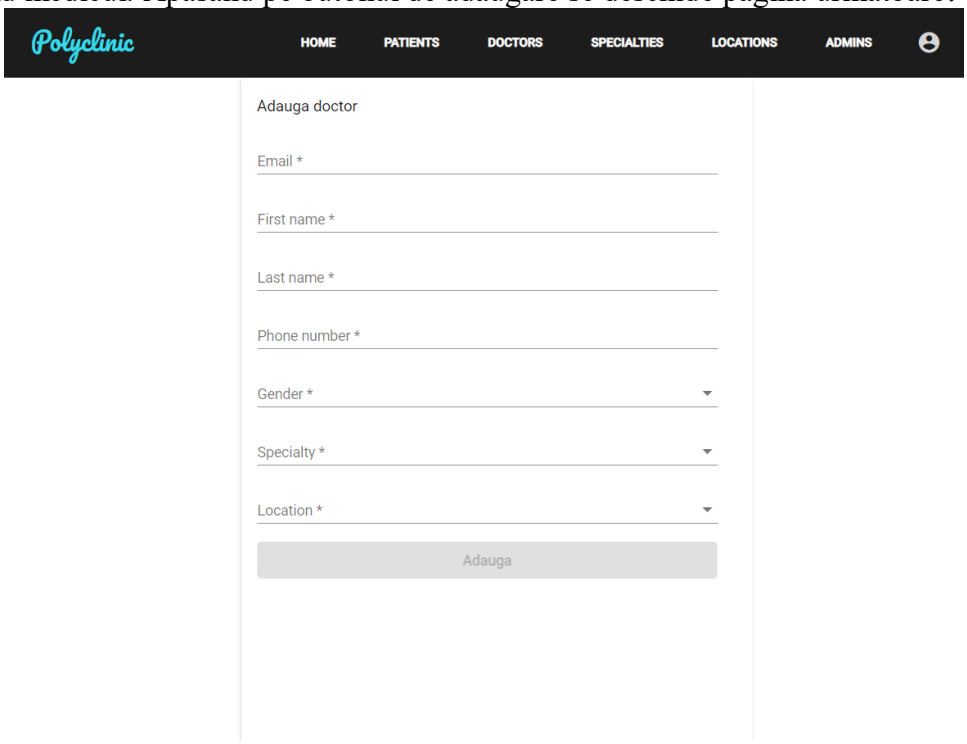
Figură 7.19 Antetul utilizatorului

Pentru toate acestea, administratorul are drept de vizualizare, editare, adăugare și ștergere. În continuare se va ilustra exemplu paginilor de administrare accesând paginile de doctors apăsând pe butonul cu același nume din partea de header.



Figură 7.20 Pagina cu lista doctorilor

În această pagină, se poate apăsa pe butonul „+” pentru a adăuga un nou medic și se poat apăsa pe butoanele din dreptul fiecărui medic, primul este folosit pentru a fi redirecționat către pagina cu serviciile medicale, următoarele butoane editând sau ștergând medicul. Apăsand pe butonul de adăugare se deschide pagina următoare:



Figură 7.21 Pagina pentru adăugarea unui medic

Pentru a adăuga un medic trebuie completate toate câmpurile corect și apăsat pe butonul „Save”. Pagina de editare este la fel ca și aceasta, dar fiecare câmp are deja valorile unui medic completate, aceastea putând fi doar schimbate prin editare.

Orice altă pagină a unui administrator are aceleași funcționalități, mai exact vizualizare, adăugare, editare și ștergere a datelor din sistem.

Capitolul 8. Concluzii

În acest capitol se vor prezenta obiectivele, rezultatele atinse și se vor descrie dezvoltările ulterioare ale aplicației.

8.1. Contribuții personale și rezultate obținute

Proiectul are ca și obiectiv dezvoltarea unei platforme medicale destinate unei policlinici care să fie utilizată de trei tipuri de actori, mai exact adminitrator, pacient și doctor. Aplicația dorește îmbunătățirea modului de comunicare dintre pacient și medic, îmbunătățirea modului de găsire a serviciilor medicale sau a medicilor, procesul prin care se desfășoară sistemul de programări și notificarea pentru acestea și de asemenea modul în care se păstrează evidența istoricului unui pacient. Nu în ultimul rând, aplicația ajută la gestionarea datelor într-un mod facil de către administrator. Toate aceste obiective au fost îndeplinite prin următoarele rezultate:

- S-a implementat o aplicație ușor de folosit prin care se găsesc foarte simplu medici și servicii medicale în funcție de mai multe criterii, precum specializare sau locație
- S-a îmbunătățit procesul de realizare a programărilor medicale, orice utilizator poate să vizualizeze programul unui medic și să aleagă o oră disponibilă. Totodată, poate vizualiza și în propriul program toate rezervările realizate
- S-a dezvoltat o mesagerie în cadrul aplicației, orice pacient având posibilitatea să comunice direct cu orice medic
- În aplicație s-a implementat un sistem de notificări prin email prin care va scădea numărul pacienților care nu se prezintă la programări
- S-a creat un proces de a vizualizare și adăugare a recenziilor pentru orice doctor, astfel orice utilizator va căpăta încredere în momentul în care va alege un doctor, acesta fiind mult mai bine informat
- În aplicație s-a dezvoltat istoricul medical pentru orice pacient, în urma consultațiilor medicii au posibilitatea să adauge rezultatele medicale obținute
- S-a creat o interfață prin care se pot gestiona ușor datele sistemului de către administratori, date precum medicii, specializările, serviciile medicale, locația și pacienții

8.2. Dezvoltări ulterioare

Aplicația implementată reprezintă o platformă medicală pentru gestionarea unei policlinici, acest sistem având multe alte funcționalități noi care pot fi dezvoltate, medicina reprezentând o arie în continuă dezvoltare. Printre funcționalitățile care pot fi dezolvate se prezintă următoarele:

- Înregistrarea în aplicație - se poate dezvolta înregistrarea în aplicație, pe lângă procesul normal de adăugare a unui cont, utilizatorii vor avea posibilitatea să se înregistreze prin intermediul serviciilor Google și

Facebook. Pe lângă butonul de "Sign up", vor exista alte două butoane pentru serviciile menționate

- Sistem de plată - în cadrul aplicației poate fi dezolvat un sistem de plată, orice pacient putând să selecteze mai multe servicii medicale, la final fiecare având posibilitatea să plătească direct în aplicație pentru serviciile selectate
- Generarea rezultatelor și documentelor medicale în aplicație – se poate dezvolta un proces prin care medicii vor putea folosi un sistem prin care să genereze documente medicale direct în această aplicație, nefiind nevoie de un alt sistem pentru acest lucru
- Aplicație mobilă – aplicația propusă momentan este disponibilă doar pe un browser, aceasta putând fi ulterior dezvoltată și pentru dispozitivele mobile
- Sistem de notificări în aplicație – momentan notificările pentru programări se primesc pe email, în viitor se va putea implementa primirea de notificări direct în aplicație. Pe lângă notificările legate de programări, se vor putea primi notificări pentru orice mesaje noi primite de la medici sau pacienți

Bibliografie

[1] G Eysenbach, *What is E-Health?*, J Med Internet Res., 2001, de pe site-ul NCBI – National Center for Biotechnology Information, disponibil online: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1761894/

[2] J. Moreira, *Manual Vs. Automated Appoinment Scheduling – The Time and Monetary Saving of Switching to Online Scheduling Software* – articol despre comparația dintre sistemele de programări manuale și cele automate, disponibil online: https://www.appointmentplus.com/wp-content/themes/appointmentplus-theme/dist/img/pdf/manual_vs_automated_appointment_scheduling.pdf

[3] C. Rudnic, *Online Medical Appointment Scheduling System*, Lucrare de diploma, Escola Tècnica Superior d'Enginyeria Informàtica, Universitat Politècnica de València, disponibil online: https://www.coursehero.com/file/44482624/CARA-Sistema-de-cita-online-para-una-consulta-m%C3%A9dicapdf/

[4] I. Sommerville, *Software engineering*, Pearson Education Limited, 2016, disponibil online: https://dinus.ac.id/repository/docs/ajar/Sommerville-Software-Engineering-10ed.pdf

[5] Craig Walls, *Spring boot in action book*, Manning Publications Co., 2016, disponibil online: https://doc.lagout.org/programmation/Spring%20Boot%20in%20Action.pdf

[6] R. Shaik, *Spring Boot Security + JSON Web Token Tutorial*, disponibil online: https://dzone.com/articles/spring-boot-security-json-web-tokenjwt-hello-world

[7] M. Dînșoreanu – Proiectare software – note de curs UTCN 2019

[8] I. Salomie, T. Cioara, I. Anghel, T.Salomie, *Distributed Computing and Systems: A practical approach*, Albastra, Publish House, 2008, ISBN 978-973-650-234-7

[9] M. Antal, C. Pop, D. Moldovan, T. Petrican, C. Stan, I. Salomie, T. Cioara, I. Anghel, *Distributed Systems – Laboratory Guide*, Editura UTPRESS ClujNapoca, ISBN 978-606-737-329-5, 2018, https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/329-5.pdf

[10] I. Costin – Sisteme de calitate în tehnologia informației – note de curs UTCN 2020

Anexa 1 – Lista figurilor

Figură 3.1 Screenshot pagină detalii doctor site Docbook....10
Figură 3.2 Screenshot detalii programare consultație oftalmologică 11
Figură 3.3 Exemplu de realizare a unei programări pentru Simple Practice 12
Figură 3.4 Exemplu de realizare programare în aplicația MyHealth1st 13
Figură 4.1 – Cazuri de utilizare pentru administrator....19
Figură 4.2 – Cazuri de utilizare pentru pacient 19
Figură 4.3 – Cazuri de utilizare pentru doctor....20
Figură 4.4 – Diagrama flowchart ștergere doctor al utilizatorului admin....22
Figură 4.5 Diagrama flowchart realizare programare al utilizatorului pacient 24
Figură 4.6 Diagrama flowchart adaugare serviciul medical propriu 26
Figură 5.1 Arhitectura Client-Server 32
Figură 5.2 Arhitectura conceptuală a sistemului 33
Figură 5.3 Arhitectura Layered 34
Figură 5.4 Arhitectura Layered a sistemului propus 35
Figură 5.5 Diagrama de clase a pachetului controller 36
Figură 5.6 Diagrama de clase a pachetului service 37
Figură 5.7 Diagrama de clase a pachetului repository....37
Figură 5.8 Diagrama de clase a pachetului model....38
Figură 5.9 Diagrama de clase a pachetului converter....39
Figură 5.10 Diagrama de clase a pachetului model....39
Figură 5.11 Diagrama de clase a pachetului security 40
Figură 5.12 Diagrama de clase a pachetului configuration....41
Figură 5.13 Structura logică a componentei frontend....41
Figură 5.14 Diagrama de pachete a componentei frontend....42
Figură 5.15 Diagrama de pachete a paginilor *account-pages* 43
Figură 5.16 Diagrama de pachete a paginilor *admin-pages*....44
Figură 5.17 Diagrama de pachete a paginilor *doctor-pages* 45
Figură 5.18 Diagrama de pachete a paginilor patient-pages 46
Figură 5.19 Diagrama de pachete a componentelor model....47
Figură 5.20 Diagrama de pachete a serviciilor....47
Figură 5.21 Diagrama de deployment a sistemului 48
Figură 5.22 Diagrama bazei de date 51
Figură 6.1 Exemplu de testare în Postman pentru înregistrarea cu succes 55
Figură 6.2 Exemplu caz de testare pentru exemplificarea mesajului de eroare 55
Figură 6.3 Exemplu de afișare a erorilor în aplicație....56
Figură 7.1 Pagina de login a aplicației....59
Figură 7.2 Pagina de înregistrare....59
Figură 7.3 Pagina *Home* a aplicației....60
Figură 7.4 Pagina de My profile a unui pacient 60
Figură 7.5 Pagina de My profile a unui doctor....61
Figură 7.6 Pagina cu programul unui pacient sau al unui medic....61
Figură 7.7 Pagina cu mesageria unui pacient....62
Figură 7.8 Pagina cu istoricul medical al unui pacient....62
Figură 7.9 Pagina cu lista serviciilor medicale ale unui doctor....63

Figură 7.10 Editarea unui serviciu medical....63
Figură 7.11 Adăugarea unui serviciu medical....63
Figură 7.12 Mesajul din mail-ul unui medic în momentul în care a fost adăugat.64
Figură 7.13 Pagina cu specializari....64
Figură 7.14 Lista serviciilor medicale ale specializarii oftalmologie....65
Figură 7.15 Lista medicilor....65
Figură 7.16 Pagina de profil a unui medic....66
Figură 7.17 Realizarea unei programari....66
Figură 7.18 Notificare din email pentru o programare a unui pacient....67
Figură 7.19 Antetul utilizatorului....67
Figură 7.20 Pagina cu lista doctorilor....68
Figură 7.21 Pagina pentru adăugarea unui medic....68

Anexa 2 – Lista tabelelor

Tabel 3.1 Comparație între sistemul propus și sistemele similare 15
Tabel 4.1 Cerintele functionale ale sistemului....16
Tabel 6.1 Tabel rezultate teste de performanță folosind JMeter 57

Anexa 3 – Glosar de termeni

Termen	Definiție
AWS	Amazon Web Services
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DTO	Data Transfer Object
HTML	HyperText Markup Language
JSON	Javascript Object Notation
JWT	JSON Web Token
REST	Representational State Transfer
S3	Simple Storage Service
SES	Simple Emailing Service
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
SQL	Structured Query Language
XML	EXtensible Markup Language