MHealth

Modulul web și Android - componenta personală

LUCRARE DE LICENŢĂ

Absolvent: **Andreea IFRIM**

Coordonator ştiinţific: **Senior Lector Ing. Cosmina IVAN**

2020

Cuprins

Capitolul 1. Introducere

În acest prim capitol se face o conturare a domeniului în care se încadrează proiectul și o prezentare generală a metodei prin care se pot gestiona cazurile de urgență înainte de sosirea unui echipaj medical, dar și rezolvarea problemelor de caracter personal, chiar dacă nu este o urgență, comunicând cu un medic de specialitate de la distanță.

1.1. Motivație

Pe zi ce trece, domeniul medical se dezvoltă din ce în ce mai mult, mai ales pe partea de digitalizare a acestuia pentru a ușura comunicarea și a micșora timpii perduți, mai ales că în acest domeniu timpul reprezintă unul dintre cei mai importanți factori.

Timpul în contextul urgențelor medicale este esențial și orice îmbunătățire a lui poate face diferența. Acest serviciu este unul foarte solicitant și se încearcă ca timpul de răspuns, adică momentul de când a fost anunțată urgența prin apel la 112 până la sosirea ambulanței la locul unde a fost solicitat ajutorul, să fie cât mai scurt. În funcție de caz, acest timp poate varia din mai multe motive (locație mai greu de accesat, condții meteo dificile, etc) și timpul de răspuns să fie mai lung decât cel așteptat. De aceea, intervenția unei persoane cu pregătire medicală, care se află aproape de zona în care este urgența, pentru a acorda primul ajutor până la sosirea echipajului solicitat ar putea aduce numai beneficii.

În contextul actual și perioada de pandemie din cauza noului virus COVID-19 prin care s-a trecut, s-a observat cât de importantă este cominicarea la distanță și rezolvarea problemelor de interes personal în mediul digital pe cât este posibil în funcție de necesități. Unele cazuri care nu reprezintă o urgența majoră pentru a suna la 112, dar ar fi de ajutor niște sfaturi din partea unui specialist s-ar putea rezolva prin convorbirea cu un medic, prin intermediul unui chat.

În zilele nostre, termenul de *mhealth* (mobil health) nu este unul necunoscut pentru domeniul medical. Acest termen se referă la folosirea dispozitivelor mobile, cum sunt smart pohone-urile, tabletele, ceasurile inteligente pentru comunicare, servicii de sănătate, schimb de informații și colectarea datelor. Telefonul mobil a ajuns să fie un obiect indispensabil pentru majoritatea persoanelor, de aceea ar putea sa fie de mare folos și în domeniul medical, atât în contextul raportării urgențelor, cât și în rezolvarea problemelor cu caracter personal: comunicarea unor rezultate ale analizelor, adăugarea unei programări, cererea unor sfaturi de la medici specialiști în anumite situații și multe altele.

Fiind prezentate aspectele de mai sus, s-a dorit implementarea unui sistem care să le includă pe toate acestea pentru a le putea accesa ușor, avându-le în același loc. Așadar, o aplicație mobilă și una web sunt disponibile pentru a acoperi atât gestionarea urgențelor și notificarea persoanelor cu pregătire medicală din apropierea locului de unde a fost raportată urgența, cât și aspectele personale alea utilizatorilor aplicațiilor, cum este comunicarea cu un cadru medical pentru sfaturi sau transmitere de fișiere.

1.2. Conținutul lucrării

În acest subcapitol se prezintă o scurtă descriere a conținutului care se regăsește în fiecare capitol din cele 8 ale acestei lucrări.

- *Capitolul 1* - **Introducere** - în acest capitol se face o încadrare a aplicației în domeniul medical și se realizează o prezentare generală a aspectelor atinse prin aceast proiect

- *Capitolul 2* - **Obiectivele proiectului** - în acest capitol se face o prezentare generală a întregului sistem și obiectivele atinse de acesta, iar mai apoi o detaliere a componentei *personael* ce s-a implementat

- *Capitolul 3* - **Studiu bibliografic** - prezintă ce influență a adus tehnologia asupra domeniului medical, statistici legate de evoluția chatului și detalii despre comunicarea digitală, cât și prezentarea unor aplicații deja existente similare cu sistemul MHealth dezvoltat și comparații între ele

- *Capitolul 4* - **Analiză și fundamentare teoretică** - acest capitol ilustrează arhitectura sistemului, cerințele sistemului, atât funcționale, cât și cele nonfuncționale, cazurile de utilizare pentru toate tipurile de utilizatori și tehnologiile care au fost folosite pentru realizarea sistemului

- *Capitolul 5* - **Proiectare de detaliu și implementare** - acest capitol conține descrierea implementării componentelor și prezentarea unor diagrame relevante sistemului, cum ar fi: diagrama bazei de date, flowchart-uri, diagrama de secvență etc.

- *Capitolul 6* - **Testare și validare** - în acest capitol de regăsesc diferite cazuri de testare și exemplificările lor cu secvențe concrete din lucrare

- *Capitolul 7* - **Manual de Instalare și Utilizare** - în acesta se prezintă resursele și aplicațiile necesare pentru instalare și un manual de utilizare specific pentru fiecare utilizator

- *Capitolul 8* - **Concluzii** - sunt rezumate obiectivele propuse și câteva dezvoltări ulterioare ce se pot realiza asupra sistemului

Capitolul 2. Obiectivele Proiectului

În acest capitol se specifică obiectivul general al sistemului printr-o scurtă descriere și obiectivele atinse în dezvoltarea componentei *personael* implementate.

2.1. Obiectivul general

Sistemul MHealth are ca obiectiv general să facă posibilă comunicarea a celor două module: aplicație web și aplicație mobilă Android cu scopul de a realiza gestionarea urgențelor și rezolvarea problemelor cu caracter personal.

Un utilizator care are instalată aplicația MHealth pe telefon poate să raporteze o urgență trmițând informații de la locul incidentului sub formă de poză, video sau descriere și datele vor fi procesate de aplicația web și trimise către un alt utilizator Android care are pregătire medicală, se află în apropiera urgenței și dorește să intervină până la sosirea ambulanței. Partea de personal este implementată atât în aplicația mobilă, cât și în cea web și va fi detaliată în subcapitolul ce urmează.

2.2. Obiective specifice

Sistemul poate fi delimitat în 3 părți:

a) partea web pentru gestionarea urgențelor
b) partea Android pentru gestionarea urgențelor
c) partea personală

Obiectivele propuse inițial au fost atinse și duse la bun sfârșit pentru fiecare parte. Sitemul MHealth a fost dezvoltat în colaborare cu doi colegi, Cornea Georgiana-Bianca și Dascălu Cosmin-Ștefan, fiecare implementând câte o parte din proiect.

	Modulul Android	**Modulul Web**
Utilizatori	Obișnuiți și cu pregătire medicală	Doctori
Componenta gestionării urgențelor	Utilizatorul obișnuit poate raporta o urgență adăugând dintr-o interfață prietenoasă o descriere text, o poză, un video, o înregistrare audio și un nivel de severitate al situației estimat de către el. Locația este transmisă în mod automat. Se trimite o notificare cu adresa urgenței din apropiere și un email cu informațiile necesare.	Datele sunt preluate de la aplicația mobilă și prelucrate în mod automat. Se folosesc servicii Cloud, împreună cu Machine Learning pentru prelucrarea imaginilor, textului, înregistrărilor video și a celor audio, iar rezultatele sunt afișate în interfață. Doctorul logat poate valida informațiile și poate să adauge detalii ce vor ajuta în administrarea cazului. În plus, are la dispoziție date despre toate cazurile active, o hartă interactivă și un istoric al cazurilor.
Componenta personală	Detalii despre acest modul sunt prezentate mai jos	

Tabel 2.1 Componentele și modulele sistemului MHealth

Modulul personal, prezentat în această lucrare, este dezvoltat și în aplicația web și în cea mobilă și se regăsesc 2 tipuri de utilizatori:

- doctor - are acces la aplicația web
- utilizator obișnuit/pacient - are acces la aplicația mobilă

S-au propus următoarele obiective pentru **doctor**, utilizatorul aplicației web:

- ✓ **Programări** - poate vizualiza, adăuga, șterge și modifica programări în funcție de dată, oră și numele pacientul
- ✓ **Adăugare perioadă de vacanță** - selectarea unei perioade de indisponibilitate și alegerea unui doctor înlocuitor
- ✓ **Distribuire fișiere** - vizualizare/trimitere de fișiere de diferite formate către pacienți

Pentru utilizatorul obișnuit carea are acces pe aplicația Android s-au indeplinit următoarele obiective:

- ✓ **Programări** - vizualizare, adăugare și ștergere unei programări în funcție de dată și specializarea doctorului
- ✓ **Distribuire de fișiere** - după selectarea specializării și a doctorului se pot distribut fișiere existente deja în telefon

Cele două module, cel al aplicației Android și cel al aplicației web au baza de date comună stocată în Firebase și orice modificare adusă ei se face instant, așa datele vor fi mereu actualizate și se vor putea vizualiza într-un mod corect.

Chat-ul stă la baza comunicării dintre module, fiind implementat atât pe web, cât și pe Android. Platforma PubNub asigură persistența mesajelor și trimiterea asincronă a lor. Așadar, doctorul și pacientul vor comunica în timp real și vor vizualiza mesajele imediat cum au fost trimise.

Securitatea sistemului a fost un alt obiectiv propus prin dechiderea unei sesiuni cu token după autentificarea utilizatorului pe aplicația web și resetarea parolei realizându-se cu un cod de securitate trimis pe email.

Capitolul 3. Studiu Bibliografic

În acest capitol se descrie cum a influențat tehnologia domeniul medical, statistici despre comunicarea digitală și evoluția chat-ului, cât și câteva aplicații similare cu sistemul MHealth și comparații între ele.

3.1. Influența tehnologiei în domeniul medical

Inovațiile tehnologice continuă să crească, schimbând majoritatea industriilor și ajutându-le să evolueze. În medicină, tehnologia joacă un rol important aproape în toate procesele ei, de la înregistrarea pacienților până la monitorizarea datelor și de la rezultatele testelor de laborator până la aplicații pentru îngrijire personală.

Scopul tehnologiei este de a ușura interacțiunea între pacienți și doctori și de a ajuta victemele în cazul unui incident să primească primul ajutor într-un timp cât mai scurt prin anunțarea cadrelor medicale prin intermediul aplicațiilor, telefoanelor și a internetului.[1]

Digitalizarea datelor medicale realizată prin aplicarea standardelor EHR (Electronic Health Records) a fost o adevarată provocare în domeniul medical pentru că a fost necesară o înlocuirea a datelor învechite de pe hârtii cu înregistrarea lor într-un sistem centralizat. Conform unui studiu realizat de Universitatea din Michigan transferul de la hârtie la date medicale înregistrate electronic a redus costul îngrijirii unui pacient cu 3%. [3]

În domeniul medical o cantitate mare de date trebuie stocată și accesată de mai multe persoane și acest fapt ar putea aduce mai multe probleme, dar *Cloud*-ul vine cu o soluție pentru că folosește componente hardware și software pentru a transmite servicii prin internet și datele pot fi accesate de oriunde, folosind aplicații și dispozitive conectate la internet.

Telemedicina[2] oferă servicii de sănătate folosind dispozitive digitale cum sunt calculatoarele și smartphone-urile. Se poate accesa serviciul de îngrijire virtuală prin intermediul aplicațiilor dedicate. În contextul actual, al pandemiei provocate de virusul COVID-19 aceasta ramură a fost de mare ajutor pentru că s-au putut susține programări online, s-au putut prescrie sau reînoi rețete și s-au putut oferi sfaturi de la medici pentru anumite probleme întâmpinate în această perioadă. Telemedicina aduce și alte beneficii precum: scurtarea timpului de așteptare al pacienților, îmbunătățește accesul la zonele rurale prin comunicarea mai ușoară cu persoanele din acele zone.

Un procent de 45.4 % din populația lumii[3] deține un smartphone și acesta tinde să crească, de accea termenul de *Mobile Health* poate fi folosit în contextul influenței tehnologiei asupra domeniului medical. Aplicațiile medicale create pentru telefonul mobil oferă o flexibilitate mare pentru toți pacienții, o comunicare mai ușoră și mai rapidă cu doctori, gestionarea programărilor, a datelor medicale personale, diagnosticuri, fitness etc.

[1] https://www.aimseducation.edu/blog/the-impact-of-technology-on-healthcare/
[2] https://www.medicalnewstoday.com/articles/telemedicine-enefits#disadvantages
[3] https://www.oberlo.com/statistics/how-many-people-have-smartphones

Așadar, tehnologia fiind schimbătoare și mereu îmbunătățită, se pot găsi soluții pentru reducerea timpilor pierduți și ușurarea comunicării în domeniul medical prin digitalizare, telemedicină, mobile health etc.

3.2. Chat - comunicarea digitală

Chat-ul online poate fi numit și mesagerie instantanee pentru că oferă transmisie în timp real al textelor prin Internet. Mesajele sunt transmise între două părți sau mai multe dacă se face transmisie multicast, denumită "chat room" sau grup. Aplicațiile dedicate mesageriei instantanee pot folosi tehnologie de tip push pentru a genera texte și mesaje în timp real, conferințe video, transmitere de fișiere și secvențe audio (VoIP - voice over Internet Protocol).

3.2.1. Mecanismul publish-subscribe

Mecansimul **publish-subscribe**[4] este o metodă de comunicare asincronă în care mesajele sunt transmise între aplicații fără să se știe identitatea expeditorului sau a destinatarului.

Modelul publish-subscribe este compus din 4 concepte de bază:

- Topic - este un canal intermediar care menține o listă de abonați (subscribers) pentru a trimite mesajele de la editori (publishers)
- Mesaj - mesajele serializate sunt trimise către un topic de către publisher care nu știe detalii despre subscribers
- Publisher - o aplicație care transferă un mesaj către topic
- Subscriber - o aplicație care se întregistrează la topicul dorit pentru a primi mesajele corespunzătoare

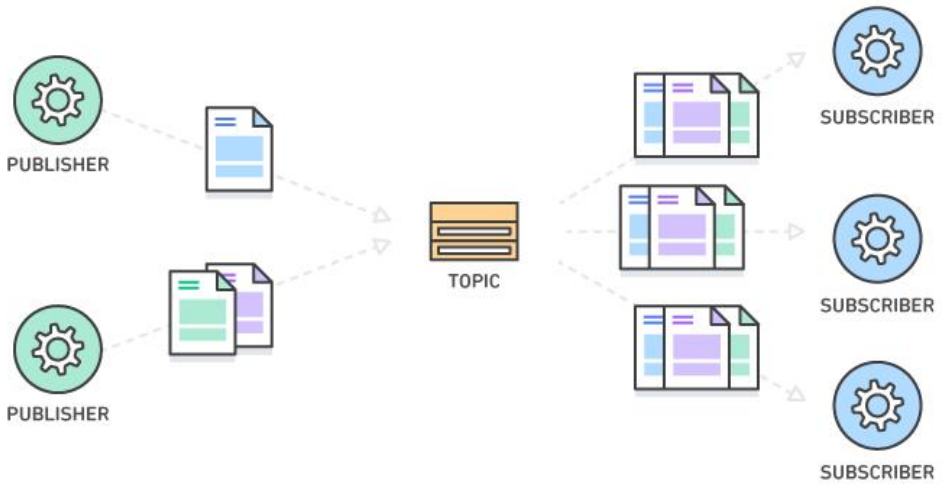

Figura 3.1 Mecansimul publish-subscribe[5]

[4] https://blog.stackpath.com/pub-sub/
[5] https://aws.amazon.com/pub-sub-messaging/

Printre avantajele acestui mecanism se regăsesc și **scalabilitatea** și asocierea redusă dintre componente (**loose coupling**).

Are o asociere scăzută între părți pentru că aplicațiile de tip publisher nu știu detalii despre aplicațiile subscriber, de aceea ele pot opera independent fără a ține cont una de cealaltă. În general un client nu poate trimite mesaje către un server dacă acesta nu rulează. Cu modelul publish-subscribe, nu se mai ține cont dacă procesele rulează sau nu pe un server.

Datorită funcționalităților din cadrul modelului publish-subscribe (operații executate în paralel, message caching, rutare de tip arbore etc.) se pote scala un volum mai mare de mesaje față de capacitatea unui singr data center tradițional. Cu toate aceste, cu cât crește numărul de noduri și mesaje, cu atât mai mult cresc șansele să încetinească procesul de transmitere.

Din punct de vedere medical, în articolul [1] este prezentat cum un model publish-subscribe bazat pe cloud poate îmbunătăți calitatea transmisiei, capabilitățile de stocare și interogare a imaginilor DICOM (standard ISO - Digital Imaging and Communications in Medicine). Datorită lui eficința de trimitere și primire a datelor a crescut cu aproximativ 35%. Rezultatele au sugerat că folosirea unei imagini medicale prin mecansimul de publish-subscribe poate îmbunătăți eficința transmisiei, permițând echipelor de specialiști să comunice în timp real chiar și atunci când sunt la distanță.

3.2.2. Evoluția platformelor de comunicare

Aplicațiile de comunicare, de tip chat, își fac apariția în fiecare an în lista celor mai descărcate aplicații. Din anul 2016 până în anul 2019, pe primul loc s-a aflat o platformă de comunicare, mai exact WhatsApp.[6]

Aplicațiile chat sunt integrate în activitățile noastre zilnice și ajută la o comunicare mai ușoară atât în viața personală, cât și în cea profesională, dar începuturile nu au arătat exact așa.

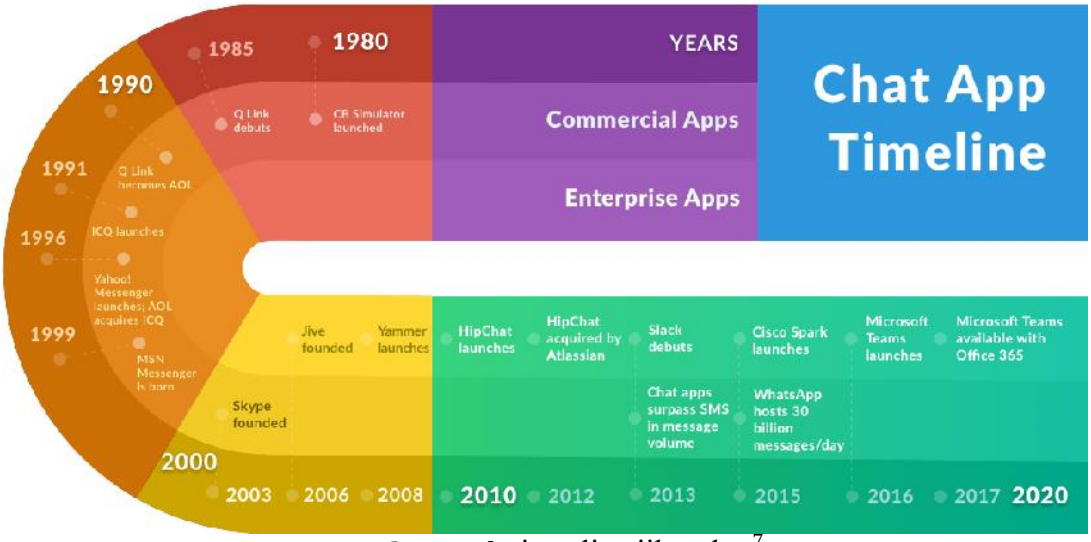

Figura 3.2 Evoluția aplicațiilor chat[7]

[6] https://www.visualcapitalist.com/ranked-most-downloaded-apps/
[7] https://blog.workato.com/2017/05/history-chat-apps/#.XvTqrigzbDf

În figura 3.2 se prezintă în funcție de ani aplicațiile care au contribuit la evoluția chat-ului până în prezent.

Platformele de mesagerie instantanee au fost asociate cu anii 90, dar aplicațiile comerciale de tip chat datează încă din 1980, când a fost lansat un serviciu online Q-Link (Commodore's Quantum Link) care permite utilizarea unui chat, email, distribuire de fișiere și jocuri.

De-a lungul timpului, aplicațiile s-au dezvoltat din ce în ce mai mult, față de camerele de chat din aplicațiile vechi care au fost dezvoltate doar pentru persoane care își petrec timpul în fața unui calculator sau laptop, cele actuale își pun bazele pe utilizarea telefonului sau a tabletei. Internetul și viteza transmiterii datelor s-au dezvoltat foarte mult pe parcursul timpului, așa și experiența de comunicare s-a îmbunătățit și ea prin adăugarea de funcționalități (schimb de fișiere, adăugare de imagini, conferințe audio/video etc.).

În viitor se preconizează că majoritatea aplicațillor deja existente vor incorpora o componentă de chat pentru o comunicare mai rapidă, fapt care deja se implementează sau atașarea unui chatbot care oferă informații și răspunsuri automate, în funcție de întrebările introduse.

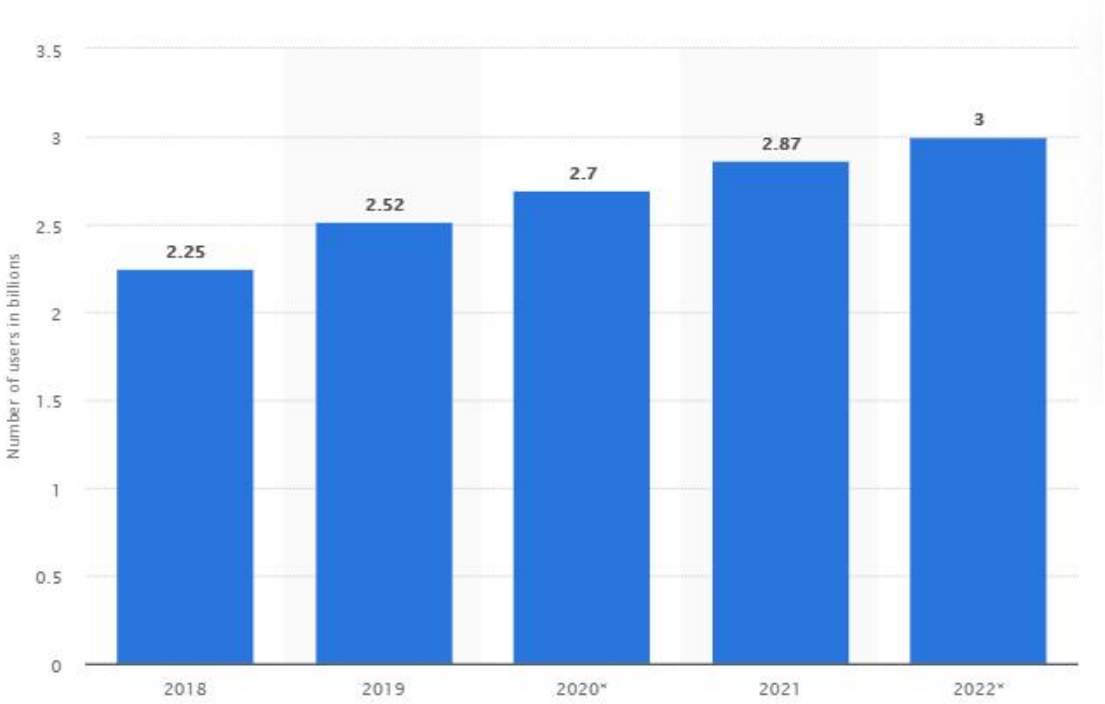

Figura 3.3 Numărul utilizatorilor de aplicații mobile de comunicare [8]

Se observă în figura 3.3 că numărul utilizatorilor de aplicații chat crește în fiecare an, în anul 2019 atingând un număr de 2.52 de miliarde de utilizatori în întreaga lume. Pentru anul acesta și următorii doi s-a făcut o estimare pe baza statisticilor din anii precedenți.

[8] https://www.statista.com/statistics/483255/number-of-mobile-messaging-users-worldwide/

3.2.3. PubNub

Platforma PubNub asigură o infrastructură "out-of-the-box", SDK-uri și unelte necesare pentru a construi aplicații scalabile în timp real. PubNub garantează 99.999% fiabilitate și oferă 100 ms latență oriunde în lume. [9]

Această platformă pune la dispoziție și un chat care poate fi customizat pentru aplicații mobile sau web. Se pot adăuga ușor noi funcționalități potrivite pentru soluțiile care se doresc implementate.

Sunt puse la dispoziție numeroase SDK-uri pentru tehnologii, printre care cele mai populare sunt: JavaScript, NodeJS, Android, Swift, C#, Python, Unity etc. În implementarea chat-ului disponibil în sistemul MHealth s-au folosit SDK-urile destinate tehnologiilor JavaScript și Android.

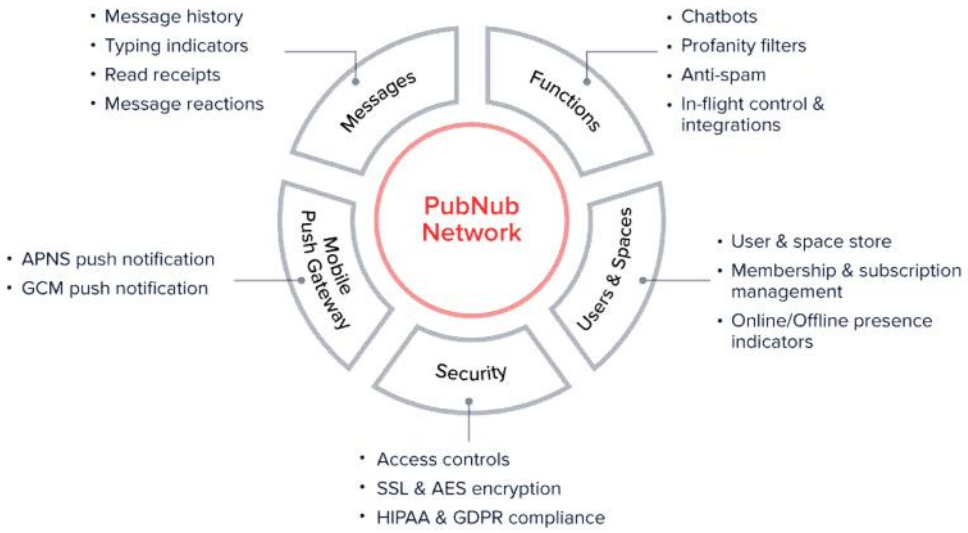

Figura 3.4 Conceptele principale ale platformei PubNub[10]

În figura 3.4 sunt cuprinse conceptele de bază din tehnologia PubNub în care SDK-urile specifice chat-ul sunt construite peste rețeaua PubNub, care este un Data Stream Network cu peste 15 puncte de interes în întreaga lume.

Utilizatorii se conectează la platformă prin intermediul aplicațiilor. Aceștia sunt identificați unic cu *uuid* (universally unique identifier) setat de către aplicație cu scopul de a asigura corectitudinea facturilor și stocarea datelor. În cazul aplicațiilor de chat, utilizatorii sunt chiar persoane care pot să adauge și chatbots.

Crearea de cont este gratuită, la fel și utilizarea platformei până la 1 milion de tranzacții și capacitate de stocare până la 1 GB de date. După ce se depășește acest prag este posibilitatea de a actualiza contul și se va plăti în funcție de utilizare.

[9] https://www.pubnub.com/docs
[10] https://www.pubnub.com/docs/chat/concepts

Utilizatorii pot stoca detalii cum sunt numele de utilizator, URL-ul profilului și adresa de email. Aceste se pot gestiona prin intermediul unui dashbord pus de dispoziție de PubNub în care se pot adăuga și proprietăți adiționale legate de cont.

Când se crează o aplicație nouă se generează un set de chei cu care te poți conecta de la dispozitive la ea. Fiecare operație PubNub necesită o inițializare cu aceste chei. Setul de chei este compus din[11]:

- **Publish key** - necesară pentru inițializarea PubNub la aplicația proprie și pentru a putea publica mesaje
- **Subscribe key** - cheie pentru a te putea abona la aplicația PubNub și a trimite mesaje. Atât publish key, cât și subscribe key sunt necesare pentru conectare și publicare de mesaje
- **Secret key** - cheie folosită pentru a efectua garanturi atunci când se utilizează funcționalitățile Access Manager și ar trebui folosită doar într-un mediu securizat de server, nu de pe aplicația de tip client

Canalele sunt camere de chat virtuale în care utilizatorii se alătură pentru a comunica unii cu alții. Mesajele trimise prin intermediul PubNub sunt mereu transmise printr-un canal. Aceste canale nu trebuie definite în avans, ele sunt create automat prin publicarea mesajelor în ele. Opțional pot fi stocate cu diferite caracteristici: nume, descriere, imagine etc.

Cnalele sunt unic indentificate printr-un ID setat de către aplicație. Acest ID este folosit pentru diferite operații puse la dipoziție de PubNub: publicare, abonare, accesare istoric al canalului și pentru a trimite și primi mesaje.

Prin abonarea la un canal se inițiază o conexiune în timp real cu PubNub. Această conexiune rămâne deschisă atât timp cât utilizatorul rămâne abonat de pe o aplicație client. De aceea, prin intermediul PubNub nu conează de pe ce dipozitiv se funizează mesaje, atăt timp cât aplicația este abonată la canalul corespunzător.

Mesajele sunt pachete de date care sunt publicate pe un canal. Ele pot conține date serializate, incluzând obiecte, numere, string-uri și vectori. Dimensiunea maximă a unui singur mesaj permisă de PubNub este de 32 kib.

Chiar dacă mesajele pot fi transmise în orice format, serializarea lor în format JSON rămâne cea mai bună variantă pentru că SDK-ul PubNub automat le transformă în string înainte de a le publica.

Sercuritatea și controlul accesului sunt aspecte importante și critice pentru orice aplicație chat, iar PubNub este construit ținând cont de aceste elemente fundamentale. Platforma are măsuri de securitate la nivelul rețelei, mesajelor, canalelor, utilizatorilor și cheilor care se pot casifica în:

- **Criptare** - PubNub suportă criptare de rețea TLS (Transport Layer Security) point-to-point și criptare de mesaje AES (Advanced Encryption Standard) end-to-end. În acest caz, PubNub nu va avea acces la datele transmise ceea ce este foarte important mai ales pentru o aplicație chat de tip medical unde se pot transmite informații cu caracter sensibil

[11] https://www.pubnub.com/developers/tech/admin-dashboard/keys-object/

- **Autorizare** - Access Manager asigură citire granulară și control al accesului la scriere cu abilitatea de gestiona permisiuni. Accesul poate fi controlat pentru utilizatori individuali și canale specifice
- **Validarea mesajelor** - funcțiile PubNub permite mesajelor trimise de către clienții neautorizați să fie validate înainte de a fi stocate în istoric și trimise altor utilizatori.
- **Prevenirea atacurilor** - PubNub deține un data center de rutare pentru a prevedea atacurile regionale

Funcțiile puse la dispoziție de PubNub din platforma Function-as-a-Service permite construirea propriilor microservicii și incorporarea logicii în timp real pentru procese de rutrare, transformare și agregare a mesajelor. Funcțiile PubNub se ocupă de scalarea, implementarea globală, redundanța și operațiile necesare pentru implementarea taskurilor care se doresc implementate.

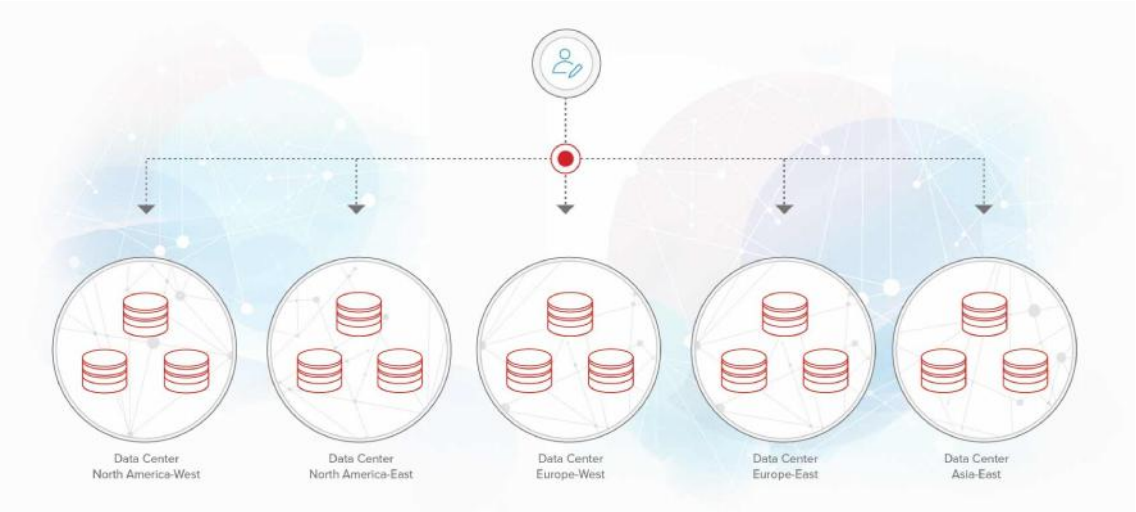

Figura 3.5 Infrastructura rețelei PubNub[12]

PubNub operează cu 15 PoP (points-of-presence) cu nivele de redundanță pe fiecare PoP. Serverul DNS are mai mulți furnizori de cloud din diferite regiuni.

Dacă un dispozitiv nu se poate conecta la cel mai apropiat data center, se poate conecta la următorul data center cel mai apropiat și automat se actualizează mesajele care nu au putut fi trimise, prin intermediul modelului "Store and Forward".

Rețeaua Data Stream este foarte rapidă și rutarea unui mesaj prin intermediul rețelei PubNub durează între 4 și 9 milisecunde.

Chiar dacă PubNub este o tehnologie complexă, cu mai multe funcționalități decât cele implementate în sistemul MHealth și este mai scalabil decăt el, s-a dorit integrarea unui chat cu ajutorul ei pentru a testa o tehnologie nouă și de ultimă generație în defavoarea unei tehnologii mai des implementate cum este RabbitMQ.

[12] https://www.pubnub.com/developers/tech/network-infrastructure/

3.3. Aplicații similare

În acest subcapitol se prezintă diferite aplicații atât mobile, cât și web care conțin funcționalități asemănătoare cu sistemul MHealth și se face o scurtă descriere a lor și o analiză comparativă între ele și aplicația implementată.

3.3.1. Descrierea aplicațiilor asemănatoare

Siren GPS

SirenGPS[13] este o aplicație mobilă disponibilă pe Google Play, cât și pe App Store care conține un buton de panică 911 cu posibilitatea de a selecta serviciul dorit (ambulanță, pompieri sau poliție) cu scopul de a îmbunătăți gestionarea urgețelor și timpul de răspuns fără a schimba serviciul de aplelare la 911.

Sunând prin intermediul aplicației se trimite sub formă de notificare și locația persoanei care a raportat urgența. Se formează câte un geofence pentru fiecare locație specificată pentru a trimite notificări peroanelor din zona urgenței, asemănător cu funcționalitatea implementată în sistemul MHealth.

Persoanele care vor sa intervină pot sa comunice cu victima prin intermediul unui chat pentru a cere informații în plus și aceasta poate să dețină și un istoric medical în aplicație dacă dorește să-l completeze.

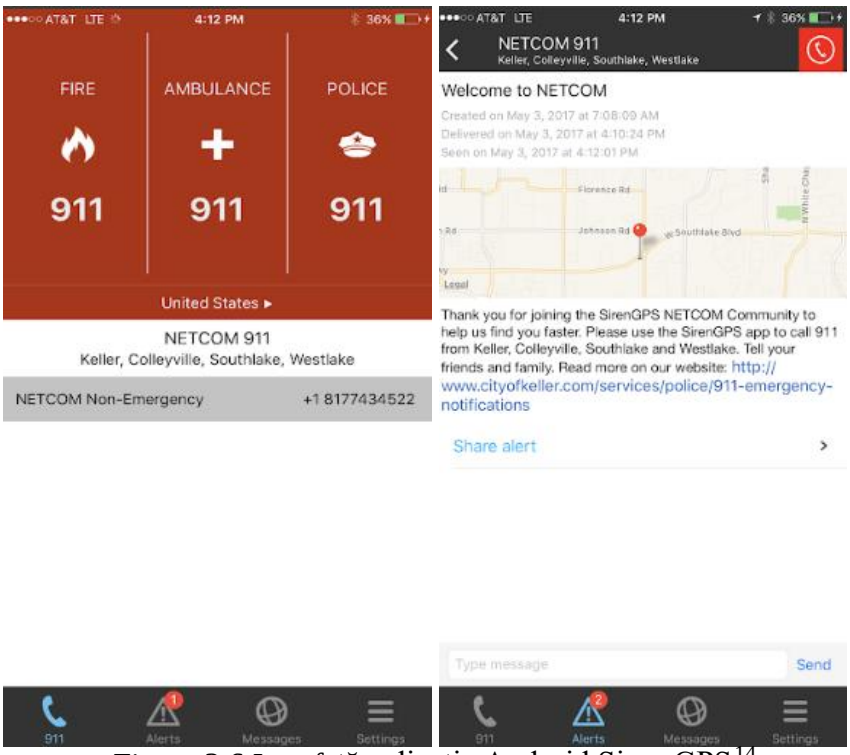

Figura 3.6 Interfață aplicație Android Siren GPS[14]

[13] https://www.sirengps.com/product/siren-911
[14] https://play.google.com/store/apps/details?id=com.sirengps.mobile&hl=ro

WebMD

WebMD: Check your symptoms[15] este o aplicație medicală disponibilă atât pe mobil,cât și pe web cu scopul de a verifica simptoamele pacienților, de a căuta informații depsre anumite medicamente, tratamente și diagnosticuri, de a căuta medici în funcție de locația pacienților și se pot seta remidere pentru medicamente.

Aspectul comun cu sistemul MHealth ar fi acela că aplicația WebMD folosește localizarea GPS pentru a găsi doctori de o anumită specialitatea în funcție de zona pacienților și pot adăuga o programare cu doctorul slectat, pot vizualiza o hartă interactivă cu locația doctorului, pot inița o convobire cu acesta prin intermediul unui chat și distribui fișiere în funcție de necesități.

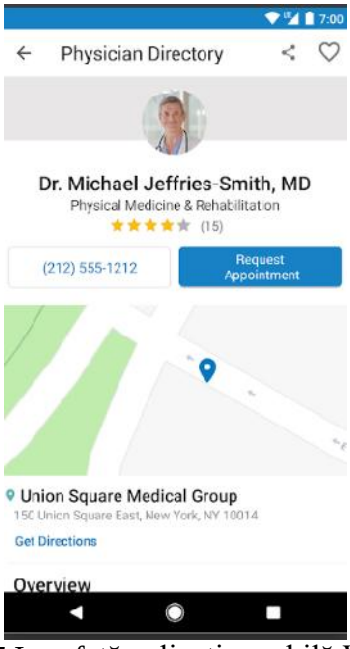

Figura 3.7 Interfață aplicație mobilă WebMD[16]

Doctor On Demand

Doctor On Demand[17] este o aplicație mobilă disponiblilă 24/7 care oferă visite online cu doctori de specialitate. Se pot adăuga programări și consultul va fi ca unul față în față cu examinarea istoricului medical, al simptoamelor și recomandarea unui plan de tratament, doar că în mediul virtual.

Aplicația dispune de selectarea doctorului în funcție de specializare, o hartă intercativă cu cabinetele doctorilor și laboratoarele, în cazul în care programarea virtuală nu este suficientă, un chat pentru comunicarea dintre pacient și doctor, istoric medical al pacientului și a programărilor trecute și posibilitatea distribuirii fișierelor între pacient și doctor în funcție de necesități.

[15] https://play.google.com/store/apps/details?id=com.webmd.android&hl=ro
[16] https://play.google.com/store/apps/details?id=com.webmd.android&hl=ro
[17] https://www.doctorondemand.com/

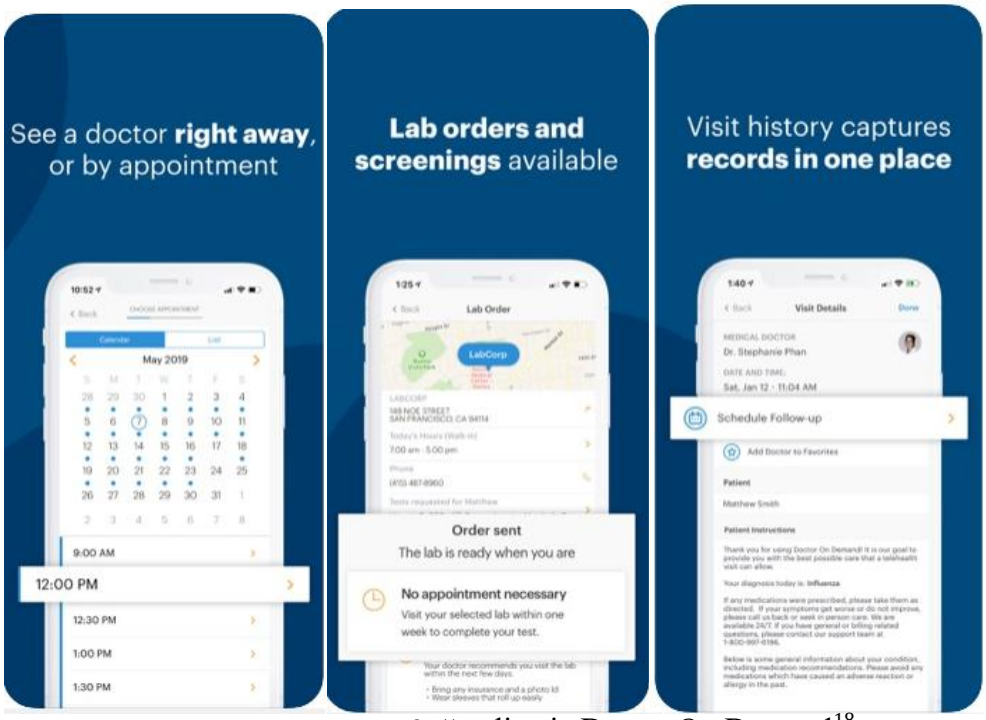

Figura 3.8 Interfață aplicație Doctor On Demand[18]

3.3.2. Analiza comparativă a sistemului

Tabelul de mai jos reprezintă o analiză comparativă între aplicațiile prezentate mai sus și sistemul MHealth. După cum putem observa, nu am găsit o aplicație care să conțină atât componenta de raportare a urgențelor cât și componenta cu caracter personal ce conține interacțiunea între doctor și pacient. Cu verde sunt marcate funcționalitățile care se regăsesc în aplicații și cu roșu cele care nu sunt prezente sau nu au fost menționate.

[18] https://apps.apple.com/us/app/doctor-on-demand/id591981144

Funcționalitate	Include	Siren GPS	WebMD	Doc. on demand	M-Health
Componenta mobila		✓	✓	✓	✓
Componenta web		×	✓	×	✓
Raportare urgență	Descriere text	✓	×	×	✓
	Imagine	×	×	×	✓
	Video	×	×	×	✓
	Audio	×	×	×	✓
	Nivel de severitate	×	×	×	✓
Localizare GPS		✓	✓	×	✓
Traducere în timp real		×	×	×	✓
Autodetecție limbă		×	×	×	✓
Autentificare biometrică		×	×	×	✓
Creare cont cu automatizare date buletin		×	×	×	✓
Push notifications folosind geofencing		✓	×	×	✓
Procesare urgențe	Cluster urgențe	×	×	×	✓
	Procesare imagini	×	×	×	✓
	Procesare text	×	×	✓	✓
	Procesare audio	×	×	×	✓
Vizualizare harta interactiva		×	✓	✓	✓
Stocare date in blockchain		×	×	×	✓
Vizualizare istoric		✓	×	✓	✓
Date statistice		×	✓	✓	✓
Chat		✓	✓	✓	✓
Programare consultație	Gestionare progamari pentru doctor	×	×	✓	✓
	Adaugare programari pentru pacient	×	✓	✓	✓
Distribuire fișiere intre doctor si pacient		×	✓	✓	✓
Gestionare perioada concediu	Programare concediu	×	×	×	✓
	Alegere înlocuitor	×	×	×	✓
	Disponibilitate in caz de urgență	×	×	×	✓

Tabel 3.1 Analiză comparativă a sistemului

Capitolul 4. Analiză şi Fundamentare Teoretică

4.1. Arhitectura conceptuală a sistemului

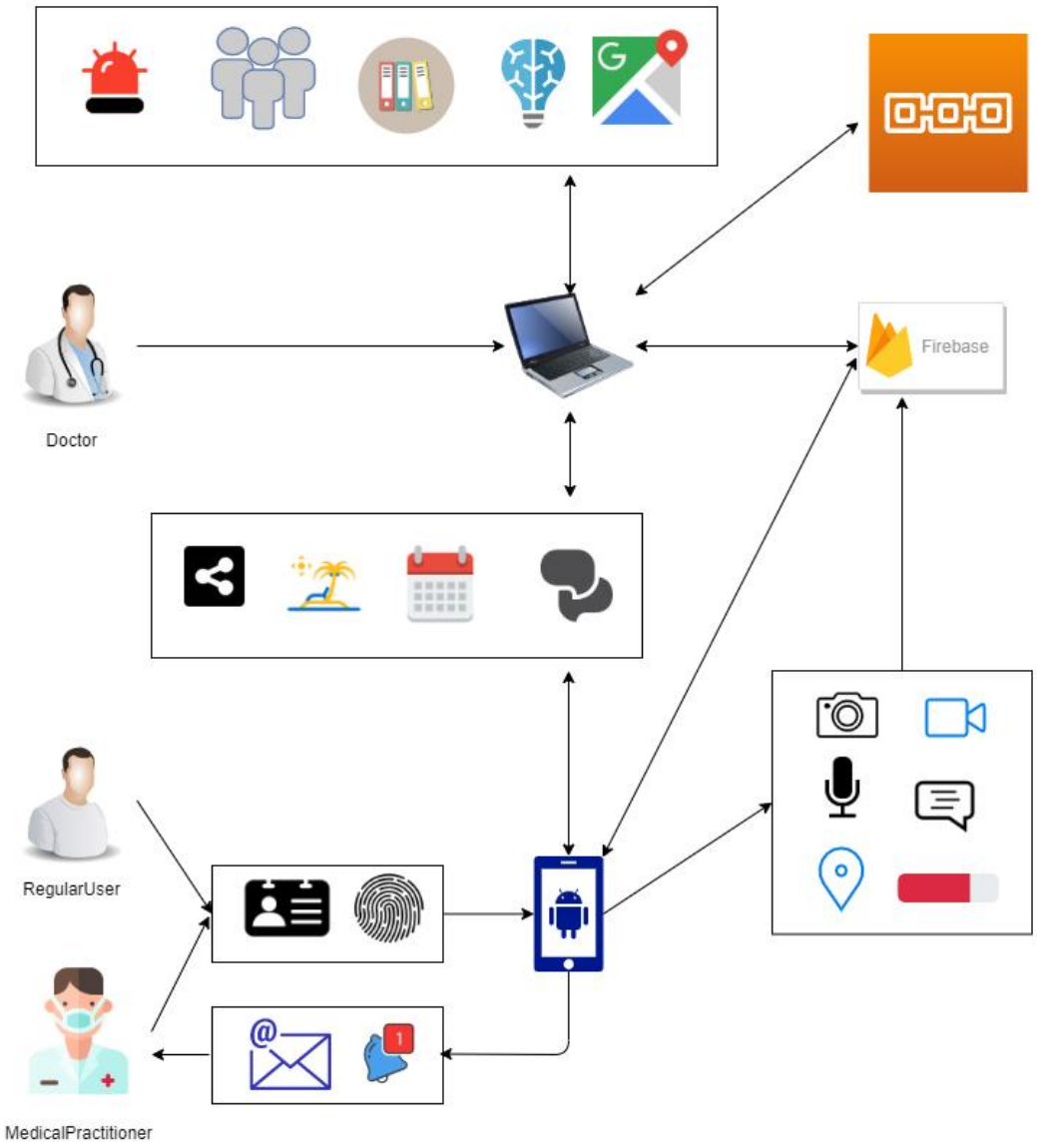

Figura 4.1 Arhitectura conceptuală a întregului sistem

Arhitectura conceptuală a întregului sistem este reprezentată in Figura 4.1. Aceasta se împarte în două componente mari: componenta aplicației web și componenta aplicației mobile Android și în 3 tipuri de utilizatori: doctor, utilizator obișnuit și utilizator cu pregătire medicală. La aplicația Android au acces utilizatorul obișnuit și cel cu pregătire medicală, iar la aplicația web are acces doar doctorul.

Din punctul de vedere al funcționalităților și acestea pot fi delimitate în două categorii: personală și raportare urgențe. Categoria personală va fi exemplificată în figura ce urmează.

Raportarea de urgențe se va face de pe aplicația Android de către utilizatorul obișnuit trimițând informații de la locul accidentului, cum ar fi: secvențe video, înregistrare audio, imagini, grad de severitate și locația va fi trimisă automat atunci când se va raporta urgența. Logarea în aplicația Android se face prin intermediul amprentei telefonului, iar înregistrarea unui nou cont se realizează pe baza buletinului, efectuând o poză acestuia și prin procesarea imaginii se extrag datele necesare. După raportarea urgenței, orice utilizator cu pregatire medicală care are aplicația instalată pe telefon și se află în apropierea urgenței primește o notificare cu locul accidentului și dacă dorește să intervină asupra cazului va primi mai multe detalii pe email. Această parte a fost implementată și documentată de Cosmin Dascălu.

Pe partea de web, toate informațiile primite de la aplicația Android vor fi procesate și rezultatele împreună cu specificațiile extra aduse de către doctorul logat vor fi trimise pe mail la utilizatorul cu pregătire medicală interesat de urgență. Toate aceste informații cu caracter sensibil vor fi pătrate într-un blockchain. Doctorul are posibilitatea să vizualizeze urgențele, istoricul lor, cât și cele actuale. Poate vizualiza o hartă cu locațiile urgențlor și poate gestiona persoanele care pot să ajungă la locul urgenței. De această parte s-a ocupat Georgiana Cornea.

Comunicarea dintre componente a fost posibilă prin intermediul Firebase, având baza de date comună Cloud Firestore. Aplicația web are deploymentul realizat pe Cloud, ceea ce aduce înregului sistem un plus pentru că este ușor de accesat și nu ține cont de performanțele dispozitivelor utilizatorilor.

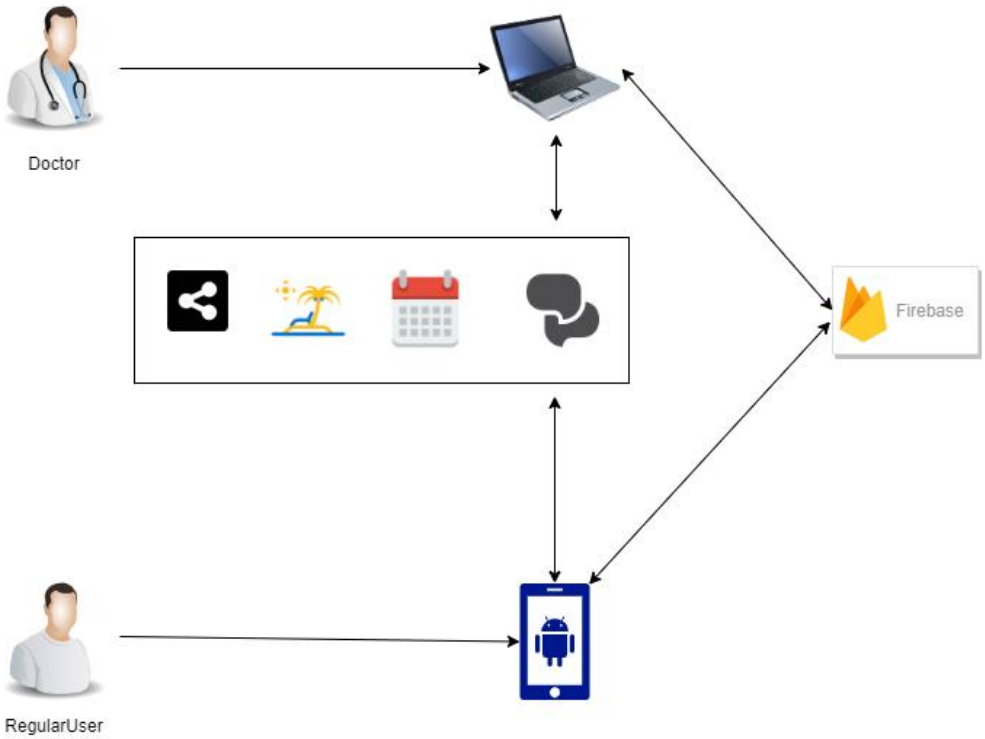

Figura 4.2 Arhitectura conceptuală a modulului implementat

Figura 4.2 reprezintă arhitectura modulul implementat, mai exact partea de personal. Acest modul poate fi folosit și într-un context obișnuit, nu numai în cadrul unei urgențe. Se împarte la rândul lui în partea de aplicație web, unde are acces doar doctorul si de aplicație mobilă care este controlată de un utilizator obișnuit. La fel ca în întreagul sistem, toate datele sunt stocate în Cloud Firestore.

Acțiunile specifice acestui modul sunt:

- distribuire de fișiere între pacient si doctor, atât pe parte de aplicație web, cât și mobilă
- posibilitatea doctorului de a marca perioada de concediu și asignarea unui alt doctor înlocuitor
- gestionarea programărilor din punctul de vedere al doctorului, cât și a pacientului
- chat între doctor și pacient care asigură și cominicarea permanentă dintre cele două componente: Android si web

4.2. Cerințe de sistem

În acest capitol se prezintă o descriere a cerințelor sistemului, mai exact cele funcționale, non-funcționale și cele tehnologice a modulului implementat.

4.2.1. Cerințe funcționale

Aceste cerințe se împart în doua categorii, cele specifice pentru partea personală de web si cea a aplcației mobile. Pe partea de web are acces doctorul, iar pe mobil un utilizator obisnuit. CF reperezintă prescurtarea pentru cerinței funcționale.

Nr. Crt	User	Cerință
CF1	doctor	**Logare** - utilizatorul se poate loga pe platforma web prin intermediul unui username si a unei parole deja existente
CF2	doctor	**Adaugare cont** - dacă nu are deja un cont, utilizatorul poate să-și creeze unul introducând numele, username-ul, parola, email si specialitatea în care profesează. Dacă email-ul introdus este folosit deja în alt cont, utilizatorul va primi un mesaj de atenționare
CF3	doctor	**Resetare parola** - se realizeaza prin trimiterea unui email cu un link care se accesează și se introduce parola nou dorițș și un cod de securitate primit deja tot pe email
CF4	doctor	**Vizualizare meniu personal** - afișearea opțiunilor cu posibilitatea de selectare a acțiunilor care doresc a fi efectuate
CF5	doctor	**CRUD programare** - utilizatorul are posibilitatea de a vizualiza, adăuga, modifica și șterge programări

CF6	doctor/utilizator android	**Distribuire fişiere** - utilizatorul poate să vizualizeze şi să distribuie fieşiere (de ex: pdf, word, jpg) către pacienți/doctori adăugând informații utile lângă acestea
CF7	doctor	**Adăugare vacanță** - utilizatorul poate să îşi aleagă o perioadă de concediu şi să selecteze dacă este disponibil în caz de urgență și trebuie să adauge un alt doctor care să-l înlocuiască
CF8	doctor/ utilizator android	**Trimitere mesaje** - utilizatorul poate să comunice cu pacienții/doctorii prin intermediul unui chat
CF9	utilizator android	**Programari** - utilizatorul poate să creeze o programare nouă, să le vizualizeze şi să le şteargă pe cele care nu le poate onora

Tabel 4.1 Cerințe funcționale

4.2.2. Cerinţe non-funcţionale

Cerințele non-funcționale (CNF) specifică calitatea atributeleor sistemului software. Față de cerințele funcționale, care descriu comportamentul sistemului, cele non-funcționale sunt un set de standarde care ne arată o operație specifică a sistemului. Cerințele non-funcționale vin ca raspuns la întrebari de tipul: "Care este nivelul de securitate al sistemului?" sau "Sistemul poate integra şi alte componente?" etc.

Mai jos este reprezentat un tabel cu câteva dintre cerințele non-funcționale regăsite în modulul implementat.

Nr. Crt.	Cerință	Descriere
CNF1	Securitate	Toată aplicația va avea un sistem de securitate ridicat pentru că aceasta conține multe date legate de pacienți, nu numai programări, dar și fișiere cu conținut sensibil, cum sunt rezultatele analizelor, fişe medicale sau plan de tratament. Această securizare este obținută prin folosirea de **JWT token-uri** generate la fiecare logare a utilizatorilor şi la fiecare request este verificat daca acel utilizator are o sesiune deschisă în acel moment, iar parola lui este criptată în baza de date Firebase. Un alt aspect ar fi *resetarea parolei* care se face prin email cu un **cod de securitate** atașat mail-ului pentru a asigura faptul că niciun utilizator nu poate să schimbe parola altui utilizator.
CNF2	Integrabilitate	Este procesul prin care sunt aduse împreuna componentele subsistemelor intr-un singur sistem şi se asigură faptul că aceste subsisteme funcționează ca un **întreg**. Putem considera modulul web ca fiind un subsistem, iar cel al aplicației android un alt subsistem. Din punctul de vedere al comunicării dintre pacient şi doctor, cele două subsisteme formează unul singur, pentru că **chat-ul** poate fi accesa de pe amândouă şi mesajele se pot vedea în timp real.

CNF3	Capacitate	Reprezintă cerința non-funcțională care ne arată câtă informație poate fi transmisă printr-un canal de comunicare. În contextul modulului de chat implementat prin tehnologia *PubNub* putem vorbi de un canal de comunicare. Conform documentației *PubNub* nu se restricționează[19]: • numărul de utilizatori care se pot conecta concurent în platformă • numărul de utilizatori care se conectează cu aceeași cheie • numărul de canale care pot fi folosite cu aceeași cheie
CNF4	Fiabilitate	Este cerința non-funcțională care se descrie ca abilitatea sistemului de a răspunde cât mai corect și fără eșec la acțiunile efectuate în platformă. De aceea sistemul este dezvoltat să ia în calcul cât mai multe scenarii și dacă anumite acțiuni nu se pot efectua datorită unor date introduse greșit sau utilizatorul încearcă să efectueze anumite acțiuni și nu este îndreptățit să le facă, acesta va primi mesaje de atenționare sau mesaje de succes când se pot realiza acțiunile. În acest fel, prin acoperirea cât mai multor scenarii posibilitatea de eșec a sistemului este din ce în ce mai mică.

Tabel 4.2 Cerințe non-funcționale

4.3. Cazuri de utilizare

Un caz de utilizare este o listă de acțiuni care definește interacțiunea dintre un actor și sistem pentru a atinge scopul dorit și ne ajută să explicăm cum se comportă sistemul în timpul procesului.

Sistemul implementat se poate împarți în două scenarii diferite: cazuri de utilizare pentru aplicația web și cazuri de utilizare petru aplicația Android. Implicit, se regăsesc și două tipuri de utilizatori, numiți actori: doctorul care are acces doar la aplicația web și utilizatorul obișnuit care este actorul specific aplicației Android.

[19] https://www.pubnub.com/docs/chat/resources/limits

4.3.1. Cazuri de utilizare doctor

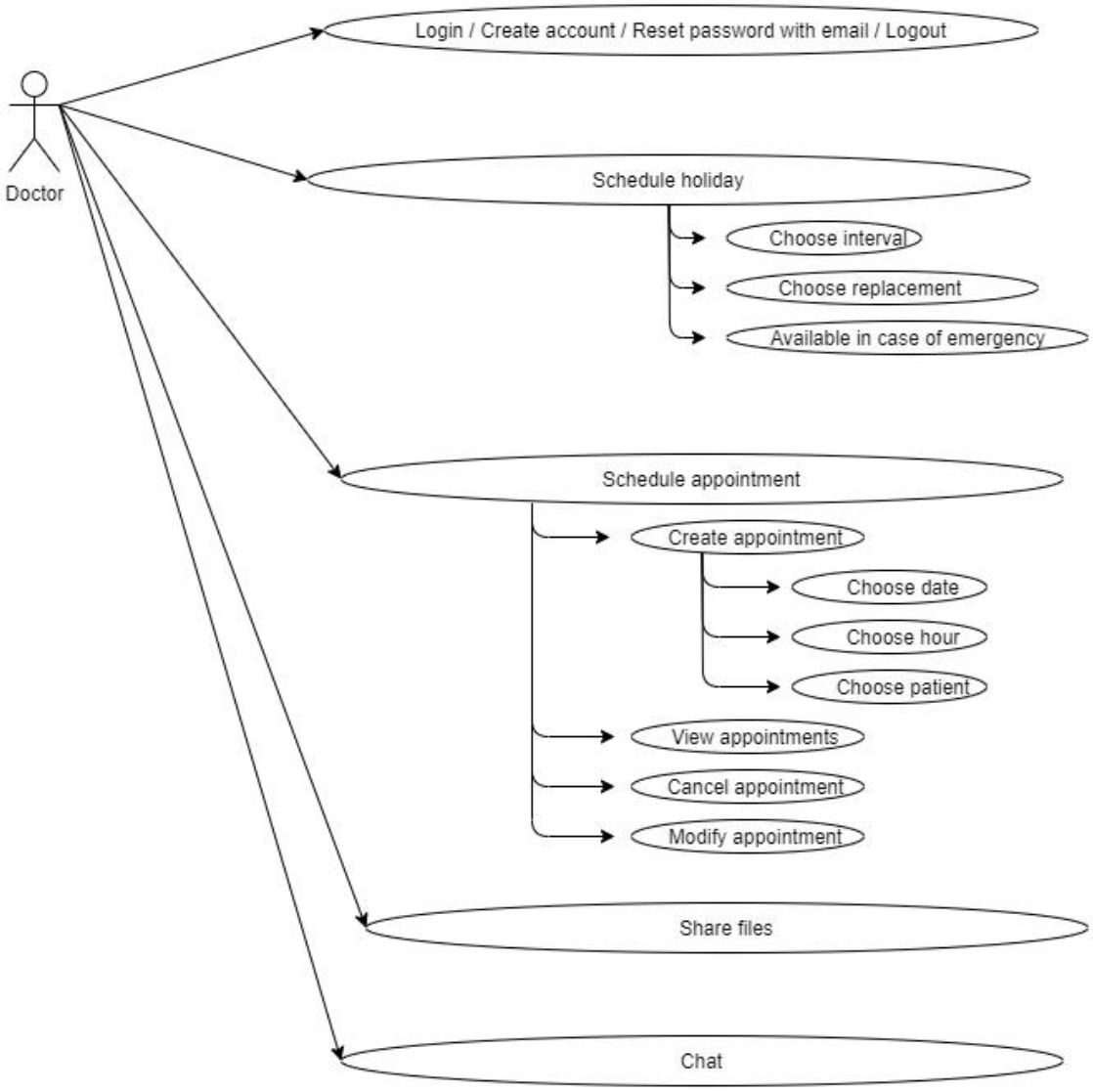

Figura 4.3 Cazuri de utilizare pentru utilizatorul aplicației web

În figura 4.3 sunt prezentate cazurile de utilizare ale doctorului în conextul aplicației web.

Cazul de utilizare 1

Nume: <u>Autentificare în platformă</u>
Actori: doctor
Precondiții:

- Actorul are deja un cont creat în sistem
- Actorul are o conexiune bună la internet

Postcondiții:
- Utilizatorul este logat cu succes
- Utilizatorul este redirecționat către pagina principală a aplicației

Scenariu principal:
a) Actorul accesează link-ul specific aplicației
b) Actorul introduce numele de utilizator și parola
c) Actorul apasă butonul de login

Scenariu alternativ:
- Dacă numele de utilizator sau parola sunt introduse greșit, atunci acestea nu sunt deja înregistrate în platformă
- Se va primi un mesaj de atenționare
- După ce se va citi mesajul se va face redirecționare tot la pagina de pornire, cea de login

Cazul de utilizare 2

Nume: <u>Înregistrare cont nou</u>
Actori: doctor
Precondiții:
- Actorul nu are deja un cont corespunzător email-ului înregistrat în platformă
- Actorul are o conexiune bună la internet

Postcondiții:
- Un cont nou a fost adăugat cu succes
- Utilizatorul are posibilitatea de a accesa aplicația
- Utilizatorul este redirecționat către pagina de login a aplicației

Scenariu principal:
a) Actorul apasă butonul de "Create new account"
b) Actorul introduce datele necesare: nume, nume de utilizator, parola, specialitatea de care face parte și email
c) Actorul apasă pe butonul de "Create" după ce a introdus toate datele

Scenariu alternativ:
- Dacă mai există un cont în platformă cu același email ca cel introdus se va primi un mesaj de atenționare și nu se va putea crea un nou cont
- Actorul nu va putea crea un nou cont dacă nu a completat toate câmpurile din secțiunea de înregistrare cont

Cazul de utilizare 3

Nume: <u>Resetare parolă</u>
Actori: doctor
Precondiții:
- Actorul trebuie să aibă deja un cont existent în platformă cu email-ul specificat
- Actorul are o conexiune bună la internet

Postcondiții:
- Parola contului a fost actualizată cu succes
- Actorul se poate loga cu noile date actualizate

Scenariu principal:
a) Actorul apasă butonul de "Reset password"
b) Actorul introduce email-ul cu care si-a facut contul existent deja în platformă (dacă nu, va fi scenariul alternativ 1)
c) Actorul primește un mail-ul pe adresa introdusă cu un link și un cod de securitate
d) Actorul accesează link-ul și este trimis la o pagină unde își poate completa datele
e) Actorul introduce email-ul și noua parolă și codul de securitate format din cifre primit pe email (dacă nu, va fi scenariul alternativ 2)
f) După ce termină de introdus datele necesare va apăsa butonul de "Reset"
g) Actorul primește un mesaj de confirmare că parola a fost modificată

Scenariu alternativ:
1. Dacă adresa de email introdusă este greșită, adică nu există deja in platformă va primi un mesaj de atenționare și nu va putea reseta parola contului. În schimb, va putea sa-și creeze un nou cont cu această adresă
2. După ce accesează link-ul primit pe email și introduce un alt cod de securitate decât cel trimis, nu va putea să-și reseteze parola

Cazul de utilizare 4

Nume: Logout
Actori: doctor
Precondiții:
- Actorul trebuie să fie conectat în aplicație
- Actorul are o conexiune bună la internet

Postcondiții:
- Utilizatorul va fi deconectat
- Utilizatorul la fi redirecționat de pagina de login

Scenariu principal:
a) Din orice submeniu s-ar afla utilizatorul acesta are acces la opțiunea de logout
b) Actorul apasă butonul de "Logout"

Scenariu alternativ:
- Dacă nu există o conexiune bună la internet nu se va putea realiza deconectarea utilizatorului
- Dacă sesiunea utilizatorului a expirat, acesta va fi deconectat direct, fară a ține cont de apăsarea butonului de logout

Caz de utilizare 5

Nume: Programare concediu
Actori: doctor

Precondiții:

- Actorul trebuie sa fie autentificat în platformă
- Actorul are o conexiune bună la internet

Postcondiții:

- Concediul a fost înregistrat cu succes
- Un alt doctor este asignat pentru perioada selectată

Scenariu principal:

a) Actorul accesează meniul de personal și selectează submeniul specific vacanței

b) Actorul introduce perioda în care va avea loc concediul sub forma unui interval selectând data de început și cea de sfârșit (dacă nu, va fi scenariul alternativ 1)

c) Are posibilitatea de a selecta dacă este disponibil în caz de urgențe sau nu

d) Actorul selectează un al medic care să-l înlocuiască în acea perioadă (dacă nu, va fi scenariul alternativ 2)

e) Actorul primește un mesaj care specifică că perioada de concediu a fost înregistrată

Scenariu alternativ:

1. Nu va fi posibilă înregistrarea dacă data de început a vacanței nu este mai mică decât data de sfârșit

2. Actorul va fi nevoit să selecteze alt doctor înlocuitor dacă cel inițial ales are și el concediu înregistrat în perioada predefinită

4.3.2. Cazuri de utilizare utilizator obișnuit

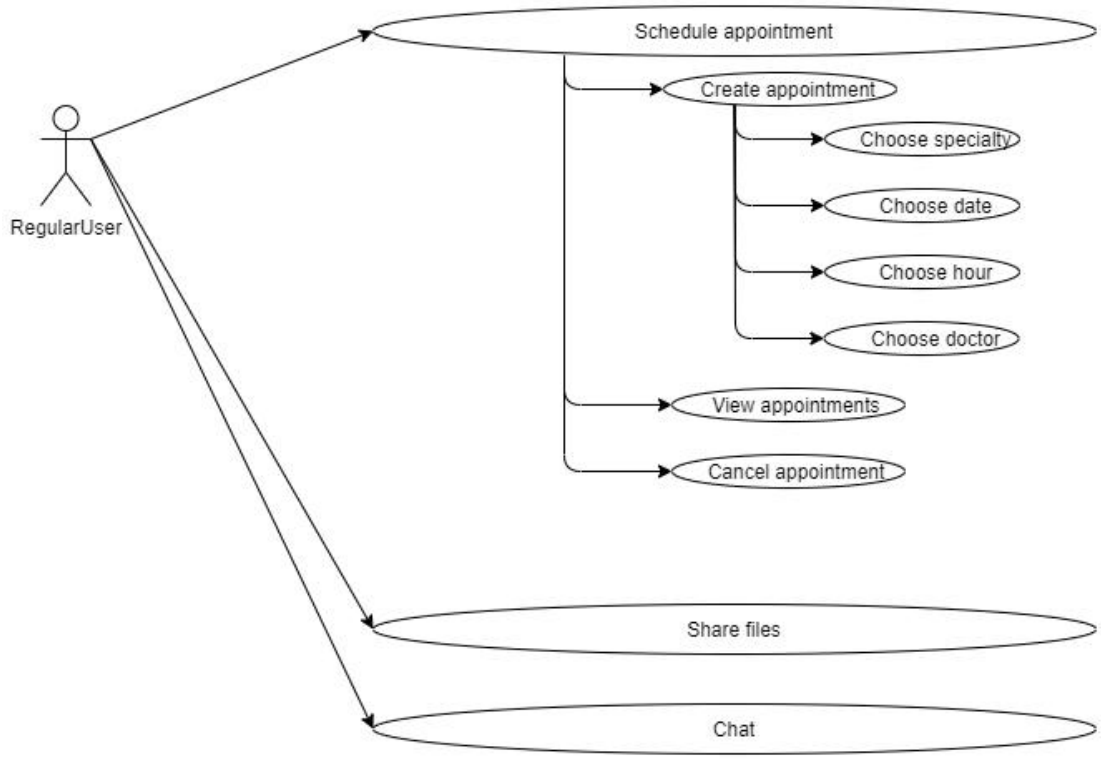

Figura 4.4 Cazuri de utilizare pentru utilizatorul aplicației mobile

În figura 4.4 sunt reprezentate cazurile de utilizare specifice aplicației Android efectuate de către utilizatorul obișnuit. Mai jos am să descriu mai în detaliu aceste cazuri. Ele sunt asemanătoare cu cele specifice aplicației web, de accea le voi prezenta doar o singură dată în cele ce urmează.

Caz de utilizare 6

Nume: Programare
Actori: doctor, utilizator obișnuit
Precondiții:
- actorul trebuie să fie autentificat în aplicație: doctorul în aplicația web, iar utilizatorul obișnuit în aplicația mobilă
- actorul trebuie să aibă o conexiune bună la internet
- actorul se află pe meniul de personal în submeniul specific programării

Postcondiții:
- Acțiunea specifică programării a fost îndeplinită cu succes
- Nu se mai pot efectua alte programări la data și ora creată

Scenariu principal:
a) Pentru adăugare programare
  - Doar dacă utilizatorul este obișnuit atunci se va alege și specialitatea doctorului la care vrea să-și facă programarea
  - Actorul selectează doctorul/pacientul
  - Actorul selectează data când vrea să fie efectuată programarea și apasă butonul de "ok" (dacă nu, va fi scenariul alternativ 1 sau 4)
  - Se vor afișa orele disponibile și se va alege o varintă
  - Actorul primește mesaj că programarea a fost înregistrată

b) Pentru vizualizare programări
  - Utilizator obișnuit: se afișează toate programările specifice pacientului logat
  - Doctor: se selectează data se apasă butonul "ok" și se afișează toate programările din acea dată (dacă nu, va fi scenariul alternativ 2)

c) Pentru ștergere programare
  - În tabelul de vizulizare a programărilor, actorul are posibilitatea de a șterge o programare viitoare acționând butonul de "delete" (dacă nu, va fi scenariul alternativ 3)
  - Actorul primește mesaj că programarea a fost ștearsă cu succes

Scenariu alternativ:
1. Dacă în data selectată nu sunt ore disponibile atunci nu se va putea efectua programarea și trebuie aleasă o altă zi
2. Dacă în data selectată nu sunt programări atunci tabelul va fi gol
3. Nu se va putea șterge o programare din trecut care a fost deja onorată
4. În cazul utilizatorului obișnuit, dacă în data pentru programare doctorul selectat este în concediu se face redirecționare către un doctor înlocuitor sau are posibilitatea să schimbe data

Caz de utilizare 7

Nume: <u>Distribuire fișiere</u>
Actori: doctor, utilizator obișnuit
Precondiții:
- Actorul are o conexiune bună la internat
- Actorul este logat în platformă
- Actorul se află pe meniul de personal în submeniul specific distribuirii de fișiere

Postcondiții:
- Un nou fișier a fost adaugat cu succes în platformă și distribuit persoanei selectate
- Actorul poate vizualiza într-un tabel toate fișierele distribuite, plus cel nou adăugat

Scenariu principal:
a) Actorul selectează pacientul/doctorul la care vrea să-i distribuie un fișier (dacă nu, va fi scenariul alternativ 1)
b) Actorul adaugă fișierul dorit din calculator sau telefon
c) Actorul acționează butonul de "Send"
d) Actorul primește un mesaj că fișierul a fost distribuit cu succes

Scenariu alternativ:
1. Fișierul nu va putea fi distribuit dacă nu va fi ales un doctor sau un pacient

Caz de utilizare 8

Nume: <u>Chat</u>
Actori: doctor, utilizator obișnuit
Precondiții:
- Actorul trebuie să fie autentificat în platformă
- Actorul se află în meniul personal, în submeniul dedicat chat-ului
- Actorul are o conexiune bună la interne t (dacă nu, va fi scenariul alternativ 1)

Postcondiții:
- Actorul a facăut schimb de mesaje cu persoana dorită
- Mesajele au fost distribuite cu succes

Scenariu principal:
a) Actorul alege din meniul superior pacientul/doctorul cu care vrea să comunice
b) Actorul compune mesajul și după finalizarea lui apasă butonul de "Send"
c) Se va afișa schimbul de mesajele

Scenariu alternativ
1. Dacă utilizatorul nu are o conexiune bună la internet sau dacă nu este autentificat în platformă, acesta nu poate să folosescă chat-ul

4.4. Viziune tehnologică

Sistemul MHealth a fost dezvoltat cu o gamă largă de tehnologii care s-au îmbinat ușor și au ajutat la realizarea acestuia.

Modulul *Personal* a fost implemetat folosind tehnologii specifice pentru mobil, web (backend și frontend), bază de date și comunicare chat (PubNub), cum se pot observa și în figura 4.5.

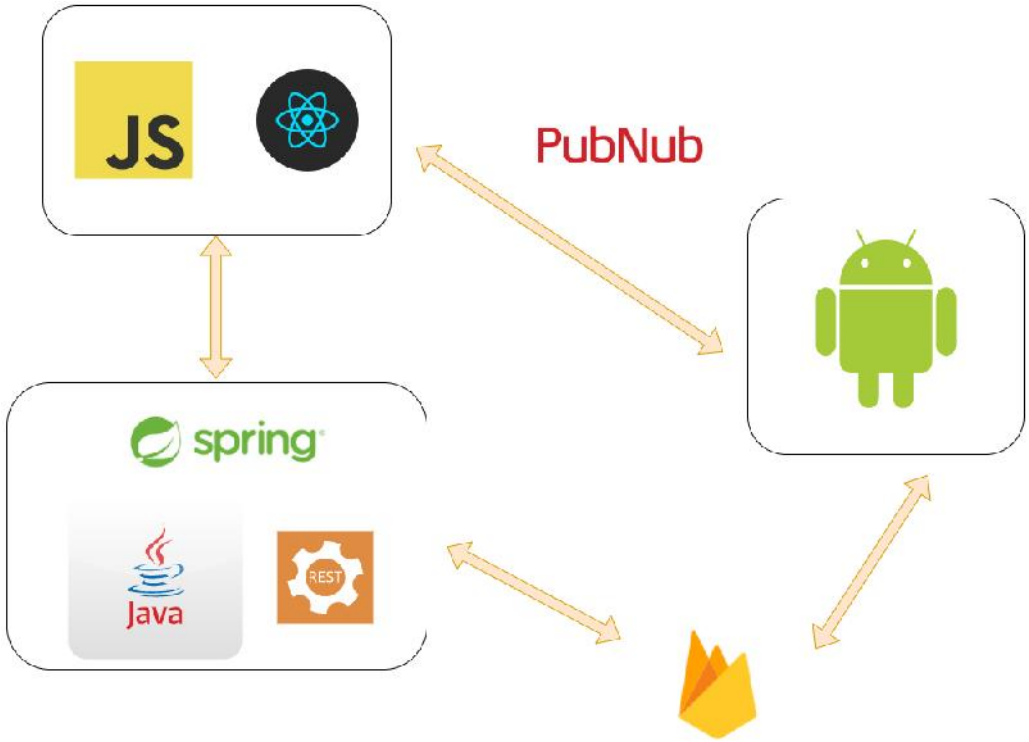

Figura 4.5 Tehnologiile folosite

4.4.1. PubNub

PubNub este o platformă de comunicare în timp real, care dispune de un SDK ușor de folosit care ajută la integrarea unui chat în aplicații atât web, cât și mobile.

Se folosește mesagerie de tip "publish-subscribe" între unul sau mai multe dispozitive. Oferă suport gratuit până la un milion de tranzacții și 1 GB de persistență a datelor.

Utilizatorii se conectează la același canal de comunicare pentru a face schimb de mesaje în timp real, având conversații unu la unu sau dacă sunt mai mulți, se va forma un grup. Se vor introduce un set de chei pentru a putea comunica mai multe dispozitive prin același chat.

Un dashboard pentru admin este disponibil pentru a putea crea chei și subchei, a urmări utilizarea contului, gestiona informații legate de plată și pentru a adjusta setările de configurare. Acest dashboard este exemplificat în figura 4.6.

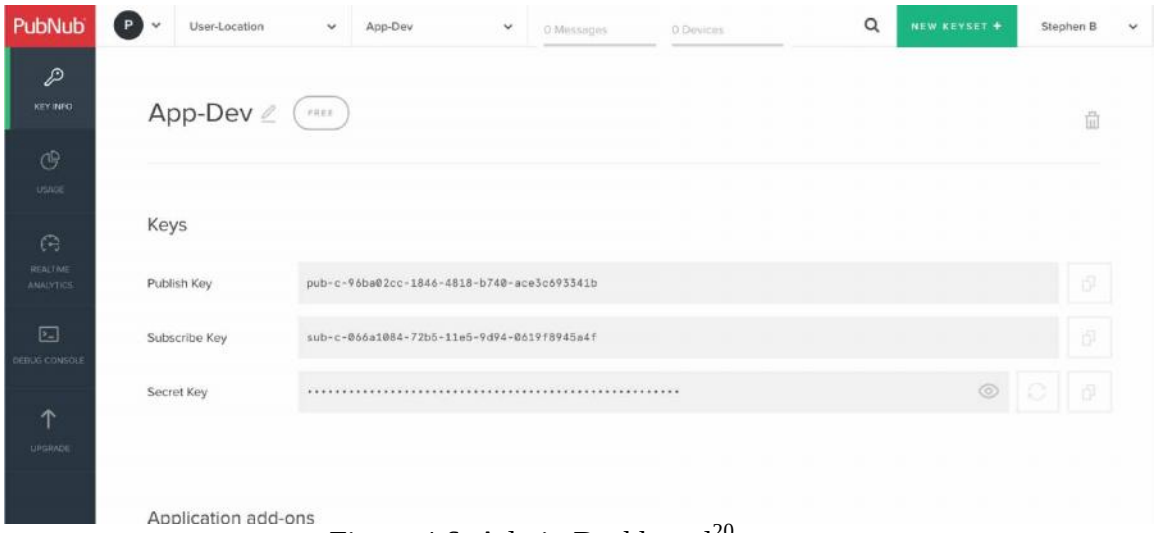

Figura 4.6 Admin Dashboard[20]

4.4.2. Java

Java este un limbaj de programare orientat pe obiect folosit în multe aplicații web și mobile. Programele java se execută în JVM, care reprezintă mașina virtuală java și transformă codul sursă într-un cod de octeți care poate fi interpretat de către codul mașină specific calculatoarelor. Aceste programe pot fi rulate pe orice platforme care au deja instaleta mșina virtuală. Față de alte limbaje mai vechi, Java are un grad de portabilitate mai ridicat.

IntelliJ IDEA este o unealtă software care susține dezvoltarea programelor scrise în Java, creată de JetBrains. Este un mediu ușor de utilizat de programatori și susține numeroase tehnologii, printre care amintim și: Maven, Spring, JUnit, Android, Tomcat, Node.js etc.

4.4.3. Spring

Spring este cel mai popular framework care poate fi folosit de orice aplicație Java și conține extensii pentru a construi aplicații web peste platforma Java EE.

Printre atributele pe care le are acest framework *open source* se numără și:

- *flexibilitate* - extensii și librări terțe ajută dezvoltare unor aplicații din ce în ce mai complexe. Inversion of Control (IoC) și Dependency Injection (DI) reprezintă baza pentru multe funcționalități.

[20] https://www.pubnub.com/developers/tech/admin-dashboard/

- *productivitate* - **Spring Boot** ajută la accelerarea dezvoltării aplicațiilor și combină necesitățile aplicațiilor și autoconfigurările pentru a dezvolta microservicii cât mai sigure.
- *securitate* - **Spring Security** este un framework care se axează pe autentificarea și autorizarea aplicațiilor java.[21]

4.4.4. REST

Representational state transfer (**REST**) este un stil arhitectural în care datele și funcționalitățile sunt considerate resurse și sunt accesate folosind *Uniform Resource Indentifiers* (**URI**), link-uri tipice pentru web.[22]

Metodele HTTP sunt folosite în cadrul aplicațiilor care folosesc serviciile REST pentru a face apeluri către server. Se amintesc mai jos câteva dintre aceste metode:

- *GET* - metodă folosită pentru a primi informații sau reprezentări ale resurselor
- *POST* - metodă folosită pentru a adăuga o nouă resursă
- *PUT* - metodă folosită pentru a modifica o resursă deja existentă
- *DELETE* - metodă folosită pentru a șterge o resură deja existentă

4.4.5. JWT

JSON Web Token **(JWT)** este un standard de internet pentru a crea date cu o semnătură opțională și se trimit date între părți sub formă de obiect de tip JSON.

JWT-urile pot fi criptate astfel încât să se mențină confidențialitatea între părți prin *token*-uri. JSON Web Tokens se folosesc în general pentru:

- *Autorizare* - este scenariul cel mai comun, după ce user-ul este logat, fiecare cerere către server va include JWT, permițând utilizatorului să acceseze rute, servicii și resurse care sunt valibile cu acel token.
- *Schimb de informații* - o modalitate bună pentru a transmite într-un mod sigur informații între părți

Strucura JSON Web Token este formată din 3 părți

- *Header* - format din tipul token-ului și algoritmul
- *Playload* - a doua parte din token care poate să fie: înregistrat, public sau privat
- *Signature* - formată din heder-ul criptat , playload criptat și algoritmul specificat în header.[23]

Cu ajutorul JWT s-a realizat securitatea întregii aplicații web, fiecare request având ca header un token pentru a se realiza apeluri autorizate către backend.

[21] https://spring.io/why-spring
[22] https://restfulapi.net/
[23] https://jwt.io/introduction/

4.4.6. NodeJs

NodeJS este o platformă construită într-un mediu runtime JavaScript pentru a construi aplicații scalabile și rapide. Se execută cod JavaScript în afara unui browser web.

NodeJs asigură librării vaste care conțin module JavaScript care simplifică implementarea aplicațiilor web.

Toate API-urile din librăria Node.js sunt asincrone și pentru ca este construită peste Google Chrome V8 JavaScriptEngine este foarte rapidă în executarea codului.[24]

4.4.7. React

React este o librărie de tip JavaScript pentru realizarea interfețelor utilizator. Folosește o abordare declarativă pentru a dezvolta UI interactive. Construiește componente încapsulate și își gestionează propria stare pentru a crea interfețe utilizator complexe.

Codul scris în React este format din entități numite componente. Acestea pot fi randate pentru un anumit element particular în DOM (Document Object Model) folosind librăria React DOM. [25]

Metode care se ocupă de ciclul de viață al componentelor:

- *componentDidMount* - este apelată atunci când componenta a fost creată în interfața utilizator
- *render* - este cea mai importantă metodă a ciclului de viață și este necesară în fiecare componentă. Este apelată de fiecare dată când starea componentei este modificată

Visual Studio Code este un editor realizat de Microsoft care suportă mai multe limbaje de programare printre care și JavaScript și ajută programatorii să dezvolte aplicații destinate realizării interfețelor de tip utilizator într-un mediu ușor de folosit cu opțiuni de debugging, completarea inteligentă a codului, evidențierea unor cuvine cheie din cod și posibilitatea integrării Git-ului.

4.4.8. Android

Android este un sistem de operare utilizat pentru dispozitive mobile *touchscreen* cum sunt smartphone-urile și tabletele. Este un software *open-source*, gratis ce permite dezvoltatorilor folosirea limbajului de programare Java.

Această platformă se bazează pe nucleul Linux și ultima verisiune stabilă lansată este Android 10. Printre caracteristicile sistemului se regăsesc și următoarele: bluetooth, Wi-Fi hotspot, suport pentru media streaming, adăugarea unui card microSD oentru stocare externă, GPS, accelerometru, barometru etc.

[24] https://nodejs.org/en/about/
[25] https://reactjs.org/

În figura 4.7 se ilustrează o schemă generală a arhitecturii sistemului în formă de stivă de la nivelul cel mai de jos, reprezentat de nucleul Linux, pâna la aplicații.

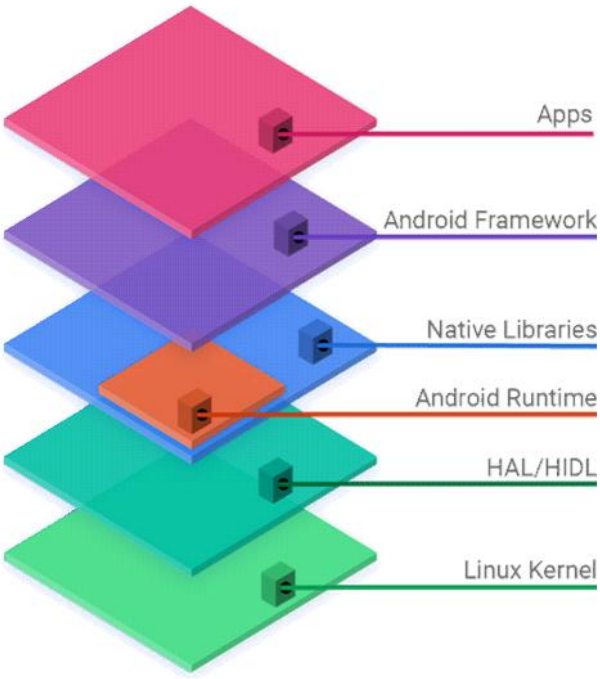

Figura 4.7 Arhitectura de bază a platformei Android[26]

Android Studio este mediul de dezvoltare oficial pentru sistemul de operare Android. Acesta a fost construit peste soft-ul IntelliJ IDEA pentru dezvoltatori pentru a realiza aplicații mobile într-un mediu ușor de folosit, utilizând limbajul de programare Java.

4.4.9. Postman

Postman este o unealtă software care vine în ajutorul programatorilor pentru a testa apelurile către API-uri. Utilizatorii Postman introduc date, alegând tipul de metodă (GET, POST, PUT, DELETE) și URL-ul care se dorește a fi testat. Datele sunt trimise către server-ul web și se afișează răspunsurile de succes dar și dacă sunt erori. [27]

4.4.10. Git

Git este un versiune distribuită de sistem de control, open-source, realizată pentru a susține cu eficiență și rapiditate proiecte de toate dimensiunile.[28]

[26] https://developer.android.com/guide/platform
[27] https://www.postman.com/
[28] https://git-scm.com/

Este proiectat pentru a ajuta programatorii să-și gestioneze munca și să-și urmărească schimbările care se execută asupra proiectului, mai ales dacă este unul realizat împreună cu mai multe persoane.

Unul dintre avantajele principale ale utilizării git-ului este acela că există o ramură principală, numită **master** care conține cod funcțional și de cea mai bună calitate și va reprezenta o însumare a mai multor ramuri de lucru.

Feature branch, reprezintă o ramură de lucru, un mediu izolat în care se pot face schimbări asupra codului de bază, adăugând îmbunătăți acestuia. Când se vrea implementarea unei noi funcționalități se va crea un nou branch.

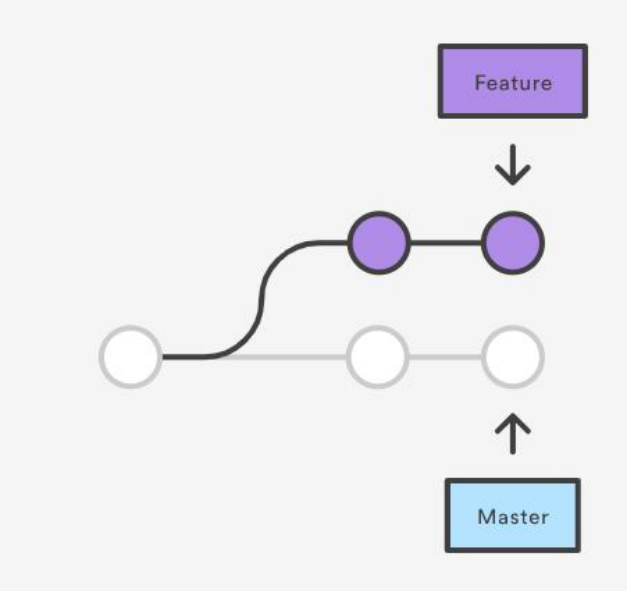

Figura 4.8 Fluxul de lucru a unui feature branch[29]

Pull request se realizează atunci când schimbările făcute asupra unui branch vor să fie vizibile pentru toți colaboratorii acelui proiect. Dacă nu există conflicte, se va face **merge**, adică toate schimbările vor fi aduse fluxului principal de lucru.

GitHub Desktop este un editor care ajută utilizatorii să relizeze schimbările asupra unui proiect printr-o interfață intuitivă și mai ușor de folosit decât utilizarea linie de comandă pentru a face acțiuni precum: pull request, merge branch, commit, etc.

Git-ul a venit în ajutor pentru a ne putea organiza mai ușor ca echipă, având mereu pe branch-ul de master doar cod funcțional. Așa toți 3 am avut acces la ultima variantă a proiectului.

[29] https://www.atlassian.com/git/tutorials/why-git#git-for-developers

4.5. Tehnologiile luate în considerare

Problematica	Sursa solutiei	Evoluția tehnologiei alese +killer usecase
Securizare	Java	**Criptarea obiectelor de tip Emergency** - s-a găsit o metodă mai ușor de implementat și mai sigură **Blockchain**
Deploy Cloud	Google Cloud	**App Engine**
Procesare Imagini	Google Cloud	**Vision API**
Procesare Text	Google Cloud	**Natural Language API** - rezultatele furnizate nu sunt suficient de precise pentru scopul aplicației si este necesara antrenarea unui agent specializat **AutoML Language API**
Procesare Audio	Google Cloud	**Speech-to-text Language API**
Chat - web+mobile	PubNub	**RabbitMQ** - s-a dorit utilizarea unei tehnologii de ultimă generație chiar dacă sistemul MHealth nu beneficiază de toate funcționalitățile la dispoziție de către PubNub **PubNub Chat**
Push notifications	-	**Google Nearby Messages API** - aria de acoperire oferită de acest API este prea mică, în jur de 20m maxim, în timp ce sistemul MHealth își propune oferirea notificărilor indiferent de locația utilizatorilor **Firebase Cloud Messaging** - deși această tehnologie oferă funcționalitatea dorită de a trimite notificări indiferent de locație, nu se poate filtra utilizatorii în funcție de distanța față de urgență **Geofencing API**
Traducere Text	Firebase	**ML Kit**
Translatare din coordonate in adresa și invers	-	**Geocoding API**
Stocare Fisiere	Firebase	**Firebase Storage**
Stocare date personale	Firebase	**Firebase Database**
Preluare date buletin	Firebase	**ML Kit**

Tabel 4.3 Tehnologiile folosite în sistemul MHealth

Capitolul 5. Proiectare de Detaliu si Implementare

În acest capitol se prezintă detalii depre proiectare și implementare, prin ilustrarea și descrierea diagramelor care reprezintă sistemul și comportamentul lui. În partea a doua a capitolului se exemplifică implemetarea modulelor din sistem.

5.1. Diagramele sistemului

Vor fi descrise în acest subcapitol digrame specifice ambelor componente unde este cazul, pentru componenta mobilă, cât și pentru cea web.

5.1.1. Flowchart pentru utilizatorul aplicației mobile

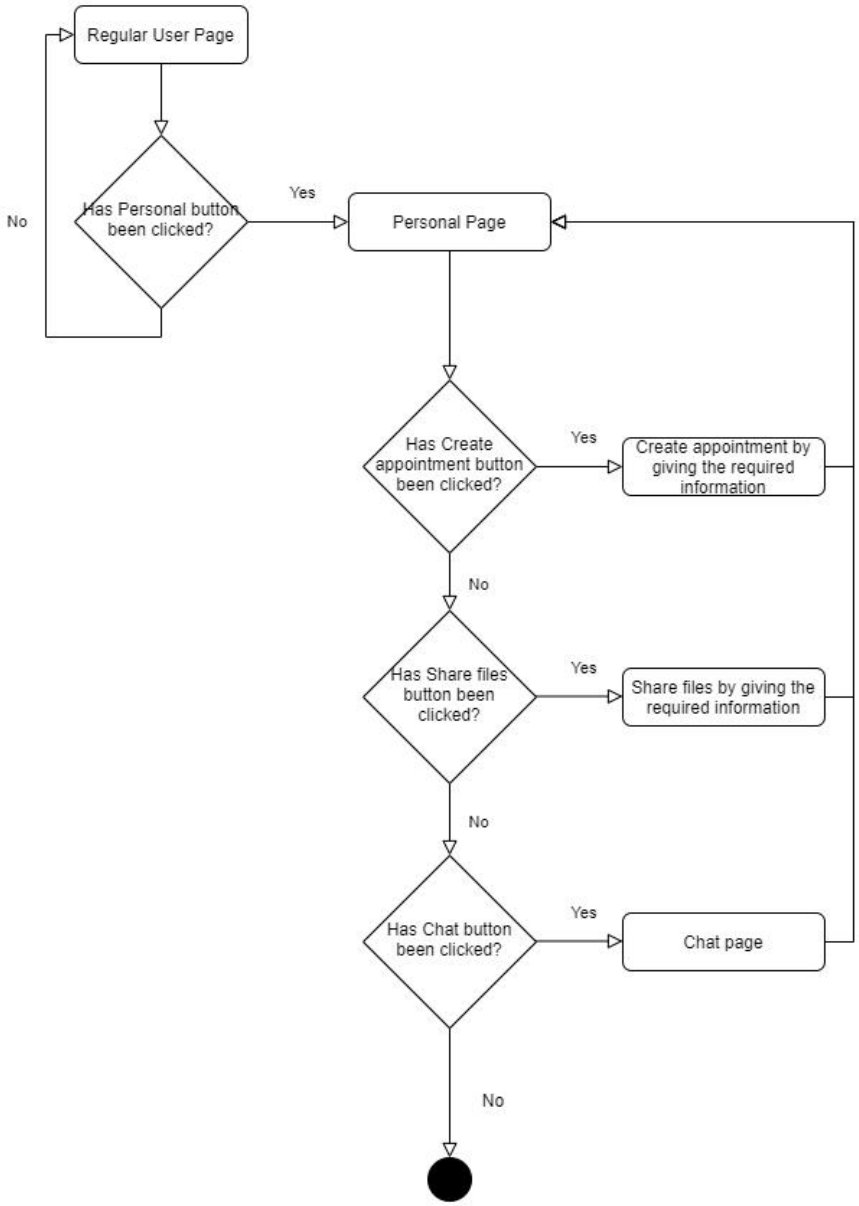

Figura 5.1 Flowchart pentru utilizatorul aplicației mobil

În figura 5.1 este ilustrată diagrama de flux de date specifică aplicației Android. Este prezentată în mare modul în care se poate interacționa cu aplicația și pașii care sunt urmați atât pentru o acțiune pozitivă, cât și pentru una negativă. După partea de autentificare, este afișat un meniu din care poate fi selctat modulul dorit: cel de raportare a urgențelor sau cel personal. Modulul personal are la bază o pagină cu butoane sub forma unor imagini care te redirecționează către diferite operații (programare, distribuire de fișiere și chat) de unde trebuie completate informațiile cerute.

5.1.2. Flowchart pentru utilizatorul aplicației web

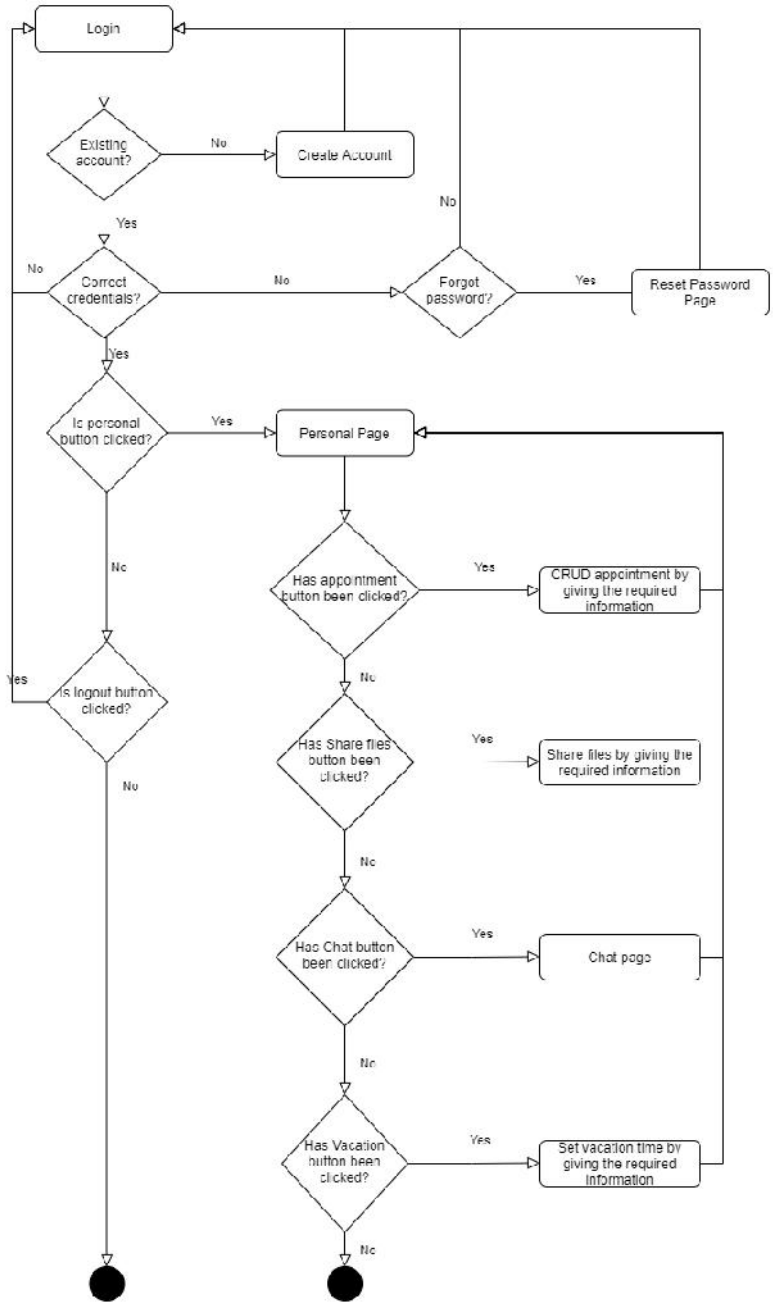

Figura 5.2 Flowchart pentru utilizatorul aplicației web

Fluxul de date specific aplicației web este prezentat în figura 5.2. Sunt prezentate opțiunile pe care utilizatorul le are dacă răspunde pozitiv, dar și negativ la acțiunile posibile din aplicațile. La începutul interacțiunii cu aplicația se găsesc elemente legate de login și de cont, mai exact adăugare de cont și resetare parolă în cazul în care utilizatorul și-a uitat parola sau vrea să o schimbe. Aceste funcționalități vor fi descrise în subcapitolul destinat descrierii implementării. De pe pagina principală de personal se pot selecta mai multe opțiuni de getionare a comunicării dintre doctor și pacient, completând câmpurile specificate.

5.1.3. Diagrama bazei de date

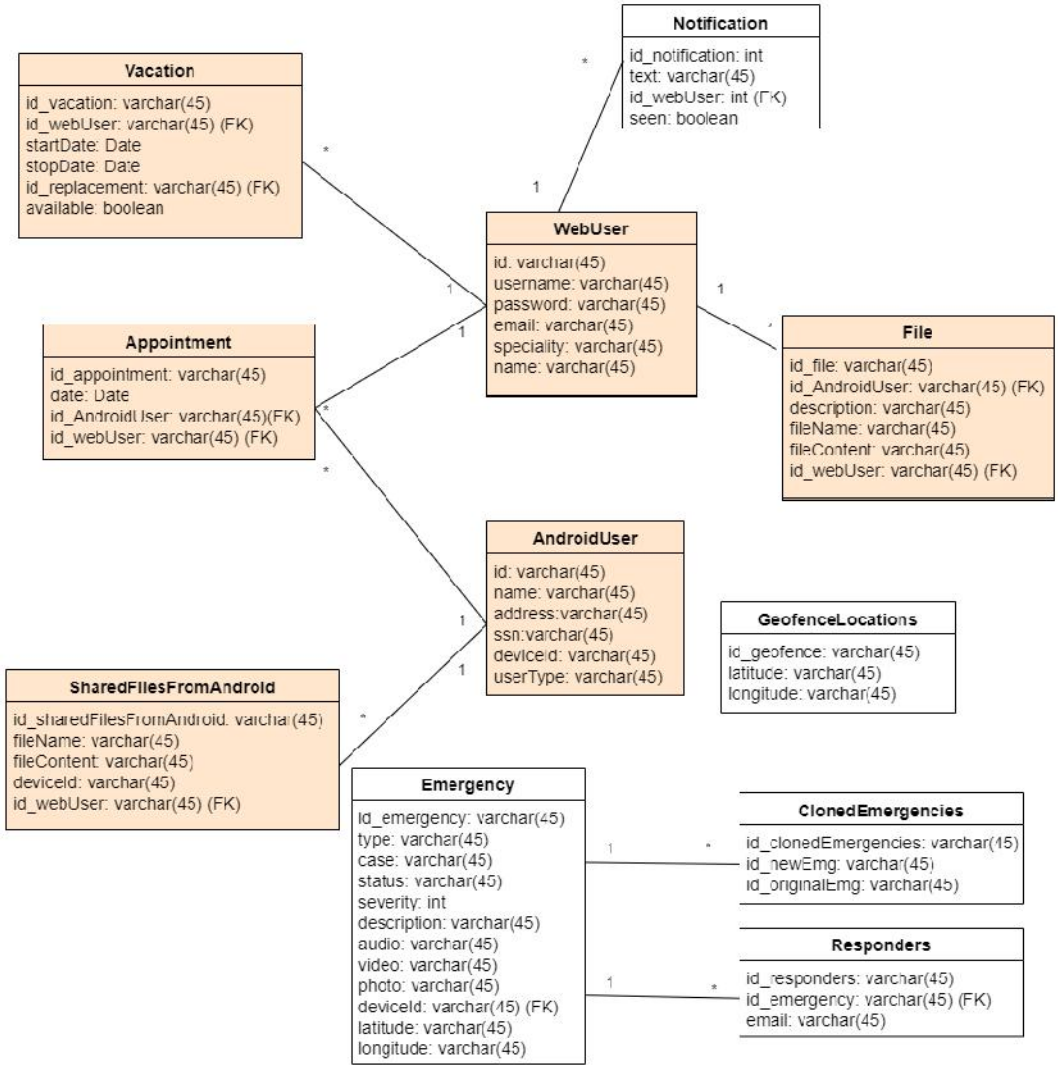

Figura 5.3 Modelul de date utilizat în modulul *personal*

Pentru stocarea și sincronizarea datelor sistemului s-a ales o bază de date NoSQL bazată pe cloud, *Firebase Realtime DataBase*.[30] Datele sunt sincronizate în timp real cu toți clienții și rămân valabile când aplicația este offline.

[30] https://firebase.google.com/docs/database

Datele sunt stocate sub formă de JSON și toți clienții conectați primesc automat cele mai noi modificări pe care le-au suferit datele stocate în baza de date.

S-a optat pentru dezvoltarea unei baze de date NoSQL pentru a avea tot timpul ulltima versiune a datelor atât pe partea de web, cât și pe Android și este mult mai scalabilă și mai performantă față de o baza de date relațională. NoSql permite scalare orizontală între diferite servere, față de baza de date relațională care oferă scalarea verticală care nu este optimizată pentru a stoca o cantitate mare de informații.

Modelul de date a întregului sistem MHealth este prezentat în figura 5.4, iar tabelele colorate sunt cele folosite în cadrul modulului personal. Aceste tabele sunt folosite atât pentru componenta web, cât și pentru cea Android. S-a realizat o schemă în varintă relațională pentru a fi mai ușor de urmări și pentru a arăta relațiile dintre tabele.

Mai jos vor fi descrise conținutul colecțiilor implementate. Toate conțin un id autogenerat de către Firebase.

WebUser - aici se stochează datele despre toți utilizatorii de pe web, adică doctorii înregistrați în sistem. La crearea unui cont sunt necesare completarea următoarelor câmpuri:

- username, password - necesare pentru logare
- email - este verificat înainte de crearea de cont și dacă este înregistrat deja în sistem nu se va putea un alt cont cu un email deja existent. Dacă utilizatorul nu își mai știe detalii legate de cont, își poate reseta parola cu un cod de securitate primit pe email.
- name - numele complet al doctorului
- speciality - specialitatea în care profesează doctorul

AndroidUser - colecție destinată stocării datelor despre utilizatorii Android, Aceste informații sunt colectate de pe o poză de buletin și adăugate în Firebase. Informațiile colectate sunt: numele, adresa, ssn (de pe un buletin românesc va fi cnp), deviceId, care reprezintă codul intern al telefonului și userType diferențiază un utilizator normal de unul cu pregătire medicală.

Appointment - este colecția care deține informațiile despre programările dintre doctori și pacienți, utilizatori ai aplicației mobile, de aceea există câmpuri cu id-urile utilizatorilor aplicației mobile și web. Colecția mai conține câmpul de dată care este stocată sub formă de *timestamp* ce conține atât data, cât și ora.

Vacation - stochează perioada în care doctorul și-a stabilit perioada de concediu și conține câmpurile:

- id_webUser - id-ul doctorului care a solicitat perioada de vacanță
- startDate, stopDate - data de început și de final a concediului
- id_replacement - id-ul doctorului înlocuitor cu aceeași specializare ca a doctorului solicitant
- available - este un boolean care marchează dacă doctorul ar fi disponibil în cazul unei urgențe chiar dacă este în perioada de concediu

File - colecție destinată stocării fișierelor trimise de pe web către pacienți. Acesta conține id-urile doctorului și a pacientului și:

- fileName - numele fișierului distribuit
- fileContent - URL-ul specific fișierului care poate fi accesat pentru descărcarea lui
- description - descrierea dată de către doctor înainte de trimiterea fișierului cu informații extra sau detalii despre acesta

SharedFilesFromAndroid - este similară cu cea de File doar că este destinată stocării fișierelor trimise de către utilizatorii Android. Conține id-ul utilizatorului web și id-ul device-ului de pe care a fost trimis fișierul, numele fișierului și URL specific mapat în câmpul fileContent.

Pentru o reprezentare non-relațională a bazei de date, s-a folosit o unealtă de modelare a schemelor numită Hackolade.[31] Această reprezentare este ilistrată în Anexa 5 a documentului. S-a dorit folosirea unui asemenea tool pentru a încerca ceva nou și pentru a evidenția colecțiile, cămpurile acestora și relațiile dintre ele.

5.1.4. Diagramă secvență

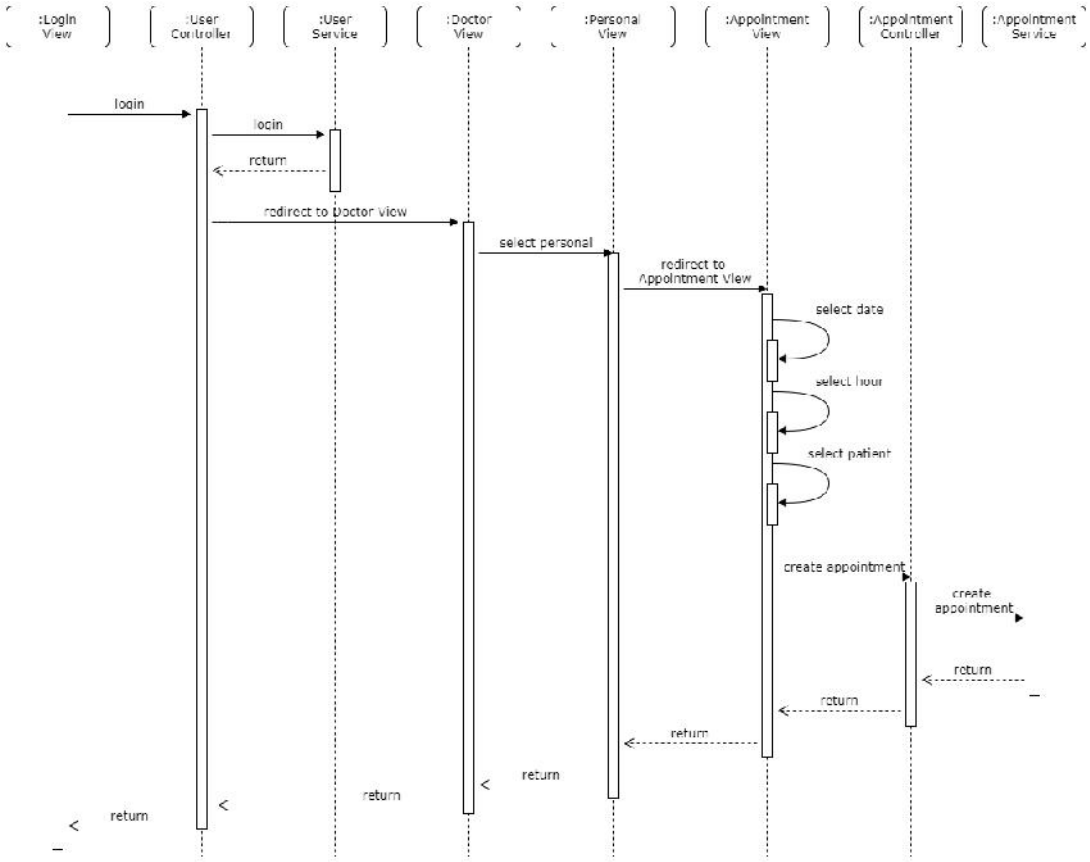

Figura 5.4 Diagramă de secvență "Create appointment" pe partea de web

[31] https://hackolade.com/

În figura 5.4 este ilustrată o diagramă de secvență specifică unei funcționalități de pe web, mai exact cea de adăugare de programare. S-au urmărit toți pașii de la partea de view, din frontend până la nivel de serviciu, unde se face adăugare în Firebase Database.

După un parcurs complet de la view până la service, pentru funcționalitatea de Login, se selectează din pagina specifică doctorului modulul personal de unde se poate adăuga o programare, selectând data, ora și pacientul printr-un apel către backend.

5.1.5. Diagramă de pachete

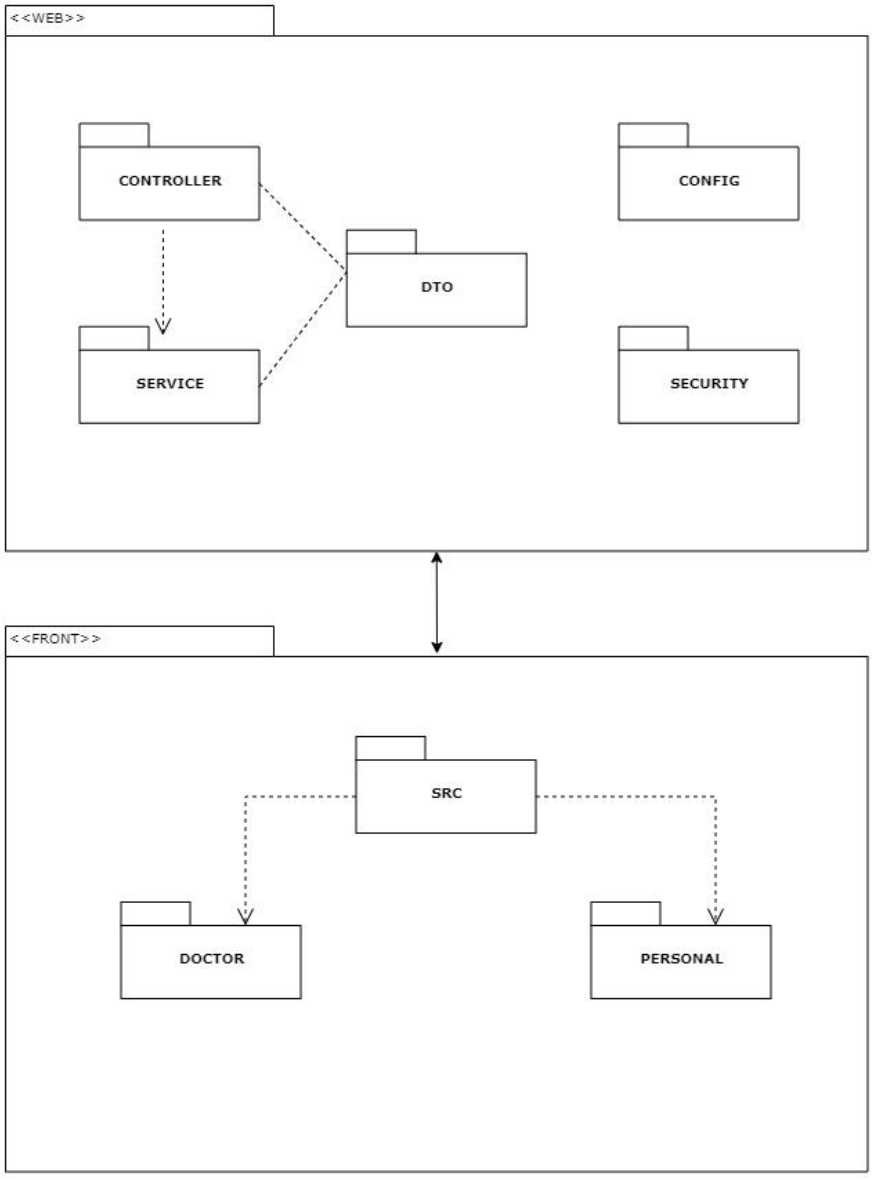

Figura 5.5 Diagrama de pachete a aplicației Web

În figura 5.5 sunt prezentate pachetele aferente aplicațiilor de frontend și de backend a modulului web. Frontend-ul este realizat în React și conține fișiere de javascript și css. S-a împarțit sursa proiectului în 2 foldere pentru a diferenția partea personală de cea a doctorului menită procesării informațiilor venite de la urgențele raportate de pe aplicația mobilă.

Partea de backend a aplicației web este împarțită în pachete de:

- config - conține clasele de inițializare a proiectului cu baza de date Firebase
- security - pachetul de securitate care se ocupă de sesiuni şi serviciul de stocarea a codurilor de securitate pentru resetarea parolei
- dto - toate mapările de clase necesare transmiterii frontend - backend
- controller - se află clasele cu endpont-urile specifice aplicației
- service - toate clasele cu rol de service în care se fac şi apelurile către Firebase database

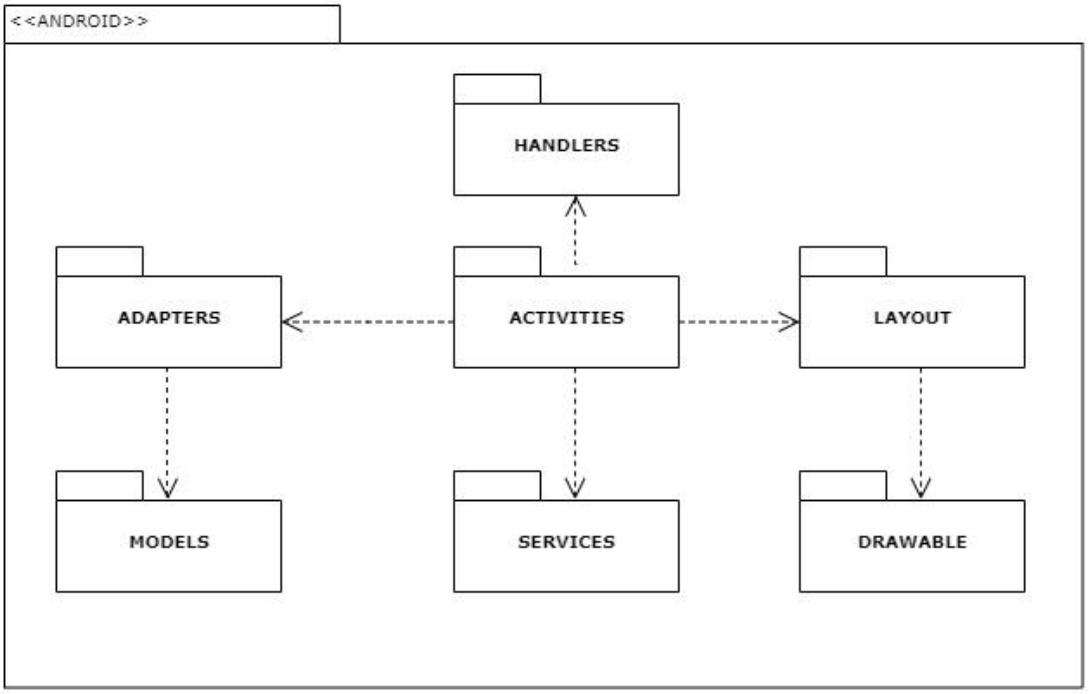

Figura 5.6 Diagrama de pachete a aplicației Android

În aplicația Android pachetele sunt denumite ca în figura 5.6. Codul este organizat pe activități pentru a delimita funcționalitățile. Partea de aspect se regăsește în pachetul *layout*, care este gestionat de către activități. Pachetul *drawable* conține toate pozele care apar în aplicație. Pachetul de *handlers* conține clase care se ocupă de logica unor alte clase cum este cea de detinată getionării informațiilor din tabele. Clasele de tip Adapter se află în interiorul pachetului de *adapters* și gestionează accesul la datele componentelor din view. Clasele care definsec obiecte și ce conține acestea sunt stocate în pachetul de *models*.

5.1.6. Diagramă de deployment

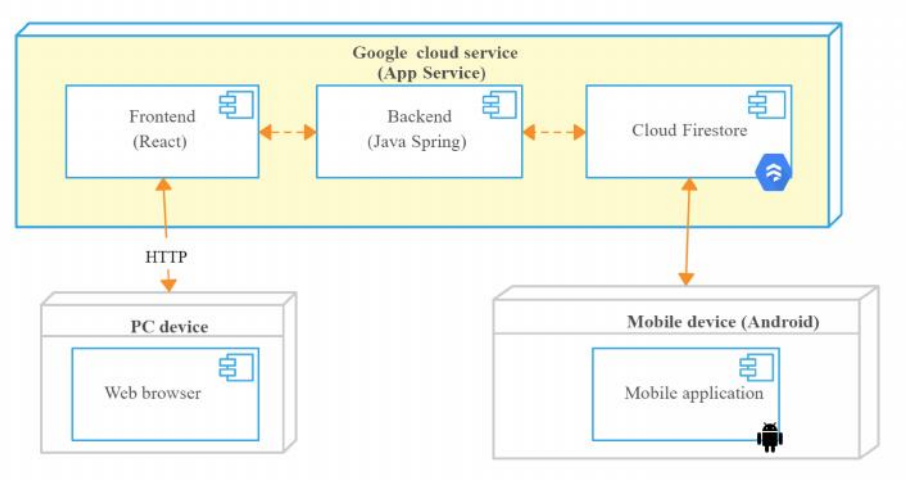

Figura 5.7 Diagramă de deployment

Aplicația de web este găzduită pe Cloud, adică frontend-ul și backend-ul aplicației, iar baza de date este una NoSql care face parte din Cloud Firestore. De pe browser se face apel cu ajutorul protocolului HTTP.

Aplicația mobilă nu este găzduită momentan pe nimic, dar ar putea fi adăugată pe GooglePlay pentru o dezvoltare ulterioară. Se fac apeluri pentru gestionarea datelor către Firebase database.

5.1.7. Diagramă de componente

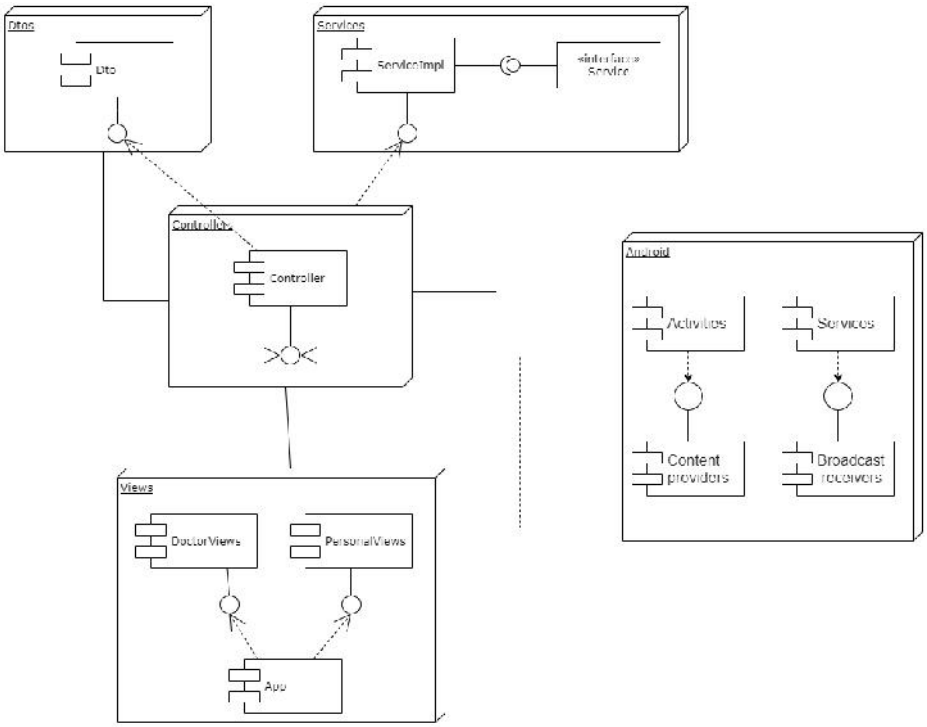

Figura 5.8 Diagrama de componente a aplicației

În figura 5.8 sunt prezentate componentele aplicației atăt pe partea de web, cât și pe cea de Android. Frontend-ul conține o componentă mare de *Views* care este împarțită în partea de doctor pentru getionarea urgențelor și una de personal. Partea de backend a aplicației web este împarțită în 3: dto. controller și service care implementează interfețele aferente. Aplicația Android conține componentele clasice a unui astfel de aplicații: activități și servicii care folosesc Content providers și Broadcast receiver.

5.2. Descrierea implementării modulelor

Login

Autentificarea în aplicație se face pe baza unui set de date format din username şi parolă. Acestea sunt stocate în Firebase Database. După apăsarea butonului de "Login" se va verifica dacă username-ul există şi dacă parola introdusă este identică cu cea deja existentă în baza de date. Dacă nu se respectă aceste condiții, utilizatorul va primi un mesaj de avertizare. In caz afirmativ, se va crea o sesiune pentru acest utilizator şi va fi unică prin asignarea unui token.

Sesiunea este creată pentru a ne asigura ca utilizatorul este autorizat si poate efectua modificări în platformă (modificări, adăugări, ştergeri etc). Aceasta se va încheia când utilizatorul se va deloga de la aplicație.

Termenul de sesiune este folosit pentru a descrie timpul cât stă logat un vizitator pe site. Reprezintă timpul de când a accesat platforma până când apasă butonul de "Logout".

Fiecare sesiune are un ID de sesiune (la noi în aplicație denumit token) unic generat aleator. De fiecare dată când un utilizator face o acțiune, acest ID va fi setat ca header în requestul care se face către server. Dacă acest ID nu există, acțiunea va fi respinsă. Mai jos, în figura este descrisă funcționarea unei sesiuni cu ID.

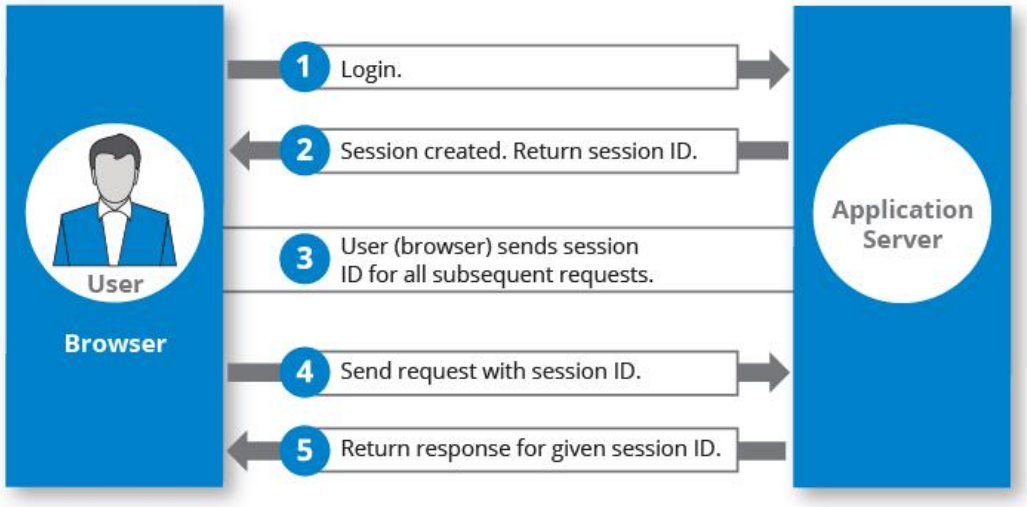

Figura 5.9 Descrierea unei sesiuni web[32]

[32] https://hazelcast.com/glossary/web-session/

Resetare parolă

Dacă un utilizator și-a uitat parola de la cont, atunci acesta o poate reseta cu ajutorul adresei de email pe care a introdus-o în momentul în care și-a creat contul. Dacă adresa nu este corectă, se va afișa un mesaj de atenționare și nu se va putea realiza resetarea. Dacă adresa se află stocată în baza de date, atunci utilizatorul va primi un mail care conține un link către o pagină unde își poate introduce parola nouă dorită. Pe lângă acel link, va primi si un cod de securitate, generat aleator.

Pentru a finaliza resetarea parolei, utilizatorul trebuie să inroducă și acel cod primit în mail. Astfel, prin această metodă, nu poate un alt utilizator să schimbe parola altei persoane, pentru că nu poate accesa mail-ul și să afle codul de securitate. Pentru trimiterea de email s-a utilizat *JavaMailSender* din pachetul pus la dispoziție de către Spring specific trimiterii de mail.

Programări și vacanță

Atât partea de programări, cât și cea de selectare a perioadei de indisponibilitate au avut la bază aceleași principii de implementare. Pe partea de frontend utilizatorul completează datele necesare și se realizează un apel către backend pentru a vizualiza, modifica, șterge sau adăuga informații. (operații CRUD).

Se pot adăuga programări doar la orele disponibile, doarece se verifică în baza de date dacă mai sunt programări în ziua selectată și la ce ore. Data și ora sunt stocate în Firebase sub formă de Timestamp.

Perioada de indisponibilitate se adăugă selectând o dată de început și una de sfârșit verificând că cea de sfârșit să fie mai mare decât cea de început, altfel nu se va putea realiza adăugarea de vacanță. Pacienții care pot fi selectați au aceeași specialitate ca doctorul care relizează selectarea perioadei de vacanță și au posibilitatea să bifeze dacă este disponibili în cazul unei urgențe.

Toate aceste informații sunt adăugate în Firebase database pentru a fi disponibile pentru pacinții conectați la aplicația Android care vor fi atenționați dacă în perioada selectată de ei doctorul ales, în funcție de specialitate și preferințe nu este disponibil și au posibilitatea de a participa la consultație cu doctorul înlocuitor sau de a alege o altă dată în care doctorul inițial selectat este disponibil.

Firebase

Când s-au creat proiectele de web și Android a fost necesară configurarea lor la baza de date NoSQL Firebase database. S-au folosit SDK-urile disponibile pentru acest fapt, atât pe partea de web, cât și pentru Android.

În figura 5.5 este prezentată o secvență de cod de la inițializarea firebase cu serverul web. Pentru a beneficia de serviciile Firebase a fost nevoie de generarea unui fișier de tip cheie privată în format JSON. Mai apoi, s-a setat url-ul bazei de date create.

Documentația Firebase[33] a fost de mare ajutor pentru gestionarea datelor, inițializarea proiectelor și metodele de citite, scriere, modificare și ștergere a datelor în Firebase Database, pentru partea de web, dar și Android.

```
@PostConstruct
public void initialize(){
    try {
        InputStream serviceAccount = getClass().getResourceAsStream( name: "/cloudfirebase.json");

        FirebaseOptions options = new FirebaseOptions.Builder()
                .setCredentials(GoogleCredentials.fromStream(serviceAccount))
                .setDatabaseUrl("https://mhealth-angeco.firebaseio.com")
                .build();

        FirebaseApp.initializeApp(options);

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figura 5.10 Inițializare Firebase pe partea de web

Distribuirea de fișiere a fost realizată atât pe aplicația web, cât și pe aplicația mobilă. Pentru amândouă a fost necesară stocarea fișierelor în Firebase Cloud Storage[34]. Se folosesc sdk-uri Firebase pentru Cloud Storage pentru a adăuga și descărca fișiere direct de la clienți, atât pe web, cât și pe mobil cu sdk-uri specifice. Cloud Storage se scalează automat și nu este nevoie de o altă migrare către un alt provider.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_FILE_GET && resultCode == RESULT_OK) {
        Bitmap thumbnail = data.getParcelableExtra( name: "data");
        Uri fullFileUri   data.getData();
        File file= new File(fullFileUri.getPath());
        fileName = file.getName();

        storageReference.child("files/" + file.getName()).putFile(fullFileUri)
                .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                        taskSnapshot.getMetadata().getReference().getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                            @Override
                            public void onSuccess(Uri uri) { addFile(uri, fileName); }
                        });
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText( context: ShareFilesActivity.this, text: "Could not upload file!", Toast.LENGTH_SHORT).show();
                    }
                });
```

Figura 5.11 Adăugare fișier în Firebase Cloud Storage din Android

[33] https://firebase.google.com/docs/
[34] https://firebase.google.com/docs/storage

Pentru realizarea distribuirii de fișiere este nevoie mai întâi de stocarea fișierului în Firebase Storage cum este ilistrat și în figura 5.6 și mai apoi este posibilă extragerea url-ului și stocarea lui în Firebase Database în colecția de SharedFilesFromAndroid pentru a putea fi afișate fișierele distribuite cu posibilitatea de descarcare a lor. Se pot distribui fișiere cu orce tip de format.

Chat

Comunicarea dintre pacient și doctor s-a realizat prin adăugarea unui sdk pus la dispoziție de platforma PubNub pentru customizarea unui chat propriu ponind de la niște tutoriale disponibile în documentația oficială. Implementarea s-a efectuat atât în aplicația web, cât și în cea Android. A fost necesară crearea unui cont gratuit pentru și a unui aplicații pentru a fi înregistrată în sistem. Această aplicație generează un set de chei specifice, prin care te poți conecta la aplicația creată și care doar cu el ai permisiunea de a vizualiza și publica mesaje.

Avantajul folosirii unei astfel de tehnologii este acela că datele sunt persistente și stocarea lor se realizează în platforma PubNub. Din dashboard-ul pus la dispoziție se pot seta cât timp datele rămăn stocate în sistem de la 1 zi până la o perioadă nelimitată. Varianta gratuita a contului suportă până la un milion de tranzacții și un spațiu de stocare de 1 GB.

Chat-ul trebuie inițializat cu cheile de publish și subscribe ale aplicației setate în dashboard. A fost nevoie de setarea utilizatorilor la configurarea pubnub cu utilizatorii aplicației noastre MHealth, adică utilizatorii web, reprezentați de doctori, dar și utilizatorii Android.

Pentru comunicarea între doi utilizatori este nevoie de un canal de comunicare care se generează din cod, nu trebuie prestabilit și nu există o limită pentru generarea lor. Ca transmiterea să funcționeze trebuie ca cei doi utilizatori să fie conectați la același canal, mai exact PubNub să dea *subscribe* la canalul dorit și un *event listener* care urmărește orice modificare și publicare de mesaje pe canalul specificate.

Pentru a genera un canal unic pentru fiecare pereche de utilizatori, formată dintr-un doctor și un pacient, s-a denumit în funcție de id-urile lor din firebase database sub forma:

src > pubnub > config > JS chat.js > [@] <unknown> > forestChatChannel

```
1   var actualUser = JSON.parse(localStorage.getItem('actualUser'));
2   var selectedUser = JSON.parse(localStorage.getItem('selectedChatUser'));
3   var id_webUser = "";
4   var id_androidUser = "";
5   if (actualUser !== null && selectedUser !== null) {
6     id_webUser = actualUser.id_webUser;
7     id_androidUser = selectedUser.id_androidUser;
8   }
9   module.exports = {
10    forestChatChannel: 'demo-' + id_webUser + id_androidUser,
11  };
12
```

Figura 5.12 Denumire canal de chat

Pe partea de web chat-ul a fost introdus în componenta de frontend, PubNub punând la dispoziție sdk pentru mai multe tehnologii printre care și React, dar și Android. PubNub este o tehnologie complexă, foarte scalabilă și chiar dacă sistemul MHealth nu o poate folosi la capacitate maximă, s-a dorit testarea și încercarea ei pe parte de chat.

Traducere

Pe partea de Android s-a realizat traducerea mesajelor dintr-un set larg de limbi în engleză înainte de a se trimite în chat și de a se publica pe canal.

S-au folosit două api-uri puse la dispoziție de sdk-ul ML Kit specifice pentru Android, de indentificare a limbii dintr-un text și cel de traducere.

ML Kit pentru identificarea limbii[35] extrage limba care are scorul de potrivire cel mai mare dintr-o propoziție dintr-un text. Acesta recunoaște mai mult de 100 de limbi diferite, printre care și limba română. A fost nevoie de adăugarea unor dependințe înainte de folosirea api-urilor.

Traducerea de text pusă la dispoziție de ML Kit[36] traduce textul între mai multe limbi, peste 50.

```
public void onSentClick(String message) {
    // Log::ignore_]
    FirebaseLanguageIdentification languageIdentifier = FirebaseNaturalLanguage.getInstance().getLanguageIdentification();
    languageIdentifier.identifyLanguage(message).addOnSuccessListener(new OnSuccessListener<String>() {
        @Override
        public void onSuccess(@Nullable String languageCode) {
            if (languageCode != "und" && languageCode != null) {
                FirebaseTranslatorOptions options = new FirebaseTranslatorOptions.Builder()
                        .setSourceLanguage(FirebaseTranslateLanguage.languageForLanguageCode(languageCode))
                        .setTargetLanguage(FirebaseTranslateLanguage.EN)
                        .build();

                FirebaseModelDownloadConditions conditions = new FirebaseModelDownloadConditions.Builder().build();

                final FirebaseTranslator translator = FirebaseNaturalLanguage.getInstance().getTranslator(options);
                translator.downloadModelIfNeeded(conditions).addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void v) {
                        translator.translate(message).addOnSuccessListener(new OnSuccessListener<String>() {
                            @Override
                            public void onSuccess(@NonNull String translatedText) {
                                String auxMessage = translatedText;
                                if (TextUtils.isEmpty(translatedText)) {
                                    if (BuildConfig.DEBUG) {
                                        StringBuilder messageBuilder = new StringBuilder("");
                                        messageBuilder.append(UUID.randomUUID().toString().substring(0, 8).toUpperCase(Locale.US));
                                        messageBuilder.append("\n");
                                        messageBuilder.append(Helper.parseDateTime(System.currentTimeMillis()));
                                        auxMessage = messageBuilder.toString();
```

Figura 5.13 Implementare recunoaștere limbă și traducere

[35] https://developers.google.com/ml-kit/language/identification/android?fbclid=IwAR2oU1WTBiEc1DScqyUYPETWSZdz1WntIlIb15UGvM8FDeqtJdt99AOkHn8

[36] https://developers.google.com/ml-kit/language/translation/android?fbclid=IwAR0w00naJj_eULP_-a0mgnoV2dvHpOjCmJjbKUpYDHLQYto_NAB0wjVuJQQ

În figura 5.8 este prezentat cum au fost apelate api-urile pentru identificarea limbii și traducerea textului unui mesaj înainte de a fi transmis pe canalul PubNub. Se crează un obiect de tip *FirebaseTranslator* și se steză ca traducerea să se facă în engleză din limba identificată. La identificarea limbii se va descărca un dicționar specific traducerii, de exemplu dacă limba detectată este română, se va descărca dicționar român-englez (ro-en).

Managementul sistemului

Pentru managemntul sistemului s-a utilizat platforma GitHub pentru o gestionare mai ușoară a modificărilor aduse codului. Sistemul MHealth fiind creat din două componente, una de web și una de Android, dar cea de web fiind la rândul ei compusă din frontend realizat în React și backend implementat în Spring, s-au realizat trei repository-uri private la care am avut acces toți cei trei membri dezvoltatori ai sistemului.

S-au stabilit de la început convenții de nume pentru brach-uri specifice fiecărui repository, fiind sugetive cu funcționalitățile implementate. Pe branch-ul principal, cel de master conține doar cod testat, funcțional. Modificările ajung pe acest branch în urma efectuării unui pull request. Pentru a gestiona cu ușurință schimbările aduse codului s-a folosit aplicația GitHub Desktop.

De exemplu, în figura 5.14 sunt ilustrate branch-urile create în aplicația de frontend, respectând convenția de nume.

S-a realizat o fișă de evoluție pentru urmărirea progresului comună toți membrilor. Această fișă a fost actualizată pe parcurs ce se implementa sau documenta o funcționalitate. Fișa poate fi găsită la Anexa 1.

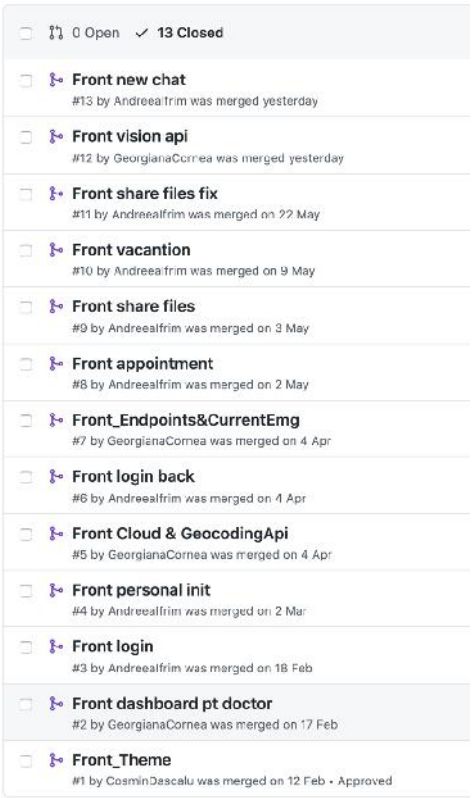

Figura 5.14 Branch-urile aplicației de frontend

Capitolul 6. Testare şi Validare

Testele vin în ajutorul dezvoltatorului de proiect pentru a înțelege ce trebuie să facă pentru a implementa un sistem cât mai corect și de calitate.

Clasificarea testelor se face după două criterii: în funcție de nivel și de tip.

- **Nivel**: se implementează teste rapide, pe porțiuni mici ale sistemului și arată cât de aproape este testul de codul sursă
- **Tip**: se realizează verificarea funcționalităților și se specifică obiectivele testelor

6.1. Cazuri de testare

Provocări la testare au fost aduse în general de formatul de dată pentru că tot timpul trebuia verificat pentru a se potrivi cu cel deja existent în baza de date Firebase. S-au folosit teste pentru a vedea dacă anumite răspunsuri la request-uri erau cele aşteptate. Nu s-au putut testa datele trimise prin intermediul chat-ului şi răspunsul anumitor request-uri pentru că acesta a fost dezvoltat prin intermediul tehnologiei PubNub care are propriul spațiu de stocare al mesajelor în mod persistent și apelul API-ului s-a facut din partea de front-end a aplicației.

6.1.1. Unit testing

Pentru realizarea unit testelor am folosit JUnit 5 care este format din trei componente:

- JUnit Platform - reprezintă baza framework-ului de lansare a testării
- JUnit Jupiter - este o combinație dintre modelul de programare și cel de extensie pentru a scrie și executa teste în JUnit 5
- JUnit Vintage - conține *TestEngine* pentru a rula testele de bază JUnit 3 și JUnit 4[37]

Primul caz de testare se referă la adăugarea de programare efectuată de către doctor din aplicația web în Firebase database. Metoda testată este cea implementată în clasa de *AppointmentService*, cea de *createAppointment*.

Codul testului și rezultatul lui sunt prezentate în figura 6.1. Am ales cazul de utilizare în care programarea pe care am vrut să o introduc există deja înregistrată în sistem și nu se va realiza adăugarea de programare, se va returna următorul mesaj: "Another appointment at that time".

Metoda testată avea ca parametru un obiect de tip *AppointmentDto*, așa că setul de date conține:

- String - id-ul utilizatorului web
- String - numele utilizatorului Android
- Date - data la care vrea să se facă programarea

[37] https://junit.org/junit5/docs/current/user-guide/

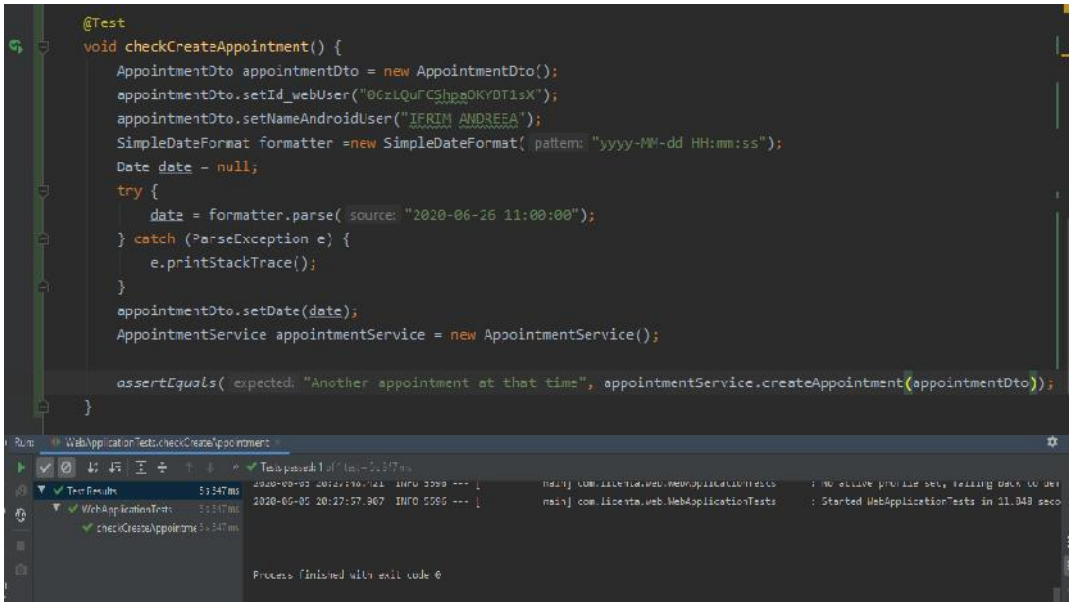

Figura 6.1 Primul unit test

Un alt caz de testare a fost făcut tot pe o metodă din cadrul *AppointmentService*, mai exact *modifyAppointment*.

Rezultatul este unul pozitiv și codul testului este prezentat mai jos. Ca date de intrare s-au folosit: id-ul programării, data și ora programării. S-a știut că rezultatul este cel bun, pentru că am verificat că modificările în firestore s-au realizat și mesajul primit a fost cel așteptat.

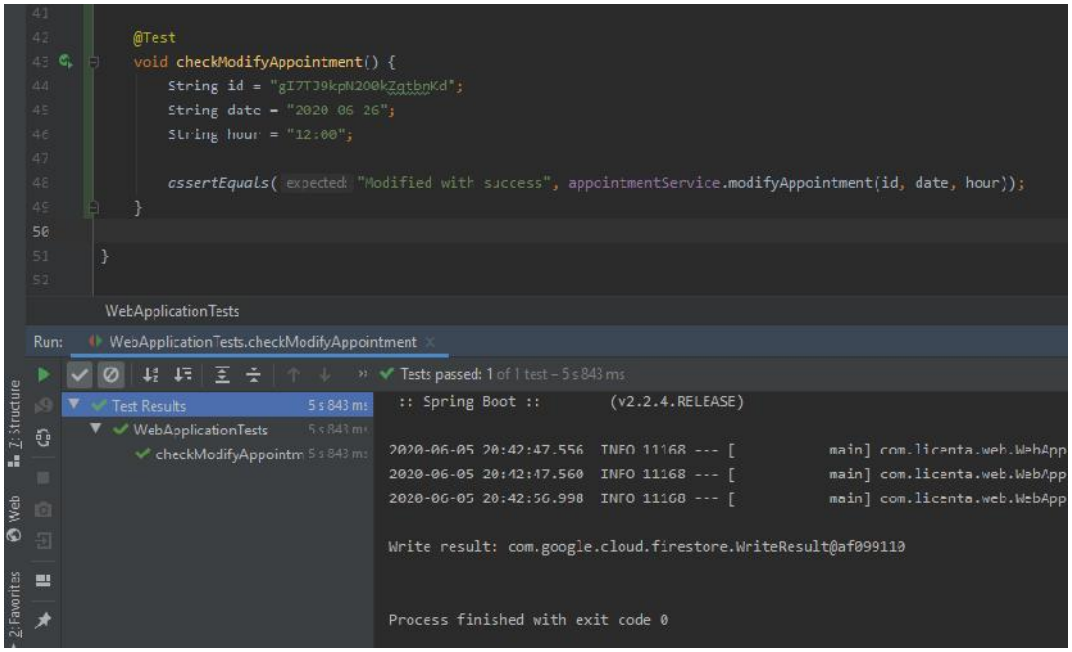

Figura 6.2 Al doilea unit test

În cazurile de test efectuate s-a folosit tipul de test data-driven pentru că s-a verificat dacă s-a creat o legătură între setul de date de intrare și valorile de ieșire așteptate. Nu a fost o testare de tip combinatorial pentru că nu s-a putut realiza executarea unui număr mare de teste.

6.1.2. Integration testing

Testarea de integrare se realizează pe mai multe unități, pentru a vedea dacă apar erori la interacțiunea dintre ele. Pentru că aplicația are integrate servicii REST s-a realizat testarea la nivel de controller. Din controller se apelează metode din service în care se fac apeluri către elemente din firestore.

Primul test a fost realizat pentru metoda *createAppointment* din *AppointmentController* . Metoda din service care este apelată a fost deja testată cu ajutorul unit testelor.

```
    @Test
    public void testAddAppointment() throws URISyntaxException {
        String token = HexUtils.toHexString(UUID.randomUUID().toString().getBytes());
        initSession( id: "sdbnXKVFU6j2rnLE9u2n", token);
        AppointmentDto appointmentDto = new AppointmentDto();
        appointmentDto.setId_webUser("06zLQuFCShpaOKYBT1sX");
        appointmentDto.setNameAndroidUser("IFRIM ANDREEA");
        SimpleDateFormat formatter =new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss");
        Date date = null;
        try {
            date = formatter.parse( source: "2020-06-26 11:00:00");
        } catch (ParseException e) {
            e.printStackTrace();
        }
        appointmentDto.setDate(date);
        String baseUrl = "http://localhost:"+ port +"/create-appointment";
        URI uri = new URI(baseUrl);
        HttpHeaders headers = new HttpHeaders();
        headers.set("token", token);
        HttpEntity<AppointmentDto> request = new HttpEntity<>(appointmentDto, headers);
        ResponseEntity<String> result = this.restTemplate.postForEntity(uri, request, String.class);
        assertEquals( expected: 200, result.getStatusCodeValue());
    }
}
```

Figura 6.3 Primul caz de integration testing

Testarea s-a finalizat cu succes pentru că codul statusului este cel așteptat. La request a fost adăugat și header cu token pentru că se face verificare de sesiune pentru securitate.

Un alt caz de testare a fost realizat pe o metodă tot din *AppointmentController* care returnează toate programările de la data specificată în request. Setul de date conține data, dar și id-ul utilizatorului web, pentru ca lista returnată să conțină programările care aparțin utilizatorului logat.

Testul a fost realizat cu succes tot prin verificarea valorii codului care este 200, reprezentând faptul ca cererea a fost îndeplinită cu succes.

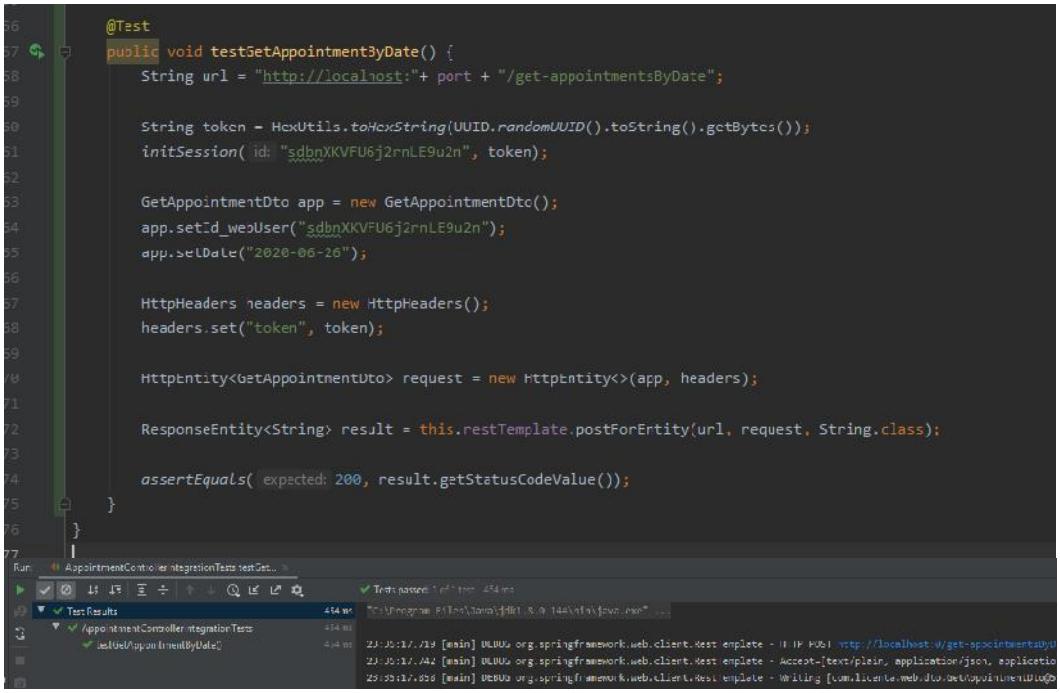

Figura 6.4 Al doilea caz de integration testing

S-au utilizat test drivers folosit la bottom-up integration testing pentru că s-a simulat comportamentul nivelelor superioare, mai exact cel implementat la nivelul controller-ului.

Toate tehnicile de testare prezentate în această secțiune sunt de tip ***funcțional*** pentru că se face verificarea dacă sistemul răspunde conform așteptărilor. În schimb cele de tip ***non-funcțional*** sunt folosite pentru a testa alte atribute ale aplicației cum sunt performanța și securitatea. Aceste teste nu au putut fi executate asupra sistemului pentru că ține de felul în care a fost dezvoltată aplicația.

6.1.3. System testing

Testarea sistemului se referă la integrarea mai multor componente pentru a forma un întreg. Are ca scop testarea întregului sistem mai ales în contextul trecerii fluxului de la o componentă la alta.

Această testare este de tip black box și a fost realizată manual. La nivel de echipă am ales un caz de utilizare potrivit pentru a acoperi toate modulele implementate de fiecare pentru a testa întregul sistem și interacțiunea dintre componente.

Acest caz de utilizare este descris mai jos:

- Un utilizator Android raportează o urgență prin trimiterea de fotografie și descriere de tip text împreună cu locația de la locul incidentului
- Datele trimise sunt procesate în cadrul aplicației web și un doctor le validează
- Un alt utilizator Android preia urgența după primirea unui notificări cu locația incidentului și primește informațiile procesate de aplicația web pe email.
- Utilizatorul care raportează urgența inițiază o conversație cu un doctor prin intermediul chat-ului pentru a primi sfaturi legate de incident

Cazul de utilizare a fost finalizat cu succes astfel fiind testat fluxul de la o componentă la alta, modulele funcționând împreună ca un întreg.

Capitolul 7. Manual de Instalare si Utilizare

În acest capitol sunt prezentate resursele necesare pentru instalarea aplicațiilor web și Android și modul în care utilizatorul poate interacționa cu aplicațiile din punctul de vederea al doctorului și al pacientului.

7.1. Resurse necesare pentru instalare

Pentru instalarea sistemului MHealth trebuie luat în considerare instalarea unei aplicații pe telefonul mobil destinată pacienților și accesarea unui link pentru a interacționa cu aplicația web care are funcționalitățile specifice doctorilor, fiind găzduită pe Cloud.

Înainte de instalare trebuie verificate anumite resurse și sepcificații. Pentru funcționarea în prealabil a **aplicației mobile** destinate pacienților se recomandă:

- sistem de operare Android cu o versiune de 5.1 sau peste
- spațiu liber de memorie de minim 50 MB
- conexiune stabilă la internet (Wi-Fi sau date mobile)

Resursele necesare pentru funcționarea corespunzătoare a **aplicației web** folosită de doctor sunt:

- conexiune stabilă la internet
- existența unui browser

Dacă toate aceste condiții sunt îndeplinite, atunci este posibilă instalarea aplicației mobile pe un telefon Android și dechiderea aplicției web accesând link-ul specific sistemului MHealth. Astfel, se pote începe interacțiunea utilizatorilor cu sistemul cum este descris în manualul de utilizare prezentat în subcapitolul următor.

7.2. Manual de utilizare

În acest subcapitol se vor descrie pașii de utilizare a sistemului MHealth atât pentru aplicația web, cât și pentru cea Android. Se vor ilustra paginile principale și modul în care utilizatorul poate interacționa cu sistemul urmând anumiți pași.

7.2.1. Manual de utilizare doctor

Doctorul are acces la aplicația web introducând în borwser link-ul sistemului. Aplicația se deschide pe pagina de pornire, de **Login**, unde:

- se autentifică folosind numele de utilizator și parola
- se înregistrează cu un cont nou în cazul în care nu are
- își resetează parola în cazul în care nu o mai știe

Pagina principală constă dintr-un meniu în stânga unde se pot accesa paginile destinate modulului personal, dar și cel al gestionării urgențelor. Modulul personal conține o serie de butoane în partea de sus a paginii care prin acționarea lor deschid alte pagini destinate opțiunilor puse la dispoziție de către sistem.

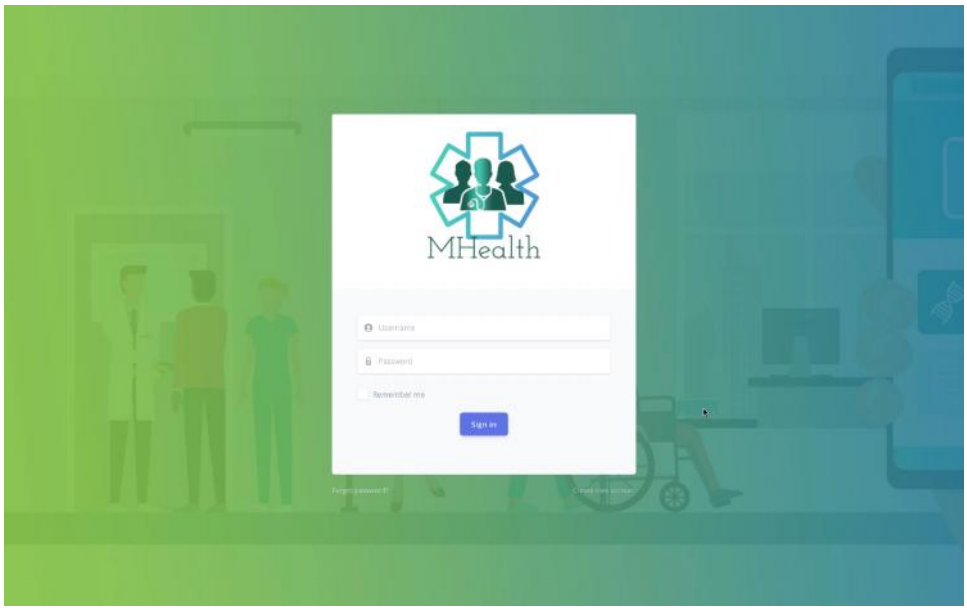

Figura 7.1 Login pe partea de web

Pentru **crearea de nou** cont este necesară completarea tuturor câmpurilor de nume, nume de utilizator, parola, specializarea în care profesează și email. Dacă există un cont cu email-ul introdus, se va afișa un mesaj de avertizare.

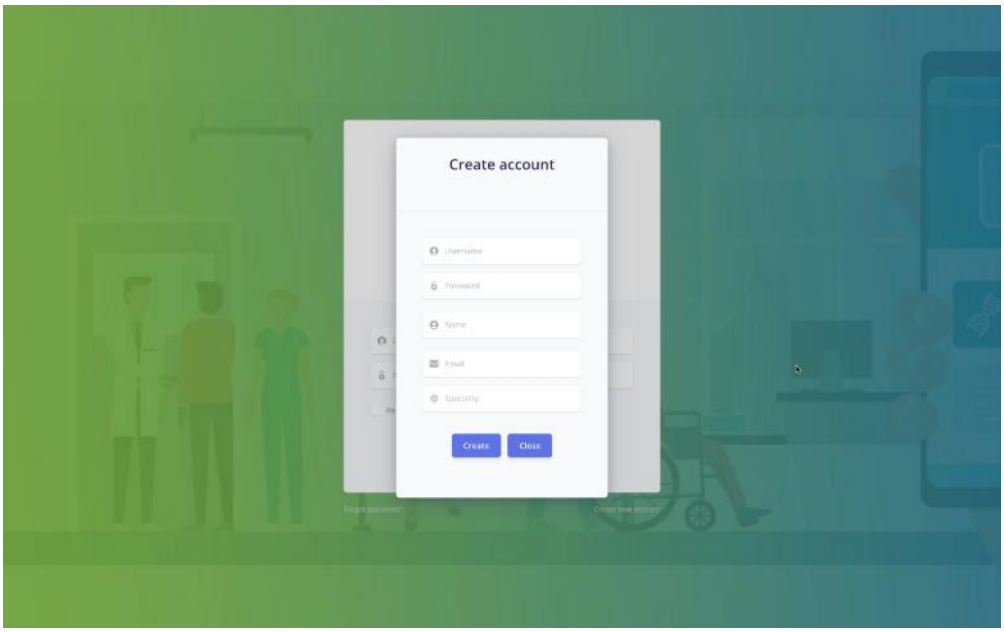

Figura 7.2 Creare cont în aplicația web

Resetarea parolei se efectuază itroducând adresa de email a contului la care nu se mai știe parola și se va primi un email cu un link care te redirecționează la o pagină unde poate fi introdusă parola nou dorită și un cod de securitate care trebuie introdus pentru a realiza resetarea.

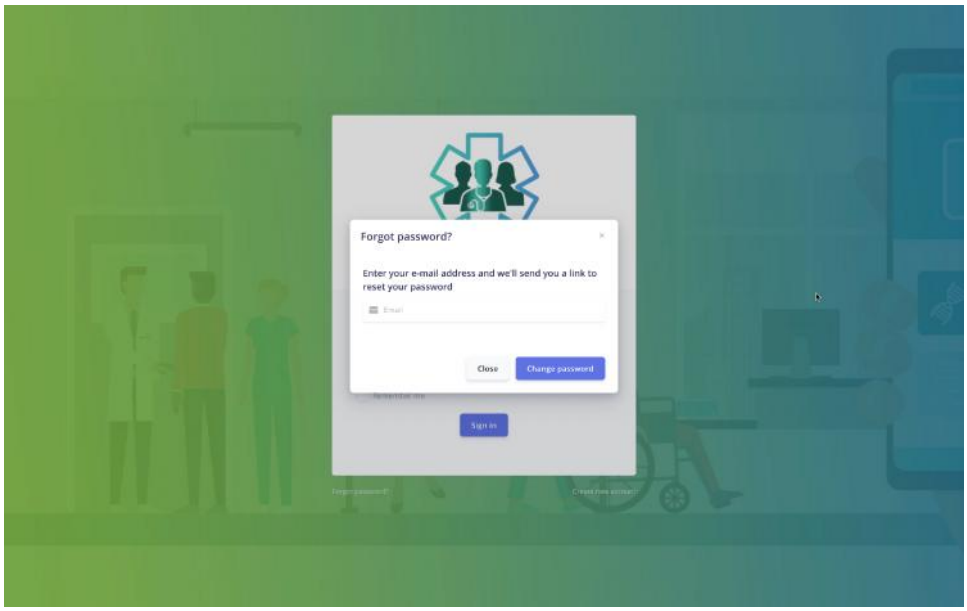

Figura 7.3 Resetare parolă cu introducere de email

Pgina de **User Profile** este destinată vizualizării informațiilor legate de doctorul logat, cum ar fi: nume, email, username și specislitatea de care aparține. Profilul poate fi vizualizat și prin apăsarea butonului din meniul din stânga,din secțiunea de "My Profile" de sub iconița utilizatorul dar și apăsând pe mesajul de bun venit de pe site.

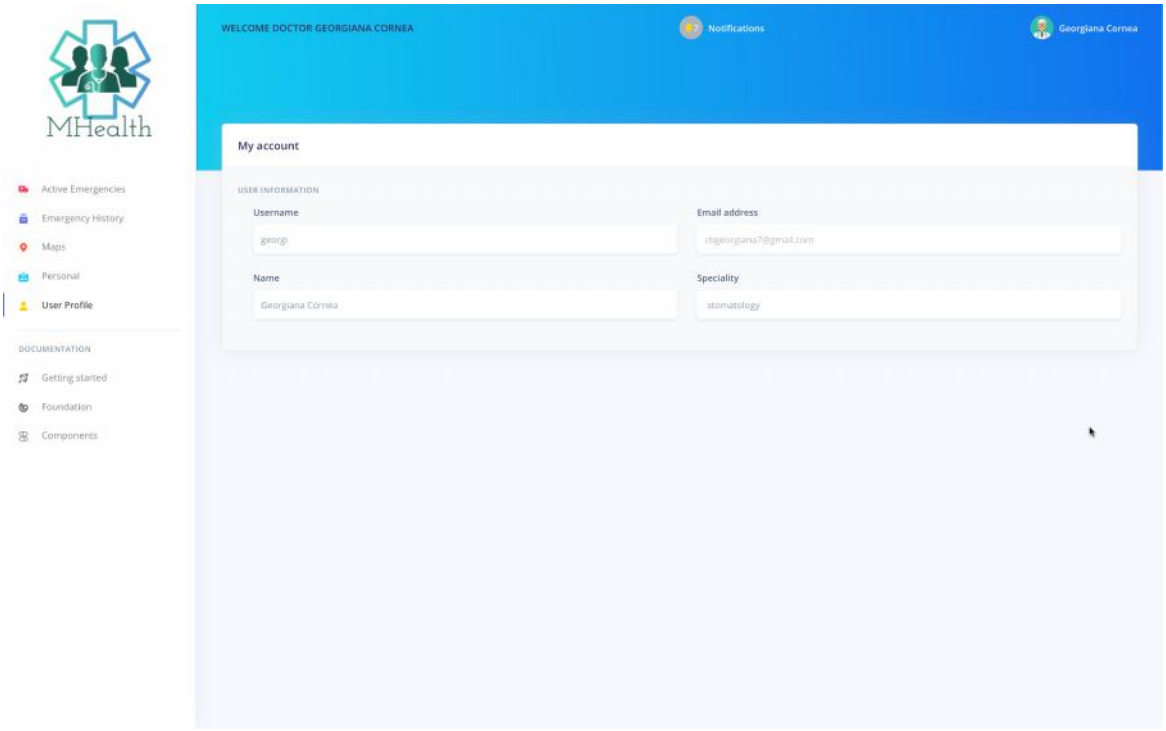

Figura 7.4 Vizualizare date profil

Chat with a patient este pagina în care doctorul poate să selecteze un pacient și începe o convorbire cu acesta prin trimitre de mesaje sau emoticoane prin apăsarea butonului de *Send.*

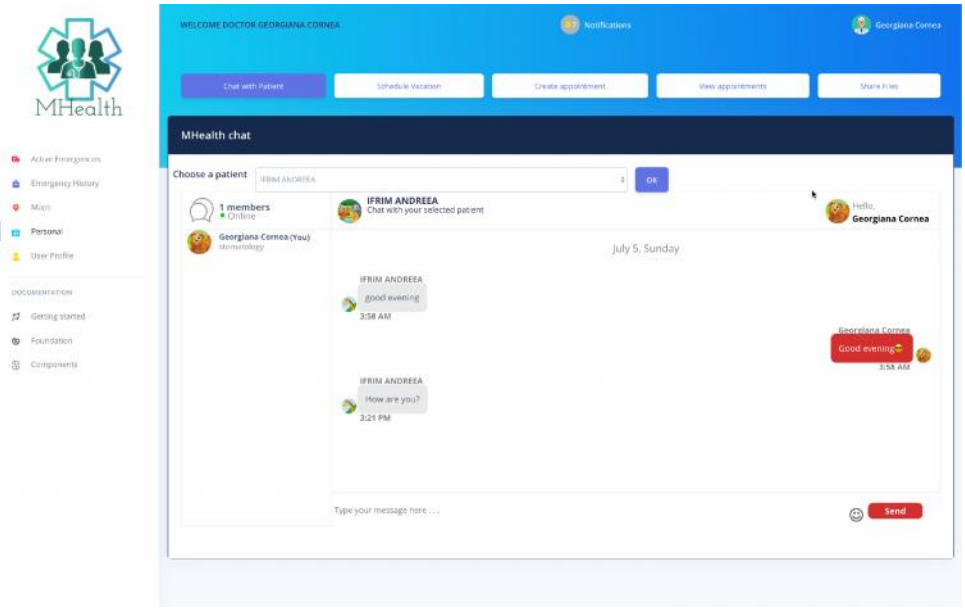

Figura 7.5 Chat cu pacientul selectat

Create appointment este opțiunea de a adăuga o nouă programare selectând data dorită și după apăsarea butonului de "OK" se vor afișa orele disponibile din acea zi. Se va selecta ora potrivită și numele pacientul. Trebuie selectate toate cămpurile și o oră existentă pentru a realiza programarea, dacă nu se vor primi mesaje de atenționare.

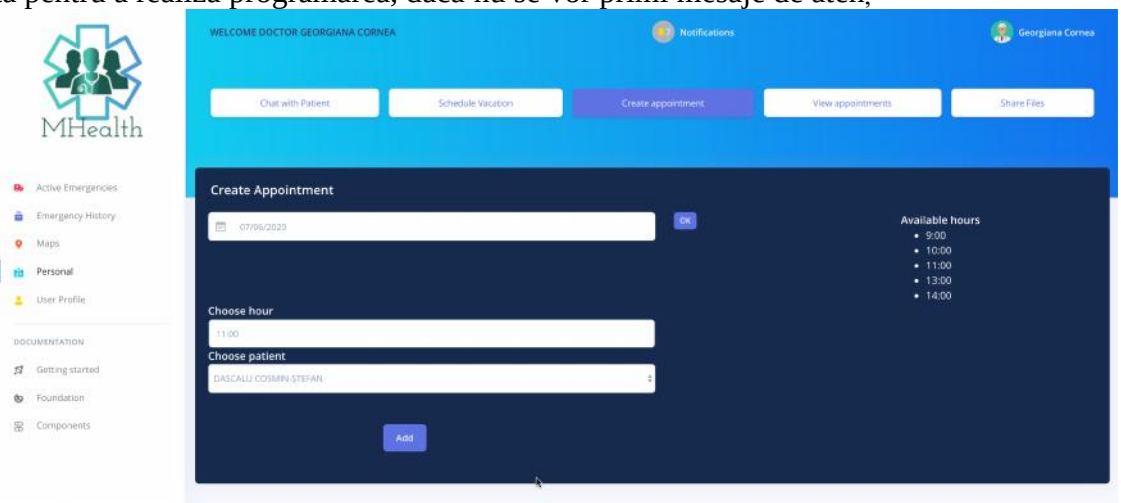

Figura 7.6 Adăugare programare în aplicația web

Prin apăsarea butonului de **Show appointments** se va deschide o pagină cu un tabel care arată toate programările din ziua selectată. Există opțiunile de a *șterge* o programare sau de a *modifica* una acționând ultimele două butoane din tabel. Pentru modificare se va deschide o secțiune pentru completarea datelor.

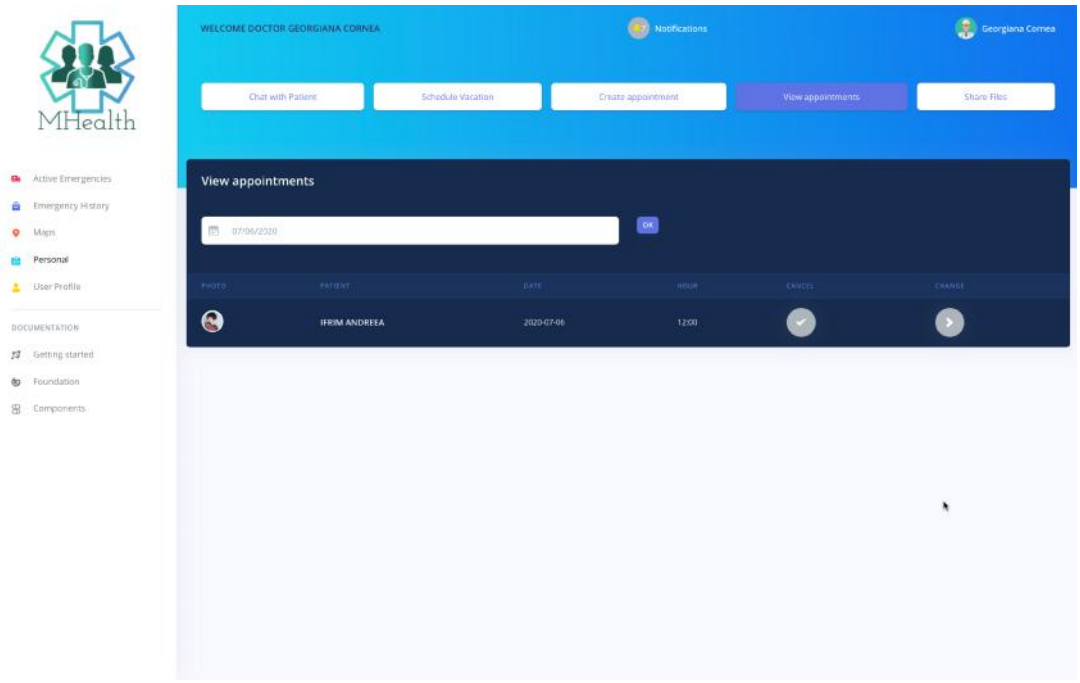

Figura 7.7 Vizualizare programări în aplicația web

Prin selectarea **Vacantion** se va deschide o secțiune în care doctorul poate selecta o perioadă de indisponibilitate prin alegerea unei date de început și de sfârșit, poate bifa dacă este disponibil în caz de urgență și trebuie să aleagă un doctor înlocuitor pentru a prelua programările din acea perioadă. Va fi necesara completarea și selectarea tuturor câmpurilor pentru a fi întregistrată perioada de vacanță.

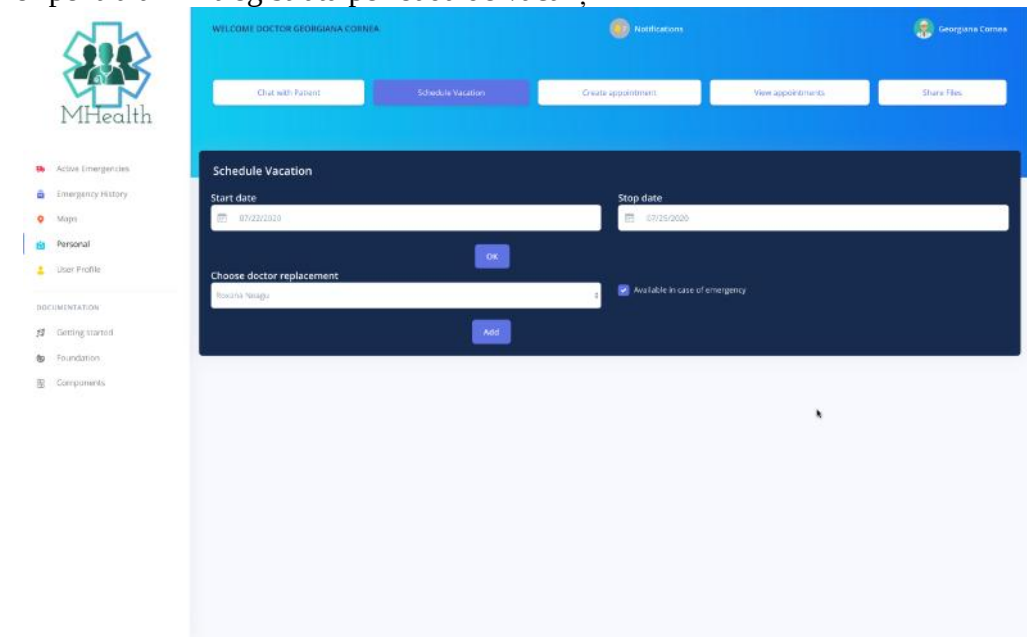

Figura 7.8 Selectare perioadă de vacanță

Share files este opțiunea care permite utilizatorului să trimită un fișiere către pacienți selectând din calculator fișierul dorit pentru a fi transmis, numele pacientului și are posibilitatea de a adăuga o descriere atașată pentru a furniza date suplimentare pacientului.

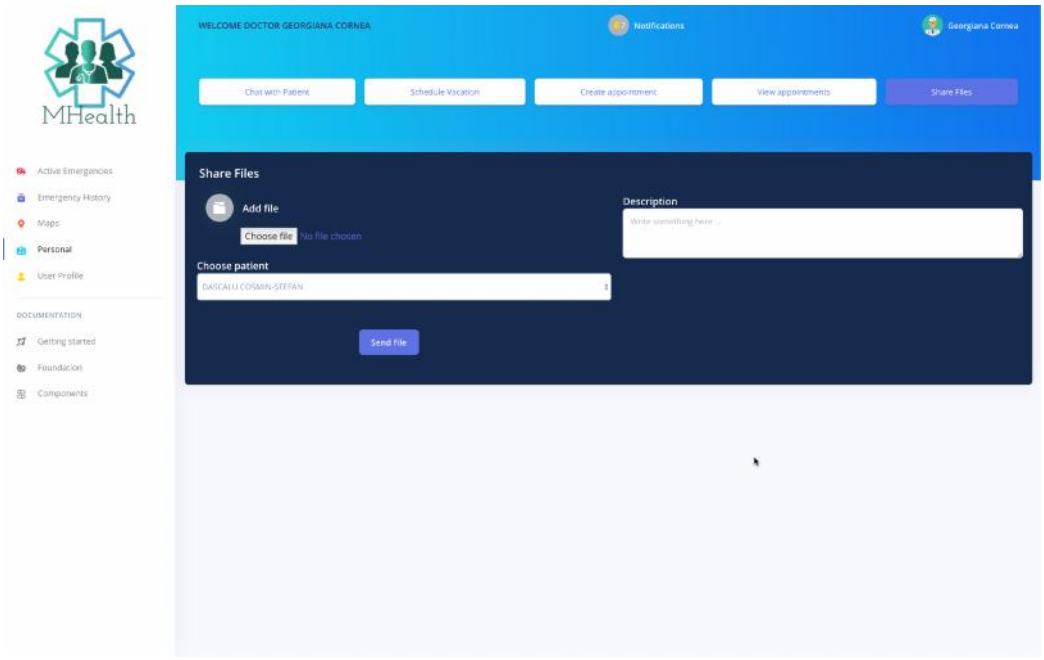

Figura 7.9 Distribuire de fișier din aplicația web

7.2.2. Manual de utilizare pacient

Un utilizator obișnuit (pacient) poate accesa aplicația mobilă deja instalată pe telefon. După autentificare este diponibil un meniu cu modulele aplicației, cel de gestionare a urgențelor și cel personal. În aplicație majoritatea opțiunilor puse la dispoziție sunt prezentate prin imagini sugestive.

După accesare butonului destinat modulului personal este afișat un meniu prezentat în figura 7.1 de unde se pot selecta opțiunile de:

- Gestionare a programărilor
- Chat cu un doctor
- Dostribuire de fișiere

Figura 7.10 Meniul modulului personal

În secțiunea de programări se pot **vizualiza programările** pe care le are de onorat pacientul logat, având diferite informații despre numele doctorului data, ora programării și posibilitate de a anula o programare prin apăsarea butonului de "DELETE".

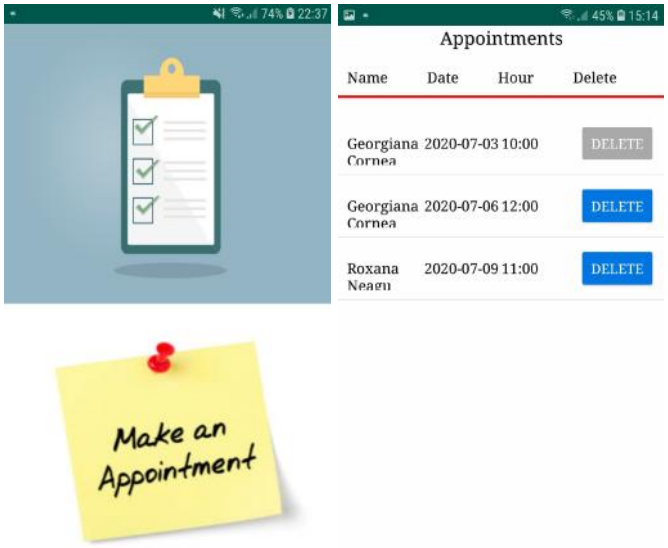

Figura 7.11 Vizualizarea programărilor din aplicația Android

Pentru a **adăuga o programare** se slectează specialitatea medicală potrivită pentru problema care se vrea rezolvată prin programare și se vor afișa doctorii cu acea specializare înregistrați în sistem. După selectarea doctorului se va redirecționa utilizatorul către o pagină în care se intoduce data și ora dorită în funcție de disponibilitate. Dacă doctorul selectat este indisponibil în acea zi, pacientul are optiunea de a efectua programarea în ziua și ora specificată dar cu un alt doctor înlocuitor sau poate schimba ziua sau doctorul.

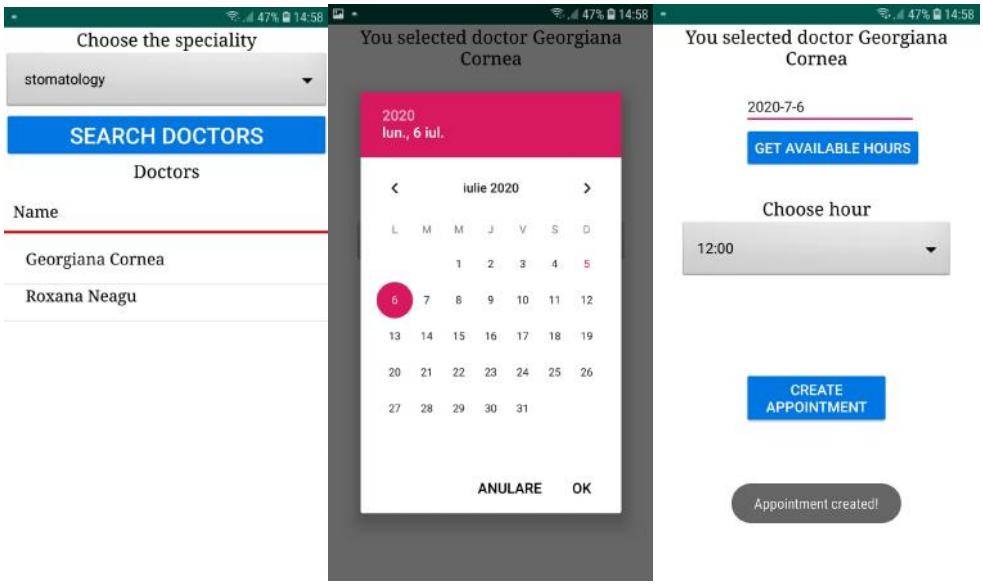

Figura 7.12 Adăugare programare în aplicația Android

Utilizatorul poate vizualiza **fişierele distribuite** prin apăsarea butonului de "Shared files" cu informații adiționale pe lângă fișierul propriu zis despre numele doctorului, descrierea adăugată de doctor şi numele fişierului.

Pentru a transmite un fișier se va selecta doctorul în funcție de specialitate și se va alege un fişiere deja existent în telefon care poate să fie o imagine, pdf, word etc.

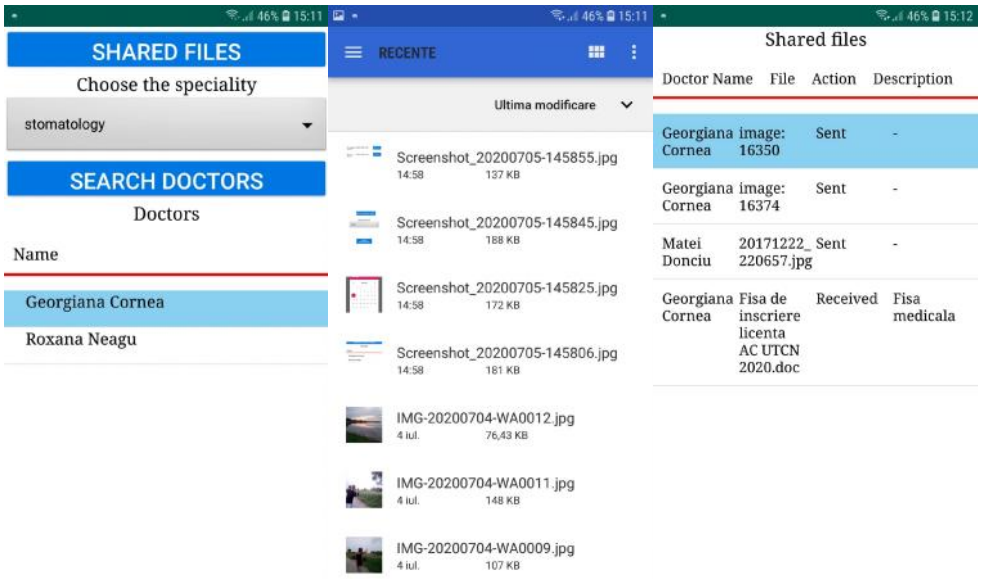

Figura 7.13 Distribuire şi vizualizare fişiere în aplicația Android

Chat-ul este destinat pentru comunicarea între pacient şi un doctor selectat. Se pot transmite mesaje și adăuga emoticoane prin scrierea lor în spațiul marcat și trimiterea lor prin apăsarea butonului de "Send" din dreapta jos. Dacă pacientul introduce ceva în altă limbă se va face translate automat în engleză.

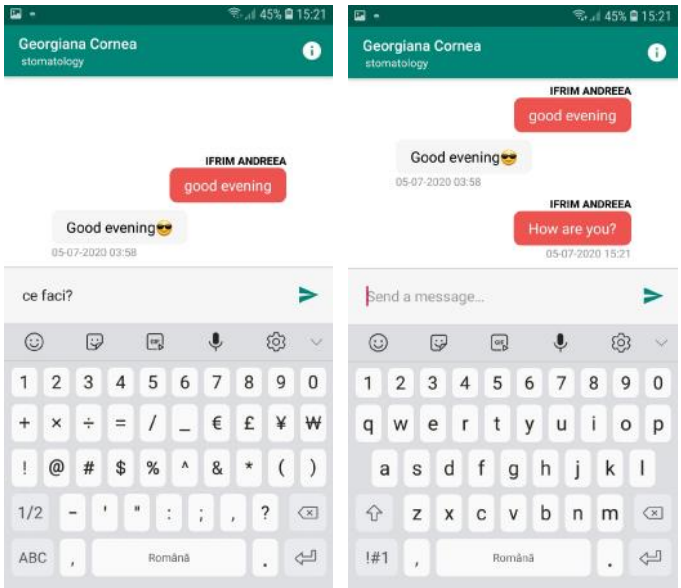

Figura 7.14 Chat între pacient şi doctor Android

Capitolul 8. Concluzii

În acest capitol se prezintă un rezumat al sistemului, ce rezultate s-au obținut prin implementarea funcționalităților propuse, cât și alte concepte și funcții care ar putea fi adăgate sistemului pentru a putea îmbunătăți interacțiunea utilizatorului cu aplicațiile.

8.1. Rezultate obținute

Sistemul MHealth a pornit de la ideea de a putea gestiona mai ușor urgențele cu posibilitatea de intervenție a unuor specialiști care se află în zona incidentului până la sosirea ambulanței, fiindu-le furnizate informații utile de la doctori și completând cu posibilitatea folosirii sistemului şi într-o manieră personală, fără a fi o urgență, interacționând cu medici.

S-au realizat toate funcționalitățile propuse spre implementare de la inițierea temei, atât pe aplicația web, căt și pe cea mobilă. S-a dorit realizarea pe partea de web a unei aplicații destinate doctorilor şi cea mobilă pentru utilizatori obișnuiți care își pot instala aplicația pe telefon.

Modulul personal este destinat pentru a ușura interacțiunea și comunicarea între pacienți și medici. Doctorii pot gestiona programările, pot crea altele noi și modifica sau şterge pe cele existente deja. Medicii au posibilitatea de a alege o perioadă de indisponibilitate, marcând dacă pot prelua urgnțe în acea perioadă și selecta un doctor înlocuitor care poate efectua programările. Atât utilizatorii web, cât și utilizatorii aplicației mobile pot distribui între ei fișiere necesare pentru consultație, fișe medicale sau poze reprezentative cu problema pe care o întâmpină.

Chat-ul vine în completarea acestor funcționalități pentru a face comunicarea cât mai ușoară între pacient şi doctor. S-a optat pentru utilizare unei tehnologii noi PubNub care conține mai multe funcționalități decât sistemul MHealth, dar care poate veni în ajutorul unor dezvoltări ulterioare a chat-ului.

Așadar, sistemul implementat este unul complex care îmbină aplicații web și mobile pentru getionarea cazurilor de urgență prin raportarea lor de către un martor ce utilizează aplicația mobilă, datele fiind procesate pe partea de web, adăugând modulul personal care simplifică comunicarea între utilizatori prin intermediul unui chat.

8.2. Dezvoltări ulterioare

Modulul personal al sistemul medical MHealth descris prezintă o posibilitate de gestinoare a programărilor medicale şi face conexiunea dintre doctori și pacineți să fie mai ușor de realizat. Adăugarea mai multor funcționalități poate aduce doar beneficii asupra sistemului simplificând interacțiunea utilizatorilor cu aplicațiile web și mobile. Câteva funcționalități care se pot adăuga sistemului pentru dezvoltarea lui sunt:

- **Conferință video** - posibilitatea de a participa la o consultație virtuală folosind camera web a laptopului și cea a telefonului. S-ar putea realiza prin folosirea unor API-uri deja existente, cum este și cel dezvoltat de PubNub pentru comunicare video între browser şi telefon, în parteneriat cu framework-ul WebRTC.

- **Stilizare calendar programări** - vizualizarea programărilor într-un calendar prin marcarea zilelor pe lângă un tabel cu programări. În partea de UI să fie colorate zilele pline cu programări și marcate cele în care doctorul este indisponibil.
- **Notificare programare** - primirea unui notificări pe aplicația mobilă cu o zi înainte de data programării pentru a reaminti pacientului de ora programării. Notificarea se poate realiza similar modului în carea este implementată notificare urgențelor din apropiere, declanșator nefiind locația urgenței, ci data programării.
- **Distribuirea de fișiere integrată în chat** - pentru o utilizare mai rapidă s-ar putea introduce funcționalitatea de distribuirea a fișierului în cadrul chat-ului, adăugând un buton destinat distribuirii de fișiere în cadrul conversației și funcționalitatea să fie asemănătoare cu cea deja implementată în aplicația MHealth. Se poate realiza și prin tehnologia PubNub, stocând datele în platforma ei, dar această funcționalitate este diponibilă doar dacă este înregistrat un cont premium.
- **Grupuri de chat** - posibilitatea de a crea un grup cu mai mulți doctori pentru a comunica. Acest lucru s-ar putea realiza tot prin tehnologia PubNub, mai mulți utilizatori să se inscrie pe același canal, așa toți pot vizualiza toate mesaje trimise și pot publica și la rândul lor.

Bibliografie

[1] Vinicius Facco, R., Euclides Palma, P., Rafael, K., Rodolfo Stoffel, A., Cristiano André, d. C., & Rodrigo, d. R, "Exploring publish/subscribe, multilevel cloud elasticity, and data compression in telemedicine". *Computer Methods and Programs in Biomedicine*, 2020.
DOI: 10.1016/j.cmpb.2020.105403
Available at:
https://www.sciencedirect.com/science/article/abs/pii/S0169260719300604?via%3Dihub

[2] Vespignani, H., Soufflet, C., Masnou, P., Medjebar, S., Sakkat, E., & Frouin, P.-Y," Is telemedicine an adequate solution to perform and interpret EEGs?" *Elsevier*, 2018.
DOI: 10.1016/j.neucli.2018.06.064
Available at:
https://www.sciencedirect.com/science/article/pii/S0987705318301886

[3] Milstein, J.-A., Salzberg, C., Franz, C., Orav, E. J., Newhouse, J. P., & Bates, D. W, "Effect of Electronic Health Records on Health Care Costs: Longitudinal Comparative Evidence From Community Practices". *Annals Of Internal Medicine*, 2013.
DOI: 10.7326/0003-4819-159-2-201307160-00004
Available at: https://www.acpjournals.org/doi/10.7326/0003-4819-159-2-201307160-00004

[4] López, C., Valenzuela, J. I., Calderón, J. E., Velasco, A. F., & Fajardo, R, " A telephone survey of patient satisfaction with realtime telemedicine in a rural community in Colombi". *PubMed*, 2010.
DOI: 10.1258/jtt.2010.100611
Available at: https://journals.sagepub.com/doi/10.1258/jtt.2010.100611

[5] Kolsoum, D., Kambiz, B., & Seyed, M. T, "Teleconsultation and Clinical Decision Making: A Systematic Review". *Acta Inform Med*. 2016.
DOI: 10.5455/aim.2016.24.286-292
Available at: https://www.bibliomed.org/mnsfulltext/6/6-1468676794.pdf?1593182541

[6] Vikram K., B., Katherine H, D., & Evelyn Jane, B, "Emergency Response in the Ambulatory Surgery Center". *Division of Ambulatory Anesthesiology*, 2019.
DOI: 10.1016/j.anclin.2019.01.012
Available at: https://www.anesthesiology.theclinics.com/article/S1932-2275(19)30012-6/fulltext

[7] Sami, E., Mehmet Burak, A., Sinan, A., Kenan, K., & Fevzi Nuri, A,"Efficiency of Instant Messaging Applications in Coordination of Emergency Calls for Combat Injuries: A Pilot Study". *Ulus Travma Acil Cerrahi Derg*, 2017
DOI: 10.5505/tjtes.2016.37897
Available at: https://www.journalagent.com/travma/pdfs/UTD-37897-CLINICAL_ARTICLE-ASIK.pdf

[8] Liu X, Sutton PR, McKenna R, et al, "Evaluation of Secure Messaging Applications for a Health Care System: A Case Study". *Appl Clin Inform*, 2019.
DOI: 10.1055/s-0039-1678607
Available at: https://www.thieme-connect.de/products/ejournals/abstract/10.1055/s-0039-1678607

[9] Bautista JR, Lin TTC, "Nurses' use of mobile instant messaging applications: A uses and gratifications perspective". *Int J Nurs Pract*, 2017.
DOI: 10.1111/ijn.1257
Available at: https://onlinelibrary.wiley.com/doi/abs/10.1111/ijn.12577

[10] Iribarren SJ, Cato K, Falzon L, Stone PW, "What is the economic evidence for mHealth? A systematic review of economic evaluations of mHealth solutions". *Journal.pone*.0170581, 2017.
DOI: 10.1371/journal.pone.0170581
Available at:
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0170581

[11] E. Tîrziu şi M. Gheorghe-Moisii, Caracteristici de calitate ale aplicațiilor eHealth, *Romanian Journal of Information Technology and Automatic Control*, 2018.
Available at: https://rria.ici.ro/wp-content/uploads/2018/07/art.3-Tarziu-Moisii-2018.pdf?fbclid=IwAR1Tyv3kX3Xr4DWeF7DdF1PCner_45HDRp9z5c31oCIQVsXXmLny8J6wOIc

Anexa 1 - Fișa de evoluție

Data	Membru	Feature
01.11.2019	-	Emiterea temei
05.11.2019	-	Documentare inițială în legătură cu contextul temei
12.12.2019	-	Analiza funcționalități sistem posibile
05.01.2020	-	Împărțirea aplicației în componente și module
16.02.2020	-	Crearea proiectelor inițiale (Spring, React, Android)
19.02.2020	-	Implementarea interfeței inițiale a aplicației web și mobile
20.02.2020	-	Setarea resurselor şi uneltelor necesare pentru managementul aplicației (GitHub + GitHubDesktop + Google Drive)
01.03.2020	Andreea	Integrarea temei pe frontend
02.03.2020	Georgiana	Crearea proiectului Google Cloud
03.03.2020	Cosmin	Integrare servicii Firebase în proiect
05.03.2020	Andreea	Login - implementare și legatura backend-frontend
07.03.2020	Cosmin	Login în aplicația Android folosind senzorul de amprentă
12.03.2020	Cosmin	Funcțiile de înregistrare video/audio, realizare fotografie în aplicația Android
14.03.2020	Georgiana	Deploy backend pe Cloud
16.03.2020 18.03.2020	-	Realizarea capitolului 3
20.03.2020	-	Schema bazei de date (draft)
21.03.2020	Georgiana	Integrare Firebase la aplicația de pe Cloud
22.03.2020	Andreea	Adăugare funcționalități la Login pe Web (create account și forget password)
27.03.2020	Georgiana	Deploy frontend pe Cloud
28.03.2020	Andreea	Create account backend și legatura cu frontend

29.03.2020	Cosmin	Completare funcționalitate “Report Emergency” (locație, descriere text, nivel de severitate), stocare foto/video/audio pe firebase cloud storage, salvare urgență în firebase cloud firestore
29.03.2020	-	Actualizarea schemei bazei de date
31.03.2020	Andreea	Reset password cu trimitere de email și cod de securitate + redirecționare pentru confirmare parola nouă (back + front)
01.04.2020	Georgiana	Funcții backend pentru citirea urgențelor din baza de date; Redeploy backend
02.04.2020	Georgiana	Enable Geocoding API pentru transformarea din coordonate in adresa exactă (frontend)
04.04.2020	Cosmin	Folosire ML Kit pentru traducerea textului folosit ca descriere pentru urgență
05.04.2020	-	Realizare capitolelor 1 și 2
07.04.2020	Cosmin	Realizare creare cont
12.04.2020	Cosmin	Folosire ML Kit pentru preluarea datelor de pe poza cu buletinul
17.04.2020	Andreea	Implementare frontend și backend adăugare de appointment
18.04.2020	-	Propunere Cuprins
19.04.2020	-	Actualizare TF și adăugare NF
20.04.2020	-	Project Management
22.04.2020	Georgiana	Integrare Google Maps
23.04.2020	-	Diagrame UseCase
24.04.2020	-	Diagrama Conceptuală a sistemului
25.04.2020	-	Flowcharts
26.04.2020	Cosmin	Informare despre structura CI https://en.wikipedia.org/wiki/National_identity_cards_in_the_European_Economic_Area
27.04.2020	-	Informare EHR Standards

28.04.2020	-	Subfoldere LucrareLicenta
29.04.2020	Andreea	View, delete, update appointment backend și frontend
01.05.2020	Georgiana	Găsire set de date și antrenare + deploy model Language AutoML pentru procesarea textului
03.05.2020	Cosmin	Gestionare servicii foreground cu event listeners pentru tabelul de geofences
06.05.2020	Andreea	Implementare funcționalitate share files pe partea de web
07.05.2020	Georgiana	Implementare procesare imagine cu Vision API pentru extragerea etichetelor și a textului din imaginile stocate în baza de date
08.05.2020	Cosmin	Implementare Geofencing Api și Geocoding Api pentru anunțarea unei urgențe în apropiere și afișarea unei notificări cu adresa
09.05.2020	Andreea	Adăugare vacanță și asignare doctor înlocuitor backend și frontend
10.05.2020	Cosmin	Implementare funcționalitate Medical Practitioner
11.05.2020	Georgiana	Analiza cont Cloud pentru billing (servicii din free tier / servicii always free)
12.05.2020	-	Documentare parțială CF / TF
14.05.2020	Andreea	Implementare share files Android cu salvare in Firebase Storage + modificare salvare fisiere pe web
16.05.2020	Georgiana	Adăugare eventListener pentru verificarea urgențelor inserate (marcarea lor ca și clone în alt tabel)
17.05.2020	Cosmin	Îmbunătățiri Android (repornire servicii după restart smartphone, resetare componenta SharedPreferences etc)
19.05.2020	Andreea	CRUD appointment Android cu redirecționare la alt doctor în caz de indisponibilitate la data aleasă
21.05.2020	Georgiana	Implementare Speech-to-Text API
23.05.2020	-	Împărțire diagrame existente și adăugarea lor în lucrări + diferențiere cuprins
24.05.2020	-	Migrare text din latex in word (TF Cloud si Studiu bibliografic)

27.05.2020	-	Diagrame de pachete, secvențiere
29.05.2020	-	Diagrama de componente, deployment
31.05.2020 01.06.2020	-	Scris în lucrarea de licenta: CF, NF, arhitectura conceptuală, cazuri de utilizare
05.06.2020 06.06.2020	-	Realizarea capitolului 6 - Testare
08.06.2020	-	Modificare diagrame Descrierea tehnologiilor
10.06.2020	Georgiana	Documentare si implementare blockchain
11.06.2020	Andreea	Documentare și implementare chat în componenta web
12.06.2020 13.06.2020	-	Realizare capitolului 4
14.06.2020	Andreea	Documentare și implementare chat în componenta mobile
15.06.2020	Georgiana	Implementare funcțiilor de sortare și trimitere mail
17.06.2020	-	Selectarea referințelor și materialelor potrivite pentru studiul bibliografic
20.06.2020	-	Verificarea funcționalității sistemului
21.06.2020 22.06.2020 23.06.2020	-	Realizarea capitolului 5 și 7
24.06.2020	Cosmin	Optimizare și refactorizare componente Android
25.06.2020	Andreea	Acoperirea unor noi funcționalități pe web
26.06.2020	Georgiana	Deploy-ul versiunilor finale de backend și frontend
28.06.2020	-	Realizarea capitolului 8
01.07.2020	-	Finalizarea temei

Anexa 2 - Tabel de figuri

Anexa 3 - Listă de tabele

Anexa 4 - Glosar de termeni

Termen	Explicație
MHealth	Denumirea întregului sistem implementat
REST	Representational state transfer - stil arhitecturat prin care datele sunt transmise printr-un URI
JWT	JSON Web Token - standard de internet pentru a crea date cu o semnătură opțională
EHR	Electronic Health Records - protocol de stocare a datelor la nivel digital
API	Application Programming Interface - set de interfețe pentru programarea de aplicații
CRUD	Create, Read, Update, Delete - cele 4 funcționalități specifice stocării datelor
NoSQL	Mecansim de stocare pentru un model de date non-relațional
SDK	Software Development Kit - set de unele ce conțin biblioteci sau API-uri ce pot fi apelatede către programatori
UUID	Universally Unique IDentifier - identificator unui universal de 128 de biți

Anexa 5 - Instrument de modelare a bazei de date cu Hackolade