MHealth

Modulul Android - componenta de gestionare a urgențelor

LUCRARE DE LICENŢĂ

Absolvent: **Cosmin-Ştefan DASCĂLU**

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

2020

Cuprins

Capitolul 1. Introducere – Contextul proiectului

În acest capitol se va realiza o scurtă prezentare a contextului proiectului, o introducere în domeniul în care se situează sistemul realizat.

1.1. Motivație

Sistemul medical este într-o continuă evoluție datorată îmbunătățirilor aduse din punct de vedere tehnologic în diferite domenii care influențează în mod direct acest domeniu.

Astfel, un termen de comparație care merită a fi luat în calcul este timpul de răspuns la urgențele medicale. În acest sens, domeniul informatic a fost cel care și-a pus amprenta semnificativ în acest context care poate face diferența pentru pacientul ce necesită ajutor medical.

Un mare avantaj pe care l-a adus evoluția tehnologiei este posibilitatea de a primi și trimite un volum mare și variat de informații într-un mod rapid si sigur. Orice dispozitiv cu o conexiune la internet poate comunica cu celelalte, schimbul de informații fiind realizat fără limitări. Dacă în ceea ce privește calculatoarele personale lucrurile sunt stabile și nu au evoluat semnficativ, dispozitivele mobile de tip smart au primit de-a lungul timpului numeroase îmbunătățiri care au dus la posibilitatea de a folosi aceste dispozitive în scopuri medicale, lucru care nu era considerat fezabil înainte.

MHealth își propune să vină în ajutorul sistemului medical, în special în contextul urgențelor. Astfel, perioada de timp dintre apelul inițial la 112 și momentul în care ambulanța ajunge la locația urgenței este gestionată folosind sistemul MHealth. Schimbul de informații și feedback-ul legat de acestea constituie ideea centrală a întregului sistem.

Urgențele medicale vor putea fi acum gestionate mai eficient, dat fiind faptul că mai multe persoane pot oferi informații cruciale din diferite perspective. Astfel, importanța timpului în care ambulanța ajunge la locația urgenței este teoretic redusă, situația problematică putând fi gestionată parțial prin feedback-ul specializat la informațiile oferite și prin ajutorul acordat de persoanele cu pregătire medicală implicate independent în contextul urgenței prin sistemul MHealth.

Implementarea sistemului face ca orice urgență, atât cele publice, cât și cele personale să se poată folosi de MHealth. Comunicarea digitală, schimbul de fișiere sau oferirea informațiilor de orice natură sunt aspecte care fac ca gestionarea urgențelor să se realizeze ușor și eficient, în anumite cazuri apelul la 112 nefiind necesar. Utilizatorul obișnuit va avea toate uneltele necesare raportării urgenței într-o singură aplicație mobilă, în timp ce personalul specializat va avea la dispoziție o aplicație web unde datele sunt centralizate, feedback-ul putând fi astfel oferit având toate informațiile necesare la dispoziție.

În concluzie, sistemul MHealth este un ajutor, un sprijin oferit sistemului medical deja existent, și nu un substituent, scopul principal fiind îmbunătățirea și completarea acestui domeniu extrem de important în activitățile cotidiene.

1.2. Conținutul lucrării

Acest subcapitol are scopul de a prezenta aspectele care sunt descrise în următoarele capitole ale acestei lucrări:

- **Capitolul 2 –** Acest capitol prezintă scopul principal al sistemului implementat, dar și ținte mai exacte, mai specifice pe care MHealth si-a propus să le îndeplinească.

- **Capitolul 3 –** Acest capitol face o descriere a contextului real al aplicației, o descriere a modului în care tehnologiile folosite influențează respectivul context, dar și o comparație cu alte sisteme deja existente care fac parte din același context.

- **Capitolul 4 –** Acest capitol prezintă analiza realizată în implementarea sistemului și informațiile necesare în acest sens. Vor fi prezentate cerințe de sistem, cazuri de utilizare ale sistemului și setul de tehnologii utilizat.

- **Capitolul 5 –** Acest capitol prezintă detalii de implementare și modul din punct de vedere tehnic în care au fost îndeplinite obiectivele stabilite, folosind diagrame UML pentru a evidenția structura sistemului.

- **Capitolul 6** – Acest capitol prezintă modul în care a fost realizată testarea sistemului, descrierea tipurilor și testelor aplicate, cât și rezultatul lor.

- **Capitolul 7 –** Acest capitol prezintă manual de utilizare al sistemului. Vor fi prezentate resursele neceare pentru fiecare componentă a sistemului, dar și pentru fiecare tip de utilizator.

- **Capitolul 8 –** Acest capitol prezintă concluziile personale după realizarea aplicației, posibile dezvoltări ulterioare, dar și rezultatele obținute.

Capitolul 2. Obiectivele Proiectului

În acest capitol vor fi prezentate obiectivele generale ale întregului sistem Mhealth, dar în același timp se vor prezenta și obiectivele specifice componentelor, în special cele care țin de modulul Android, componenta de gestionare a urgențelor.

2.1. Obiectivul general

Sistemul MHealth are ca obiectiv principal digitalizarea gestionării urgențelor. În acest sens, s-a intenționat îmbinarea tehnologiei în domeniul medical prin realizarea unui sistem a cărui componente să poată interacționa indiferent de factori externi.

Astfel, sistemul constă din două module: un modul Android destinat dispozitivelor mobile și un modul Web destinat calculatoarelor personale la care vor avea acces personalul medical specializat. Ambele module au atât o componentă responsabilă cu gestionarea urgențelor, cât și o componentă responsabilă cu gestionarea situațiilor personale.

Schimbul de informații, fișiere sau feedback-ul sunt aspectele în jurul cărora gravitează implementarea și arhitectura generală a sistemului. Pentru îndeplinirea obiectivului general al aplicației este nevoie de integrarea celor trei părți:

- Componenta de gestionare a urgențelor pentru modulul Web – realizat de Georgiana-Bianca Cornea
- Componenta personală atât pentru modulul Web, cât și pentru modulul Android – realizat de Andreea Ifrim
- Componenta de gestionare a urgențelor pentru modulul Android – realizat de Cosmin-Ștefan Dascălu

În cele ce urmează vor fi prezentate obiectivele specifice ale modulului de gestionare a urgențelor pentru componenta Android, descris în această lucrare.

2.2. Obiective specifice

Componenta de gestionare a urgențelor pentru modulul Android este responsabilă de oferirea funcționalităților specifice utilizatorului obișnuit și funcționalităților specifice utilizatorului cu pregătire medicală.

Toți utilizatorii aplicației mobile MHealth au parte de o securitate sporită, sistemului folosind date biometrice. Astfel, autentificarea se realizează folosind senzorul de amprentă. De asemenea, fiecare dispozitiv are atribuit un cont, acesta fiind realizat prin extragerea automată a datelor de pe o poză conținând cardul de identitate.

În ceea ce privește realizarea obiectivelor specifice utilizatorului obișnuit autentificat în cadrul aplicației, se pot remarca următoarele:

- Posibilitatea raportării unei urgențe, realizând și trimițând conținut de tip foto, video, audio, text
- Traducerea automată a textului în limba engleză
- Localizarea automată a urgenței, oferirea coordonatelor GPS

- Selectarea gradului de severitate

În ceea ce privește realizarea obiectivelor specifice utilizatorului cu pregătire medicală, se pot remarca următoarele:

- Primirea unei notificări cu adresa exactă a unei urgențe din apropiere dacă utilizatorul a petrecut cel puțin perioada prestabilită în aria de acoperire a urgenței
- Posibilitatea preluării urgenței prin acționarea notificării atribuite urgenței respective
- Posibilitatea introducerii unei adrese de email în cazul preluării unei urgențe, adresă pe care se va primi un email cu toate informațiile necesare

Tabel 2.1 Componentele sistemului MHealth

	Modulul Android	**Modulul Web**
Utilizatori	Obișnuiți și cu pregătire medicală	Doctori
Componenta gestionării urgențelor	Detaliile despre acestă componentă al modulului Android au fost descrise anterior.	Datele sunt preluate de la aplicația mobilă şi prelucrate în mod automat. Se folosesc servicii Cloud, împreună cu Machine Learning pentru prelucrarea imaginilor, textului, înregistrărilor video şi 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	Se ocupă de punerea în legătura a pacienților cu medicii de familie sau cu alți doctori de specialitate pentru sfaturi medicale, schimb de fișiere sau realizarea de programări. De asemenea, pune la dispoziție un chat pentru a facilita comunicarea.	

Capitolul 3. Studiu Bibliografic

În acest capitol se vor prezenta diferite aspecte ale studiului bibliografic realizat cu scopul întocmirii acestei lucrări.

3.1. Impactul tehnologiei asupra medicinei

Trecerea timpului a adus odată cu ea și o evoluție a tehnologiei, evoluție care a avut un impact pozitiv asupra multor domenii printre acestea numarându-se și domeniul medical.

Una dintre cele mai importante și entuziasmante ramuri ale medicinii este „mHealth". Deși nu există o versiune standardizată a definiției, organizația mondială a sănătății[1] definește această ramură ca fiind practici medicale sau practici ce țin de sănătatea publică susținute de dispozitive mobile ca smartphone-urile, dispozitivele de monitorizare a sănătății sau alte dispozitive wireless.

Un avans tehnologic impresionant și vizibil în viața cotidiană este cel realizat în rândul dispozitivelor de tip „wearable". Astfel, monitorizarea activităților fizice, parametrilor ce țin de sănătatea generală a organismului cum ar fi ritmul cardiac sau oferirea informațiilor despre somn s-au transformat din concepte în fapte realizabile la îndemâna oricui prin intermediul unei brățări inteligente sau a unui ceas inteligent.

Un alt mod în care tehnologia poate îmbunătății serviviile medicale existente este reducerea cazurilor în care un pacient trebuie să se deplaseze fizic pentru un anumit control. În acest sens, dispozitivul care vine în ajutorul pacientului este chiar propriul smartphone. Astfel, detecția unei infecții în zona urechii a devenit un proces care se poate realiza chiar acasă prin atașarea unui otoscop[2]. Aceeași metodă a fost utilizată și în domeniul dermatologiei, diagnosticarea mai multor afecțiuni fiind realizată prin atașarea unui dermatoscop la camera foto a dispozitivului mobil.

Atașarea diferitelor dispozitive sau utilizarea unor senzori nu sunt singurule moduri prin care un smartphone poate avea un impact pozitiv asupra medicinei. Un mare avantaj al smartphone-ului față de telefoanele mobile clasice este posibilitatea folosirii unor aplicații realizate de dezvoltatori terți. Astfel, personalul medical specializat are posibilitatea realizării unor intervenții de la distanță. Acest scenariu este întâlnit în contextul militar, unde gestionarea unor traume trebuie realizată în ciuda absenței unui cadru medical. Aici intervin aplicații medicale terțe, care oferă asistență foto, video sau sub formă de instrucțiuni text.

Cu toate astea, impactul tehnologiei asupra medicinii se dorește a fi remarcat cel mai clar în numărul de vieți salvate și pentru asta tehnologia trebuie să îmbunătățească poate cea mai importantă ramură a medicinei, mai exact serviciile de gestionare a urgențelor. Deși conform organizației mondiale a sănătății timpul de răspuns ideal la urgențe este de aproximativ 8 minute, chiar și în țări dezvoltate precum Austria, timpul de răspuns real este departe de această cifră. În anul 2015, Viena a raportat un timp mediu de răspuns de 15 minute, aproape dublu față de cel recomandat. Importanța gestionării

[1] https://www.who.int/goe/publications/goe_mhealth_web.pdf?

[2] https://www.forbes.com/sites/singularity/2012/07/16/now-your-smartphone-can-be-used-to-diagnose-ear-infections-at-home/#7feccba16811

urgenței în acest interval crește astfel semnificativ, fiecare minut fiind crucial în șansele de supraviețuire. Aici intervine tehnologia, aplicațiile mobile integrate într-un ecosistem medical putând face diferența. Posibilitatea comunicării, schimbului de fișiere și de informații cu ajutorul smartphone-ului personal este poate cea mai mare îmbunătățire pe care a adus-o tehnologia în domeniul medical, scopul principal al sistemului MHealth fiind chiar acesta, gestionarea urgențelor în intervalul de timp care trece de la notificare la sosirea cadrului medical.

3.2. Senzorii dispozitivelor mobile în aplicațiile medicale

Odată cu trecerea timpului și cu evoluția tehnologiei, smartphone-ul a devenit cel mai important mijloc de comunicare al omenirii. Astfel, conform studiilor[3] 3.5 miliarde de oameni folosesc un smartphone, număr ce reprezintă 44.98% din populația lumii.

Motivul principal pentru care dispozivele mobile sunt foarte atractive în domeniul medical este faptul că acestea dispun de foarte mulți senzori, aproape fiecare dintre acești fiind capabil să joace un rol important în gestionarea unei urgențe.

Senzorii integrați în smartphone-uri se pot împărți în două categorii: senzori de mediu și senzori de localizare. Printre cei mai importanți senzori de mediu se numără microfonul și camera.

Microfonul este un senzor de mediu prezent pe toate telefoanele mobile inteligente. Cazul de utilizare principal al acestui senzor este comunicarea. În ramura „mHealth" a medicinii, microfonul este utilizat pentru a înregistrarea unei descrieri, poate chiar a întregului mediu ce este subiectul utilizării aplicației mobile sau la schimbul de informații între utilizator și personalul specializat.

Avansul tehnologic a dus de asemenea la utilizarea microfonului și ca suport pentru diagnosticarea automată. Un exemplu în acest sens este descris în articolul [6]. Miotonia este o anomalie musculară caracterizată printr-o decontractare anormal de lentă. Conform articolului menționat anterior, utilizatorii vor putea folosi un jurnalul automat, interactiv, bazat pe voce, pentru a monitoriza frecvența și severitatea simptomelor: oboseală, durere, slăbiciune musculară sau rigiditate musculară. O dată pe saptămână timp de opt săptămâni, utilizatorii vor trebuie sa înregistreze aceste date despre simptome. Sistemul automat va grupa și clasifica simptomele, reducând astfel numărul de vizite necesare și permițând monitorizarea pacienților fără ca spitalizarea să mai fie necesară.

În cazul sistemului MHealth, microfonul este utilizat pentru a oferi posibilitatea de a adăuga conținut audio în cadrul raportării unei urgențe.

Camera este un alt senzor de mediu prezent pe toate telefoanele mobile inteligente. Odată cu avansul tehnologic realizat în ceea ce privește smartphone-ul, calitatea camerelor a crescut substanțial. Dat fiind interesul enorm al clienților pentru camere calitative pe smartphone-uri, producătorii se află într-o concurență continuă în ceea ce privește numărul de funcționalități pe care camerele le oferă, dar și calitatea generală a acestora.

[3] https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

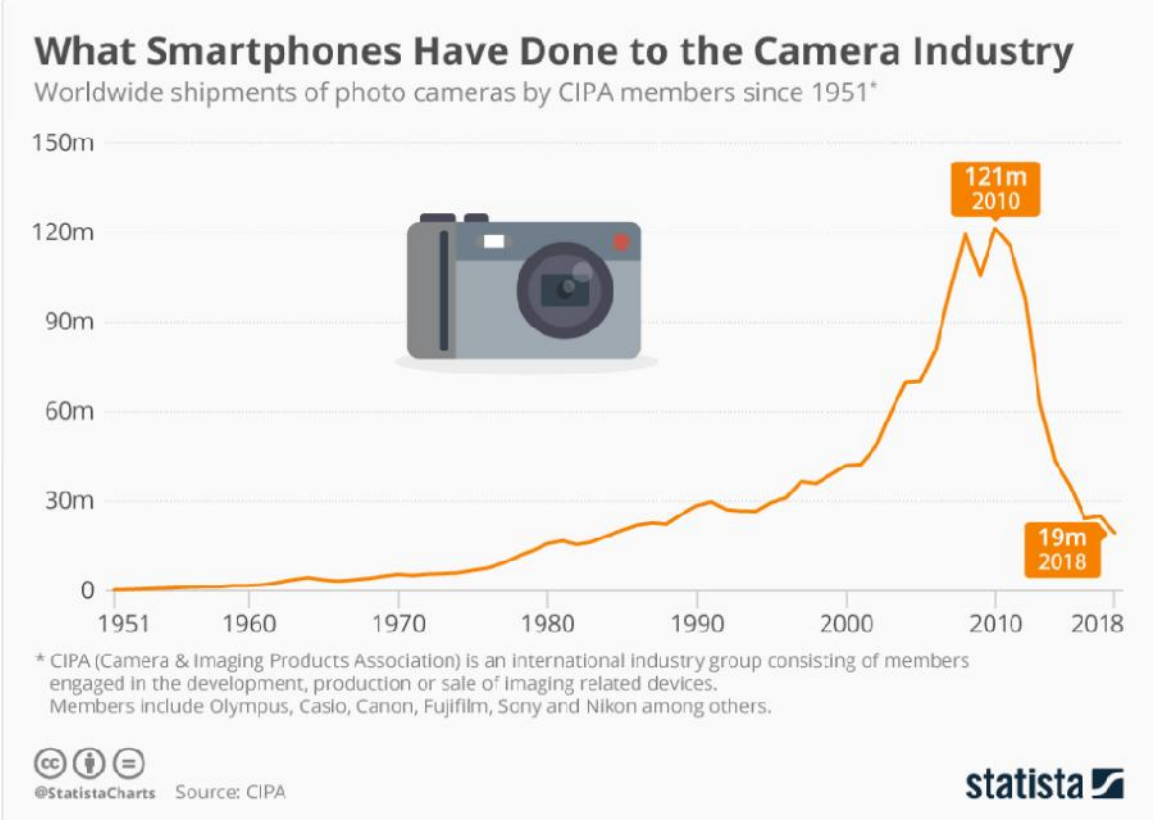


Figura 3.1 Evoluția vânzărilor de aparate foto[4]

Figura 3.1 prezintă modul în care avansul tehnologic, mai ales cel în domeniul telefoanelor mobile inteligente a afectat numărul de vânzări al aparatelor foto. Dacă în anul 2010, industria aparatelor foto era la apogeu, numărul vânzărilor depășind 120 de milioane de unități, în anul 2018 numărul de vânzări nu atinge nici 20 de milioane de unități ajungând la o cifră apropiată de cea înregistrată in 1985. Acest fenomen este datorat faptului că oamenii nu mai justifică utilizarea unui dispozitiv separat pentru a realiza conținut foto-video, când telefonul mobil inteligent dispune de un senzor similar, poate chiar mai bun în anumite situații.

Această evoluție face ca utilizarea camerei în aplicațiile mobile medicale să joace un rol foarte important. Prima modalitate în care camera de pe telefoanele mobile inteligente a fost folosită în scop medical a fost în contextul consultațiilor medicale realizate la distantă. În acest moment, există o multitudine de aplicații, nu neapărat medicale, care pot oferi platforma suport pentru realizarea acestor consultații: WhatsApp, Messenger, Skype etc.

În ceea ce privește un mediu specializat pentru realizarea acestor consultații, ClickMedix[5] oferă o aplicație în domeniul dermatologiei. Astfel, utilizatorii sunt instruiți cu scopul de a realiza fotografii legate de probleme ale pielii, aceste date sunt trimise

[4] https://www.statista.com/chart/15524/worldwide-camera-shipments/

[5] https://clickmedix.com/wp-content/uploads/2012/05/ClickMedix-Brochure-v1.1-1.pdf

către un server, date la care doctorii au acces și pe urma cărora pot analiza problema și pune un diagnostic, urmând prescrierea unui tratament corespunzător.

O altă ramură în care camera de pe telefonul mobil inteligent poate fi folosită este cardiologia. Cardiio[6] este o aplicație mobilă care poate măsura ritmul cardiac folosindu-se de cameră. Utilizatorul trebuie să acopere cu un deget întreaga suprafață a senzorului. La fiecare bătaie a inimii, fluxul de sânge crește fapt ce duce la o absorbție mai mare a luminii. Între bătăile inimii, mai puțină lumină este absorbită. Detectând schimbările subtile în modul în care lumina este absorbită, aplicația poate calcula pulsul.

În contextul sistemului MHealth, camera de pe dispozitivul mobil este utilizată pentru a oferi posibilitatea de a adăuga conținut foto sau video în cadrul raportării unei urgențe. De asemenea, în procesul de creare a unui cont asociat cu dispozitivul mobil, camera este folosită pentru a realiza o fotografie conținând cardul de identitate. Datele vor fi extrase automat și confirmate de utilizator înainte ca asocierea cu un cont să fie realizată.

GPS[7], Global Positioning System este un sistem disponibil pe toate telefoanele mobile inteligente folosit în scopuri de localizare. Se folosește de undele radio trimise de la sateliți la receiverul din interiorul dispozitivului mobil pentru a stabili locația exactă. Trimiterea de date de la dispozitivul mobil la sateliți nu este necesară, este nevoie doar de primirea lor cu succes de cel puțin 4 sateliți destinați localizării din cei 28 posibili.

Dat fiind faptul că datele GPS sunt lente, localizarea putând dura chiar și un minut, iar receiverul din interiorul dispozitivului mobil folosește foarte multe resurse, AGPS este preferat. GPS rămâne totuși metoda mai precisă de localizare.

AGPS, Assisted Global Positioning System este sistemul preferat când se dorește localizarea. Folosește mai puține resurse, consumând astfel mai puțin bateria dispozitivului mobil și este mai rapid. Un alt avantaj este faptul că, spre deosebire de GPS, nu este obstrucționat de clădiri înalte sau alte obstacole care pot preveni primirea de date. AGPS se folosește de datele celulare pentru a obține locația. Acest lucru se realizează folosind turnurile de telefonie mobilă ale furnizorului de servicii mobile. Deși acest sistem, spre deosebire de GPS, trimite date de la telefonul mobil, aceste date nu sunt suplimentare, ele fiind transmise oricum către turnurile de telefonie mobilă. Locația obținută are o eroare de aproximativ 50 de metri, dar în momentul în care se obțin date GPS de la sateliți, locația afișată este actualizată cu valoarea mai precisă.

MHealth obține locația utilizatorului aplicației mobile în mod automat, aceasta fiind inclusă în cadrul raportului.

Senzorul de amprentă este folosit în special ca mod de autenficare a utilizatorului. Acest mod de autenficare și recunoașterea facială sunt cele mai noi metode de securizare a telefoanelor mobile inteligente.

[6] https://apps.apple.com/us/app/cardiio-heart-rate-monitor/id542891434
[7] https://www.androidcentral.com/how-does-gps-work-my-phone

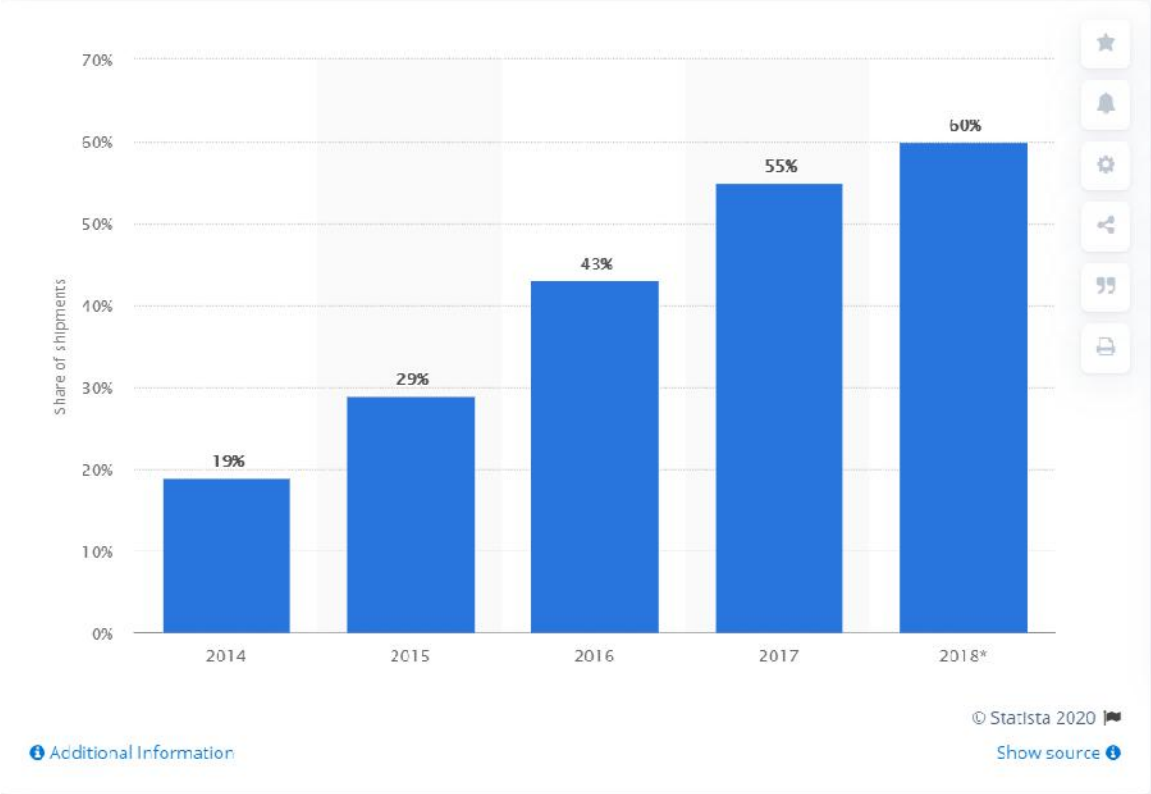


Figura 3.2 Procentul de smartphone-uri cu senzor de amprentă[8]

Figura 3.2 prezintă creșterea anuală constantă a procentului de telefoane mobile inteligente ce conțin un senzor de amprentă. Conform articolului[9], raportându-ne tot la anul 2018, procentul estimat de smartphone-uri ce folosesc recunoașterea facială este de 40%, cifră considerabil mai mică decât cea a dispozitivelor mobile ce dispun de senzorul de amprentă. Din acest motiv, componenta Android a sistemului MHealth folosește ca metodă de autentificare senzorul de amprentă, scopul fiind ca aplicația să fie atât sigură, cât și disponibilă cât mai multor utilizatori.

Senzorii de amprentă utilizați pe smartphone-uri se împart în 3 categorii[10]: senzori capacitivi, senzori optici și senzori ultrasonici.

Senzorii optici sunt cei mai vechi dintre cei menționați. Se realizează o fotografie a suprafeței degetului și se folosesc algoritmi de detecție a șabloanelor unice. Senzorii optici folosesc leduri pentru a lumina aria de interes, deoarece prezența degetului întunecă fotografia.

[8] https://www.statista.com/statistics/804269/global-smartphone-fingerprint-sensor-penetration-rate/

[9] https://www.biometricupdate.com/201802/counterpoint-estimates-more-than-1-billion-smartphones-to-be-shipped-with-facial-recognition-in-2020

[10] https://www.androidauthority.com/how-fingerprint-scanners-work-670934/

Marele dezavantaj al senzorilor optici este nivelul de securitate scăzut, deoarece fotografia realizată este 2D și acest design poate fi păcălit chiar și cu o altă imagine de calitate înaltă.

Senzorii capacitivi sunt cei mai folosiți la ora actuală pe telefoanele mobile inteligente. Spre deosebire de senzorii optici, cei capacitivi nu realizează o fotografie 2D, ci se folosesc de niște condensatori electrici ce preiau datele, le analizează, le salvează și le compară ulterior.

Această metodă este mult mai sigură, dar și mai scumpă, prețul componentelor necesare fiind încă ridicat, deși mult mai scăzut decât la momentul inițial. Acești senzori pot fi folosiți și în alte scopuri. Un bun exemplu este folosirea senzorului pentru acțiunea de glisare în galerie, funcționalitate disponibilă pe telefoanele mobile inteligente mai noi.

Cea mai nouă tehnologie de autentificare folosind amprenta se bazează pe senzorii ultrasonici. Acești senzori au două componente majore: un transmițător ultrasonic și un receiver. Un puls ultrasonic este transmis către degetul poziționat peste scanner. O parte din acest puls este absorbit, restul fiind trimis înapoi și recepționat de receiver. Acest fenomen depinde de caracteristicile unice al amprentei, reproducția 3D a amprentei facând ca această metodă să fie cea mai sigură dintre cele trei metode prezentate.

Senzorii ultrasonici se regăsesc cel mai des integrați sub ecran pe cele mai noi dispozitive. Totuși, fiind cea mai nouă tehnologie, ea nu este încă perfecționată, prezentând două dezavantaje evidente: viteza, procesul de trimitere și recepționare a pulsului ultrasonic fiind încă lent și compatibilitatea precară cu foliile de protecție, acestea obstrucționând comunicarea senzorului cu degetul.

Figura 3.3 Senzor ultrasonic integrat sub ecran[11]

[11] https://www.androidauthority.com/how-fingerprint-scanners-work-670934/

3.3. Importanța notificărilor de tip push

O notificare de tip push[12] este un mesaj scurt sau o alertă trimisă de o aplicație către toți utilizatorii care au acea aplicație instalată și care au permis trimiterea acestui tip de notificări. Pentru a oferi accesibilitate ridicată, aplicația nu trebuie să fie pornită la momentul trimiterii notificării.

Notificările de tip push oferă multiple avantaje în ceea ce privește trimiterea informațiilor esențiale, față de alternativele clasice: emailurile tradiționale și mesajele SMS. Când se trimite un email, acesta pleacă de la un inbox, ajunge în alt inbox și asteaptă acolo până când utilizatorul decide să-l citeasca. În cazul în care emailul ajunge în folderul „spam”, este posibil ca mesajul să nu fie citit vreodată. În ceea ce privește mesajul SMS, acesta ajunge direct în telefonul mobil al destinatarului, dar numărul de destinatari posibili este limitat. Un mare dezavantaj pe care îl prezintă emailul și mesajul SMS este faptul că mulți oameni nu doresc ca datele de contact ale acestora, mai exact adresa de email și numărul de telefon, să fie publice sau folosite în scopuri comerciale.

Statisticile[13] arată că aproximativ 25% dintre utilizatori dezinstalează o aplicație după doar o utilizare și doar 16% dintre utilizatori folosesc o aplicație de mai multe de 2 ori. Notificările de tip push poate schimba în mod semnificativ modul în care un utilizator interacționează cu o aplicație, ținând cont că[14] 76% dintre utilizatorii cu vârste între 18 și 34 de ani au setat ca aplicațiile să poată trimite notificări de tip push.

Un factor decisiv în succesul acestor notificări este rata de accesare a lor. Astfel, studiile arată că notificările de tip push au o rată de accesare cu 50% mai mare decât rata de accesare a emailurilor.

Cu toate acestea, modul în care informația este transmisă prin intermediul notificărilor de tip push contribuie decisiv cifrele prezentate anterior.

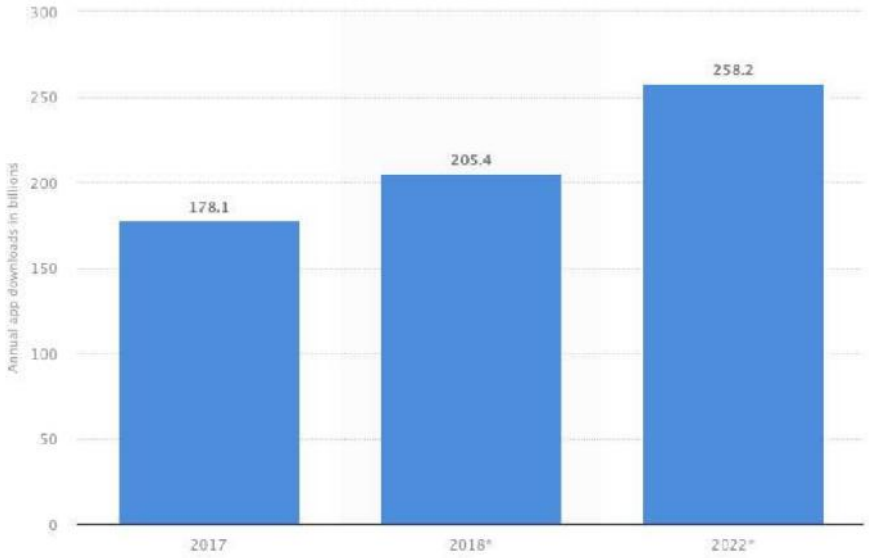


Figura 3.4 Numărul de miliarde de aplicații descărcate anual

[12] https://buildfire.com/what-are-push-notifications/

[13] https://techcrunch.com/2016/05/31/nearly-1-in-4-people-abandon-mobile-apps-after-only-one-use/

[14] https://www.entrepreneur.com/article/234875

Figura 3.4 prezintă creșterea evidentă în ceea ce privește numărul de aplicații descărcate în fiecare anual. Chiar dacă 90% din ele sunt deschise o singură dată, cu cât numărul de aplicații descărcate crește, cu atât cresc șansele dezvoltatorilor să facă aplicațiile atractive, un mod de a realiza acest lucru fiind notificările de tip push.

CEO-ul și co-fondatorul Hivemapper, Ariel Seidman[15], a spus: „Este greu să nu supraevoluezi puterea notificărilor de tip push. Pentru prima oară în istorie, poți bate pe spate milioane de oameni în același timp."

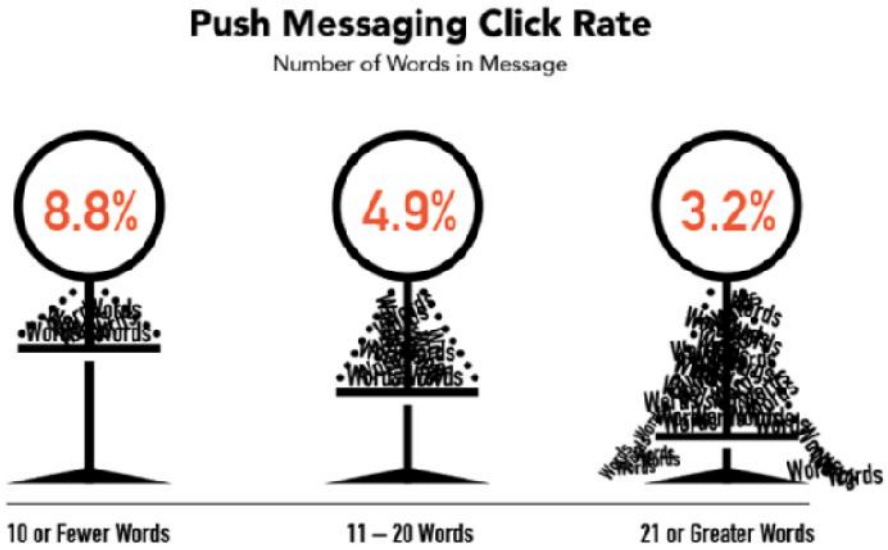


Figura 3.5 Rata de accesare a notificărilor push[16]

În figura 3.4 este prezentat modul în care numărul de cuvinte dintr-o notificare de tip push afectează de rata de accesare. Se poate observa faptul că rata de accesare tinde să crească odată cu reducere numărului de cuvinte folosite.

În contextul sistemului MHealth s-au luat în calcul mai multe tehnologii pentru a trimite notificări de tip push: Firebase Cloud Messaging, Google Nearby Messages și Geofencing.

Firebase Cloud Messaging este un serviciu cross-platform care oferă posibilitatea dezvoltatorilor de a trimite mesaje de la un server către aplicațiile realizate. FCM înlocuiește fostul Google Cloud Messaging, folosind aceleași servere Google pentru a trimite mesaje, dar adaugă funcționalitatea de a trimite notificări push web[17].

Implementarea[18] acestui serviciu necesită două componente principale: un server securizat care să se ocupe de gestionarea și trimiterea mesajelor, de exemplu Firebase, și o aplicație client (Android, iOS, Web) care să primească mesajele.

Design-ul serviciul FCM prin care se trimit mesaje a rămas identic cu cel al GCM, așa cum este prezentat în figura 3.5.

[15] https://blog.pushengage.com/why-mobile-push-notifications-are-important/
[16] https://info.localytics.com/blog/ideal-push-message-length
[17] https://www.pushmaze.com/how-fcm-push-notification-works/
[18] https://firebase.google.com/docs/cloud-messaging

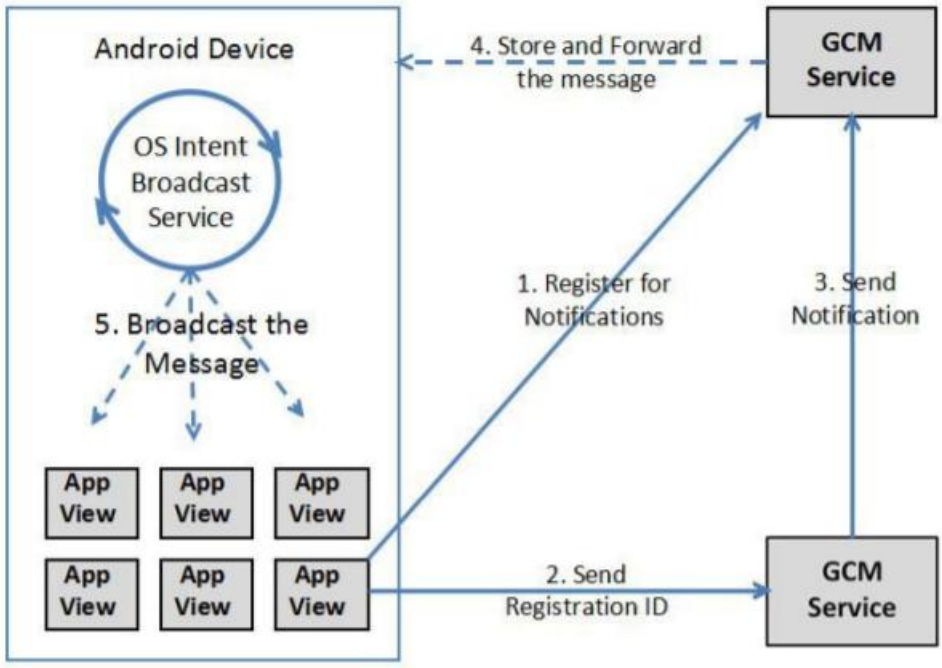


Figura 3.6 Flow-ul FCM [9]

În contextul sistemului MHealth implementat nu s-a ales Firebase Cloud messaging, deoarece notificările push se doresc a fi filtrate în funcție de distanța fiecărui utilizator de o locație raportata.

Google Nearby Messages este un API de tip publish-subscribe lansat de cei de la Google care permite dispozitivelor mobile conectate la internet să facă schimb de date. Dispozitivele trebuie doar să fie conectate la internet, nu neapărat la aceeași rețea.

API-ul folosește[19] o combinație între Bluetooth, Wi-Fi pentru a comunica un cod unic între dispozitive. Când un dispozitiv primește un cod de la un dispozitiv din apropiere, trimite codul înapoi către server pentru validare care după facilitează trimiterea de mesaje între dispozitive.

Marele dezavantaj al acestui API, mai ales în cazul sistemului MHealth, este faptul că distanța maximă între dispozitive este de aproximativ 20 de metri. Această distanță nu satisface cerințele sistemului MHealth, notificările trebuind să fie activate de alte dispozitive aflate la o distanță mult mai mare, astfel API-ul nu a fost folosit.

Spre deosebire de cele două tehnologii prezentate anterior, **Geofencing** oferă atât filtrare a utilizatorilor în funcție de locație, cât și posibilitatea de a notifica dispozitive aflate la distanțe mari de locația unei urgențe. Aceste amănunte au dus la folosirea API-ului pentru implementarea sistemului MHealth.

Modul în care notificările de tip push operează pe sistemele de operare Android și iOS diferă: sistemul de operare Android permite trimiterea notificărilor de tip push în mod implicit, utilizatorii fiind nevoiți să dezactiveze manual această opțiune în cazul în care nu doresc să primească notificări, în timp ce sistemul de operare iOS operează în mod total contrar.

[19] https://developers.google.com/nearby/messages/overview

3.4. Aplicații similare

În această secțiune se vor prezenta câteva aplicații din domeniul medical, aplicații care prezintă funcționalități relativ similare cu cele oferite de sistemul MHealth.

3.4.1. Descrierea aplicațiilor

ELERTS See Say[20] este o aplicație mobilă disponibilă atât pe iOS cât și pe Android. Scopul principal al aplicației este îmbunătățirea siguranței personale.

Funcționalitatea de bază a aplicației reprezintă posibilitatea raportării unei activități suspecte către un personal specializat. Raportul conține locația activității, dar poate prezenta și o descriere text sau o imagine foto realizată în acel moment.

O altă funcționalitate interesantă a aplicației este cea de „Check-in". Prin această opțiune, utilizatorul poate transmite un mesaj anumitor persoane. Mesajul poate fi trimis prin SMS, Email, Twitter sau Facebook. Odată cu mesajul se va transmite și o hartă cu locația curentă a utilizatorului.

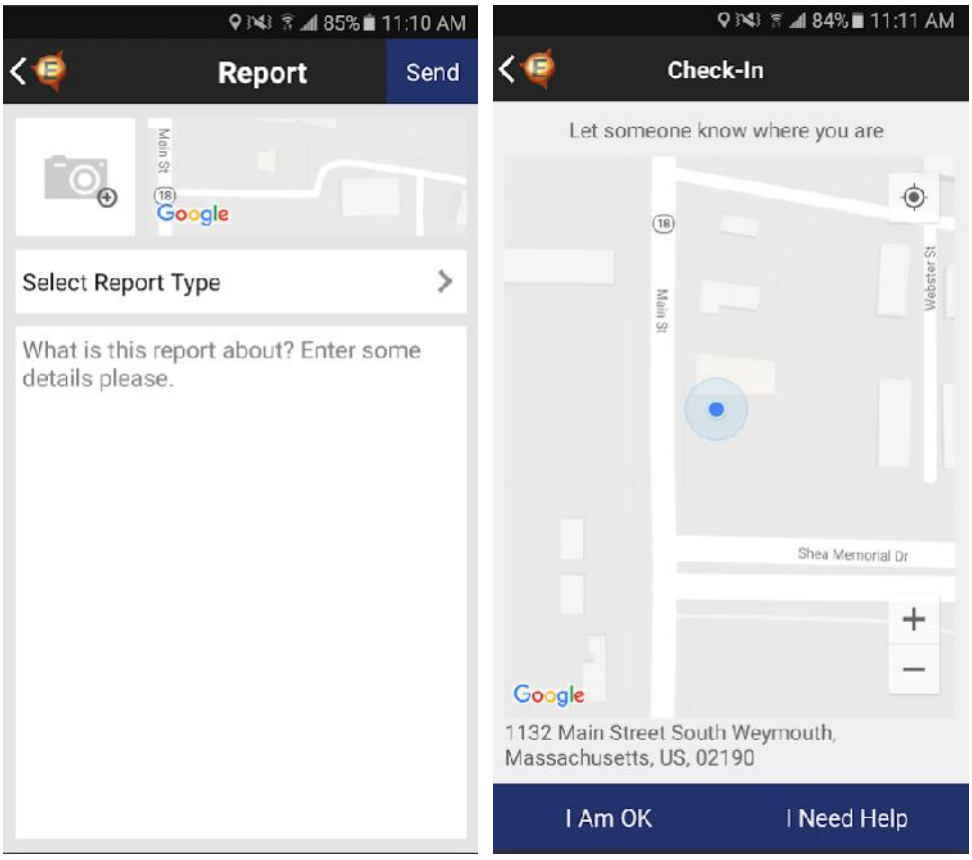


Figura 3.7 Elerts See Say Report / Check-in

[20] https://play.google.com/store/apps/details?id=com.elerts.elertscampus&hl=ro

Health Tap[21] este o platformă medicală disponibilă 24 de ore din 24, 7 zile din 7, accesibilă de pe orice dispozitiv cu conexiune la internet.

Utilizatorii pot interacționa cu doctorii printr-un chat live sau printr-o convorbire virtuală audio / video. De asemenea, se poate realiza un schimb de fișiere și există posibilitatea vizionării dosarului medical personal, incluzând aici și concluziile consulației virtuale.

Astfel se poate realiza întregul proces necesar unei consultații virtuale, de la diagnosticare, la investigații și plan de tratament.

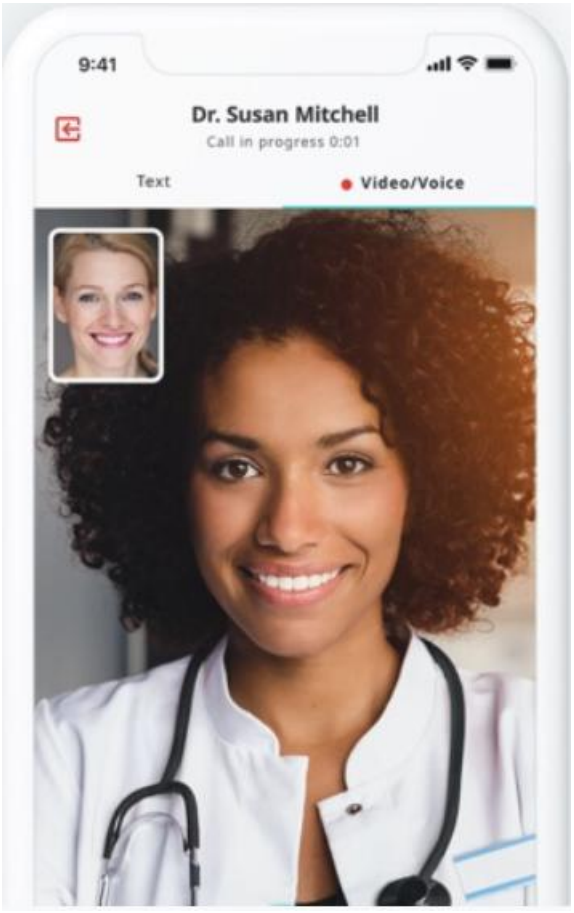


Figura 3.8 Consultație virtuală HealthTap

c-Now[22] este o aplicație mobilă destinată eficientizării gestionării urgențelor. Utilizatorii au posibilitatea de a începe o convorbire video live cu personalul specializat care face parte din ecosistemul în care se află și aplicația c-Now.

[21] https://www.healthtap.com/

[22] https://play.google.com/store/apps/details?id=com.reporty.reporty&hl=en_US&fbclid=IwAR3f_hEOBJrVhYKYx8KQSBYWiGi7TfBlmSgCG-66s6vTlRnj4ytzma3cDLs

Aplicația oferă ca alternativă și un chat live, în cazul în care convorbirea video nu este posibilă. Locația curentă utilizatorului poate fi accesată de personalul specializat.

O altă funcționalitate a aplicației o reprezintă selectarea unor contacte care vor fi notificate când utilizatorul raportează o urgență. De asemenea, aplicația oferă și o hartă live care afișează toate urgențele raportate în apropiere.

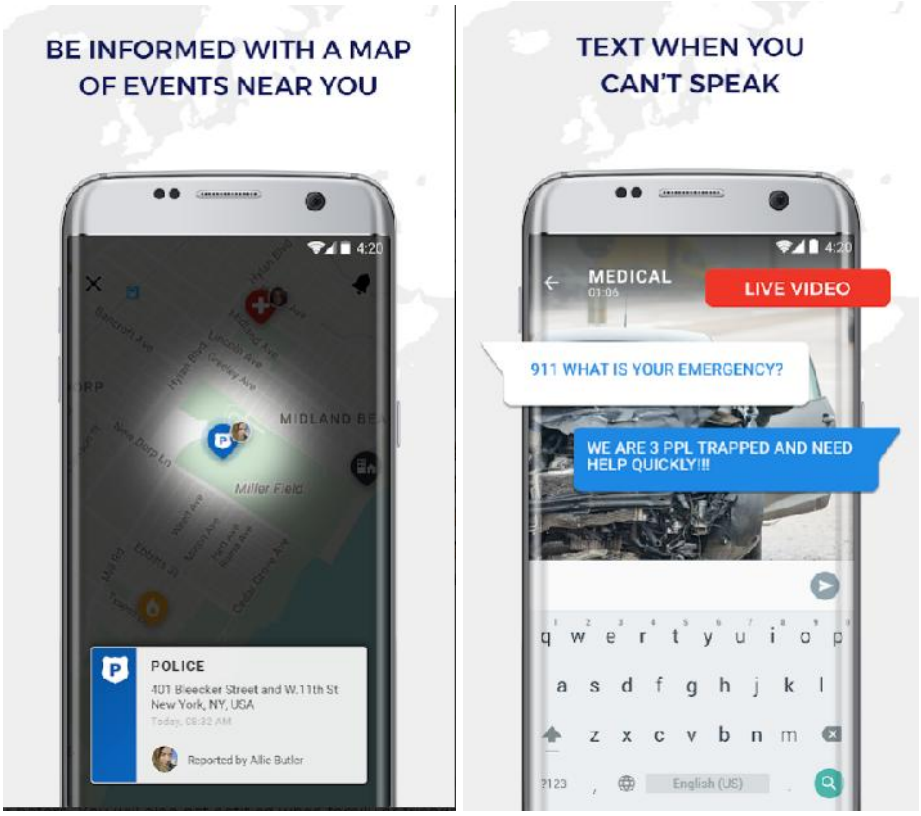


Figura 3.9 UI c-Now

3.4.2. Analiză comparativă

Cele trei aplicații prezentate mai sus fac parte din domeniul medical şi oferă câteva funcționalități similare cu cele prezentate de sistemul MHealth. Tabelul 2 prezintă diferențele între sistemul MHealth și aplicațiile anterior menționate.

Astfel, se poate observa faptul că sistemul MHealth este mult mai complex şi complet comparativ cu celelalte.

Modulul personal al sistemului MHealth conține funcționalități pentru utilizatorul doctor care sunt oferite parțial doar de HealthTap. De asemenea, componenta mobile prezintă o securitate sporită folosind date biometrice, beneficiu care nu se regăsește la celelalte aplicații. O altă diferență semnificativă o reprezintă modulul gestionării urgențelor în cadrul componentei web. Procesarea automată a informațiilor trimise de pe aplicația mobilă nu este oferită de nicio altă aplicație.

Tabel 3.1 Comparație între MHealth și aplicații similare

Funcționalitate	Include	ELERTS See Say	Health Tap	c-Now	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 (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 programări doctor	×	✓	×	✓
	Adăugare programări pentru pacient	×	✓	×	✓
Distribuire fișiere între doctor și pacient		×	✓	×	✓
Gestionare perioadă concediu	Programare concediu	×	×	×	✓
	Alegere înlocuitor	×	×	×	✓
	Disponibilitate în caz de urgență	×	×	×	✓

Capitolul 4. Analiză şi Fundamentare Teoretică

4.1. Arhitectura conceptuală

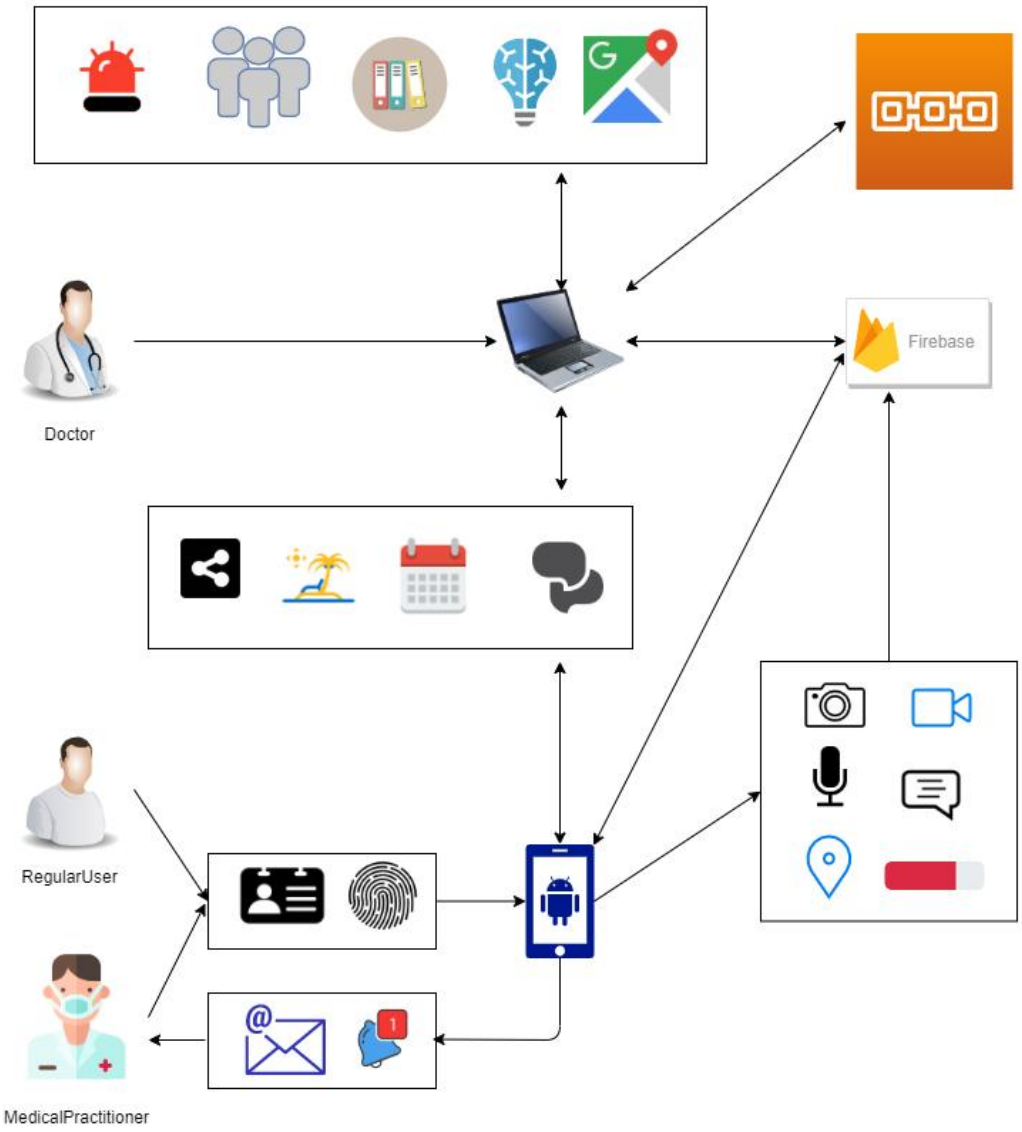


Figura 4.1 Arhitectura conceptuală a sistemului

În figura 4.1 este prezentată arhitectura conceptuală a sistemului. Se pot remarca cele trei tipuri de utilizatori: doctor, utilizator obişnuit şi utilizator cu pregătire medicală.

De asemenea, sistemul implementat conține două componente de bază: o componentă mobilă și o componentă web. Cele două interacționează prin accesul la datele comune și la baza de date, acestea fiind salvate exclusiv pe Cloud folosind platforma Firebase. Componenta mobilă este disponibilă pentru dispozitivele cu sistem

de operare Android cu versiunea minimă 5.1, în timp ce componenta web este găzduită pe Cloud pentru o performanță sporită și disponibilitate continuă.

În arhitectura conceptuală este prezentat și faptul că atât aplicația mobilă cât și aplicația web au un modul comun, cel personal.

Din punctul de vedere al doctorului, secțiunea personală a sistemului oferă posibilitatea gestionării concediilor, gestionării programărilor, schimbul de fișiere, dar și posibilitatea de comunicare prin chat. Din punctul de vedere al utilizatorului obișnuit îi permite schimbul de fișiere, crearea sau anularea programărilor, dar și comunicarea prin chat.

Al doilea modul prezentat în arhitectura conceptuală constă în modulul gestionării urgențelor. În cadrul componentei web, acest modul este reprezentat de vizualizarea urgențelor active, vizualizarea utilizatorilor cu pregătire medicală care participă la o anumită urgență, istoricul urgențelor, procesarea automată a datelor unei urgențe (procesări audio, foto, video, text) și vizualizarea pe o hartă interactivă a urgențelor active. De asemenea, acest modul este responsabil și de mutarea datelor despre urgențele finalizate de pe platforma Firebase în blockchain.

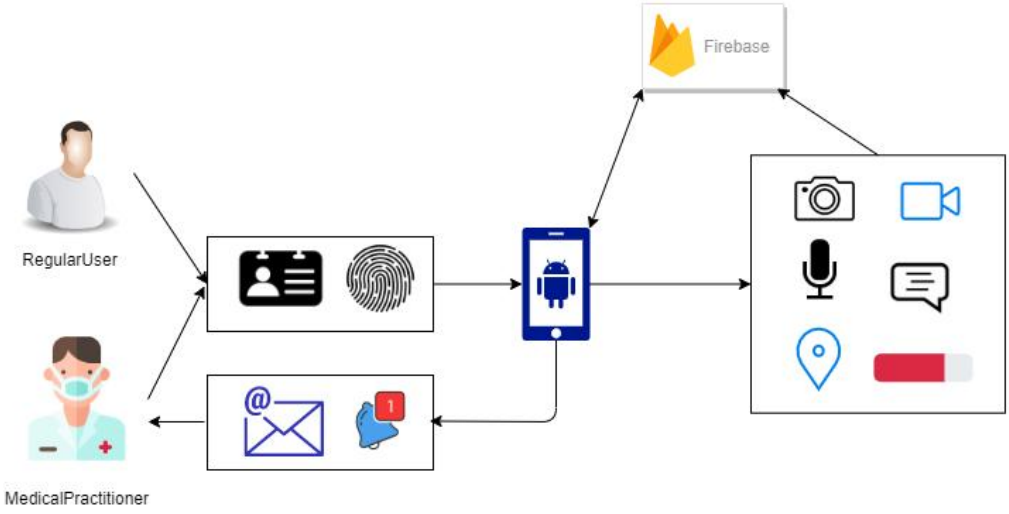


Figura 4.2 Arhitectura conceptuală a componentei implementate

În figura 4.2 este prezentată arhitectura conceptuală a componentei implementate. În această componentă sunt implicați doi actori: utilizatorul obișnuit și utilizatorul cu pregătire medicală.

Ambele tipuri de utilizatori interacționează inițial cu aplicația pe partea de securitate. Prima utilizare a aplicației redirecționează utilizatorul către activitatea de creare cont, unde va fi nevoit să realizeze o fotografie cu cardul de identitate. Datele necesare vor fi extrase automat din fotografie. Procesul de autentificare în aplicație se realizează folosind senzorul de amprentă prezent pe dispozitiv.

După acest pas, utilizatorul obișnuit are posibilitatea de a raporta o urgență. Informațiile pe care el le poate furniza sunt de mai multe tipuri: conținut foto, conținut

audio, conținut video, descriere text sau grad de severitate. Locația utilizatorului este trimisă în mod automat.

Utilizatorul cu pregătire medicală va primi o notificare cu locația respectivă în cazul unei urgențe în apropiere, moment în care el va putea prelua urgența și va primi un email cu toate informațiile necesare.

4.2. Cerințe de sistem

În această secțiune se vor descrie cerințele sistemului care duc la îndeplinirea obiectivelor setate inițial. Astfel, cerințele de sistem au fost împărțite în trei categorii: cerințe funcționale, cerințe non-funcționale si cerințe tehnologice.

4.2.1. Cerințe funcționale

Cerințele funcționale ale unui sistem descriu atât modul în care utilizatorii pot interacționa cu aplicația realizată, cât și modul în care sistemul răspunde la acțiunile efectuate. Cerințele au fost descrise din perspectiva ambelor tipuri de utilizatori: utilizator obișnuit si utilizator cu pregărire medicală.

CF1 – Autentificare: atât utilizatorul obișnuit cât și utilizatorul cu pregătire medicală au posibilitatea de a se autentifica în aplicație folosind senzorul de amprentă prezent pe dispozitivul mobil

CF2 – Creare cont: la prima utilizare a aplicației, utilizatorul este redirecționat către pagina de creare cont unde datele necesare sunt selectate automat după procesul de realizare a unei poze cu cardul de identitate

CF2.1 – Selectare tip de utilizator: după ce datele preluate sunt acceptate, se alege tipul de utilizator (utilizator obișnuit sau utilizator cu pregătire medicală) înainte ca procesul de creare a contului să fie finalizat

CF3 – Raportare urgenta: utilizatorul obișnuit are posibilitatea de a raporta o urgență

CF3.1 – Adăugare fotografie: utilizatorul obișnuit are posibilitatea de a realiza și adăuga conținut foto în cadrul raportului

CF3.2 – Adăugare videoclip: utilizatorul obișnuit are posibilitatea de a realiza și adăuga conținut video în cadrul raportului

CF3.3 – Adăugare secvență audio: utilizatorul obișnuit are posibilitatea de a realiza și adăuga conținut audio în cadrul raportului

CF3.4 – Adăugare descriere: utilizatorul obișnuit are posibilitatea de a realiza și adăuga conținut text în cadrul raportului

CF3.4.1 – Traducere descriere: descrierea adăugată de utilizator este tradusă în mod automat în limba engleză în momentul raportării urgenței

CF3.5 – Adăugare grad de severitate: utilizatorul obișnuit are posibilitatea de a seta gradul de severitate al urgenței printr-un cursor în cadrul raportului

CF3.6 – Adăugare locație: locația în care se află utilizatorul la momentul raportării urgenței se adaugă în mod automat

CF3.7 – Selectare tip de urgență: utilizatorul obișnuit are posibilitatea de a seta tipul de urgență (urgență personală sau urgență publică) în cadrul raportului

CF4 – Primire notificări: atât utilizatorul obișnuit cât și utilizatorul cu pregătire medicală vor primi o notificare cu locația unei urgențe din apropiere în cazul în care petrec cel puțin 30 de secunde în aria de acoperire a urgenței

CF5 – Preluare urgență: după primirea notificării, utilizatorul cu pregătire medicală poate prelua urgența, moment în care i se va cere o adresă de email pe care va primi toate informațiile legate de aceasta

4.2.2. Cerințe non-funcționale

Cerințele non-funcționale ale unui sistem descriu caracteristicile și modul de proiectare din punctul de vedere al aplicației și nu modul în care acesta funcționează din perspectiva utilizatorului cum este în cazul cerințelor funcționale.

Disponibilitatea – reprezintă perioada de timp în care sistemul poate fi folosit în mod corect de către utilizator.

Considerând că aplicația are ca scop principal oferirea posibilității de a raporta o urgență și detaliile ei sau de a fi notificat în cazul unei urgențe raportate în apropiere, ambele cazuri fiind direct influențate de disponibilitea aplicației, sistemul a fost proiectat astfel încât obiectivul de a fi disponibil utilizatorului 24 de ore din 24 să fie unul fezabil. În acest sens, toleranța sistemului și modul în care acesta reacționează la diferite anomalii sau la introducerea unor date eronate au fost principiile definitorii care au stat la baza dezvoltării aplicației.

De asemenea, serviciul care este responsabil de procesul de trimitere a notificărilor este activ și dacă aplicația este oprită, iar în cazul opririi sau repornirii dispozitivului mobil, serviciul va reporni automat după încheierea procesului de pornire a sistemului de operare.

Utilizabilitate – reprezintă ușurința cu care un utilizator poate realiza acțiunile pentru care folosește aplicația. Este o cerință non-funcțională extrem de importantă în ceea ce privește interacțiunea utilizatorului cu sistemul implementat.

Aplicația își propune ca designul interfeței utilizator să fie cât mai intuitivă, ușor de învățat și eficientă. În acest scop, aspectul aplicației este reprezentat de secțiuni proporționale care constau doar în butoane care sunt simbolizate prin imagini sugestive. Astfel, se poate duce la bun sfârșit un scenariu de utilizare într-un mod rapid și prietenos.

Această cerință non-funcțională a fost un detaliu de implementare foarte important, luând în considerare contextul aplicației și importanța timpului de răspuns în gestionarea urgențelor.

Integritatea datelor – reprezintă acuratețea datelor și siguranța nealterării lor în perioada în care ele sunt păstrate. Este o cerință non-funcțională de bază din punctul de vedere al siguranței, datele utilizatorului fiind un subiect extrem de sensibil în contextul aplicațiilor din domeniul medical.

În ceea ce privește siguranța datelor, sistemul este proiectat astfel încât datele transmise în momentul raportării unor urgențe să fie salvate într-o platformă de stocare a datelor în cloud, accesul la date și posibilitatea modificării lor fiind restricționat prin configurarea platformei.

Din punctul de vedere al preciziei datelor și al conformității lor la cerințele funcționale, sistemul afișează rezultatele tuturor procesărilor automate, astfel încât utilizatorul este nevoit să confirme corectitudinea informațiilor prelevate. Un bun exemplu în acest sens este cazul creării unui cont.

Din cauza posibilei calități reduse a fotografiei datorată unui senzor mai puțin performant, luminozității neadecvate sau a distanței neadecvate, sistemul întreabă utilizatorul dacă datele obținute de pe cardul de identitate sunt corecte și doar în caz afirmativ acestea sunt folosite.

Securitatea – este de departe cea mai importantă cerința non-funcțională implementată în sistemul prezentat. Contextul aplicației impune ca acest aspect al proiectării să fie documentat și realizat temeinic.

În acest sens, aplicația conține două tehnici de bază care susțin această idee. Crearea contului folosind o fotografie cu cardul de identitate asigură corectitudinea informațiilor despre fiecare utilizator al aplicației.

Unicitatea contului în cadrul instanței unei aplicații și faptul că acest cont nu este modificabil după creare oferă securitate din punctul de vedere al sursei în cazul raportării unei urgențe. În același timp, procesul de autentificare în sistem se realizează pe baza datelor biometrice, mai exact pe baza amprentei, fapt ce asigură că persoana care raportează o urgență și cea care este reprezentată prin contul creat sunt de fapt una și aceeași.

4.3. Cazuri de utilizare

Un caz de utilizare descrie modul prin care utilizatorul unei aplicații interacționează cu sistemul implementat pentru a îndeplini un obiectiv. Această descriere este reprezentată printr-o listă de evenimente sau de pași care pot fi urmați pentru a realiza cu succes scopul propus.

Cazurile de utilizare ale unui sistem se împart în funcție de tipurile de utilizatori ce au acces în cadrul aplicației. Acești utilizatori poartă numele de actori.

Sistemul MHealth prezintă trei tipuri de utilizatori: doctor, utilizator obișnuit și utilizator cu pregătire medicală. Componenta mobilă are doi actori: utilizatorul obișnuit și utilizatorul cu pregătire medicală.

4.3.1. Cazuri de utilizare pentru utilizator obișnuit

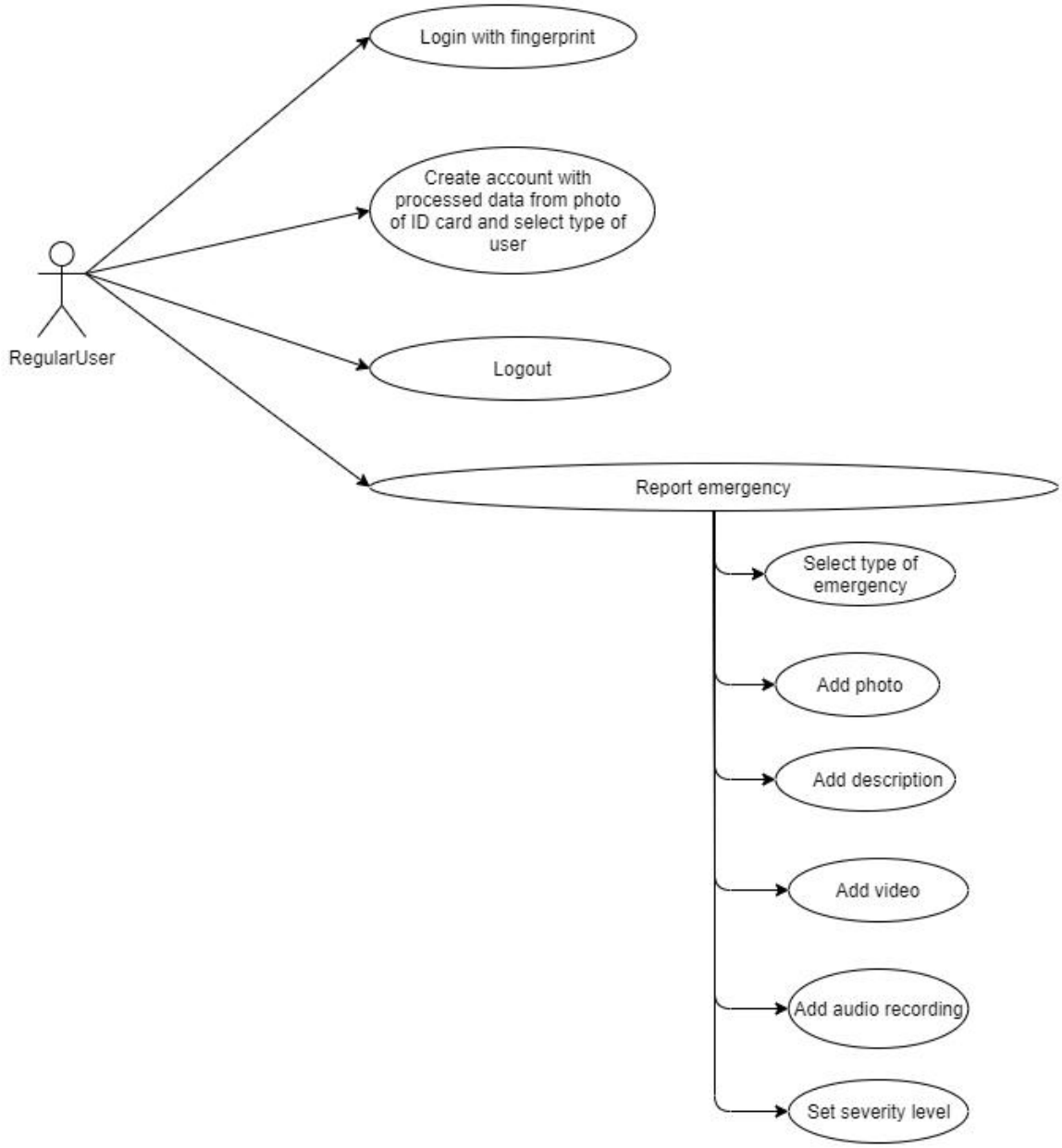


Figura 4.3 Cazuri de utilizare pentru utilizator obișnuit

În figura 4.3 sunt prezentate cazurile de utilizare are utilizatorului obișnuit, utilizator cu acces la aplicația mobilă.

Cazul de utilizare 1

Descriere: **Creare cont**

Precondiții:

1) Utilizatorul are conexiune la internet.
2) Nu există un cont creat pentru acest dispozitiv.

Postcondiții:

1) Contul este creat cu succes.
2) Actorul este redirecționat către pagina de login.

Scenariul de succes:

1) Utilizatorul pornește aplicația pentru a realiza anumite acțiuni.
2) Aplicația redirecționează utilizatorul către pagina de creare cont, după ce nu a fost găsit un cont creat pentru acest dispozitiv.
3) Utilizatorul va realiza o fotografie cu cardul de identitate.
4) După ce fotografia este acceptată de utilizator, datele necesare creării contului sunt extrase automat și afișate.
5) Actorul confirmă corectitudinea datelor și alege cărui tip de utilizator va fi atribuit contul.
6) Contul utilizatorului este creat cu succes și acesta este redirecționat către pagina de login.

Scenarii alternative:

5) Datele extrase de pe fotografia cu cardul de identitate nu sunt acceptate de către utilizator. În acest caz, utilizatorul are posibilitatea de a realiza o nouă fotografie, procesul de extragere a datelor fiind astfel repetat.
6) Utilizatorul nu are o conexiune la internet. În acest caz aplicația nu realizează cu succes procesul de creare a contului, iar utilizatorul trebuie să asigure refacerea conexiunii la internet.

Cazul de utilizare 2

Descriere: **Autentificare**

Precondiții:

1) Există un cont creat pentru acest dispozitiv.
2) Dispozitivul are cel puțin o amprentă înregistrată ca metodă de securitate.

Postcondiții:

1) Actorul este redirecționat la meniul principal corespunzător tipului de utilizator atribuit contului înregistrat pe dispozitiv.

Scenariul de succes:

1) Utilizatorul pornește aplicația pentru a realiza anumite acțiuni.
2) Aplicația redirecționează utilizatorul către pagina de login, după ce a fost găsit un cont creat pentru acest dispozitiv.
3) Utilizatorul acționează senzorul de amprentă.
4) Amprenta scanată este identică cu cel puțin o amprentă înregistrată pe dispozitiv.
5) Actorul este redirecționat la meniul principal în funcție de tipul de utilizator atribuit contului înregistrat pe dispozitiv.

Scenarii alternative:

4.1) Dispozitivul nu are cel puțin o amprentă înregistrată ca metodă de securitate. În acest caz, la acționarea senzorului, aplicația va afișa un mesaj de atenționare.

4.2) Amprenta scanată nu are un echivalent printre amprentele înregistrate în dispozitiv. Acest caz este la rândul lui evidențiat printr-un mesaj de atenționare corespunzător. Cauza acest scenariu se poate regăsi printre următoarele: degetul folosit pentru scanare nu este cel potrivit, senzorul nu a fost acoperit pe o arie îndeajuns de mare sau degetul trebuie menținut pe senzor pe o perioadă mai lungă de timp.

Cazul de utilizare 3

Descriere: **Logout**

Precondiții:

1) Utilizatorul este autentificat în cadrul aplicației.
2) Actorul se află pe meniul principal.

Postcondiții:

1) Aplicația redirecționează utilizatorul înapoi la pagina de login și nu există posibilitatea de a accesa altă pagină înainte de a se realiza cu succes procesul de autentificare.

Scenariul de succes:

1) Utilizatorul este autentificat în cadrul aplicației și se află pe pagina cu meniul principal.
2) După acționarea butonului de logout, utilizatorul este redirecționat către pagina de login.
3) Acționarea butonului fizic de întoarcere a dispozitivului fizic nu readuce utilizatorul la meniul principal.

Scenarii alternative:

- ținând cont de natura aplicației și de modul în care acest caz de utilizare modifică comportamentul sistemului, nu sunt prezente scenarii alternative

Cazul de utilizare 4

Descriere: **Raportarea unei urgențe**

Precondiții:

1) Actorul este autentificat în cadrul aplicației.
2) Utilizatorul are conexiune la internet.
3) Serviciile de localizare sunt activate.

Postcondiții:

1) Urgența este raportată cu succes și conține toate informațiile furnizate de utilizator.
2) Aplicația redirecționează utilizatorul către meniul principal.

Scenariul de succes:

1) Utilizatorul este autentificat în cadrul aplicației și accesează opțiunea de raportare urgențe.
2) Actorul selectează opțiunea de adăugare descriere și completează câmpul cu informații text. Această descriere poate fi modificată în orice moment înainte de raportarea urgenței, ultima descriere realizată fiind cea trimisă. După realizarea descrierii, utilizatorul este redirecționat către pagina de raportare urgențe.
3) Actorul selectează opțiunea de adăugare fotografie și începe procesul de realizare conținut foto. După fiecare fotografie realizată, utilizatorul este întrebat dacă este de acord ca respectiva fotografie să fie folosită, iar în caz afirmativ este redirecționat către pagina de raportare urgențe.
4) Actorul selectează opțiunea de adăugare videoclip și începe procesul de realizare conținut video. După fiecare videoclip realizat, utilizatorul este întrebat dacă este de acord ca respectivul videoclip să fie folosit, iar în caz afirmativ este redirecționat către pagina de raportare urgențe.
5) Actorul selectează opțiunea de adăugare înregistrare audio și începe procesul de realizare conținut audio. La sfârșitul înregistrării audio, utilizatorul este redirecționat către pagina de raportare urgențe.
6) Actorul modifică cursorul care semnifică gradul de severitate de la valoarea inițială de 50, la orice valoare din intervalul 1 – 100.
7) Coordonatele GPS ale utilizatorului sunt preluate în mod automat de către dispozitiv.
8) După acționarea butonului de raportare urgențe, aceasta este înregistrată cu succes și utilizatorul este redirecționat către meniul principal.

Scenarii alternative:

7) Serviciile de localizare nu sunt activate. În acest caz, urgența va fi raportată fără coordonate gps.

8) Utilizatorul nu are o conexiune la internet. În acest caz aplicația nu realizează cu succes procesul de raportare a urgenței, iar utilizatorul trebuie să asigure refacerea conexiunii la internet.

4.3.2. Cazuri de utilizare pentru utilizator cu pregătire medicală

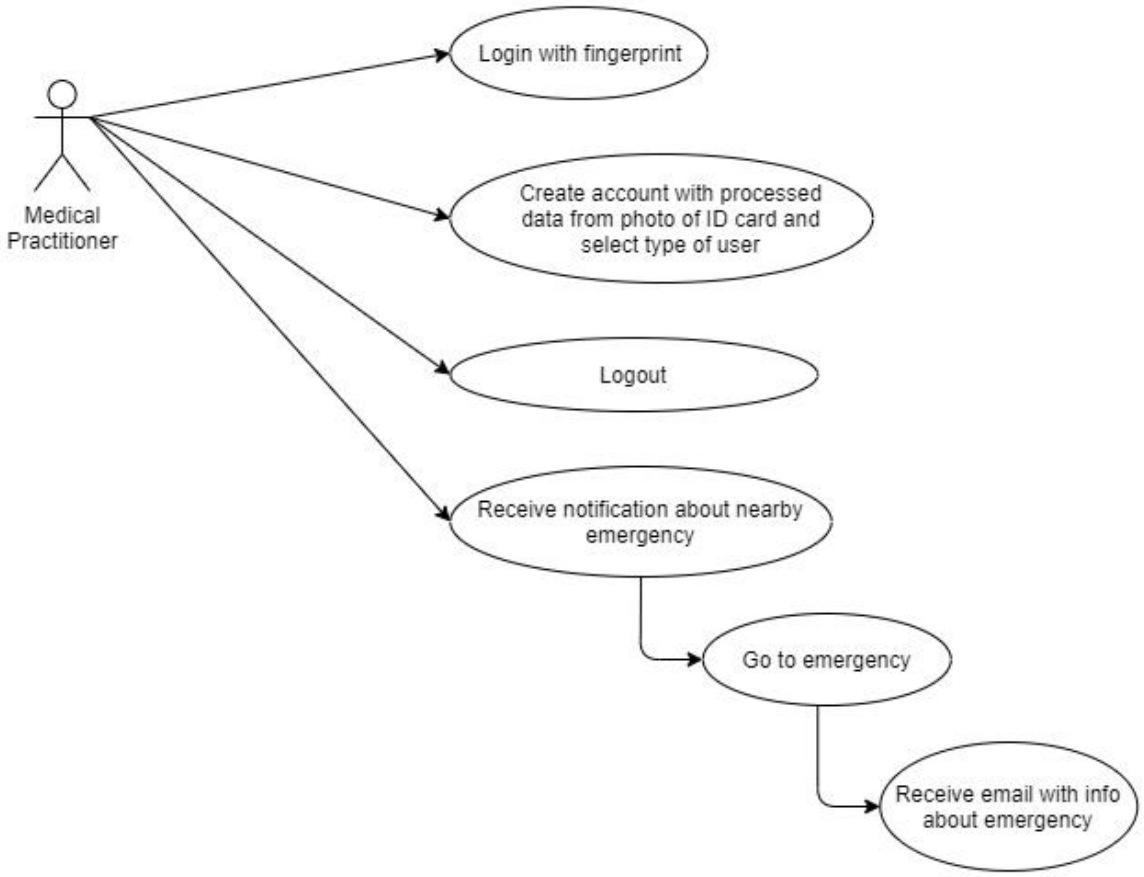


Figura 4.4 Cazuri de utilizare pentru utilizator cu pregatire medicală

În figura 4.4 sunt prezentate cazurile de utilizare are utilizatorului cu pregătire medicală, utilizator cu acces la aplicația mobilă. Primele trei cazuri de utilizare sunt asemănătoare cu primele trei cazuri ale utilizatorului obișnuit, iar din acest motiv ele nu vor fi descrise și pentru acest actor.

Cazul de utilizare 1

Descriere: **Primire notificare legată de o urgență în apropiere**

Precondiții:

1) Utilizatorul are conexiune la internet.
2) Serviciile de localizare sunt activate.
3) Serviciile de căutare urgențe în apropiere sunt activate.

Postcondiții:

1) Actorul primește o notificare cu adresa exactă a unei urgențe în apropiere.

Scenariul de succes:

1) Aplicația nu este deschisă și toate serviciile funcționează.
2) Utilizatorul se află timp de cel puțin 30 de secunde în raza de 1000m a urgenței.
3) Actorul primește o notificare cu adresa exactă a urgenței.

Scenarii alternative:

1.1) Utilizatorul nu are o conexiune la internet. În acest caz aplicația nu realizează cu succes procesul de căutare a urgențelor din apropiere, iar utilizatorul trebuie să asigure refacerea conexiunii la internet.

1.2) Serviciile de căutare urgențe în apropiere nu sunt activate. Acest scenariu poate fi datorat repornirii recente a dispozitivului mobil, serviciile neapucând să fie încă repornite la rândul lor.

2) Serviciile de localizare nu sunt activate. În acest caz, aplicația nu poate verifica dacă utilizatorul îndeplinește condițiile pentru a primi notificarea.

Cazul de utilizare 2

Descriere: **Preluare urgență**

Precondiții:

1) Utilizatorul are conexiune la internet.
2) A fost primită o notificare legată de o urgență în apropiere.

Postcondiții:

1) Urgența a fost preluată.
2) Actorul a primit un email cu toate informațiile în legătură cu urgența preluată.

Scenariul de succes:

1) Actorul a primit o notificare cu adresa exactă a unei urgențe în apropiere.
2) La selectarea notificării, se deschide aplicația.
3) Dacă nu este găsit un cont pentru dispozitiv, aplicația redirecționează utilizatorul către pagina de creare cont.
4) Actorul este redirecționat către pagina de login.
5) După autentificare, utilizatorului i se cere să introducă o adresă de email pentru a primi toate informațiile legate de urgența preluată.
6) După acționarea butonului de preluare urgență, aceasta este înregistrată cu succes și utilizatorul primește emailul respectiv.

Scenarii alternative:

1) Utilizatorul nu are o conexiune la internet. În acest caz aplicația nu realizează cu succes procesul de căutare a urgențelor din apropiere, iar utilizatorul trebuie să asigure refacerea conexiunii la internet.
5) Adresa de email introdusă de utilizator nu este validă. În acest caz, utilizatorul nu primește informațiile despre urgență.

4.4. Perspectivă tehnologică

Componenta mobilă a sistemului MHealth a fost implementată utilizând un număr semnificativ de tehnologii. Fiecare dintre aceste tehnologii a fost integrată folosind ultima versiune stabilă lansată, dat fiind faptul că un factor decisiv în funcționarea corectă a sistemului este compatibilitatea între tehnologii. Modul în care acestea interacționează este prezentat în figura 4.5.

Figura 4.5 Tehnologiile utilizate

4.4.1. Android

Android este un sistem de operare specializat pentru dispozitivele mobile, dezvoltat de Open Handset Alliance și cumpărat de Google în 2005, lansat prima oară pentru dispozitivele mobile în 2007. Este cel mai bine vândut sistem de operare pentru dispozitive mobile inteligente din 2011, și din 2013 pentru tablete.

Google lansează o versiune nouă de Android în fiecare an, ultima versiune fiind Android 10. S-a ales realizarea unei aplicații mobile folosind sistemul de operare Android ținând cont de răspândirea acestuia pe dispozitivele de tip smartphone. Astfel, la nivel mondial[23], sunt 1.6 miliarde de utilizatori Android ceea ce reprezintă un procent de 74.13% din numărul total de dispozitive mobile.

Un alt avantaj al acestui sistem de operare este faptul că oferă un IDE bazat pe cunoscutul IntelliJ IDEA, IDE ce permite dezvoltarea aplicațiilor folosind limbajul de programare Java pentru backend și XML pentru frontend, acomodarea la acest set de tehnologii fiind facilă. De asemenea, în ceea ce privește dezvoltarea ulterioară a sistemului, utilizarea acestui sistem de operare face ca tranziția aplicației către alte dispozitive (ceasuri inteligente, tablete) să fie una naturală fără ca multe modificări în arhitectura aplicației să fie necesare.

4.4.2. Android Studio

Android Studio[24] este IDE-ul (Integrated Development Environment) oficial pentru sistemul de operare Android, deținut de cei de la Google. Este bazat pe IntelliJ IDEA, IDE-ul celor de la JetBrains.

Dacă în ceea ce privește backendul aplicației, limbajele de programare utilizare cel mai des sunt Java și Kotlin (IDE-ul oferă suport și pentru C++), Android Studio abordează secțiunea de frontend într-un mod diferit de cele mai populare IDE-uri. Astfel, limbajul XML este cel utilizat, fiind posibilă atât scrierea de cod pentru realizarea unei interfețe grafice prietenoase, cât și utilizarea unui editor[25] care oferă posibilitatea folosirii tehnicii *drag and drop*.

Printre calitățile principale ale acestui IDE se numără:

- Suport pentru Google Cloud Platform, astfel integrarea platformei Firebase în aplicație este una facilă
- Suport pentru GitHub, astfel procesul de mentenanță în cadrul aplicației este unul fluent
- Editorul de cod și uneltele destinate dezvoltatorilor preluate din IntelliJ, fapt ce complementează limbajul de programare Java
- Emulator al dispozitivelor Android, pentru o testare manuală rapidă
- Framework-uri și unelte specifice Android pentru testare automată

[23] https://www.statista.com/topics/876/android/
[24] https://developer.android.com/studio/intro
[25] https://developer.android.com/studio/write/layout-editor

4.4.3. Java

Java este un limbaj de programare orientat obiect deținut de Oracle, utilizat atât în dezvoltarea aplicațiilor Android, cât și a celor Web sau Desktop. Conform Github[26], Java este al treilea cel mai popular limbaj de programare.

Android SDK (Software Development Kit) include majoritatea librăriilor standard Java necesare, cât şi câteva librării speciale Android. Astfel, OpenJDK oferit la configurarea Android Studio este versiunea de JDK (Java Development Kit) recomandată, dar această versiune poate fi ulterior modificată.

Un avantaj al limbajului care a dus la utilizarea acestuia în dezvolarea aplicațiilor mobile folosind Android Studio este faptul că orice cod Java este compilat într-un cod de biți ce poate fi rulat în orice sistem cu suport pentru JVM (Java Virtual Machine).

4.4.4. Firebase

Firebase este o platformă cloud deținută de cei de la Google ce oferă servicii pentru dezvoltarea aplicațiilor mobile, atât Android, cât și iOS. Printre cele mai importante servicii Firebase se numără:

- Google Analytics
- Firebase Cloud Messaging
- Firebase Authentication
- Firebase Realtime Database
- Cloud Firestore
- Firebase Storage
- Firebase Hosting

Aplicația Android prezentată folosește două dintre servicii: Cloud Firestore şi Firebase Storage.

Cloud Firestore[27] este o bază de date NoSQL găzduită pe Cloud care poate fi accesată folosind Android SDK. Datele sunt stocate în documente care la rândul lor sunt stocate în colecții. Documentele suportă diferite tipuri de date de la numere sau stringuri la obiecte complexe. De asemenea, în documente se pot crea și subcolecții pentru a realiza structuri de date ierarhice. Acest model folosit de Cloud Firestore este descris în figura 4.6.

[26] https://octoverse.github.com/#top-languages
[27] https://firebase.google.com/docs/firestore

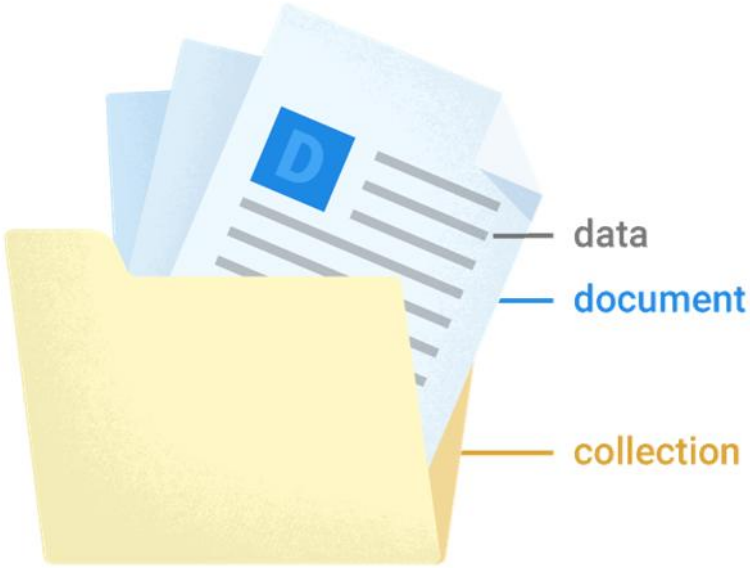


Figura 4.6 Modelul de date Cloud Firestore

Firebase Storage[28] este un serviciu de stocare și de descărcare a fișierelor direct de la client. Un avantaj al acestui serviciu este modul în care reacționează la conexiunea la internet a clientului. Astfel, dacă de exemplu conexiunea la internet se pierde, clientul poate continua descărcarea fișierelor din punctul în care a rămas atunci când conexiunea este refăcută, salvând astfel timp.

4.4.5. ML Kit

ML Kit[29] este un SDK deținut de cei de la Google care aduce API-urile machine learning pe dispozitivele mobile.

Firebase ML Kit SDK a fost inițial lansat ca o componentă a platformei Firebase, oferind atât API-uri găzduite pe Cloud, cât și API-uri găzduite pe dispozitivul mobil. În cele din urmă, produsul a fost împărțit în două produse separate: ML Kit, un SDK ce conține doar API-urile găzduite pe dispozitivul mobil și Firebase Machine Learning, utilizat pentru API-urile găzduite pe Cloud.

Un avantaj important al ML Kit este posibilitatea utilizării acestor servicii în mod offline. Printre aceste API-uri se numără:

- Scanarea unui cod de bare
- Detecția feței
- Detecția și urmărirea obiectelor
- Recunoașterea textului
- Identificarea limbii
- Traducerea textului

[28] https://firebase.google.com/docs/storage
[29] https://developers.google.com/ml-kit/guides

4.4.6. Geofencing

Geofencing[30] este API care se folosește atât de locația curentă a unui utilizator, cât și de locația unui punct de interes (Geofence).

Pentru crearea unui Geofence, trebuie specificate următoarele informații:

- Latitudine
- Longitudine
- Raza cercului care definește respectivul Geofence în metri
- Perioada de valabilitate
- Evenimentul de declanșare

Dacă celelalte informații sunt clare, în ceea ce privește evenimentul de declanșare lucrurile sunt mai complexe. Astfel, există 3 tipuri de evenimente: intrarea în aria acoperită de Geofence, ieșirea din aria acoperită de Geofence sau petrecerea unei perioade minime de timp specificată anterior.

În contextul sistemului MHealth, s-a folosit ca eveniment de declanșare petrecerea unei perioade de timp prestabilită pentru a evita situațiile în care notificarea este trimisă inutil. În figura 4.7 sunt evidențiate aceste caracteristici.

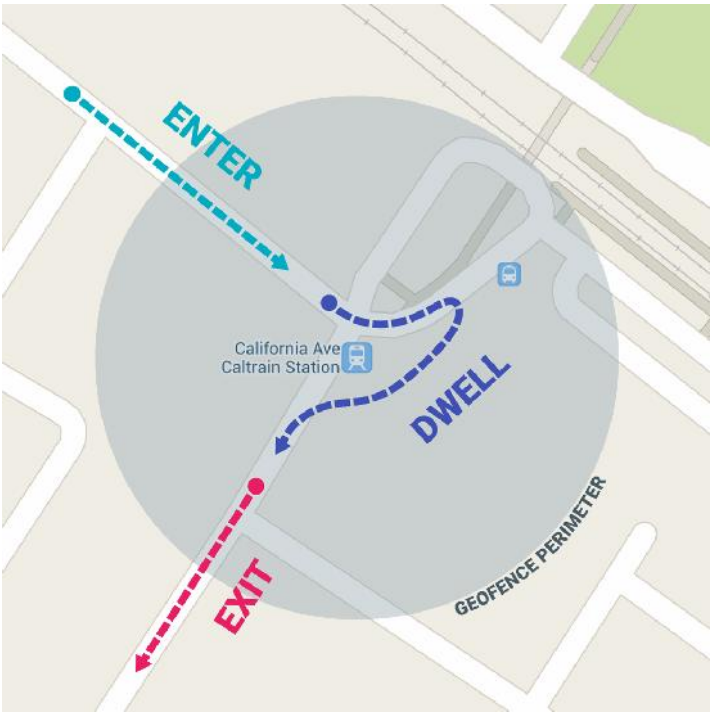


Figura 4.7 Aria și evenimentele de declanșare pentru un Geofence

[30] https://developer.android.com/training/location/geofencing

4.4.7. Geocoding

Geocoding[31] este API care se ocupă cu procesul de transformare a unei adrese sau a descrierii unei locații în coordonate (latitudine, longitudine).

Acest proces poate fi utilizat și în mod opus, transformând coordonate în adresa exactă a unei locații. În acest caz, precizia rezultatului poate varia de la întreaga adresă a celei mai apropiate clădiri, la numele orașului și codul poștal.

În contextul aplicației prezentate, s-a folosit Geocoding pentru afișarea adresei exacte a urgenței în cadrul notificării, dar și în cadrul activității care îi oferă posibilitatea utilizatorului cu pregătire medicală să preia urgența respectivă.

4.4.8. Git

Git[32] este un software open-source utilizat în urmărirea modificărilor de cod și în controlarea diferitelor versiuni ale unei aplicații. Sistemul pe care funcționează Git este motivul principal pentru care este atât de popular.

Astfel, codul sursă se află pe un server principal și fiecare developer folosește local o clonă a codului sursă. Acest concept oferă un avantaj din punctul de vedere al backup-ului, fiecare dintre aceste clone putând în orice moment să înlocuiască codul sursă aflat pe serverul principal. Figura 4.8 evidențiază acest concept.

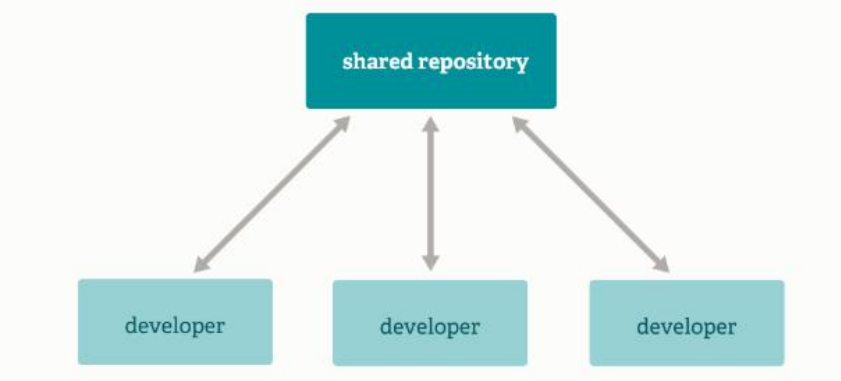


Figura 4.8 Separarea entităților[33]

Un alt avantaj oferit de sistemul folosit de Git este viteza. Fiecare modificare sau acțiune asupra codului se realizează local, nefiind necesară comunicarea cu un server. Totuși, acest fapt atrage și un dezavantaj, viteza cu care se realizează clona inițială fiind scazută considerând că fiecare clonă locală conține și tot istoricul codului sursă aflat pe serverul principal.

„Branching” este un concept care aduce Git încă un plus. Fiecare developer poate crea local un număr infinit de „branch-uri”, independente de alți developeri, fapt ce face ca gestionarea diferitelor funcționalități să se realizeze independent și încurajează experimentarea și testarea unor idei, fără ca alte versiuni ale codului sursă să fie afectate.

[31] https://developer.android.com/reference/android/location/Geocoder
[32] https://git-scm.com/about
[33] https://git-scm.com/about/distributed

4.4.9. GitHub Desktop

GitHub Desktop[34] este un client destinat în mod special utilizării funcționalităților oferite de Git.

Are scopul de a gestiona într-un mod ușor și prietenos proiectele în care Git este integrat. Utilizarea liniei de comandă nu mai este necesară, comenzile putând fi executate intuitiv prin intermediul acestui client Git.

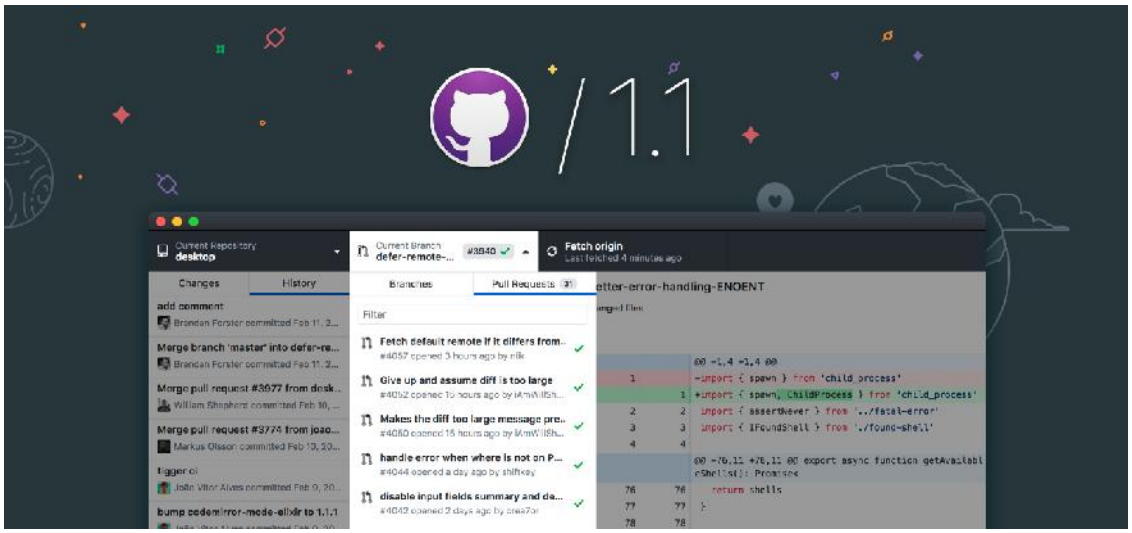


Figura 4.9 Interfața GitHub Desktop[35]

4.4.10. Espresso

Espresso[36] este un framework folosit în testarea UI. Acest framework oferă API-uri pentru scrierea testelor ce simulează interacțiunile unui utilizator cu aplicația.

Un avantaj al acestui framework este faptul că sincronizarea dintre acțiuni și UI-ul aplicației testate se realizează în mod automat. Espresso detectează când thread-ul principal este în starea idle, astfel comenzile din testul scris pot fi executate la momentul potrivit.

În cadrul aplicației prezentate a fost folosit Core API din cadrul framework-ului Espresso, motivul fiind prezența unor metode care pot facilita realizarea testelor propuse: testarea trecerii de la o activitate la alta, testarea conținutului unor componente sau testarea modificării valorii unor componente.

[34] https://github.blog/2015-08-12-github-desktop-is-now-available/
[35] https://github.blog/2018-02-26-github-desktop-1-1-is-now-available/
[36] https://developer.android.com/training/testing/espresso

Tabel 4.1 Tehnologii luate în considerare

Problematica	Sursa soluției	Evoluția tehnologiilor 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 pot filtra utilizatorii în funcție de distanța față de urgență **Geofencing API**
Traducere Text	Firebase	**ML Kit**
Translatare din coordonate în adresă și invers	-	**Geocoding API**
Stocare Fișiere	Firebase	**Firebase Storage**
Stocare date personale	Firebase	**Firebase Database**
Preluare date buletin	Firebase	**ML Kit**

Capitolul 5. Proiectare de Detaliu si Implementare

Capitolul 5 va face o prezentare a diagramelor reprezentative pentru aplicația implementată, cât și o descriere amănunțită a implementării realizate în cadrul acestei lucrări.

5.1. Diagrame reprezentative

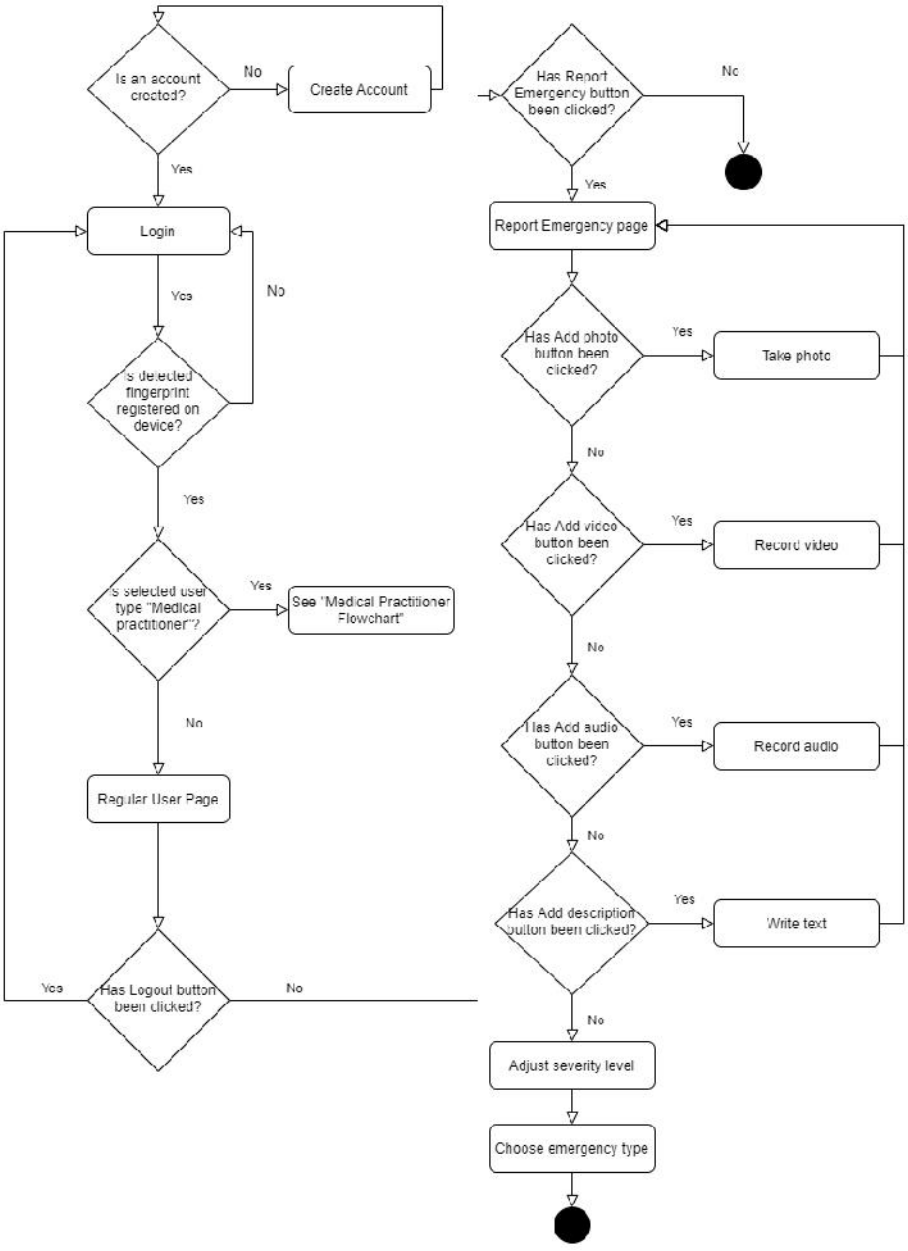


Figura 5.1 Flux de control pentru utilizatorul obișnuit

În figura 5.1 este prezentat fluxul de control pentru utilizatorul obișnuit. Astfel, la deschiderea aplicației se verifică dacă există un cont asociat cu dispozitivul folosit. În caz afirmativ, aplicația redirecționează utilizatorul către pagina de login. Dacă contul nu este găsit, utilizatorul este redirecționat către pagina de creare cont, doar ulterior către pagina de login. Utilizatorul are 3 opțiuni: raportarea unei urgențe, accesarea modulului personal sau logout. Dintre cele trei, lucrarea prezentată s-a ocupat de raportarea unei urgențe și logout. Raportarea unei urgențe îi va oferi utilizatorului o multitudine de opțiuni în ceea ce privește conținutul pe care îl poate adăuga raportului: text, audio, foto, video, grad de severitate sau tipul urgenței.

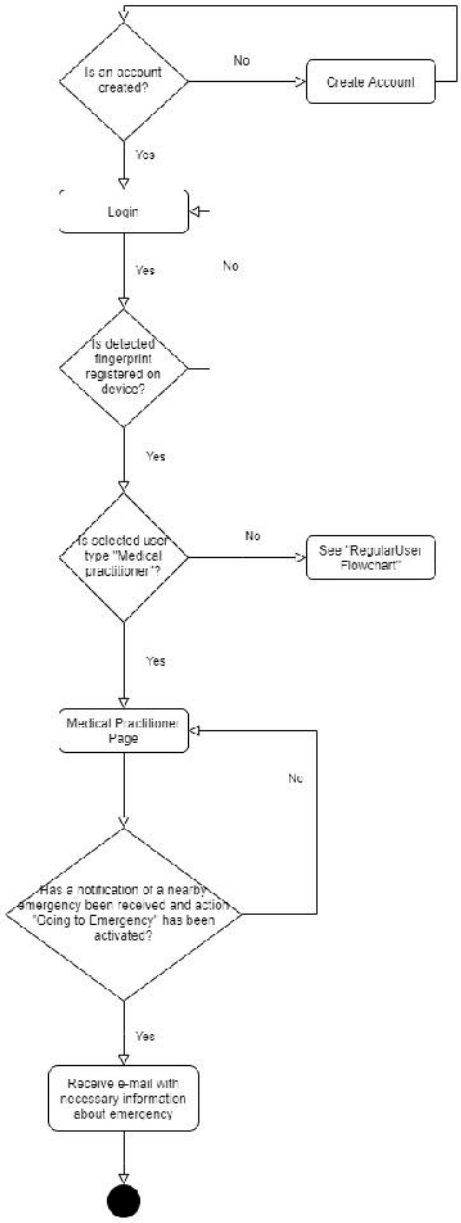


Figura 5.2 Flux de control pentru utilizatorul cu pregatire medicală

Figura 5.2 prezintă fluxul de control pentru utilizatorul cu pregătire medicală. Astfel, funcționalitățile oferite utilizatorului obișnuit pot fi accesate și de utilizatorul cu pregătire medicală. Diferența apare în momentul în care este primită o notificare de tip push. Accesarea ei de către un utilizator cu pregătire medicală duce către fluxul de început al aplicației, mai exact creare unui cont, dacă este necesar, și autentificare folosind senzorul de amprentă. Autentificarea cu succes duce utilizatorul către pagina de preluare urgență în care utilizatorul va trebui să ofere o adresă de email pentru a primi informațiile legate de urgența respectivă.

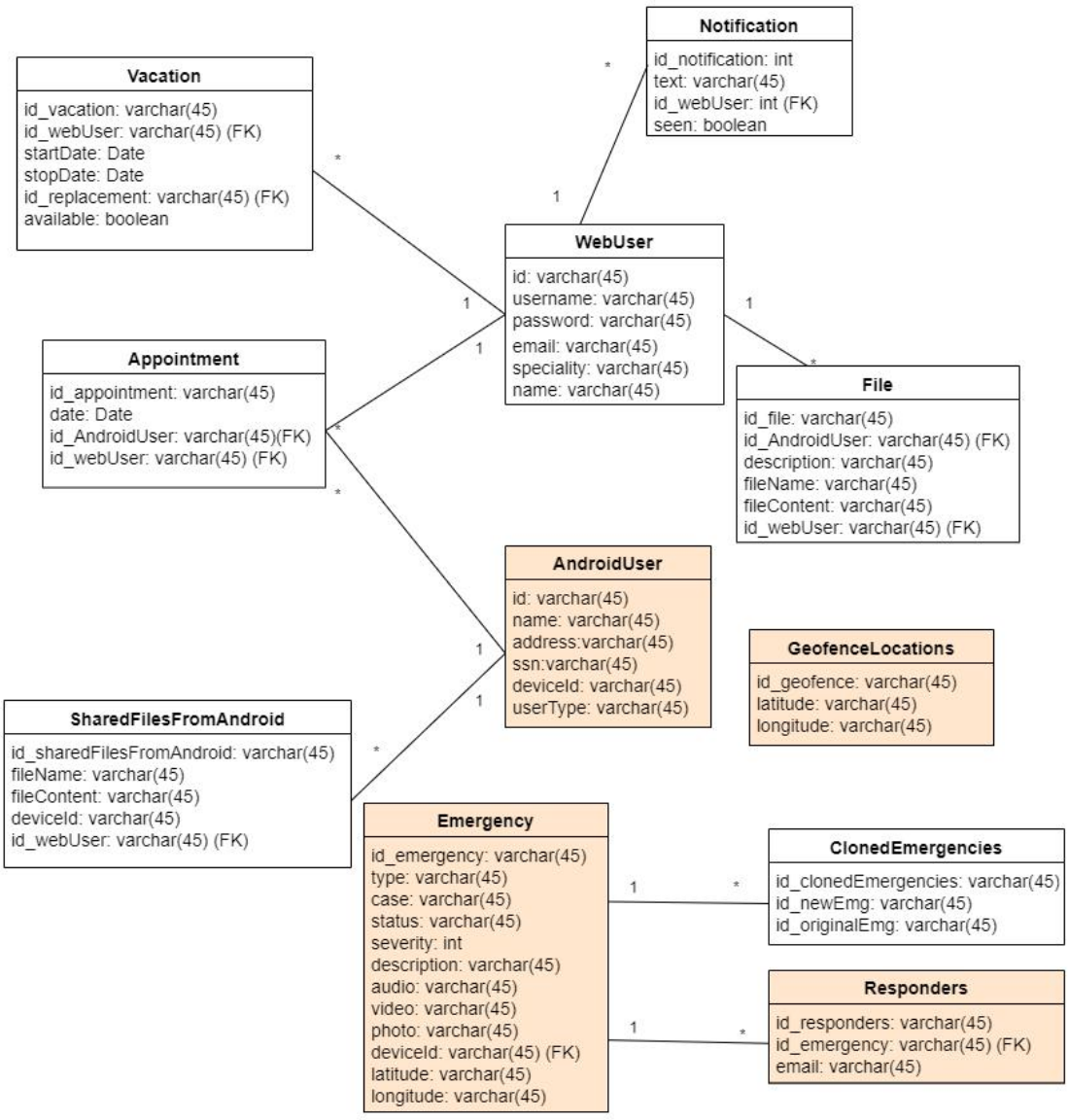


Figura 5.3 Diagrama bazei de date

În figura 5.3 este reprezentată structura bazei de date din cadrul sistemului MHealth. Contextul lucrării a dus la alegerea Firebase Cloud Firestore[37] ca soluție pentru găzduirea bazei de date.

Cloud Firestore oferă o bază de date NoSql găzduită pe Cloud, fiind succesorul Realtime Database. Oferă SDK native aplicațiile Android, iOS sau Web, fapt ce se potrivește perfect în contextul sistemului MHealth, integrarea celor două module realizându-se astfel facil.

De asemenea, un alt beneficiu vital al bazei de date NoSql Cloud Firestore este suportul pentru sincronizare și actualizări în timp real prin „Event Listeners", funcționalitate utilizată în special în raportarea și gestionarea urgențelor.

În cadrul aplicației Android, conexiunea la baza de date NoSql Cloud Firestore este realizată de asistentul Firebase[38] încorporat în IDE-ul Android Studio.

Baza de date este prezentată în figura 5.3 într-o forma convențională, relațională, doar cu scopul de a reprezenta clar legătura dintre elemente. În Anexa 5, baza de date este reprezentată folosind unealta Hackolade, specializată în ilustrarea bazelor de date de tip No-Sql.

Figura 5.3 evidențiază colecțiile care sunt utilizate în cadrul modulului Android:

- AndroidUser – colecție care conține conturile utilizatorilor aplicației mobile MHealth:
  - name: numele și prenumele utilizatorului, informație extrasă de pe buletin
  - address: adresa utilizatorului, informație extrasă de pe buletin
  - ssn: codul numeric personal al utilizatorului, informație extrasă de pe buletin
  - deviceId: id-ul unic al instanței aplicației, făcând astfel legătura între dispozitiv și contul utilizatorului
  - userType: utilizator obișnuit sau utilizator cu pregătire medicală
- GeofenceLocations – colecție care conține toate locațiile urgențelor active, conține coordonatele urgenței (latitudine și longitudine)
- Emergencies – colecție care conține toate urgențele raportate, indiferent de statusul lor:
  - type: publică sau personală
  - case: categoria din care face parte urgența
  - status: starea în care se află urgența (pending, active, closed)
  - severity: gradul de severitate pe care îl raportează utilizatorul
  - description: conținutul text pe care îl adaugă utilizatorul
  - audio: conținutul audio pe care îl adaugă utilizatorul
  - video: conținutul video pe care îl adaugă utilizatorul
  - foto: conținutul foto pe care îl adaugă utilizatorul
  - deviceId: id-ul unic al instanței aplicației, făcând astfel legătura între urgență și utilizatorul care a raportat-o
  - latitude, longitude: coordonatele locației urgenței

[37] https://firebase.google.com/docs/firestore
[38] https://developer.android.com/studio/write/firebase

- Responders – conține informații legate de preluare unei urgenței
  - id_emergency: id-ul urgenței care a fost preluată
  - email: adresa de email a utilizatorului cu pregătire medicală care a preluat urgența

Fiecare document are un id unic auto-generat la momentul adăugării într-o colecție.

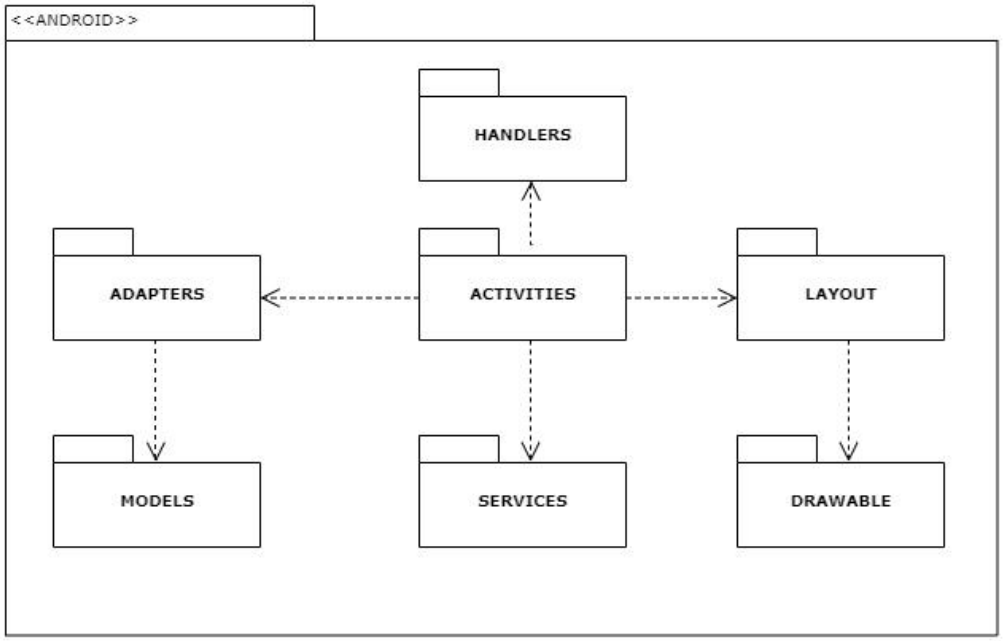


Figura 5.4 Diagrama de pachete

Modulul de gestionare a urgențelor din componenta mobilă a fost împarțită în pachete în funcție de rolul pe care îl joacă în implementare, așa cum este prezentat în Figura 5.4. Pachetele care au fost utilizate în structura aplicației prezentate în această lucrare sunt următoarele:

- **Activities**: conține toate clasele *Activity*[39]care au fost implementate în aplicația prezentată
- **Handlers**: clase Java care se ocupă cu gestionarea logicii unor componente, de exemplu logica autentificării folosind senzorul de amprentă
- **Services**: conține atât servicii implementate în sistem, de exemplu trimiterea de notificări sau gestionarea geofence-urilor, cât și *Broadcast Receivers* implementate, de exemplu repornirea serviciilor după un restart al smartphone-ului

[39] https://developer.android.com/guide/components/activities/intro-activities

- **Layout**: fișierele XML asociate cu activitățile implementate care se ocupă de interfața utilizator a fiecărei pagini din aplicație
- **Drawable**: conține toate imaginile utilizate în intefața utilizator a aplicației, cât și fișiere custom pentru anumite elemente, de exemplu cele pentru adăugarea conținutului audio

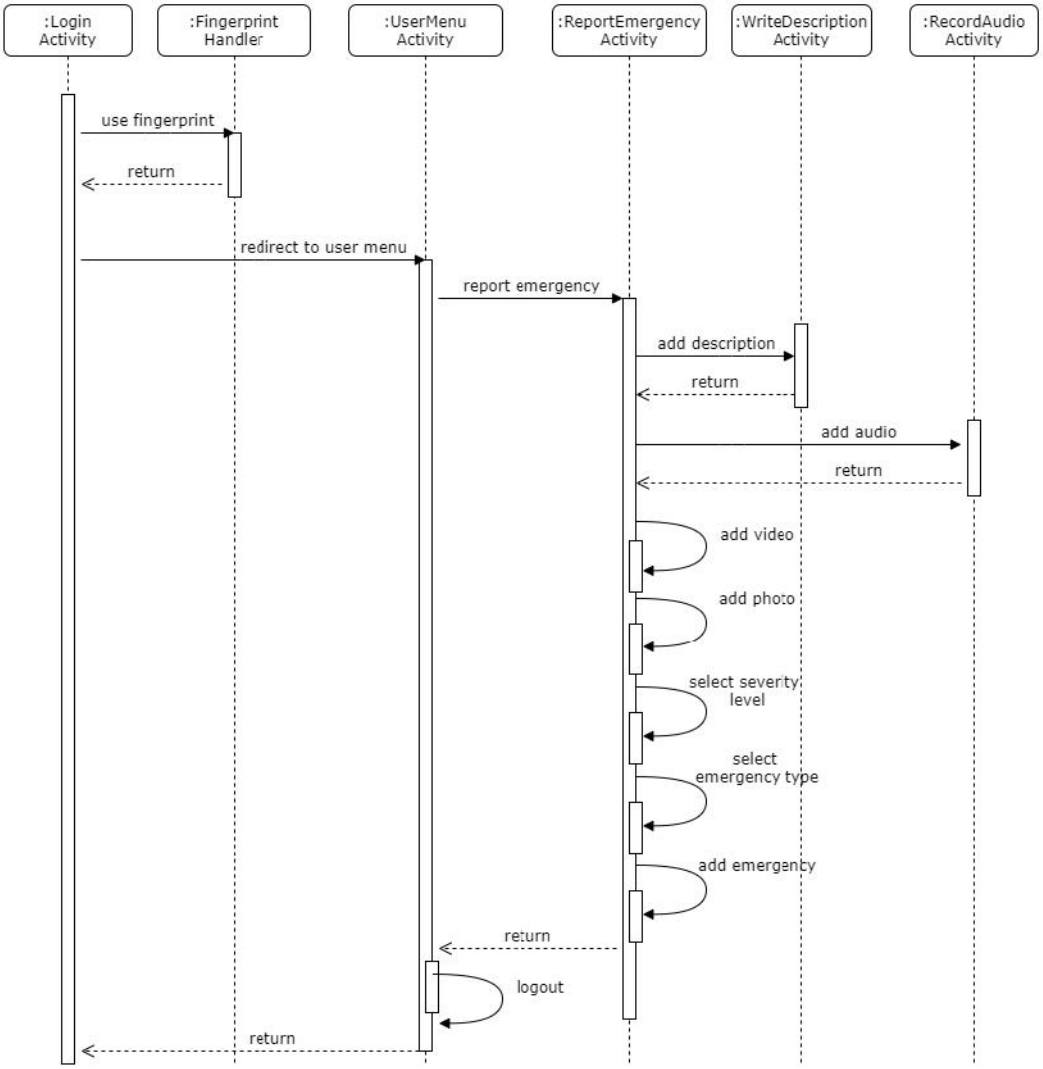


Figura 5.5 Diagrama de secvență

Figura 5.5 prezintă cazul de utilizare principal care constituie motivația principală a sistemului MHealth.

Astfel, un utilizator care și-a creat anterior un cont se autentifică în cadrul aplicației, alege opțiunea de raportare urgență și adăugă raportului toate informațiile posibile: descriere text, imagine, videoclip, înregistrare audio, grad de severitate și tipul urgenței. Locația este adăugată în mod automat.

După adăugarea informațiilor, utilizatorul raportează urgența și la sfârșit se deconectează.

În figura 5.6 se prezintă diagrama de deployment a aplicației. Baza de date la care se conectează ambele componente ale sistemului MHealth, componenta Android și componenta Web, este găzduită în Cloud, platforma folosită fiind Firebase Cloud Firestore. Aplicația mobilă se instalează local pe dispozitivul mobil, sistemul de operare Android necesitând versiunea minimă Lollipop 5.1.

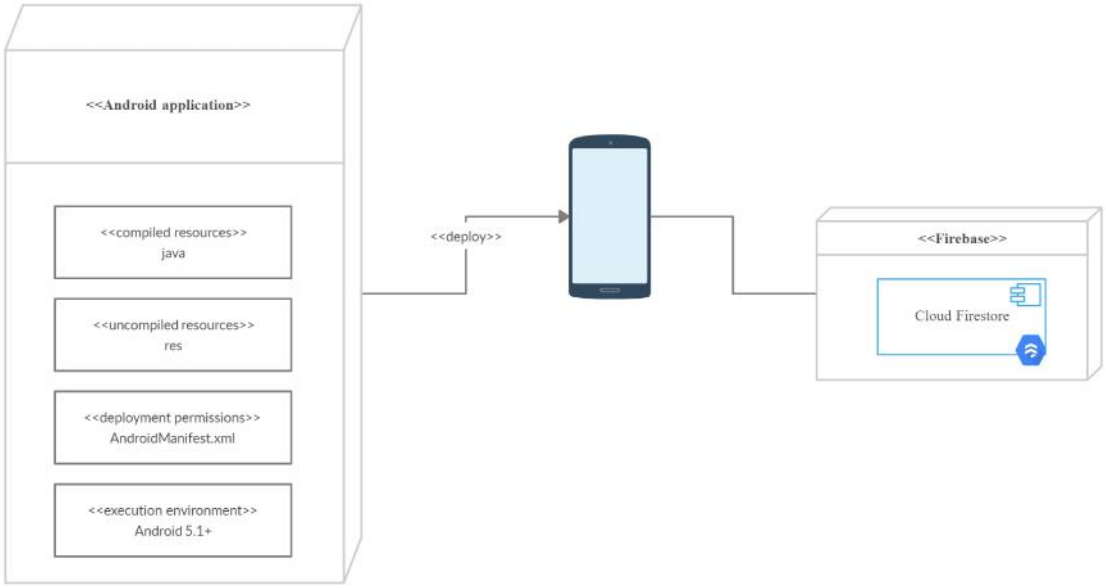


Figura 5.6 Diagrama de deployment

Aplicația implementată folosește toate componentele standard[40] ale unei aplicații Android. Figura 5.7 prezintă modul în care componente interacționează, utilizarea *Content Providers* și *Broadcast Receivers* fiind impusă de funcționalitățile implementate și de structura aplicației.

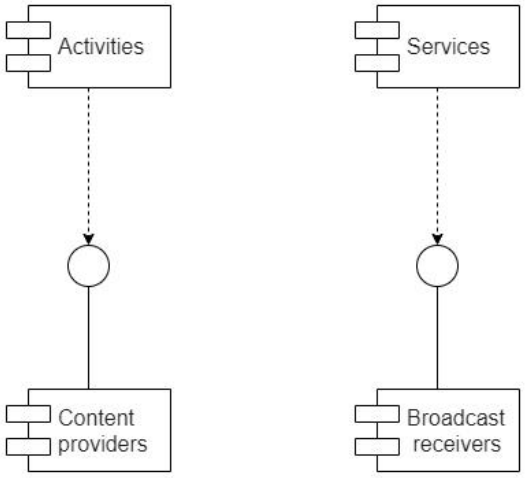


Figura 5.7 Diagrama de componente

[40] https://developer.android.com/guide/components/fundamentals

5.2. Descrierea implementării

Modulul Android pentru gestionarea urgențelor a fost implementat folosind cele mai noi tehnologii existente, scopul fiind realizarea unui sistem cât mai sigur și eficient. Implementarea funcționalităților din cadrul acestei lucrări va fi descrisă în cele ce urmează.

5.2.1. Creare cont folosind o fotografie cu un card de identitate

Contul fiecărui utilizator din cadrul modulului Android pentru gestionarea urgențelor se bazează pe o fotografie cu un card de identitate.

Astfel, la prima utilizarea a aplicației, utilizatorul este redirecționat către pagina de creare cont. Faptul că aplicația este utilizată pentru prima oară este asigurat printr-o verificare a prezenței unui cont în baza de date cu „deviceId" identic cu cel curent. Se va afișa un mesaj cu toate informațiile legate de acest proces, mesaj urmat de un buton ce redirecționează către camera foto de pe dispozitivul mobil. Utilizatorul poate realiza o multitudine de fotografii, fiind nevoit să aleagă doar una dintre acestea.

După selectarea fotografiei preferate, imaginea este salvată temporar în directorul PICTURES al aplicației, extensia fiind .jpg. Pentru accesul la conținutul foto, dar și la conținutul video prezentat în cele urmează, s-a folosit un FileProvider[41].

```
private void TakePhotoForAccount() {
    File file = null;

    try {
        file = File.createTempFile(
                prefix: "AccountPhoto_",
                suffix: ".jpg",
                getExternalFilesDir(Environment.DIRECTORY_PICTURES));
    }
    catch (IOException e) {
        e.printStackTrace();
    }

    photoUri = FileProvider.getUriForFile( context: this,  authority: "fileprovider", file);

    Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoUri);
    startActivityForResult(cameraIntent, CAMERA_INTENT_CODE);
}
```

Figura 5.8 Crearea și salvarea unei fotografii

[41] https://developer.android.com/reference/androidx/core/content/FileProvider

Ulterior, procesul automat de extragere a datelor personale incepe. În acest sens, aplicația mobilă MHealth folosește SDK-ul ML Kit al celor de la Google, Vision Text Recognition[42] fiind API-ul folosit pentru această funcționalitate.

Conținutul imaginii salvate temporar este introdus într-un obiect FirebaseVisionImage. Acest conținut este ulterior procesat, folosindu-se un obiect FirebaseVisionTextRecognizer. În momentul în care textul din imagine este obținut, imaginea originală este ștearsă din memoria telefonului.

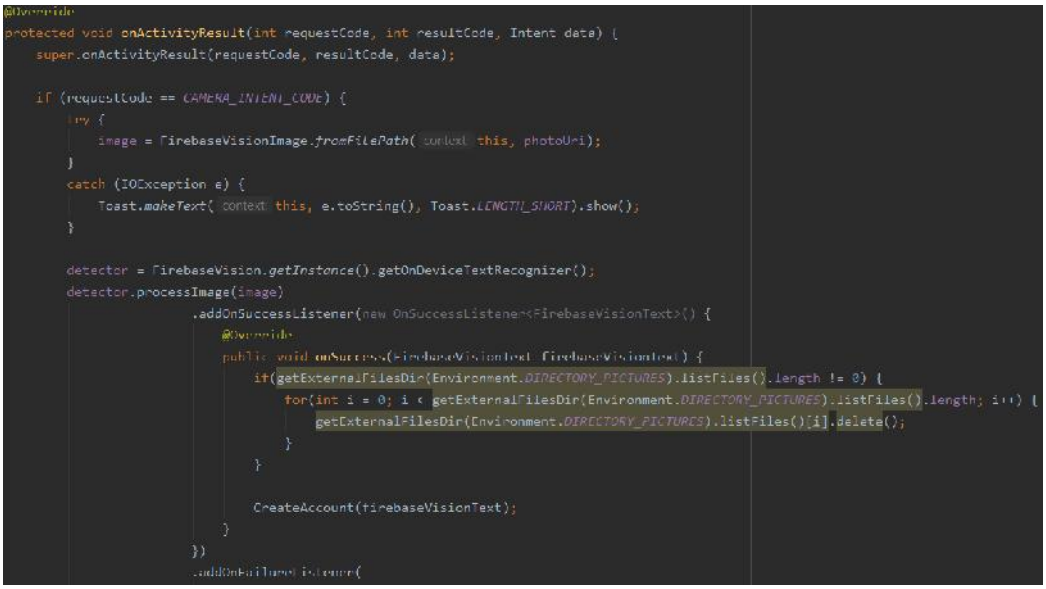


Figura 5.9 Procesarea automată a imaginii și stergerea din memorie

Din textul obținut se extrag ulterior cât mai multe informații posibile. În cazul unui buletin românesc, se obțin următoarele informații: cod numeric personal, nume, prenume, adresă. După obținerea datelor, acestea sunt afișate utilizatorului, care trebuie să confirme corectitudinea lor și să aleagă tipul de utilizator asociat contului, înainte ca acesta să fie creat. Pentru a asocia contul unui utilizatorul cu dispozitivul folosit s-a obținut id-ul unic al instanței aplicației[43].

```
account.put("DeviceId", FirebaseInstanceId.getInstance().getId());
```

Figura 5.10 Obținerea id-ului unic al instanței aplicației

5.2.2. Autentificare folosind senzorul de amprentă

Pentru aplicația mobilă a fost realizat un proces de autentificare folosind date biometrice, mai exact folosirea senzorului de amprentă de pe dispozitiv. Pentru a începe procesul de autentificare, este nevoie ca dispozitivul să îndeplinească două condiții:

[42] https://developers.google.com/ml-kit/vision/text-recognition/android

[43] https://firebase.google.com/docs/reference/android/com/google/firebase/iid/FirebaseInstanceId

- dispozitivul să conțină un senzor de amprentă funcțional
- dispozitivul să aibă cel puțin o amprentă înregistrată

Dacă dispozitivul respectă aceste două condiții, procesul va continua, pașii fiind descriși în continuare. În primul rând este nevoie de generarea unui „Encryption Key" care va fi folosit în crearea unui „Cypher", scopul fiind obținerea unui CryptoObject. Acest obiect este cel folosit de FingerprintManager în procesul de autentificare. „Encryption Key"-ul este generat și salvat în mod securizat în sistemul intern Android Keystore[44].

Pagina de login este cea care impune folosirea senzorului de amprentă pentru a continua. Această acțiune poate avea mai multe rezultate:

- succes: utilizatorul a folosit senzorul de amprentă, acesta a reușit să preia informațiile și să le compare cu amprentele deja înregistrate in dispozitiv și a obținut un rezultat de egalitate
- eșec: utilizatorul a folosit senzorul de amprentă, dar unul din următoarele cazuri a fost îndeplinit: utilizatorul nu a acoperit toată suprafața senzorului, utilizatorul nu a ținut suficient de mult timp degetul pe senzor, nu a fost gasită o amprentă echivalentă înregistrată în dispozitiv

Toate rezultatele prezentate mai sus vor fi afișate vizual utilizatorului.

5.2.3. Raportare urgență

Obiectivul principal al întregului sistem MHealth este de a eficientiza modul în care sunt gestionate urgențele în prezent. Având în vedere acest aspect, funcționalitatea de raportare urgență a fost implementată pentru a fi cât mai ușor, rapid și intuitiv de folosit.

Interfața acestei funcționalități conține câte un buton pentru fiecare tip de conținut care poate fi adăugat (foto, video, audio, text) care redirecționează către procesul de adăugare corespunzător, un slider pentru gradul de severitate și un toggle pentru tipul de urgență raportat.

Locația[45] este trimisă automat în momentul raportării. Pentru obținerea latitudinii și longitudinii s-a folosit un FusedLocationProviderClient pentru a găsi ultima locație cunoscută.

Adăugarea conținutului foto este implementată în mod asemănător cu cel din cadrul creării unui cont. Butonul corespunzător redirecționează către camera foto, unde utilizatorul poate realiza o multitudine de fotografii, fiind nevoit sa aleagă doar una, aceasta fiind salvată temporar în memoria telefonului.

Același proces este realizat și în cazul conținutului video, imaginea fiind de data asta salvată temporar în directorul MOVIES al aplicației, extensia fiind .mp4.

[44] https://developer.android.com/training/articles/keystore
[45] https://developer.android.com/training/location/retrieve-current

Butonul corespunzător adăugării descrierii text redirecționează către o pagină nouă. Aceasta conține un simplu EditText, rezultatul fiind salvat temporat în interfata SharedPreferences.

Adăugarea conținutului audio se realizează la rândul ei într-o pagină nouă, aceasta conținând un singur buton, prima oară având rolul de a începe înregistrarea, a doua oară de a opri înregistrarea. Pentru a înregistra conținutul audio, s-a folosit un obiect MediaRecorder[46], sursa fiind microfonul încorporat în dispozitiv, formatul și encoderul fiind AMR_WB, iar extensia .awb. S-au ales aceste opțiuni pentru a integra cu succes procesarea conținutul audio, folosindu-se API-ul Speech-To-Text al celor de la Google. Din nou, conținutul este salvat temporar în directorul DOCUMENTS al aplicației.

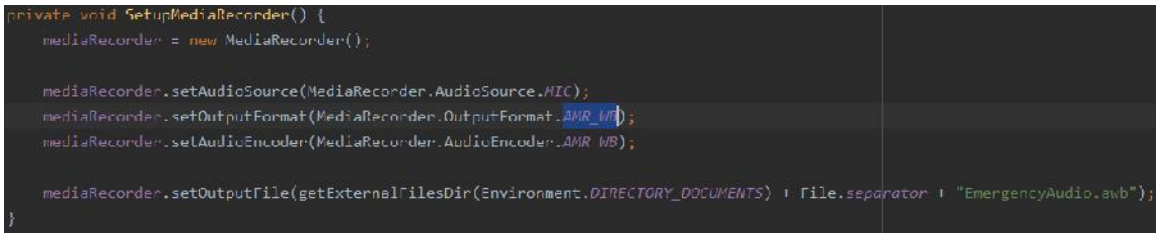


Figura 5.11 Setarea Media Recorder pentru conținutul audio

Conținutul adăugat de utilizator, indiferent de tip (foto, video, audio) este stocat în varianta originală în Firebase Storage. După finalizarea procesului, se obține un „download url”. Acest url este informația care este salvată corespunzător tipului de conținut în documentul ce reprezintă urgența raportată.

Figura 5.12 Stocarea conținutului în Firebase Storage

De asemenea, utilizatorul poate modifica și gradul de severitate al urgenței. Valorea inițială este de 50, intervalul valorilor posibile fiind între 1 și 100. Tipul urgenței este o altă informație raportată. Cele două opțiuni sunt: urgență publică (urgența de interes public) sau urgență personală. Raportarea unei urgențe în care locația ultimei

[46] https://developer.android.com/reference/android/media/MediaRecorder

poziții cunoscute a fost obținută creează automat și un document în colecția GeofenceLocations. Acest document este ulterior folosit trimitere de notificări utilizatorilor care îndeplinesc anumite condiții, scopul fiind preluarea urgenței. După ce se raportează urgența, toate fișierele create și salvate temporar în memoria telefonului vor fi șterse.

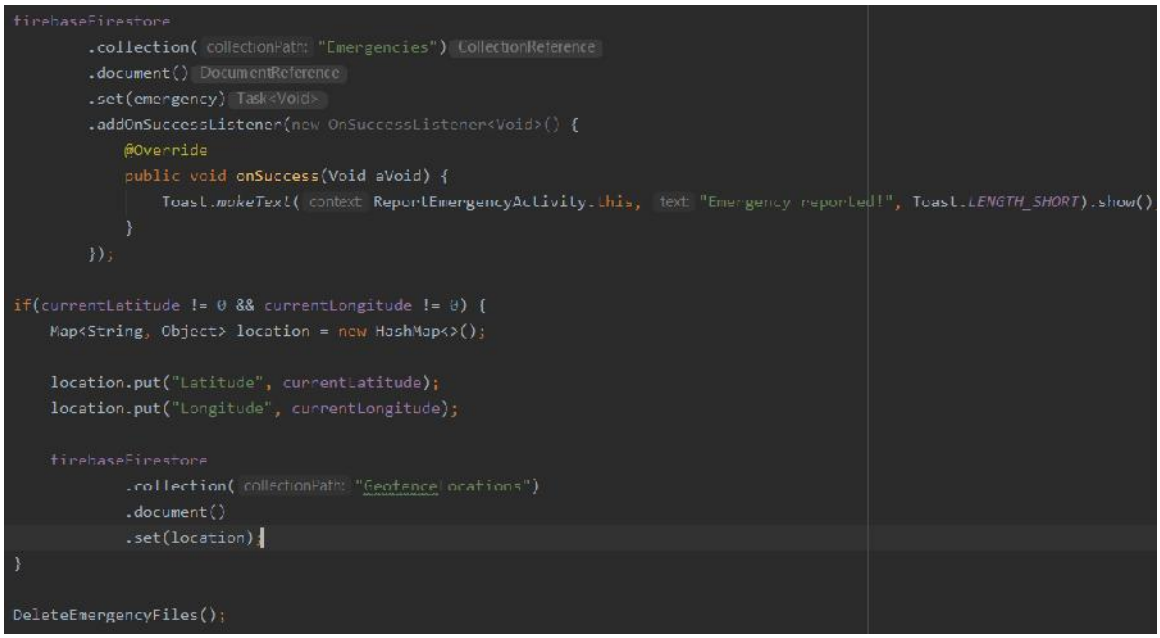


Figura 5.13 Raportarea urgenței și adăugarea geofence-ului

5.2.4. Traducerea conținutului text

Raportarea unei urgențe oferă posibilitatea utilizatorului de a adăuga conținut text. Modulul Android din cadrul sistemului MHealth traduce automat conținutul text dintr-o multitudine de limbi.

Funcționalitatea a fost realizată folosind SDK-ul ML Kit al celor de la Google, API-urile utilizate fiind Language Identification[47] și Translation[48].

În primă fază s-a folosit Language Identification API pentru a identifica limba în care a fost scris conținutul text. Acest API poate identifica peste 100 limbi. Apelul acest API returnează un cod de limbă BCP-47[49] care are o valoare de încredere de minim 0.5. În cazul în care API-ul nu a putut identifica limba, se va returna valoarea „und”.

În cazul în care s-a identificat limba, aceasta va constitui limba sursă în timp ce limba țintă este setată ca fiind engleza. Pentru traducere, API-ul Translation folosește un „translation model”, în funcție de limba sursă și limba țintă. API-ul oferă aceste pachete pentru traducere între mai mult de 50 de limbi. Acest pachet va fi descărcat local în cazul în care nu este deja existent în memoria telefonului. Pachetul poate fi descărcat atât prin

[47] https://developers.google.com/ml-kit/language/identification/android
[48] https://developers.google.com/ml-kit/language/translation/android
[49] https://en.wikipedia.org/wiki/IETF_language_tag

conexiune Wi-Fi, cât și folosind date mobile. Dimensiunea pachetului este de aproximativ 30MB.

```
private void TranslateText(final String description) {
    FirebaseLanguageIdentification languageIdentifier = FirebaseNaturalLanguage.getInstance().getLanguageIdentification();
    languageIdentifier.identifyLanguage(description).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(description).addOnSuccessListener(new OnSuccessListener<String>() {
                            @Override
                            public void onSuccess(@NonNull String translatedText) {
```

Figura 5.14 Traducerea conținutului text

5.2.5. Preluare urgență

Poate cea mai importantă cerință non-funcțională a modului Android pentru gestionarea urgențelor este disponibilitatea. Astfel, implementarea aplicației pune accent pe acest aspect, obiectivul fiind realizarea unui sistem disponibil 24 de ore din 24.

Se pune astfel problema repornirii serviciului MHealth în cazul restartării dispozitivului mobil. În acest sens, s-a implementat un BroadcastReceiver. Odată cu apariția Android 8.0 se impune ca pornirea unui serviciu în fundal fără ca aplicația să fie pornită doar dacă se folosește un ForegroundService. Luând în considerare acest aspect, BroadcastReceiver-ul implementat este un listener pentru evenimentul „on boot completed”. Astfel, în momentul în care procesul de bootare al sistemului de operare Android este finalizat, se va afișa un push notification care îndeamnă utilizatorul să acceseze notificarea pentru a reporni serviciul MHealth.

```
public class OnBootCompletedBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new NotificationChannel( id: "OnBootCompleted",  name: "OnBootCompleted", NotificationManager.IMPORTANCE_DEFAULT);
            NotificationManager notificationManager = context.getSystemService(NotificationManager.class);
            notificationManager.createNotificationChannel(channel);

            Intent openAppIntent = new Intent(context, CheckAccountActivity.class);
            PendingIntent pendingIntent = PendingIntent.getActivity(context,  requestCode: 10, openAppIntent, PendingIntent.FLAG_UPDATE_CURRENT);

            NotificationCompat.Builder builder = new NotificationCompat.Builder(context,  channelId: "OnBootCompleted")
                    .setSmallIcon(R.drawable.ic_launcher_foreground)
                    .setContentTitle("MHealth")
                    .setContentText("Click to start looking for nearby emergencies!")
                    .setAutoCancel(true)
                    .setContentIntent(pendingIntent)
                    .setOngoing(true)
                    .setPriority(NotificationCompat.PRIORITY_DEFAULT);

            NotificationManagerCompat notificationManagerCompat = NotificationManagerCompat.from(context);
            notificationManagerCompat.notify(String.valueOf(System.currentTimeMillis()),  id: 0, builder.build());
        }
    }
}
```

Figura 5.15 Notificarea despre repornirea serviciului MHealth

Sistemul MHealth își propune să răspundă în mod dinamic și prompt urgențelor raportate. Astfel, s-a implementat un ForegroundService care rulează indiferent de starea aplicației. Acest serviciu se folosește de funcționalitatea de bază a platformei Firebase Cloud Firestore integrată în aplicația mobilă, mai exact de „realtime updates”. Astfel, serviciul de căutare a urgențelor din apropiere folosește un „event listener” pe colecția GeofenceLocations din baza de date.

Pentru fiecare modificare a documentelor din colecție, serviciul reface lista de geofence-uri urmărite de aplicație. Fiecare document din colecție este reprezentat în aplicație printr-un geofence cu următoarele proprietăți:

- RequestId: id-ul geofence-ului, în acest caz fiind locația urgenței, setată prin coordonate
- CircularRegion: cercul acoperit de geofence. Se seteaza centrul cercului, care în cazul sistemului MHealth este locația urgenței raportate, datele locației fiind latitudinea și longitudinea. De asemenea, se setează și raza cercului în metri, în acest caz raza fiind setată la 500 de metri
- ExpirationDuration: perioada de timp pentru care geofence-ul este activ. Aplicația MHealth gestionează geofence-urile manual, așa că valoarea setată a fost NEVER_EXPIRE
- TransitionType: evenimentul care declanșează geofence-ul. În contextul sistemului MHealth, s-a setat GEOFENCE_TRANSITION_DWELL pentru a se limita trimiterea accidentală a notificărilor
- LoiteringDelay: perioada de timp pe care utilizatorul trebuie să o petreacă în cercul acoperit de geofence. Pentru a se asigura faptul că utilizatorul nu primește o notificare incorectă, s-a setat o perioadă de 30 de secunde

```
firebaseFirestore.collection( collectionPath: "GeofenceLocations")
        .addSnapshotListener(new EventListener<QuerySnapshot>() {
            @Override
            public void onEvent(@Nullable QuerySnapshot snapshot, @Nullable FirebaseFirestoreException e) {
                if(snapshot.getDocuments().size() > 0) {
                    List<Geofence> geofences = new ArrayList<>();

                    for (DocumentSnapshot location : snapshot.getDocuments()) {
                        geofences.add(new Geofence.Builder()
                                .setRequestId(location.get("Latitude").toString() + " " + location.get("Longitude").toString())
                                .setCircularRegion(Double.parseDouble(location.get("Latitude").toString()), Double.parseDouble(location.get("Longitude").toString()),  r: 500)
                                .setExpirationDuration(Geofence.NEVER_EXPIRE)
                                .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_DWELL)
                                .setLoiteringDelay(30000)
                                .build());
                    }

                    geofencingClient.removeGeofences(GetGeofencePendingIntent());

                    GeofencingRequest.Builder geofencingRequest = new GeofencingRequest.Builder();
                    geofencingRequest.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_DWELL);
                    geofencingRequest.addGeofences(geofences);

                    geofencingClient.addGeofences(geofencingRequest.build(), GetGeofencePendingIntent());
                }
            }
        });
```

Figura 5.16 Serviciul de actualizarea a geofence-urilor

Pentru funcționalitatea de preluare urgențe s-a mai implementat un BroadcastReceiver. Acesta este declanșat în momentul în care utilizatorul a îndeplinit toate cerințele unui Geofence.

Declanşarea acestui BroadcastReceiver duce la trimiterea unui push notification cu adresa exactă a locației din apropiere. Pentru a afișa corect această informație, s-a folosit un Geocoder[50] care primește o latitudine și o longitudine și returnează adresa locației. Precizia rezultatului poate varia de la adresa celei mai apropiate clădiri, la numele orașului și un cod poștal.

Accesarea notificării are un rezultat diferit în funcție de tipul utilizatorului asociat cu dispozitivul mobil inteligent. Astfel, un utilizator obișnuit va putea doar viziona și elimina notificarea, în timp ce un utilizator cu pregătire medicală va putea accesa notificarea, fiind redirecționat către pagina de login. După autenficarea cu succes, va fi redirecționat către pagina de preluare urgență, unde va fi afișată din nou locația și utilizatorul va trebui să introducă o adresă de email validă pe care ulterior i se va trimite un email cu toate informațiile legate de urgență. Tipul utilizatorul curent se verifică prin prezența în baza de date a unui cont cu „deviceId” identic cu cel curent.

5.2.6. Permisiuni

Implementarea modulului Android pentru gestionarea urgențelor a avut ca scop realizarea unei aplicații cât mai sigure, mai ales în contextul sistemului MHealth, dar și cât mai rapide și intuive. Luând în considerare acest fapt, aplicația mobilă are nevoie de anumite permisiuni[51] din partea utilizatorului pentru a avea acces la diferiți senzori sau resurse. Printre aceste permisiuni se numără:

- RECEIVE_BOOT_COMPLETED: permite aplicației să fie notificată în momentul în care procesul de bootare al sistemului de operare este terminat
- FOREGROUND_SERVICE: permite aplicației să folosească servicii de tip foreground
- INTERNET: permite aplicației să deschidă socket-uri
- ACCESS_BACKGROUND_LOCATION: permite aplicației să acceseze serviciile de localizare în fundal
- ACCESS_FINE_LOCATION: permite aplicației să acceseze serviciile de localizare precisă
- ACCESS_COARSE_LOCATION: permite aplicației să acceseze serviciile de localizare aproximativă
- RECORD_AUDIO: permite aplicației să înregistreze conținut audio
- WRITE_EXTERNAL_STORAGE: permite aplicației să scrie în spațiul de stocare extern
- CAMERA: permite aplicației să acceseze aplicația de camera foto disponibilă în sistemul de operare
- USE_BIOMETRIC: permite aplicației să folosească modalitățile de obținere a datelor biometrice disponibile pe smartphone
- ACCESS_NETWORK_STATE: permite aplicației să afle informații legate de rețeaua de internet la care smartphone-ul este conectat

[50] https://developer.android.com/reference/android/location/Geocoder
[51] https://developer.android.com/reference/android/Manifest.permission

5.2.7. Managementul implementării proiectului

Sistemul MHealth este format din două module majore: modulul Android și modulul Web. Fiecare dintre cele două module are atât o componentă pentru gestionarea urgențelor, cât și o componentă personală.

Pentru gestionarea implementării sistemului MHealth, s-a folosit platforma GitHub. Astfel, s-a creat câte un repository pentru fiecare proiect implementat: aplicație mobilă Android, backend Java Spring și frontend React. Fiecare membru a avut acces la aceste repository-uri.

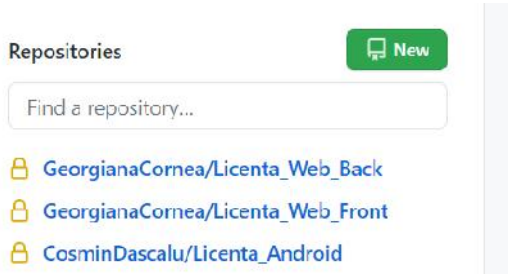


Figura 5.17 Repository-urile create

Fiecare funcționalitate implementată a fost creată și testată pe un branch separat, branch care abia după verificarea funcționării corespunzătoare a fost merge-uit în branch-ul master unde s-a păstrat mereu ultima versiune funcțională pentru fiecare proiect.

Pentru o gestionare facilă a progresului realizat, s-a folosit clientul GitHub Desktop, unde s-au putut urmări toate modificările aduse sistemului, cât și statusul implementării altor funcționalități.

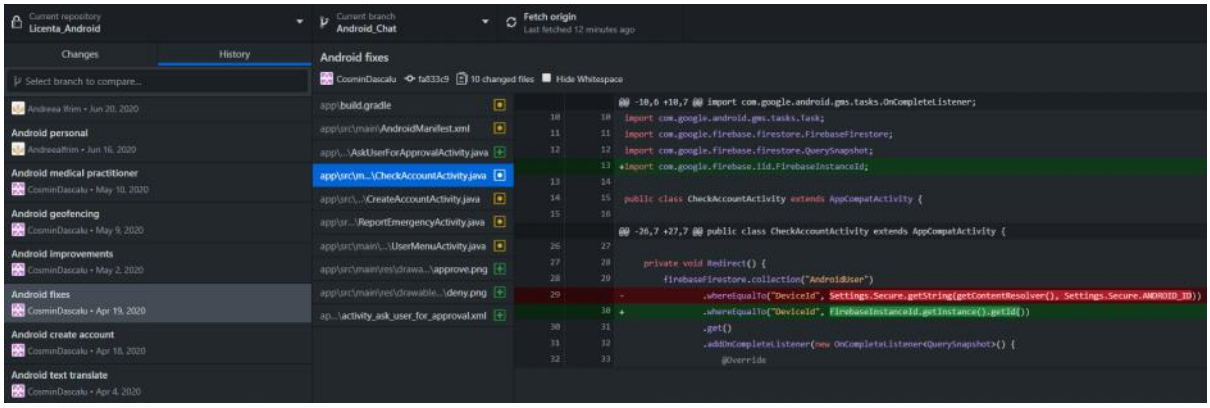


Figura 5.18 Utilizare GitHub Desktop

Pe parcursul dezvoltării sistemului MHealth a fost actualizată constant o fișa de evoluție, fișă prezentată în detaliu în Anexa 1.

Capitolul 6. Testare şi Validare

Conform documentației Android, sunt puse la dispoziție trei tipuri de testare funcțională: local unit tests, instrumented unit tests și UI tests.

Local unit tests sunt teste care rulează local, fiind compilate folosind JVM.

Instrumented unit tests sunt teste care rulează pe un dispozitiv fizic sau pe un emulator. Diferența majoră între cele două tehnici constă în accesul sporit la resurse pe care îl au instrumented unit tests și faptul că se pot folosi de dependințe complexe, de exemplu cele necesare pentru folosirea platformei Firebase.

În ceea ce privește **UI tests**, sunt puse la dispoziție mai multe frameworkuri, cel mai complex și utilizat este Espresso. Astfel, din punctul de vedere Android, local unit tests pot fi considerate testele folosite pentru Unit Testing, iar instrumented unit tests și UI tests pot fi considerate testele folosite pentru Integration Testing.

6.1. Testarea non-funcțională

Din acest punct de vedere, aplicația prezintă cerințe non-funcționale precum: utilizabilitatea, securitate, disponibilitate sau integritatea datelor. În contextul aplicației prezentate, aceste cerințe necesită testare de tip black-box, testare ce a fost realizată cu succes la momentul implementării.

Un exemplu este cazul integrității datelor unde la crearea contului folosind datele extrase de pe o fotografie cu buletinul, utilizatorului i se prezintă rezultatul, iar acesta trebuie să-și dea acordul înainte ca datele să fie înregistrate.

Un alt exemplu este cazul securității, unde pe lângă crearea contului cu datele extrase de pe o fotografie cu buletinul, autentificarea în aplicație se realizează folosind o amprentă înregistrată pe dispozitivul mobil. Ambele teste au fost realizate din perspectiva utilizatorului final la momentul implementării.

În ceea ce privește atât utilizabilitatea, cât și disponibilitatea, testarea a fost realizată în cadrul echipei MHealth. Fiecare membru a testat interfața grafică, sugestiile fiecăruia fiind luate în considerare pentru a realiza un UI cât mai prietenos și ușor de folosit. Serviciile de căutare a urgențelor și creare a geofence-urilor au fost la rândul lor testate în cadrul echipei, verificându-se funcționarea lor pe o lungă perioadă de timp și modul în care ele răspund la diferite scenarii, un exemplu fiind restartarea smartphone-ului.

6.2. Instrumented Unit Testing

A fost utilizat acest tip de testare pentru a verifica corectitudinea modului în care aplicația implementată comunică cu platforma Firebase. Astfel, s-au verificat operații de bază precum adăugarea unui user sau adăugarea de fișiere.

```
public void AddUser() {
    final FirebaseFirestore firebaseFirestore = FirebaseFirestore.getInstance();

    final Map<String, Object> testObject = new HashMap<>();
    testObject.put("Address", "Cluj");
    testObject.put("Cnp", 0);
    testObject.put("DeviceId", FirebaseInstanceId.getInstance().getId());
    testObject.put("Name", "Dascalu Cosmin-Stefan");
    testObject.put("UserType", "Regular User");

    try {
        Tasks.await(firebaseFirestore
                .collection( collectionPath: "AndroidUser") CollectionReference
                .document() DocumentReference
                .set(testObject) Task<Void>
                .addOnCompleteListener((task) → {
                        assertEquals( expected: true, task.isSuccessful());
                }));
    } catch (ExecutionException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

Figura 6.1 Testarea adăugării unui user

În figura 6.1 este prezentat modul în care s-a testat adăugarea unui user. S-a creat un obiect de tip HashMap ce conține următoarele câmpuri: Address, Cnp, DeviceId, Name, UserType. S-a creat un task de adăugare în baza de date și s-a verificat dacă operația a avut success folosind un listener OnComplete.

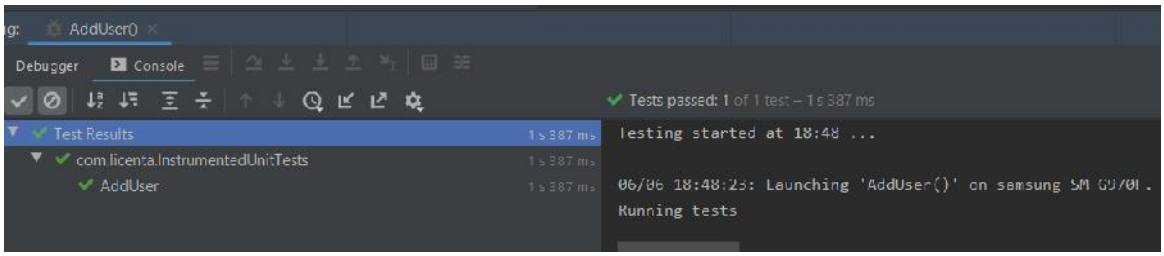


Figura 6.2 Rezultatul testului

În figura 6.2 se observă faptul că la testarea funcționalității de adăugare user, rezultatul obținut este unul pozitiv.

```
@Test
public void AddFile() {
    final StorageReference firebaseStorage = FirebaseStorage.getInstance().getReference();
    File testFile = null;
    try {
        testFile = File.createTempFile( prefix: "test", suffix: "jpg");
    } catch (IOException e) {
        e.printStackTrace();
    }

    try {
        Tasks.await(firebaseStorage
                .child("Tests/" + UUID.randomUUID() + ".jpg")
                .putFile(Uri.fromFile(testFile))
                .addOnCompleteListener((task) -> {
                        assertEquals( expected: true, task.isSuccessful());
                }));
    } catch (ExecutionException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

Figura 6.3 Testarea adăugării unui fișier

În figura 6.3 este prezentat modul în care s-a testat adăugarea unui user. S-a creat un fișier temporar gol cu extensia jpg. S-a creat un task de adăugare în baza de date și s-a verificat dacă operația a avut success folosind un listener OnComplete.

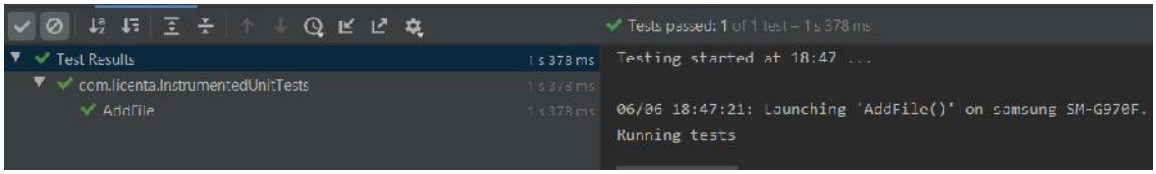


Figura 6.4 Rezultatul testului

În figura 6.4 se observă faptul că la testarea funcționalității de adăugare fișier, rezultatul obținut este unul pozitiv.

6.3. UI Testing

Acest tip de testare a fost utilizat pentru a verifica corectitudinea modului în care sistemul răspunde la inputul utilizatorului.

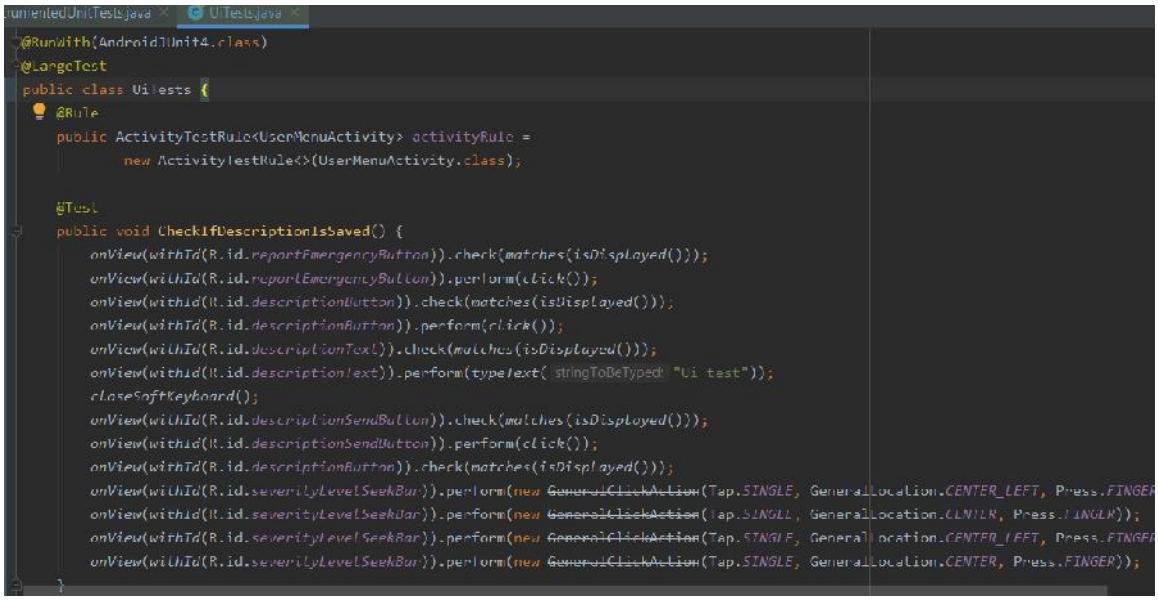


Figura 6.5 Testarea urmăririi unui flux din UI

În figura 6.5 este prezentat modul în care s-a testat o secvență de acțiuni din UI. Aplicația va porni automat activitatea UserMenu. Se va urmări următorul flux: se accesează pagina ReportEmergency, se accesează pagina AddDescription, se adaugă un string in edittextul respectiv, se salvează descrierea, se revine la pagina ReportEmergency, se schimbă valoarea cursorului pentru nivelul de securitate și se iese din aplicație.

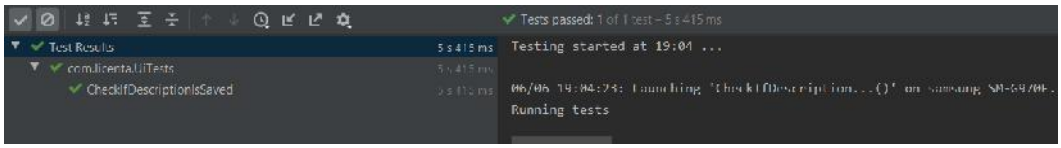


Figura 6.6 Rezultatul testului

În figura 6.6 se observă faptul că la testarea fluxului de acțiuni, rezultatul obținut este unul pozitiv.

Pentru testarea automată a UI-ului s-a folosit tool-ul Espresso. A fost realizată o captură de ecran sub formă de videoclip mp4 care evidențiază execuția testului. Viteza videoclipului a fost editată ulterior la 0.5x pentru a se observa mai bine modul în care acționează tool-ul. Acest videoclip poate fi urmărit **aici**.

6.4. System Testing

Ținând cont că fiecare dintre cele trei subsisteme care constituie platforma MHealth a fost testat individual, s-a optat pentru system testing ca modalitate de validare aplicației înainte de acceptance testing.

Astfel, în cadrul echipei s-a ales caz de utilizare amplu, care să testeze cât mai multe funcționalități din cele implementate: un utilizator obișnuit se autentifică în aplicația Android și raportează o urgență oferind locația și informații de tip text, foto;

datele sunt preluate de aplicația web, sunt procesate, iar un cadru medical completează raportul cu o listă de instrucțiuni; un utilizator Android cu pregătire medicală aflat în raza urgenței primește o notificare cu adresa exactă, preia urgența și primește un email cu toate informațiile necesare; utilizatorul care a raportat urgența folosește chatul pentru a interacționa cu un cadru medical aflând detalii suplimentare.

După realizarea testării cazului s-a observat faptul că sistemul funcționează corespunzător cerințelor funcționale ca un tot unitar. Aceeași metodă a fost aplicată și celorlalte cazuri de utilizare posibile.

Capitolul 7. Manual de Instalare şi Utilizare

În cadrul capitolul 7 se vor prezenta resursele hardware și software necesare pentru rularea aplicației mobile, cât și pașii care trebuie urmați sau modul în care aplicația este concepută pentru funcționare.

7.1. Resursele necesare pentru instalare

Componenta mobilă a sistemului MHealth necesită o serie de resurse hardware şi software pentru a funcționa în mod optim. Aceste resurse au fost alese pentru ca sistemul să fie disponibil unui număr cât mai mare de utilizatori, dar şi ca tehnologiile folosite să fie integrate cu succes. Printre resursele necesare se numără următoarele:

- Un telefon mobil inteligent cu sistem de operare Android
- Versiunea minimă a sistemului de operare Android trebuie să fie Android 5.1 Lollipop, 92.3% din dispozitivele Android au minim această versiune[52]
- Dispozitivul mobil trebuie să conțină un senzor de amprentă
- Conexiune stabilă la internet (Wi-Fi sau date mobile)
- Serviciile de localizare GPS activate
- Spațiu de stocare disponibil de minim 50MB

7.2. Manual de utilizare

În cele ce urmează se va prezenta modul în care aplicația a fost concepută, prezentând pașii care trebui urmați pentru a folosi aplicația în mod optim, în funcție de tipul de utilizator autentificat în cadrul aplicației.

7.2.1. Utilizator obişnuit

După instalarea aplicației, la prima deschidere aplicația va detecta faptul că dispozitivul curent nu este asociat cu niciun cont, astfel va redirecționa utilizatorul către pagina de creare cont.

Se va afișa un mesaj care atenționează utilizatorul că nu s-a găsit un cont asociat cu respectivul dispozitiv. De asemenea, utilizatorul va fi anunțat că pentru a realiza un cont, se va realiza o fotografie cu cardul de identitate, datele vor fi extrase automat şi poza nu va fi salvată ulterior. La finalul mesajului se vor da niște indicații legate de modul în care ar trebui realizată fotografia pentru ca rezultatul să fie cel dorit, iar datele să fie extrase în mod corect.

Butonul din secțiunea inferioară redirecționează către camera foto, unde utilizatorul poate realiza oricâte fotografii, până când este de acord cu calitatea uneia dintre ele.

Această parte a procesului de creare cont este descrisă în figura 7.1.

[52] https://www.xda-developers.com/android-version-distribution-statistics-android-studio/

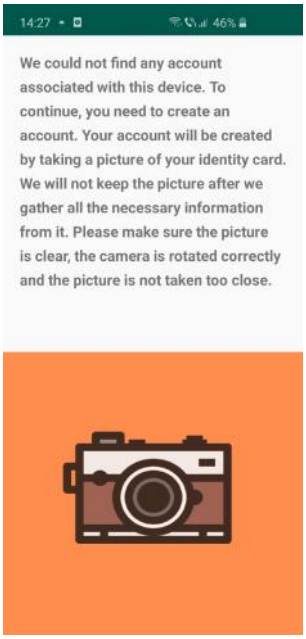


Figura 7.1 Pagina inițială în procesul de creare a contului

În figura 7.2 este prezentată partea finală a procesului de creare cont. După ce utilizatorul a confirmat una dintre fotografii, sistemul MHealth extrage automat datele necesare de pe cardul de identitate. Rezultatele sunt afișate pe această pagină, iar utilizatorul trebuie să confime corectitudinea lor și să aleagă tipul de utilizator asociat contului. Dacă datele nu sunt corecte, procesul se reia.

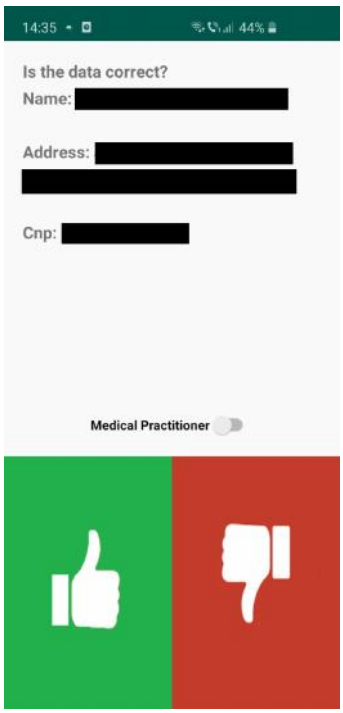


Figura 7.2 Pagina de confirmare a datelor extrase

După crearea contului, utilizatorul este redirecționat către pagina de login, unde va fi redirecționat și de fiecare dată când va folosi aplicația ulterior. Autentificarea se realizează folosind senzorul de amprentă. Pagina corespunzătoare este prezentată în figura 3.1.

Figura 7.3 Pagina de login

După autentificarea cu succes a utilizatorului, acesta este redirecționat către meniul principal. Cele trei opțiuni posibile sunt: raportare a unei urgențe, accesare a modulului personal și logout. Figura 7.4 pune în evidență interfața acestei pagini.

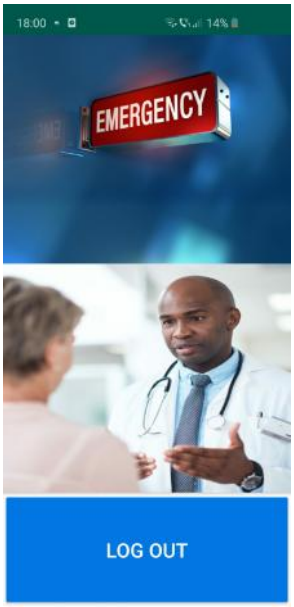


Figura 7.4 Meniul principal

Figura 7.5 prezintă pagina care este afișată în cazul în care utilizatorul dorește raportarea unei urgențe. Se pot observa astfel opțiunile oferite: adăugare de conținut text, audio, foto, video. De asemenea, utilizatorul poate seta gradul de severitate folosind un slider cu valori între 1 și 100, sau poate alege dacă urgența este de publică sau personală.

Figura 7.5 Interfața funcționalității de raportare urgență

7.2.2. Utilizator cu pregătire medicală

Utilizatorul cu pregătire medicală are acces la toate funcționalitățile utilizatorului obișnuit. Diferența între cele două tipuri de utilizatori apare în cazul funcționalității de preluare urgentă.

Astfel, toți utilizatorii care îndeplinesc cerințele vor primi notificarea de tip push care anunță prezența unei urgențe în apropiere și va afișa adresa exactă a acesteia, dar utilizatorii obișnuiți nu o vor putea accesa, notificarea are doar valoare vizuală. Utilizatorul cu pregătire medicală va putea spre deosebire accesa notificarea pentru a continua procesul de preluare a urgenței.

Figura 7.6 prezintă notificarea de tip push care apare în cazul unei urgențe în apropiere. De asemenea, în figură se poate observa și serviciul MHealth de căutare a urgențelor, serviciu care rulează independent de starea aplicației.

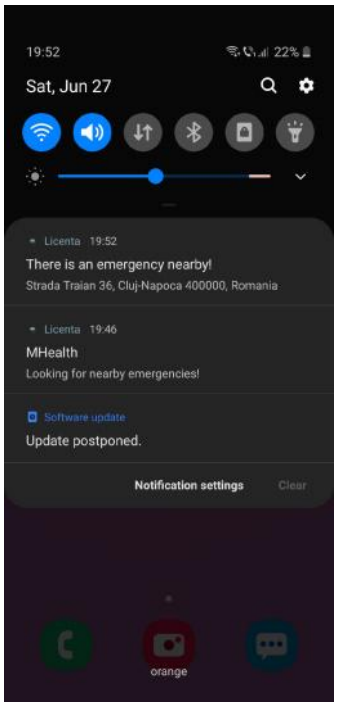


Figura 7.6 Notificarea de tip push

După accesarea notificării, utilizatorul cu pregătire medicală va fi redirecționat către pagina de login a aplicației. Autentificarea cu succes va afișa ulterior pagina de preluare urgență care va afișa locația urgenței și utilizatorul va trebui să completeze o adresă de email pe care va primi ulterior toate informațiile legate de urgentă. Pagina de preluare a urgenței este prezentată în figura 7.7.

Figura 7.7 Pagina de preluare urgență

Capitolul 8. Concluzii

În acest ultim capitol al lucrării se vor prezenta rezultatele obținute în urma implementării componentei Android pentru gestionarea urgențelor în cadrul MHealth, cât și diferite dezvoltări ulterioare posibile care ar completa sistemul propus.

8.1. Rezultate obținute

Aplicația realizată îndeplinește obiectivele setate inițial. Astfel, utilizatorul are la dispozitie o aplicație sigură, cu o interfață prietenoasă, ușor de folosit și cu toate funcționalitățile esențiale pentru ca urgențele să fie gestionate mai eficient decât în prezent.

Securitatea a fost unul dintre cele mai importante aspecte în realizarea sistemului. Astfel, autentificarea utilizatorului se bazează pe senzorul de amprentă, asigurând astfel faptul că utilizatorii neautorizați nu vor putea folosi aplicația. De asemenea, contul asociat cu un telefon mobil inteligent este creat extrăgând automat datele necesar de pe o fotografie cu cardul de identitate, fapt ce indică din nou o securitate sporită.

Funcționalitatea de raportare a unei urgențe este rapidă și eficientă, fluxul de acțiuni necesare fiind intuitiv. Adăugarea de conținut foto, video, audio sau text se face simplu, accesând secțiunea dedicată fiecărui tip de conținut. Localizarea se realizează automat, utilizatorul fiind nevoit doar să activeze serviciile GPS ale dispozitivului mobil. Înainte de a raporta urgența, utilizatorul poate selecta atât tipul de urgență raportat, cât și gradul de severitate din punct de vedere personal. Traducerea automată a conținutului text în limba engleză permite utilizatorului să se exprime în limba preferată pentru a descrie cât mai clar urgența, fără a se mai pune problema unei bariere de limbaj.

În ceea ce privește rezultatul raportării unei urgențe, toți utilizatorii aplicației vor primi o notificare cu adresa exactă a urgenței în cazul în care petrec cel puțin perioada de timp prestabilită în raza de acțiune a geofence-ului creat în jurul locației raportate. Accesarea notificării duce utilizatorul cu pregătire medicală către pagina de login unde după autentificarea cu succes, acesta poate prelua urgența, oferind o adresă de email unde va primi toate informațiile necesare legate de urgentă.

Aplicația se comportă conform așteptărilor, oferind o soluție pentru a gestiona eficient o urgență, în special în perioada de timp dintre apelul la 112 și momentul în care personalul specializat ajunge la locația urgenței.

8.2. Dezvoltări ulterioare

Deși aplicația realizată îndeplinește obiectivele setate inițial, există diverse dezvoltări ulterioare care ar putea face ca sistemul MHealth să fie și mai atractiv. Printre aceste dezvoltări ulterioare pentru aplicația mobilă se numără:

- Implementarea unei hărți interactive care să arate drumul cel mai scurt de la locația curentă la locația urgenței preluate. Google Maps API oferă suport nativ pentru aplicațiile Android, fiind astfel o soluție posibilă pentru implementarea acestei dezvoltări ulterioare

- Integrarea unui asistent vocal pentru a eficientiza și mai mult sistemul, timpul de raportare al urgenței astfel scăzând. Un bun exemplu ar fi integrarea asistentului vocal Google. Utilizatorul ar putea astfel adăuga conținut mai rapid, folosindu-se de comenzi vocale pentru a naviga prin aplicație.
- Oferirea posibilității de a crea o listă de persoane de contact care să primească o notificare în cazul utilizatorul este implicat, indiferent dacă cerințele legate de locație pentru a primi notificare sunt îndeplinite sau nu
- Implementarea funcționalității de „Check-in", pentru a notifica o listă de persoane cu starea curentă a utilizatorului. Modul în care persoanele pot fi notificate poate varia de la mesaje de tip SMS, la emailuri, la notificări de tip push
- Setarea dinamică a setărilor geofence-ului în funcție de gravitatea urgenței și de numărul de raportări din cadrul aceluiași cluster. Astfel, condițiile care trebuie îndeplinite de utilizatorii cu pregătire medicală pentru a primi notificărilor ar putea fi diminuate în cazul unei gravități ridicate și a unui număr de raportări mare

Bibliografie

[1] C. Q. Wu, Z. Wang, G. Chen și D. Ferebee, „Recent Advances and Developments in Mobile Health," *Journal of Healthcare Engineering*, 2018.
DOI: 10.1155/2018/4747593
Available at: https://www.hindawi.com/journals/jhe/2018/4747593/

[2] S. R. Steinhubl, E. D. Muse și E. J. Topol, „The emerging field of mobile health," *Sci Transl Med*, 2015.
DOI: 10.1126/scitranslmed.aaa3487
Available at: https://stm.sciencemag.org/content/7/283/283rv3

[3] M. Janda, L. J. Loescher și H. P. Soyer, „Enhanced Skin Self-examination: A Novel Approach to Skin Cancer Monitoring and Follow-up," *JAMA Dermatol*, 2013.
DOI: 10.1001/jamadermatol.2013.1218
Available at:
https://jamanetwork.com/journals/jamadermatology/fullarticle/1654871

[4] S. Iyengar, "Chapter 12 - Mobile health (mHealth)," in *Fundamentals of Telemedicine and Telehealth*, Elsevier, 2020, pp. 277-294.
DOI: 10.1016/B978-0-12-814309-4.00012-4
Available at:
https://www.sciencedirect.com/science/article/pii/B9780128143094000124

[5] E. Cabral, W. Castro, D. Florentino, D. Viana, J. Costa, R. Souza, A. Rêgo, I. Araújo-Filho and A. Medeiros, "Response time in the emergency services. Systematic review," *Acta Cirurgica Brasileira*, 2018.
DOI: 10.1590/s0102-865020180120000009
Available at:
https://www.researchgate.net/publication/330175313_Response_time_in_the_emergency_services_Systematic_review

[6] J. M. Statland, Y. Wang, R. Richesson, B. Bundy, L. Herbelin, J. Gomes, J. Trivedi, S. Venance, A. Amato, M. Hanna, R. Griggs and R. J. Barohn, "An Interactive Voice Response Diary for Patients With Non-Dystrophic Myotonia," *Muscle Nerve*, 2011.
DOI: 10.1002/mus.22007
Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/mus.22007

[7] E. Stankevich, I. Paramonov and I. Timofeev, "Mobile phone sensors in health applications," *2012 12th Conference of Open Innovations Association (FRUCT), Oulu*, 2012, pp. 1-6.
DOI: 10.23919/FRUCT.2012.8122097
Available at: https://ieeexplore.ieee.org/document/8122097

[8] M. Romano, T. Onorati, I. Aedo and P. Díaz, "Designing Mobile Applications for Emergency Response: Citizens Acting as Human Sensors," *Sensors*, 2016.
DOI: 10.3390/s16030406
Available at: https://www.mdpi.com/1424-8220/16/3/406

[9] L. Sun, "Overview and Evaluation of Notifications Systems for Existing M2M," *Helsingfors universitet*, 2014.
Available at: https://helda.helsinki.fi/handle/10138/144248

[10] L. DeNardis, „E-health Standards and Interoperability," *ITU-T Technology Watch*, 2012.
Available at:
https://www.itu.int/dms_pub/itu-t/oth/23/01/T23010000170001PDFE.pdf

[11] M. Eichelberg, J. Riesmeier, T. Aden, A. Dogac şi G. Laleci, „Electronic Health Record Standards - A Brief Overview," *Research Gate*, 2006.
DOI: 10.1109/ITICT.2006.358222
Available at:
https://www.researchgate.net/publication/267975385_Electronic_Health_Record_Standards_-_A_Brief_Overview

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 și 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

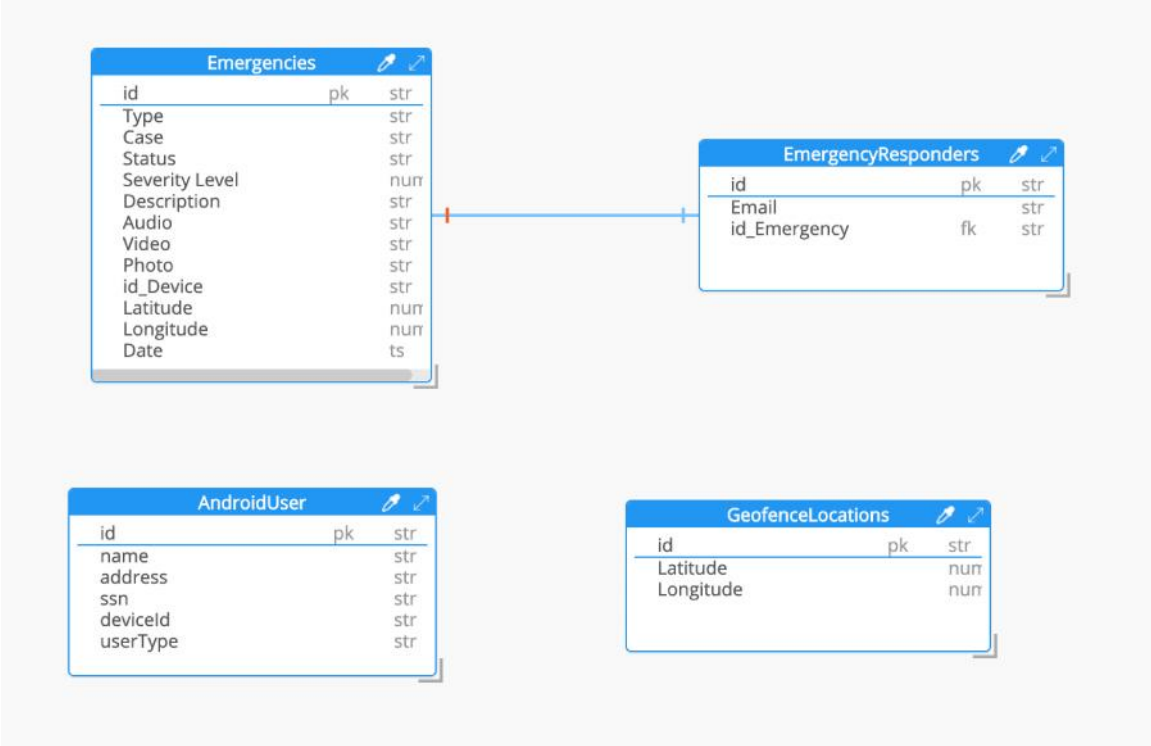
Anexa 2 – Lista figurilor

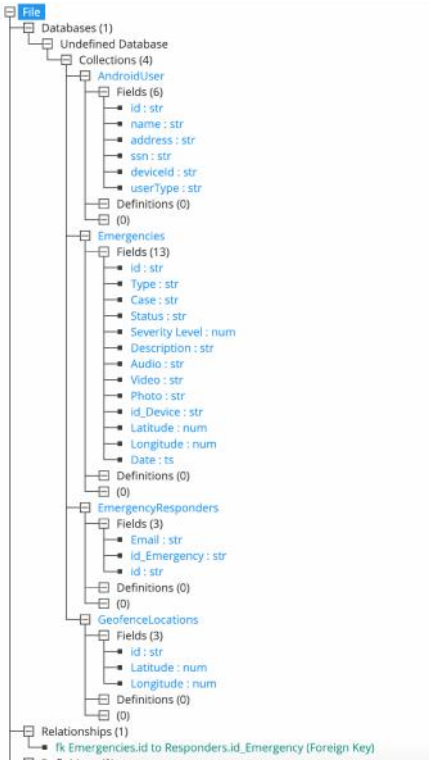
Anexa 3 – Lista tabelelor

Anexa 4 – Glosar de termeni

Termen	Descriere
MHealth	Denumirea sistemului din care face parte modulul implementat
API (Application Programming Interface)	Set de definiții de sub-programe, protocoale și unelte pentru programarea de aplicații software
SDK (Software Development Kit)	Colecție de unelte în folosul dezvoltării software
EHR (Electronic Health Records)	Colecție sistematizată de informații medicale într-un format digital
IDE (Integrated Development Environment)	Aplicație software cu rol de mediu de dezvoltare a aplicațiilor
NoSql (Not only Structured query language)	Mecanism pentru stocarea și obținerea datelor modelate în alte moduri față de bazele de date relaționale Sql
JVM (Java Virtual Machine)	Mașina virtuală care permite unui computer să execute programe Java
ML Kit (Machine Learning Kit)	SDK mobil destinat API-urilor Google pentru funcționalități bazate pe machine learning
UML (Unified Modeling Language)	Limbaj standard pentru descrierea de modele și specificații software
FCM (Firebase Cloud Messaging)	Soluție cross-platform pentru trimitere de mesaje către instanțe ale unei aplicații

Anexa 5 – Instrument Hackolade[53] pentru modelarea bazei de date





[53] https://hackolade.com/