

1. Introducere

1.1 Contextul

Într-o lume în care tehnologia evoluează tot mai mult, în care aceasta își pune din ce în ce mai mult amprenta pe viața noastră de zi cu zi, medicina este una din ariile care s-a bucurat de o atenție mare. Deși de-a lungul ultimilor ani, au apărut multe tehnologii noi care revoluționează atât speranța de viață a oamenilor cât și munca și pregătirea medicilor și a doctorilor, unele aspecte chiar foarte minore au rămas neglijate. Deși am ajuns să operăm la o precizie de microni, să avem roboți asistenți sau să învățăm pe holograme, lucruri care acum nu mult timp păreau luate din filme, am uitat un aspect foarte important: pacientul.

1.2 Motivația

În momentul actual, aplicațiile medicale care sunt oferite pe piață sunt împărțite în două mari categorii: aplicațiile pentru folosința cadrelor medicale și aplicațiile pentru folosința tuturor. Fiecare categorie are avantajele și dezavantajele sale.

Pentru cadrele medicale ce lucrează într-un mediu privat, aplicațiile sunt de obicei doar de gestiune a pacienților, de a face programări etc. fără a da prea multe opțiuni pacientului. Cele din mediul spitalelor sau clinicilor de stat sunt la fel, și deși au o bază de date comună și folositoare în multe cazuri, acestea sunt gândite în mare parte ca o aplicație de gestiune economică, fără a fi gândite pentru nevoile medicilor sau a pacienților. În plus rareori sunt acestea ținute la zi cu nevoile curente sau oferită mentenanță, și din păcate aspectul vizual și interfața cu utilizatorul lasă de dorit.

Pentru pacienți, sau simpli oameni preocupați de starea lor de sănătate în schimb, exista o multitudine de aplicații venite în folosul lor. De la aplicații embeded, web sau mobile, până la o combinație dintre acestea. Deși aceste aplicații de cele mai multe ori sunt ținute la zi cu nevoile utilizatorilor, au un aspect plăcut și sunt ușor de utilizat, acestea sunt lipsite de o bază de date specializată, completată de un cadru medical specializat. Un alt dezavantaj este faptul că de obicei aceste aplicații sunt axate pe un domeniu singular, iar pentru date mai complete ar trebui ca utilizatorul să aibă o multitudine de aplicații instalate.

Problema generală este că deși aceste două tipuri de aplicații există de sine stătătoare, ele nu funcționează concomitent sau cel puțin nu pot interacționa. Orice pacient ar trebui să poată vedea istoricul său medical, eventualul tratament primit, să poată interacționa cu cadrele medicale în cazul în care acesta este internat sau îi este oferită o anumită îngrijire medicală de un anume tip. În același timp, cadrul medical specializat trebuie să poată să își desfășoare activitatea în continuare într-un mod cât mai rapid și eficient.

1.3 Descrierea proiectului

Având în vedere nevoile generale ale utilizatorilor și a cadrelor medicale și a probleme identificate mai sus, am găsit următoarele funcționalități și caracteristici:

- **Mobilitate**
 - Aplicația trebuie să fie în variantă mobilă, atât pentru cadrele medicale cât și pentru pacienți (utilizatori).
 - Cadrele medicale ar trebuie să poată avea în orice moment acces la toate datele necesare, și să nu trebuiască să fie limitate de o anumită locație.
 - Pacienții (utilizatorii) ar trebui să poată avea acces oricând la datele lor medicale, reușind astfel să introducem o întreaga bibliotecă de acte, analize și istorice medicale într-un singur telefon sau tabletă.
- **Costuri reduse pentru platforme**
 - Având în vedere faptul că această aplicație ar trebui să poată fi accesibilă de marea masă de oameni dar și instituții medicale, aceasta trebuie implementată pe o platformă disponibilă aproape oricui.
- **Acces la propriul istoric medical**
 - Pacientul ar trebui să poată accesa istoricul medical fără a fi nevoit să îl introducă el sau să folosească multe aplicații. Acesta ar trebui să poată accesa direct baza de date folosită de cadrul medical.
- **Cererea asistenței medicale**
 - Pe timpul internării, pacientul ar trebui să poată cere asistență medicală fără a trebui să se deplaseze, mai ales că în unele cazuri această deplasare este imposibilă
- **Implementarea funcționalităților necesare cadrelor medicale**
 - Aplicația trebuie să dispună în continuare de facilitățile care le au medicii în momentul actual, poate chiar să le extindă
- **O interfață prietenoasă**
 - Aplicația trebuie să aibă o interfață ușor de înțeles și utilizat și să ofere informațiile necesare oricărui tip de utilizator.
- **Împărtășirea istoricului medical cu rude sau prieteni**
 - Aplicația ar trebui să permită împărtășirea istoricului medical cu alți membri ai familiei sau prieteni apropiați, mai ales pentru persoanele vârstnice.

Luând în considerare caracteristicile de mai sus, aplicația va avea următorul format:

- **Utilizarea sistemului Android**

- Aplicația va fi dezvoltată pentru dispozitive cu sisteme de operare Android pentru a rezolva chiar 2 caracteristici cheie:
 - Platformele ce rulează Android sunt mobile
 - Platformele ce rulează Android impun costuri reduse pentru platformă. Acestea se găsesc cu prețuri variabile dar sunt și platforme foarte ieftine ce vor putea rula aplicația. Desigur, acest lucru este în comparație cu restul platformelor populare.

- **Aplicația va avea trei tipuri de utilizatori**

- Pentru a îndeplini cerințele acces la propriul istoric medical, cererea asistenței medicale și împărtășirea istoricului medical cu rude sau prieteni, aplicația va avea trei tipuri de utilizatori, fiecare având diferite nivele de acces și funcții specifice. Cele trei tipuri de utilizatori vor fi:
 - Cadru medical
 - Pacient
 - Familie

- **Folosirea unor elemente vizuale plate**

- Pentru a asigura o interfață vizuale plăcută, aplicația va conține doar elemente plate și familiare din setul “Material Design Android”.

2. Fundamentare teoretică

2.1 Sistemul de operare Android

Android este un sistem de operare și o platformă pentru telefoane mobile, tablete, ceasuri deștepte și mai nou și alte dispozitive precum televizoare și mașini. Acesta este bazat pe nucleul Linux însă permite dezvoltatorilor să scrie aplicații în limbajul Java prin intermediul unor biblioteci dezvoltate de Google sau după caz, de alte terțe. Deși la început, sistemul de operare nu era în totalitate publicat, început cu 2008 acesta a devenit produs software cu licență de tip Open Source, motiv în plus pentru dezvoltatori să investească timp și să creeze diverse aplicații ieșite din comun. [1]

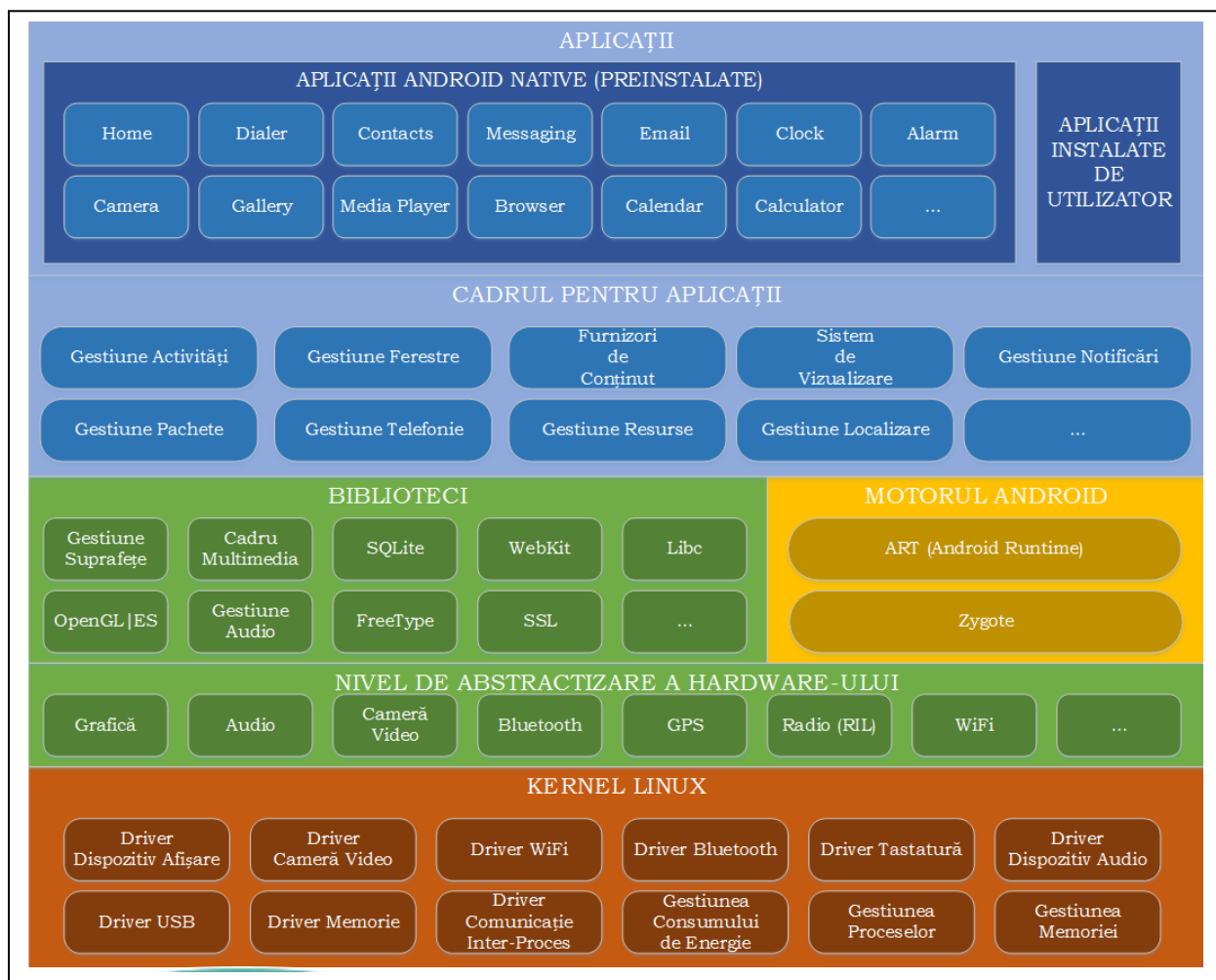


Fig. 2.1 - Arhitectura Sistemului de Operare Android [2]

Capacitățile platformei sunt numeroase, făcând această platformă și mai potrivită pentru orice tip de aplicație mobilă. Printre caracteristicile suportate actul regăsim [3] [4] :

- Configurații dispozitive - adaptabilă la configurații mai mari, VGA, biblioteci grafice 2D și biblioteci grafice 3D.
- Stocare de date
- Conectivitate – suportă tehnologii de conectivitate incluzând GSM/EDGE/HSDPA/LTE, CDMA, EV-DO, UMTS, Bluetooth și Wi-Fi.
- Mesagerie instant
- Navigatorul de web
- Mașina virtuală Dalvik
- Suport media
- Suport hardware additional - poate utiliza camere video/foto, touchscreen, GPS, accelerometru, cititor de amprente, diverse periferice și grafică accelerată 3D.
- Mediu de dezvoltare
- Piața Android
- Multi-touch
- Și altele...

2.2 Android Studio

2.2.1 Noțiuni generale

Mediul de dezvoltare Android Studio a fost anunțat în 2013 la conferința Google I/O și a fost realizat ca și o alternativă a mediului de dezvoltare Eclipse. Acesta este bazat pe mediul de dezvoltare Java a celor de la IntelliJ IDEA. În scurt timp, acesta a devenit mediul oficial de dezvoltare a aplicațiilor Android. [5]

Android Studio oferă următoarele facilități:

- Un sistem flexibil de construcție bazat pe Gradle
- Un emulator rapid și bogat în caracteristici
- Un mediu unificat în care vă puteți dezvolta pentru toate dispozitivele Android
- Rulare Instantă pentru a împinge modificările aplicației în desfășurare fără a construi un nou APK
- Șabloane de coduri și integrare GitHub pentru a vă ajuta să construiți caracteristici comune ale aplicațiilor și să importați coduri de probă
- Instrumente și cadre de testare extinse
- Instrumente pentru scanare pentru a prinde performanța, gradul de utilizare, compatibilitatea versiunilor și alte probleme
- Suport C++ și NDK
- Suport încorporat pentru Google Cloud Platform, facilitând integrarea serviciului Google Cloud Messaging și a aplicației Engine Engine

2.2.2 Structura proiectului

Fiecare proiect din Android Studio conține unul sau mai multe module cu fișiere de cod sursă și fișiere de resurse. Tipurile de module includ:

- Module de aplicații Android
- Module de bibliotecă
- Modulele Google App Engine

În mod implicit, aplicația Android Studio afișează fișierele de proiect în vizualizarea proiectului Android, după cum se arată în figura 2.1. Această vizualizare este organizată de module pentru a oferi acces rapid la fișierele sursă cheie ale proiectului.

Toate fișierele de construcție sunt vizibile la nivelul superior în cadrul scripturilor de grad și fiecare modul de aplicație conține următoarele dosare:

- Manifestul: conține fișierul AndroidManifest.xml.
- Java: conține fișierele de cod sursă Java, inclusiv codul de testare JUnit.
- Res: conține toate resursele non-cod, cum ar fi layout-uri XML, șiruri de caractere UI și imagini bitmap.

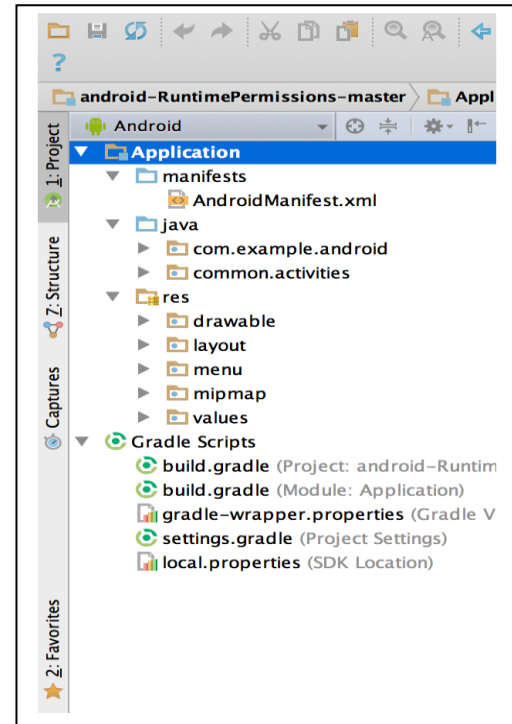


Fig 2.2 – Structura unui proiect

2.2.3 Interfața cu utilizatorul

Fereastra principală Android Studio este alcătuită din mai multe zone logice identificate în figura 2.1.

- 1 Bara de instrumente vă permite să efectuați o gamă largă de acțiuni, inclusiv difuzarea aplicației și lansarea de instrumente Android.
- 2 Bara de navigare vă ajută să navigați prin proiectul dvs. și să deschideți fișierele pentru editare. Acesta oferă o vedere mai compactă a structurii vizibile în fereastra proiectului.
- 3 Fereastra de editor este locul în care creați și modificați codul. În funcție de tipul de fișier curent, editorul se poate schimba. De exemplu, când vizualizați un fișier de aspect, editorul afișează Editorul de Layout.
- 4 Bara de ferestre a instrumentului rulează în afara ferestrei IDE și conține butoanele care vă permit să extindeți sau să restrângeți ferestrele individuale ale uneltelor.

- 5 Ferestrele de instrumente vă oferă acces la anumite sarcini, cum ar fi gestionarea proiectelor, căutarea, controlul versiunilor și multe altele. Le puteți extinde și le puteți restrânge.
- 6 Bara de stare afișează starea proiectului dvs. și IDE în sine, precum și orice avertismente sau mesaje.

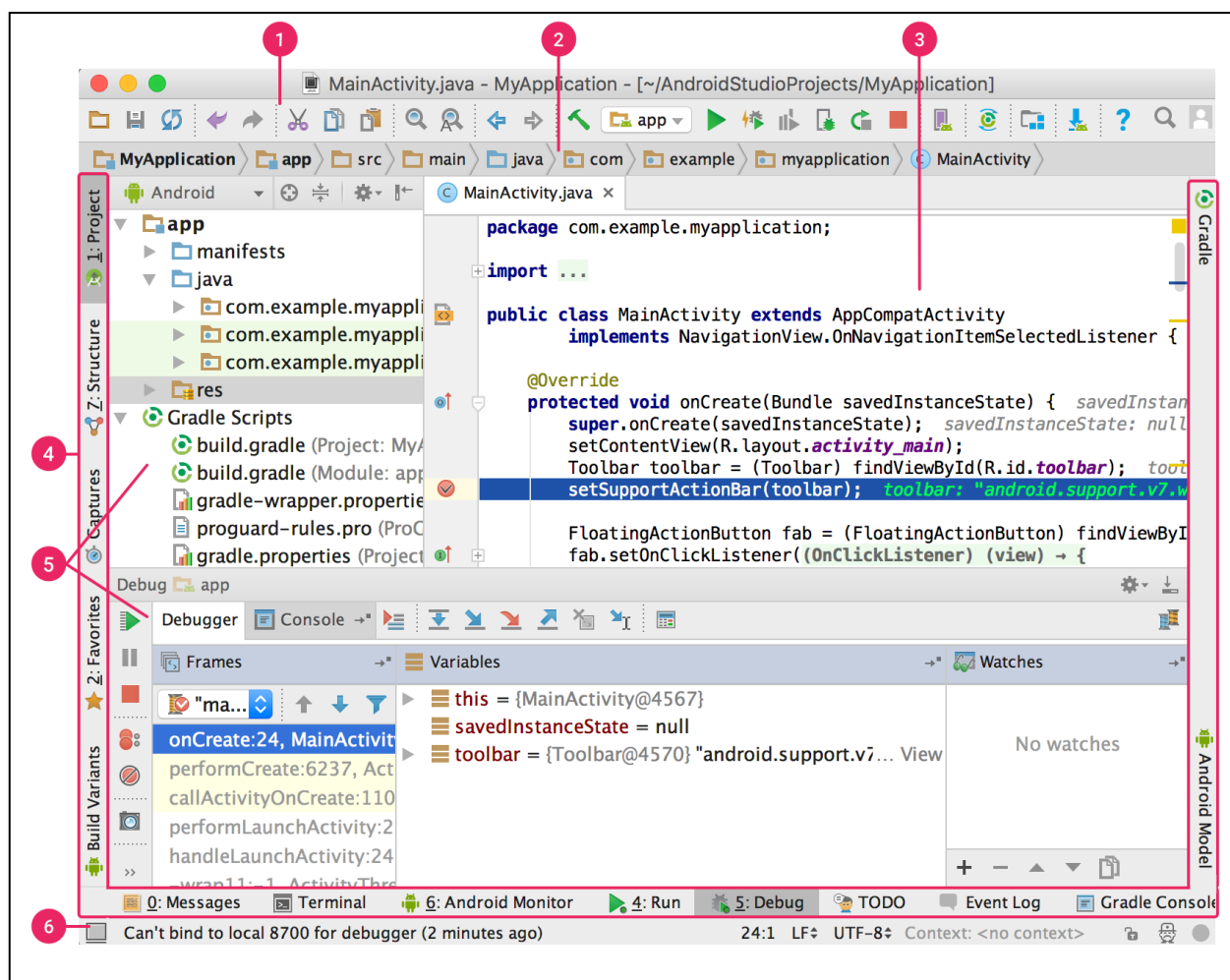


Fig 2.3 – Android Studio IDE

2.3 Limbajul de programare Java

Java este un limbaj de programare orientat-obiect, conceput de către James Gosling la începutul anilor '90, fiind lansat în 1995. Cele mai multe aplicații distribuite sunt scrise în Java, iar noile evoluții tehnologice permit utilizarea sa și pe dispozitive mobile de genul telefon, agenda electronică, palmtop etc. În felul acesta se creează o platformă unică, la nivelul programatorului, deasupra unui mediu eterogen extrem de diversificat. Acesta este utilizat în prezent cu succes și pentru programarea aplicațiilor destinate intranet-urilor [6].

Limbajul împrumută o mare parte din sintaxă de la C și C++, dar are un model al obiectelor mai simplu și prezintă mai puține facilități de nivel jos. Un program Java compilat, corect scris, poate fi rulat fără modificări pe orice platformă care e instalată o mașină virtuală Java. Acest nivel de portabilitate (inexistent pentru limbaje mai vechi cum ar fi C) este posibil deoarece sursele Java sunt compilate într-un format standard numit cod de octeți care este intermediar între codul mașină (dependent de tipul calculatorului) și codul sursă.

Mașina virtuală Java este mediul în care se execută programele Java. În prezent, există mai mulți furnizori de JVM, printre care Oracle, IBM, Bea, FSF. În 2006, Sun a anunțat că face disponibilă varianta sa de JVM ca open-source.

2.4 Baze de date

2.4.1 Noțiuni generale

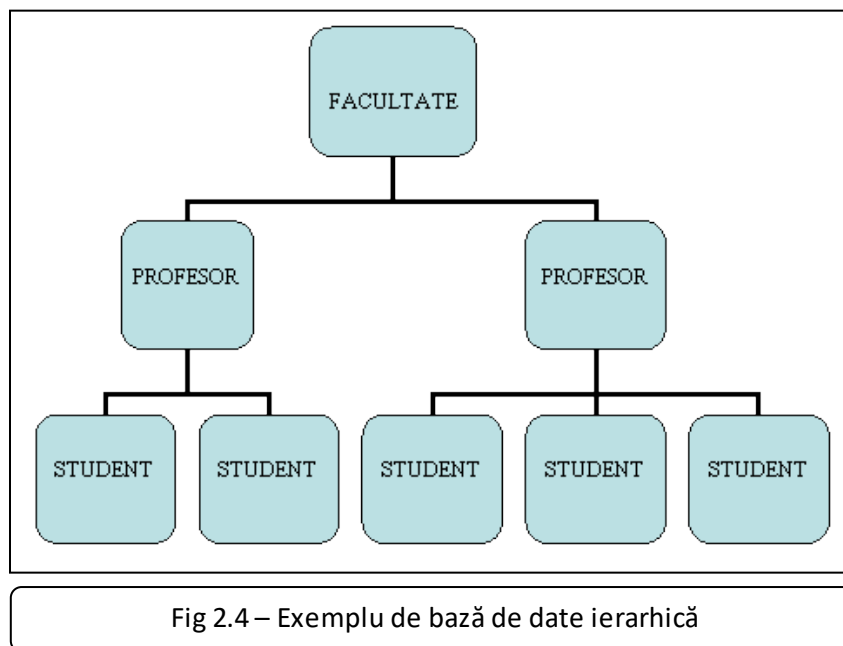
Sistemele de baze de date sunt o componentă importantă a vieții de zi cu zi în societatea modernă. Zilnic, majoritatea persoanelor desfășoară activități care implică interacțiunea cu o bază de date: depunerea sau extragerea unei sume de bani din bancă, rezervarea biletelor de tren sau de avion, căutarea unei cărți într-o bibliotecă computerizată, gestiunea angajaților dintr-o firmă, cumpărarea unor produse etc. Bazele de date pot avea mărimi (număr de înregistrări) și complexități extrem de variate, de la câteva zeci de înregistrări (de exemplu, baza de date pentru o agendă de telefon a unei persoane) până la zeci și sute de mii poate chiar milioane de înregistrări. Marea majoritate a sistemelor de baze de date existente în momentul de față sunt relaționale și există un număr mare de astfel de sisteme comerciale care pot fi achiziționate și folosite pentru propriile dezvoltări.

În sensul larg, o bază de date (database) este o colecție de date corelate din punct de vedere logic, care reflectă un anumit aspect al lumii reale și este destinat unui anumit grup de utilizatori. În acest sens, bazele de date pot fi create și menținute manual sau computerizat așa cum sunt majoritatea bazelor de date în momentul de față. O definiție într-un sens mai restrâns a unei baze de date este următoarea: O bază de date este o colecție de date centralizate, creată și menținută computerizat, în scopul prelucrării datelor în contextul unui set de aplicații. Prelucrarea datelor se referă la operațiile de introducere, ștergere, actualizare și interogare a datelor. [7]

2.4.2 Tipuri de baze de date

1. Modelul de date ierarhic

A fost primul model folosit pentru dezvoltarea bazelor de date, legaturile dintre date fiind ordonate unic, accesul se face numai prin varful ierarhiei, un subordonat nu poate avea decat un singur superior direct si nu se poate ajunge la el decat pe o singura cale [8]. (Figura 2.3)



2. Modelul de date retea

Este modelul în care datele sunt reprezentate ca într-o multime de ierarhii, în care un membru al ei poate avea oricâți superiori, iar la un subordonat se poate ajunge pe mai multe cai. Deosebirea față de modelul ierarhic constă în faptul că în modelul retea asocierile se reprezintă prin duplicarea înregistrărilor, fiecare înregistrare putând fi referită de mai multe înregistrări, ceea ce elimină redundanța datelor. [9]

Modelul retea (Network Model) folosește o structură de graf pentru definirea schemei conceptuale a bazei de date; nodurile grafului sunt tipuri de entități (înregistrări – records), iar muchiile grafului reprezintă în mod explicit asocierile (legăturile – links) dintre tipurile de entități.

3. Modelul de date relational

Este modelul de baze de date cel mai utilizat in prezent in gestiunea bazelor de date. Structura de baza a datelor este aceea de relatie-tabel.

Modelul relational (Relational Model) se bazeaza pe notiunea de relatie (relation) din matematica, care corespunde unei multimi de entitati de acelasi tip si are o reprezentare usor de inteles si de manipulat, ce consta intr-un tabel bidimensional, compus din linii si coloane. Fiecare linie din tabel reprezinta o entitate atribut corespunzand unei coloane a tabelului.

4. Modelul de date distribuite

Este rezultatul integrarii tehnologiei bazelor de date cu cea a retelelor de calculatoare, fiind baze de date logic integrate dar fizic distribuite pe mai multe sisteme de calcul.

5. Modelul de date semantice

Orientate spre obiecte, spre reprezentarea semnificatiei datelor, structura de baza folosita este cea de clasa de obiecte, definita prin abstractizare din entitatea fizica.

6. Modelul logic de date

Orice sistem de gestiune a bazei de date pentru a manipula o baza de date foloseste un anumit tip de model logic de date fundamentale (ierarhice, retea, relationale, orientate obiect) si derivate (distribuite).

7. Modelul orientat obiect

Marcheaza trecerea la a treia generatie de baze de date. El aduce bazelor de date un plus de deschidere, flexibilitate si da rezultate bune pentru probleme mari si complexe. In structura sunt acceptate toate tipurile de date cunoscute, putandu-se aplica in toate domeniile de activitate. Comunicarea intre obiecte se face prin mesaje, actualizarea metodelor, actualizarea proprietatilor, actualizarea claselor, realizarea legaturilor intre clase, actualizarea instantelor.

2.4.3 Baze de date SQL și NoSQL

1. SQL

În lumea tehnologiilor de baze de date, există două tipuri principale de baze de date: SQL și NoSQL - sau baze de date relaționale și baze de date non-relaționale. Diferența vorbește despre modul în care sunt construite, tipul de informații pe care le stochează și modul în care stochează. Bazele de date relaționale sunt structurate, cum ar fi cărțile de telefon care stochează numere de telefon și adrese. Bazele de date non-relaționale sunt orientate spre documente și distribuite, cum ar fi dosarele de fișiere care dețin totul de la adresa persoanei și numărul de telefon la preferințele de pe Facebook și preferințele de cumpărături online.

În primul rând, să aruncăm o privire asupra uneia dintre principalele caracteristici care separă aceste două sisteme: modul în care structurează datele. O bază de date relațională - sau o bază de date SQL, numită pentru limba în care este scris, Language Structured Query (SQL) - este modul mai rigid și structurat de stocare a datelor, cum ar fi o carte telefonică. Fiecare rând reprezintă o intrare, iar fiecare coloană sortează un tip foarte specific de informații, cum ar fi numele, adresa și numărul de telefon. Relația dintre tabele și tipurile de câmpuri se numește o schemă. Într-o bază de date relațională, schema trebuie să fie definită clar înainte de adăugarea oricăror informații.

Pentru ca o bază de date relațională să fie eficientă, datele pe care le stocați trebuie să fie structurate într-un mod foarte organizat. O schemă bine concepută minimizează redundanța datelor și împiedică sincronizarea tabelor, o caracteristică critică pentru multe companii, în special cele care înregistrează tranzacțiile financiare. O schemă prost concepută poate duce la dureri de organ organizaționale datorită rigidității sale. De exemplu, o coloană destinată să stocheze numere de telefon din S.U.A. poate necesita 10 cifre, deoarece este standard pentru numerele de telefon din S.U.A. Acest lucru are avantajul de a respinge orice valori nevalide (de exemplu, în cazul în care un număr lipsește un cod de zonă). Cu toate acestea, dacă trebuie să schimbați schema (de exemplu, dacă trebuie să includeți o intrare internațională cu mai mult de 10 cifre), atunci trebuie modificată întreaga bază de date. Avantajele cheie: organizația excelentă are ca rezultat un compromis în flexibilitate cu o bază de date relațională.

Limbajul structurat de interogări (SQL) este un limbaj de programare folosit de arhitecții de baze de date pentru a proiecta baze de date relaționale. SQL este un limbaj ușor, declarativ, care face o mulțime de ridicare grele pentru baza de date relațională, acționând ca o versiune a unei baze de date a unui script de server. Un avantaj special al SQL este clauza JOIN simplă, dar puternică, care permite dezvoltatorilor să recupereze date corelate stocate pe mai multe tabele cu o singură comandă.

Un alt motiv pentru care bazele de date SQL rămân populare este că se potrivesc în mod natural în multe stiluri venerabile de software, inclusiv stive LAMP și Ruby. Aceste baze de date

sunt bine înțelese și susținute pe scară largă, ceea ce poate fi un avantaj major dacă întâmpinați probleme.

2. NoSQL

Dacă cerințele de date nu sunt clare la început sau dacă aveți de-a face cu cantități masive de date nestructurate, este posibil să nu aveți luxul de a dezvolta o bază de date relațională cu o schemă clar definită. Introduceți baze de date non-relaționale, care oferă o flexibilitate mult mai mare decât omologii lor tradiționali. Gândiți-vă la bazele de date non-relaționale mai degrabă ca dosare de fișiere, asamblarea de informații conexe de toate tipurile. Dacă un blog WordPress folosește o bază de date NoSQL, fiecare fișier ar putea stoca date pentru un post de blog: plăceri sociale, fotografii, text, metrice, link-uri și multe altele.

Datele nestructurate de pe web pot include date de senzori, partajare socială, setări personale, fotografii, informații despre locație, activitate online, valori de utilizare și multe altele. Încercarea de a stoca, procesa și analiza toate aceste date nestructurate a dus la dezvoltarea alternativelor fără schemă la SQL. Luate împreună, aceste alternative sunt numite NoSQL, adică "nu numai SQL". În timp ce termenul NoSQL cuprinde o gamă largă de alternative la bazele de date relaționale, ceea ce au în comun este că acestea vă permit să tratați datele mai flexibil.

Cum funcționează bazele de date NoSQL? În locul tabelelor, bazele de date NoSQL sunt orientate spre documente. În acest fel, datele nestructurate (cum ar fi articole, fotografii, date sociale, videoclipuri sau conținut într-o postare de blog) pot fi stocate într-un singur document care poate fi găsit cu ușurință, dar nu este neapărat clasificat în câmpuri ca relațional Baza de date nu. Este mai intuitiv, dar rețineți că stocarea în bloc a datelor în acest fel necesită efort suplimentar de procesare și stocare mai mult decât datele SQL foarte bine organizate. De aceea, Hadoop, o platformă open-source de calcul și analiză a datelor capabilă să proceseze cantități uriașe de date în nor, este atât de populară în combinație cu stive de baze de date NoSQL.

Bazele de date NoSQL oferă un alt avantaj major, în special pentru dezvoltatorii de aplicații: ușurința de acces. Bazele de date relaționale au o relație plină cu aplicațiile scrise în limbi de programare orientate pe obiecte, cum ar fi Java, PHP și Python. Bazele de date NoSQL sunt adesea capabile să ocolească această problemă prin intermediul API-urilor, care permit dezvoltatorilor să execute interogări fără a trebui să învețe SQL sau să înțeleagă arhitectura care stă la baza sistemului lor de baze de date.

2.5 Google Firebase

2.5.1 Noțiuni generale

Firebase este o platformă de dezvoltare a aplicațiilor mobile și web. Firebase este alcătuită din caracteristici complementare pe care dezvoltatorii le pot amesteca și potrivi pentru a le satisface nevoile. Echipa are sediul în San Francisco și Mountain View, California. Compania a fost fondată în 2011 de către Andrew Lee și James Tamplin. Produsul inițial al lui Firebase a fost o bază de date în timp real, care oferă un API care permite dezvoltatorilor să stocheze și să

sincronizeze date către mai mulți clienți. De-a lungul timpului, și-a extins linia de produse pentru a deveni o suită completă pentru dezvoltarea aplicațiilor. Compania a fost achiziționată de Google în octombrie 2014 și un număr semnificativ de caracteristici noi au fost prezentate în mai 2016 la Google I/O. [10]

2.5.2 Servicii

1. Firebase Analytics

Firebase Analytics este o soluție gratuită de măsurare a aplicațiilor, care oferă informații despre utilizarea aplicațiilor și implicarea utilizatorilor.

2. Firebase Cloud Messaging

Fiind cunoscut sub numele de Google Cloud Messaging (GCM), Firebase Cloud Messaging (FCM) este o soluție de tip cross-platform pentru mesaje și notificările pentru aplicații Android, iOS și web, care pot fi folosite în prezent fără costuri.

3. Firebase Auth

Firebase Auth este un serviciu care poate autentifica utilizatorii folosind doar codul client-side. Acesta susține furnizorii de servicii de conectare socială Facebook, GitHub, Twitter și Google. În plus, acesta include un sistem de gestionare a utilizatorilor prin care dezvoltatorii pot permite autentificarea utilizatorilor cu datele de conectare prin e-mail și parola stocate cu Firebase.

4. Realtime Database

Firebase oferă o bază de date în timp real și un backend ca serviciu. Serviciul furnizează dezvoltatorilor de aplicații un API care permite sincronizării datelor aplicațiilor între clienți și stocate în cloudul Firebase. Compania oferă biblioteci de clienți care permit integrarea cu aplicațiile Android, iOS, JavaScript, Java, Objective-C, swift și Node.js. Baza de date este, de asemenea, accesibilă printr-un API REST și legături pentru mai multe cadre JavaScript, cum ar fi AngularJS, React, Ember.js și Backbone.js. API-ul REST utilizează protocolul Server-Sent Events, care este un API pentru crearea conexiunilor HTTP pentru primirea notificărilor push de la un server. Dezvoltatorii care folosesc baza de date în timp real își pot securiza datele utilizând regulile de securitate impuse de server. [11]

5. Firebase Storage

Firebase Storage oferă încărcări și descărcări de fișiere sigure pentru aplicațiile Firebase, indiferent de calitatea rețelei. Dezvoltatorul poate să-l folosească pentru a stoca imagini, audio,

video sau alte conținuturi generate de utilizatori. Firebase Storage este susținut de Google Cloud Storage.

6. Firebase Hosting

Firebase Hosting este un serviciu static de gazduire de active care a fost lansat pe 13 mai 2014. Suportă găzduirea fișierelor statice cum ar fi CSS, HTML, JavaScript și alte fișiere care nu se modifică dinamic. Serviciul livrează fișiere printr-o rețea de difuzare de conținut (CDN) prin criptare HTTP Secure (HTTPS) și Secure Sockets Layer (SSL). Firebase colaborează cu Fast, un CDN, pentru a furniza suportul CDN Firebase Hosting. Compania declară că Firebase Hosting a crescut din cererile clienților; Dezvoltatorii folosesc Firebase pentru baza de date în timp real, dar au nevoie de un loc pentru a-și găzdui conținutul.

7. Firebase Test Lab for Android

Firebase Test Lab pentru Android oferă infrastructură cloud pentru testarea aplicațiilor Android. Cu o singură operație, dezvoltatorii pot iniția testarea aplicațiilor lor într-o gamă largă de configurații de dispozitive și dispozitive. Rezultatele testului - inclusiv jurnalele, videoclipurile și capturile de ecran - sunt puse la dispoziție în proiect în consola Firebase. Chiar dacă un dezvoltator nu a scris niciun cod de testare pentru aplicația lor, Test Lab poate să exercite aplicația în mod automat, căutând prăbușiri.

8. Firebase Crash Reporting

Raportarea cu Crash creează rapoarte detaliate despre erorile din aplicație. Erori sunt grupate în grupuri de urme de stive similare și triajate de severitatea impactului asupra utilizatorilor de aplicații. Pe lângă rapoartele automate, dezvoltatorul poate înregistra evenimente personalizate pentru a ajuta la captarea pașilor care duc la un accident.

9. Firebase Notifications

Firebase Notifications este un serviciu gratuit care permite notificări direcționate către utilizatori pentru dezvoltatorii de aplicații mobile.

10. Firebase App Indexing

Firebase App Indexing, fostul index Google App, obține o aplicație în Căutarea Google. Adăugarea indexării aplicațiilor promovează ambele tipuri de rezultate ale aplicațiilor în Căutarea Google și oferă, de asemenea, automate de completare a interogărilor.

11. Firebase Dynamic Links

Firebase Dynamic Links sunt adrese URL inteligente care modifică dinamic comportamentul pentru a oferi cea mai bună experiență pe diferite platforme.

12. Firebase Invites

Firebase Invites este o soluție transfrontalieră pentru trimiterea invitațiilor personalizate prin e-mail și prin SMS, a utilizatorilor la bord și măsurarea impactului invitațiilor.

13. Firebase Remote Config

Firebase Remote Config este un serviciu cloud care permite dezvoltatorilor să schimbe comportamentul și apariția aplicațiilor lor fără a le cere utilizatorilor să descarce o actualizare a aplicației.

14. Adwords

AdWords este un serviciu de publicitate online Google care se integrează cu Firebase pentru a permite dezvoltatorilor să vizeze utilizatorii folosind segmentarea Firebase AnalyticsAdmob.

2.4.3 Interfață

Interfața serviciului Firebase oferită de Google este una simplă și foarte intuitivă. Aceasta arată precum în figura 2.4. Ea conține un meniu în partea stângă iar la puterea unei apăsări de click, aceasta vă va oferi detaliile prezentate în capitolul 2.4.2. De obicei acestea sunt foarte bine structurate, oferă atât detalii individuale cat și detalii în ansablu sub formă de grafice sau scheme pentru o mai bună și structurată afișare.

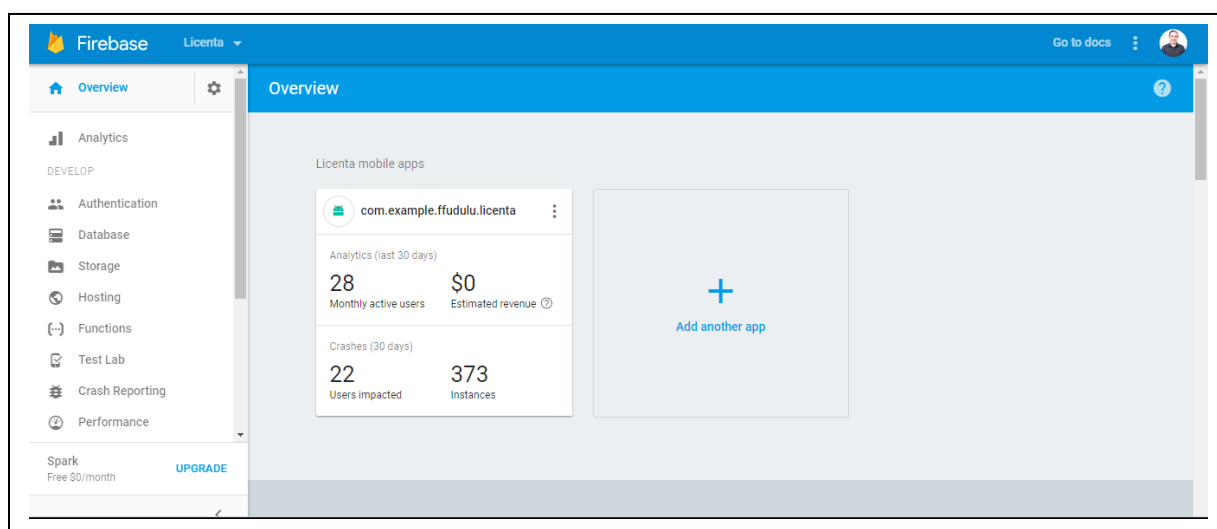


Fig. 2.5 – Pagina de pornire Google Firebase

3. Proiectarea aplicației

3.1 Analiza cerințelor

Prezentul sistem a fost inspirat din similitudinile actualului proces prin care trece un pacient la internare și în timpul internării în sistem de sănătate de stat din România. Mi-am propus ca la final să reușesc să implementez baza unui sistem mai amplu de gestionare a pacienților într-un spital și de a umple lipsa unei baze a pacientului, care poate fi vizualizată oricând și conexiunea dintre pacienți și cadre medicale într-un mod ușor și simplu. Acest sistem îmi propun a pune bazele unei aplicații modulare care să poată fi personalizată mai apoi pentru cerințele fiecărei secții.

Aplicația are trei tipuri de utilizatori:

- **Medicii (cadrele medicale)** – această categorie este împărțită în 2 subcategorii și anume:

- Doctori
- Asistenți

Această categorie va trebui să fie capabilă să facă internările într-un mod cât mai ușor și să vizualizeze: pacienții, profilul lor personal, istoricul medical sau alte date importante despre aceștia. Tot această categorie de utilizatori va trebui să poată să își vadă profilul personal, să îl modifice, să administreze noi tratamente sau să pună un diagnostic nou, să răspundă la chemarea unui pacient dacă acesta are nevoie de orice fel de asistență medicală sau să facă o externare.

Pentru orice cont nou de asistent, un doctor va trebui să valideze contul, iar pentru conturile de doctori, validarea se va face de către un administrator.

- **Pacienții** – Această categorie este formată, așa cum îi spune și numele, doar din pacienți. Aceștia au dreptul să își creeze un cont nou dar care va putea fi folosit doar după internare. Datele personale ale acestora vor fi introduse în momentul internării, de către un cadru medical specializat. Acest tip de utilizator va putea să își vadă datele personale cu care este introdus în baza de date, și să semnaleze eventuale greșeli sau modificări. Tot pacienții vor avea posibilitatea de a-și vizualiza tot istoricul medical detaliat, de la diagnostic analize până la tratamentul administrat. În plus, pe toată perioada internării, acesta poate cere asistență medicală prin simpla apăsare a unui buton, și va fi anunțat dacă un cadru medical este disponibil. În plus, acesta poate oferi sau retrage oricând accesul la istoricul său medical unui membru al familiei sau prieten. Acest lucru este folositor mai ales la persoanele vârstnice sau cele foarte tinere (copii).

- **Familia** – Această categorie a apărut în momentul în care mi-am dat seama de necesitatea împărtășirii istoricului medical în timp real cu unul din membrii familiei. De câte ori nu am auzit fraza “Și ce ți-au dat să iei?”. Da, un lucru riscant de făcut. Împărtășirea datelor personale cu alte persoane, dar cum spuneam mai sus, retragerea dreptului de a vizualiza aceste date este la puterea unei apăsări de buton. Acest cont, precum toate de mai sus, poate să își vadă datele personale și să vadă istoricul medical al unui pacient care i-a oferit accesul.

3.2 Proiectarea soluției

Sistemul prezent, la fel cum s-a menționat anterior, reprezintă o aplicație mobilă, disponibilă momentan doar pentru utilizatorii de telefoane mobile cu sistemul de operare Android. Așadar există, două componente principale și anume, partea de Client fiind aplicația mobilă, și partea de server.

Baza de date, conturile utilizatorilor cât și toate fișierele media, sunt stocate pe cloud, printr-un serviciu oferit de cei de la Google, și anume Google Firebase. Toate acțiunile din interiorul aplicație sunt făcute prin modificări la baza de date iar toate modificările făcute sunt mereu la zi pe orice dispozitiv cu acces la aplicație.

În cazul în care apare orice fel de eroare, utilizatorul este atenționat și toate erorile apărute sunt stocate online, tot cu ajutorul Google Firebase. Acesta mai apoi face statistici cu erorile apărute folosind date esențiale de genul: dispozitivul folosit, contextual, date despre internet, zona, etc. Aceste date sunt disponibile dezvoltatorului. În cazul în care aceste date nu sunt verificate, un email va fi trimis automat dezvoltatorului pentru a fi atenționat de noi erori în sistem pentru a putea fi rezolvate ulterior.

4. Proiectarea în detaliu

4.1 Arhitectura aplicației

4.1.1 Alegerea arhitecturii

Partea de client a aplicației pornește dintr-un proiect de tip Facory. A fost ales acest mod de structurare a aplicației datorită beneficiilor care le aduce. Datorită modului arhitectural, aplicația va putea fi dezvoltată în continuare fără mari probleme. Având în vedere că la momentul creării acestei aplicații mă aștept la o mărire a capacităților ei și nu știu ce fel de clase vor mai fi adăugate pe viitor, și pentru a reduce toate interfețele la cate una generală care va fi populată automat și care va arăta datele în funcție de tipurile selectate sau datele primite. Având în vedere argumentele prezentate mai sus, consider ca acest model arhitectural se mulează foarte bine pe aplicația dezvoltată.

4.1.2 Despre modelul arhitectural factory

În programarea bazată pe clasă, modelul arhitectural factory este un model creator care folosește metode factory pentru a rezolva problema creării de obiecte fără a fi nevoie să precizeze clasa exactă a obiectului care va fi creat. Acest lucru se realizează prin crearea de obiecte prin apelarea unei metode din factory - fie specificată într-o interfață și implementată de clase de copii, fie implementată într-o clasă de bază și opțional suprascrisă de clase derivate - mai degrabă decât apelând un constructor.

Metoda Factory face un design mai personalizabil și doar un pic mai complicat. Alte modele de design necesită clase noi, în timp ce Factory Method necesită doar o operație nouă. Oamenii folosesc adesea Metoda de fabricare ca modalitate standard de a crea obiecte. Dar nu este necesar dacă: clasa care este instanțiată nu se modifică niciodată, sau instanțierea are loc într-o operație pe care subclasele o pot suprascrie cu ușurință (cum ar fi o operație de inițializare).

Metodele de fabricare sunt specificate în mod obișnuit de un cadru arhitectural și apoi sunt implementate de către utilizatorul cadrului.

4.1.3 Arhitectura propriu-zisă

Activitatea principală (MainDrawer.java), va apela clasa Initialization.java. Aceasta, are o metodă care determină tipul de utilizator folosit, verifică starea (logat / nu este logat/ și altele) și creează obiectul corespunzător nefiind nevoie de a crea mai multe activități pentru această operațiune. În continuare, clasa Initialization.java va rula în fundal și va detecta orice schimbare a bazei de date. În momentul unei schimbări, aceasta va reinițializa obiectul schimbat și va fi schimbat instant și automat în toate interfețele utilizatorului. În activitatea principală, în funcție de datele preluate din baza de date în momentul inițializării, se conferă acces la anumite activități.

Ca și clase pentru stocarea datelor avem următoarele:

- Tipuri de utilizatori
 - UserMedic
 - Aici sunt stocate toate datele personale a unui cadru medical
 - Având în vedere că toate datele personale sunt obligatorii, acesta are un singur constructor și un setter pentru a stabili dacă respectivul cont a fost activat sau nu
 - UserPacient
 - Aici sunt stocate toate datele personale ale unui pacient
 - Având în vedere procedurile, aceasta are 2 constructori, unul în care se salvează doar UID-ul și adresa de email și încă unul care se apelează în momentul în care se face internarea.
 - UserFamily
 - Aici sunt stocate toate datele personale ale unui membru al familiei
 - Având în vedere ca toate datele personale sunt obligatorii, acesta are un singur constructor, apelat la înregistrare și un setter pentru a permite sau respinge accesul la istoricul medical de către pacient.
- Date medicale
 - Internare
 - Conține data internării/externării, un vector care conține linkuri către analize și o listă cu obiecte de tip Tratament care va conține toate tratamentele.
 - Acesta conține 4 tipuri de constructori. Unul care conține toate datele precizate, unul pentru momentul în care se schimbă analizele și tratamentul dar rămâne aceeași internare, unul pentru schimbarea tratamentului și unul pentru externare.
 - Tratament
 - Conține toate necesare unui tratament: diagnosticul, medicația, data aplicării tratamentului și alte detalii

- Acesta conține constructori pentru diagnostic nou cu tratament nou sau pentru un tratament nou cu același diagnostic
- Notificări
 - Conține mesajul notificării, utilizatorul care a declanșat notificarea, tipul de acces la notificare, prioritatea, dacă a fost sau nu rezolvată problema și data și ora la care s-a declanșat

Toate clasele de mai sus conțin pe lângă cele specificate și un constructor gol și getters pentru a putea fi preluate datele din baza de date.

4.2 Structura bazei de date

4.2.1 Despre baza de date Firebase

Baza de date Firebase Realtime vă permite să construiți aplicații bogate, de colaborare, permițând accesul securizat la baza de date direct din codul clientului. Datele sunt persistente la nivel local și, chiar și offline, evenimentele în timp real continuă să se declanșeze, oferind utilizatorului final o experiență receptivă. Când dispozitivul își recapătă conexiunea, baza de date Realtime sincronizează modificările de date locale cu actualizările de la distanță care au avut loc în timp ce clientul era offline, fuzionând automat conflictele.

Baza de date Realtime oferă o limbă flexibilă bazată pe expresii, numită Reguli de securitate bazate pe Firebase Realtime Database, pentru a defini modul în care trebuie să fie structurate datele dvs. și când pot fi citite sau scrise date. Atunci când sunt integrate cu Firebase Authentication, dezvoltatorii pot defini cine are acces la ce date și cum pot să le acceseze.

Baza de date Realtime este o bază de date NoSQL și ca atare are diferite optimizări și funcționalități în comparație cu o bază de date relațională. API-ul bazei de date Realtime este conceput pentru a permite doar operațiuni care pot fi executate rapid. Acest lucru vă permite să construiți o experiență mare în timp real, care poate servi milioane de utilizatori fără a compromite răspunsul. Din acest motiv, este important să vă gândiți la modul în care utilizatorii au nevoie să vă acceseze datele și apoi să le structureze în consecință.

În concluzie de ce am ales o bază de date NoSQL și de ce Firebase? Este simplu:

- Experiență foarte bună în timp real
- Capacitatea de a funcționa rapid chiar și la un număr mare de utilizatori
- Posibilitatea de validare a datelor
- Posibilitatea de extindere a bazei de date cu ușurință

4.2.2 Baza de date propriu-zisă

Deși baza de date este una de tip NoSQL, felul în care baza de date este structurată, nu este 100% tipic unei astfel de baze de date. În consecință, baza de date are 3 noduri principale (figura 4.4).

- “Notifications” conține toate notificările aplicației.
- “Tratamente” conține toate tratamentele (structurate pentru fiecare pacient).
- “Users” conține toți utilizatorii și datele lor personale.

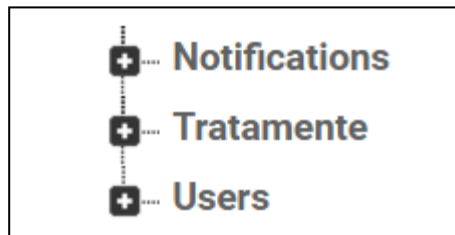


Fig. 4.4 – Nodurile principale ale bazei de date

În secțiunea de notificări, la fel cum puteți observa și în figura 4.5 , regăsim următoarele:

1. Primul nod de indentificare este practic data și ora la care a fost declanșată acțiunea. Astfel, toate notificările pot fi aranjate și găsite repede în ordinea în care au fost declanșate.
2. În secțiunea 2 din poză putem observa datele esențiale unei notificări. Putem să observăm cine are acces la această notificare, dacă cineva a rezolvat problema sau nu, mesajul ce trebuie afișat, prioritatea sau tipul de notificare.
3. Pentru un acces mai rapid la date esențiale precum datele persoanei care a declanșat notificarea, vom salva și aici un obiect de tipul utilizatorului cu datele aferente din se poate vedea și în secțiunea 3.

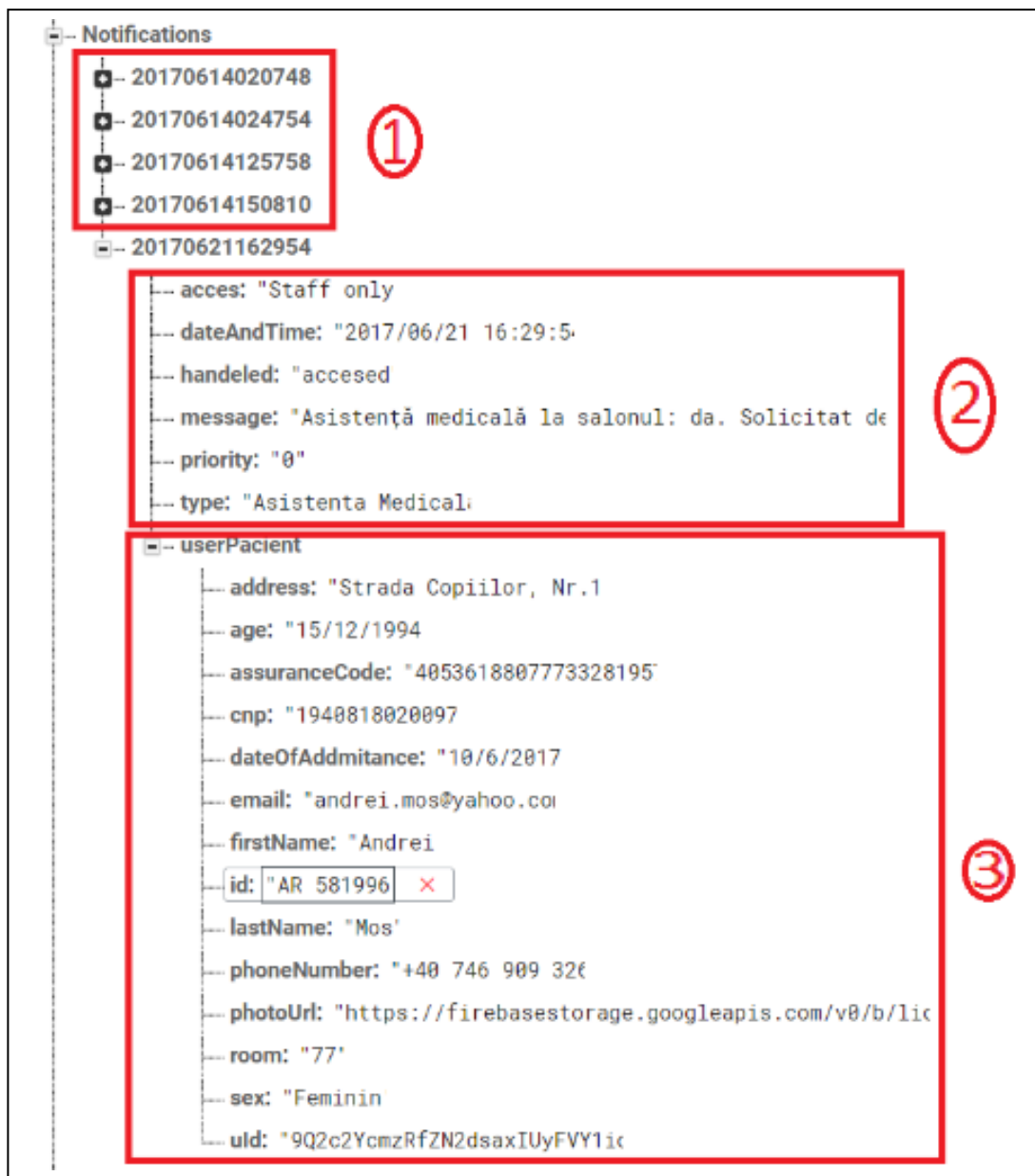


Fig. 4.5 – Structura bazei de date la nodul de notificări

În secțiunea Tratamente, după cum puteți vedea în figura 4.6 , putem regăsi următoarele:

1. Codul unic al pacientului. Orice adăugare se va face sub acest nod pentru nu a căuta în foarte multe locuri.
2. Data internării este următorul nod pentru a putea avea o evidență mai logică și ușoară bazată pe fiecare internare
3. În secțiunea 3 din figură putem observa data internării dar și data externării pacientului dacă asta exista și nu este încă internat.

4. Aici putem observa obiecte de tip Tratament aranjate într-o Listă de obiecte cu datele aferente clasei Tratament (data aplicării tratamentului, diagnostic, detalii și medicație)



Fig. 4.6 – Structura bazei de date la nodul de tratamente

În nodul Users, la fel cum îi spune și numele putem regăsi utilizatorii cu toate datele lor personale. După cum se poate observa în fig. 4.7 , regăsim:

1. Notat în figură cu 1, regăsim tipul de utilizator
2. Sub fiecare tip de utilizator regăsim o listă de utilizatori, fiecare nod reprezentând un ID unic.
3. Structura fiecărui tip de utilizator se pot observa în figurile 4.8 , 4.9 , 4.10 și 4.11



Fig. 4.7 – Structura bazei de date la nodul de utilizatori



Fig. 4.8 – Structura bazei de date la nodul asistenti



Fig. 4.9 – Structura bazei de date la nodul familie



Fig. 4.10 – Structura bazei de date la nodul de doctori

4.3 Organigrama

4.3.1 Crearea unui cont nou

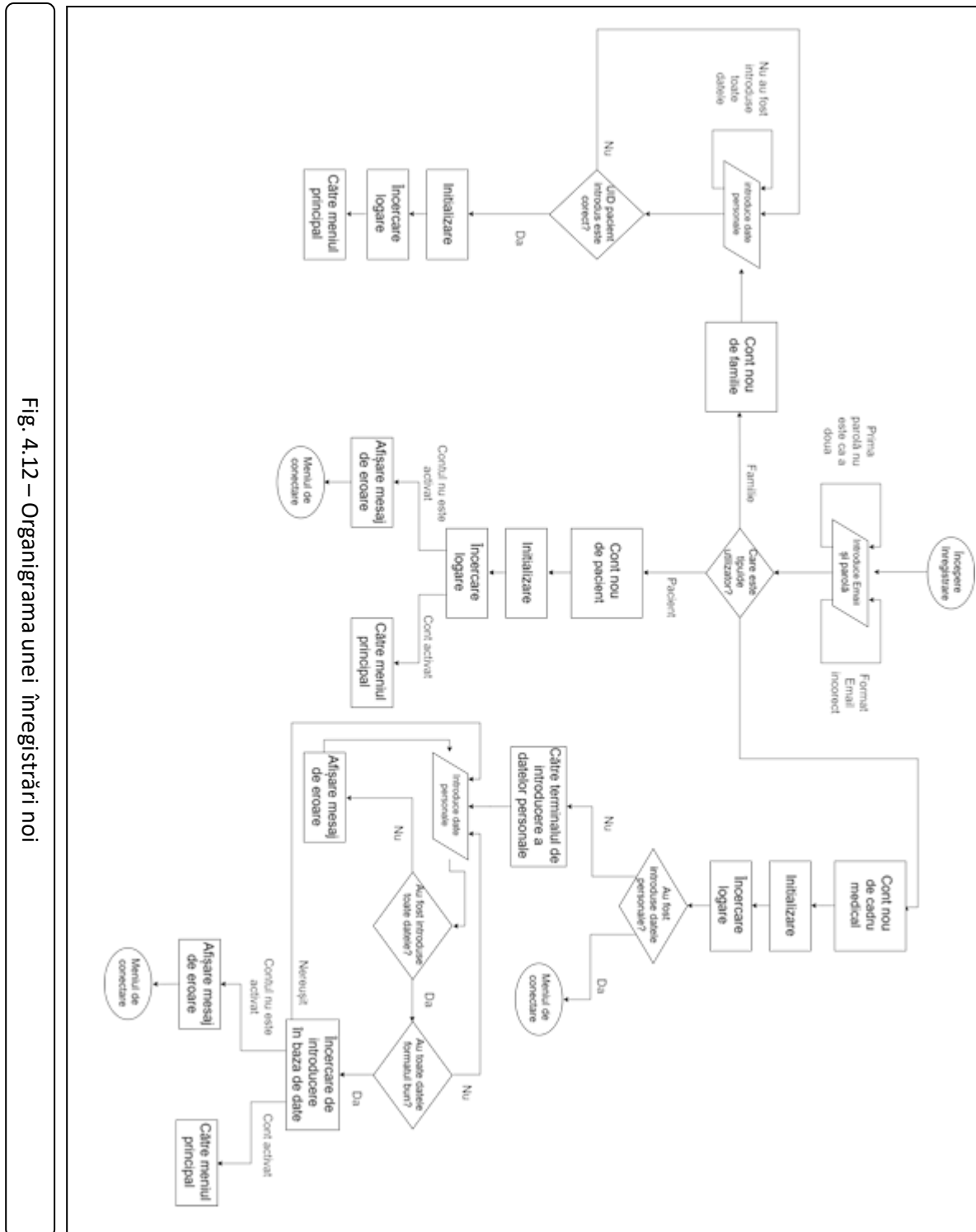


Fig. 4.12 – Organigrama unei înregistrări noi

După cum se poate observa în figura 4.12, procesul de înregistrare este unul complex mai ales pentru cadrele medicale. Procesul este unul complex în ideea validărilor multiple ce au loc în momentul unui cont nou, procesul efectiv fiind chiar unul simplu.

Ca și cont de pacient lucrurile sunt foarte simple datorită faptului ca toate datele vor fi introduse la internare iar pacientul nu va trebui să își bată capul cu problemele acestea. Ca și cont de familie practic utilizatorul trebuie doar să bage codul unic oferit de pacient iar acesta să fie corect. Ca și cadru medical, lucrurile se complică desigur. Aceștia trebuie să își introducă datele corespunzătoare foarte corect. Acestea mai apoi vor fi validate de către un administrator în cazul doctorilor sau de către un doctor în cazul asistenților.

Desigur tot acest procedeu pare complicat datorită tuturor validărilor ce se fac în fundal, însă pentru orice tip de utilizator, acest proces este chiar unul simplu. Eventualele probleme care ar fi putut apărea sunt automat tratate de către aplicație, astfel utilizatorul are o experiență plăcută.

4.3.2 Autentificarea

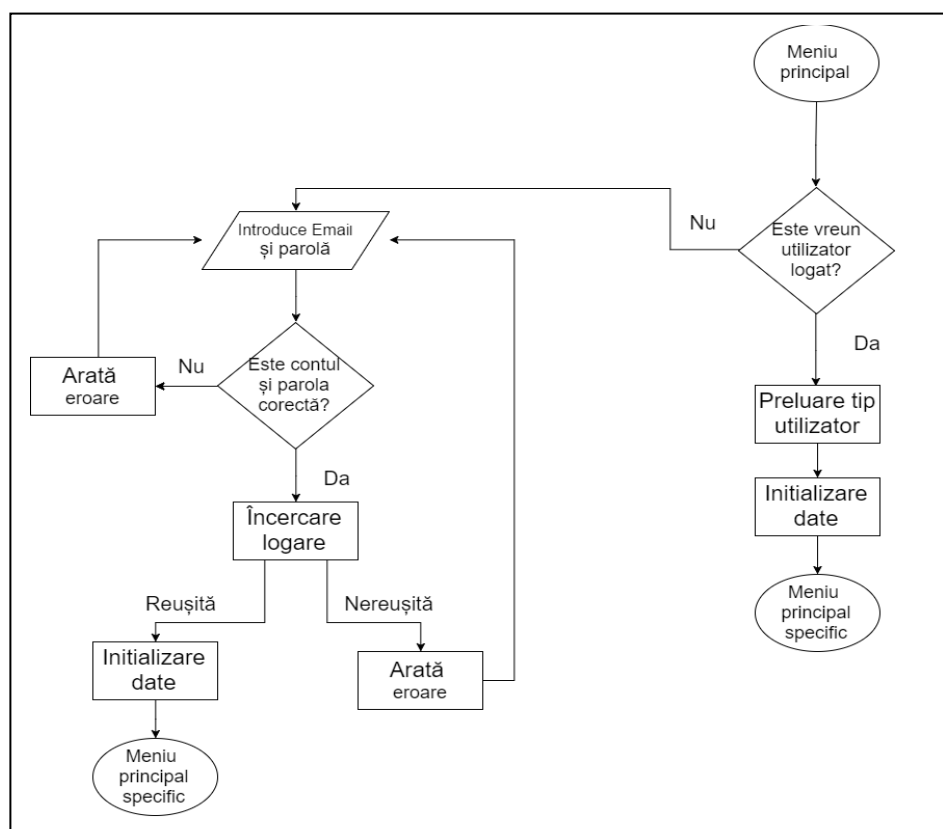


Fig. 4.13 – Organigramă intrare în cont

După cum se poate observa în figura 4.13, intrarea în cont este una standard, cu verificările specifice. După ce este confirmat ca utilizatorul și parola sunt corecte, datele vor fi inițializate, având în vedere arhitectura aleasă și folosită.

4.3.3 Cerere asistență

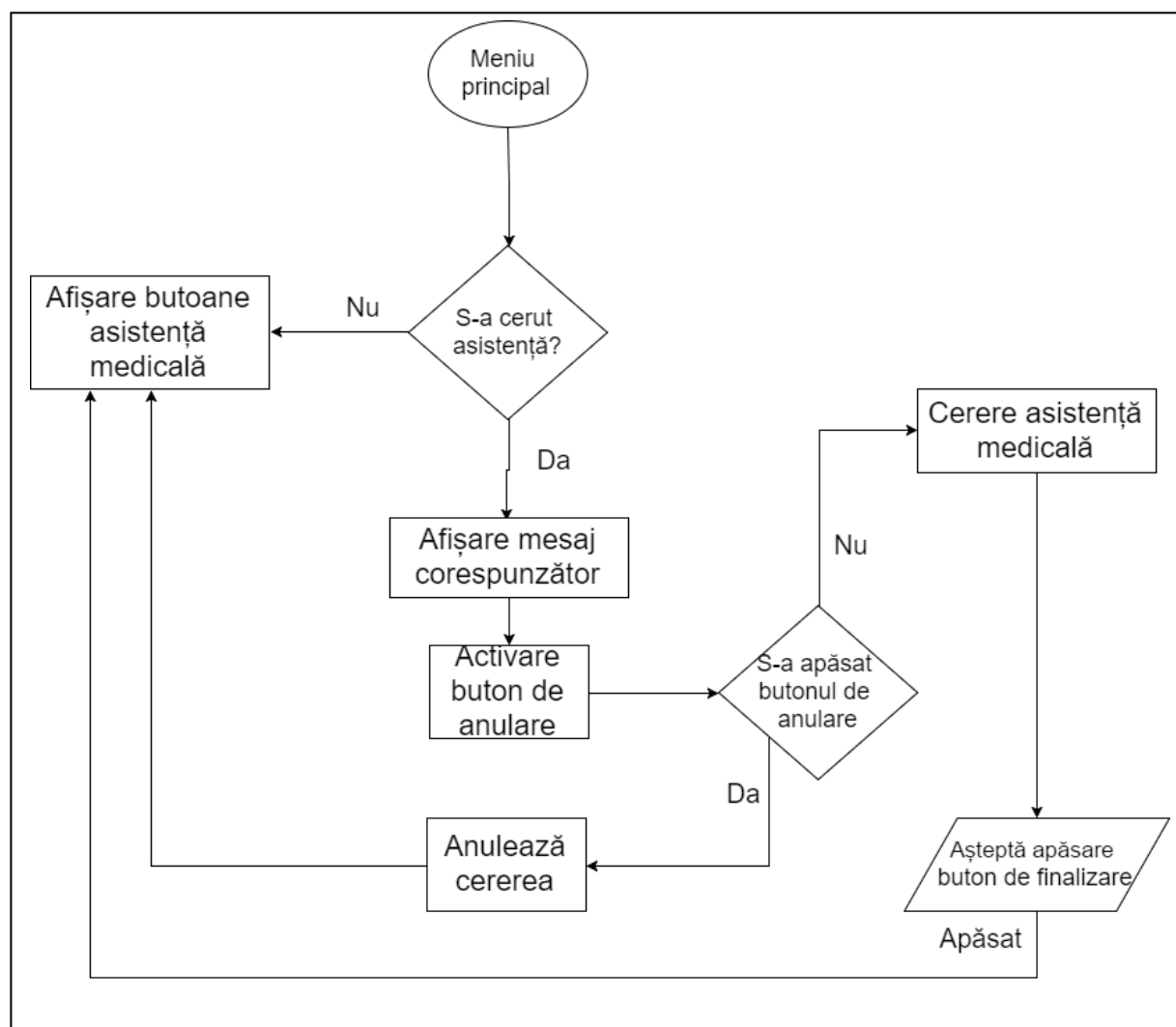


Fig. 4.14 – Organigramă cerere asistență medicală

Exact cum se vede și în organigrama de la figura 4.14, în momentul în care un pacient are nevoie de atenție medicală, acesta apasă un buton specific și notificarea este salvată în baza de date. În momentul în care un cadru medical acceptă asistența, el va primi un mesaj și va fi rugat să apese butonul de finalizare la final.

4.3.4 Acces familie

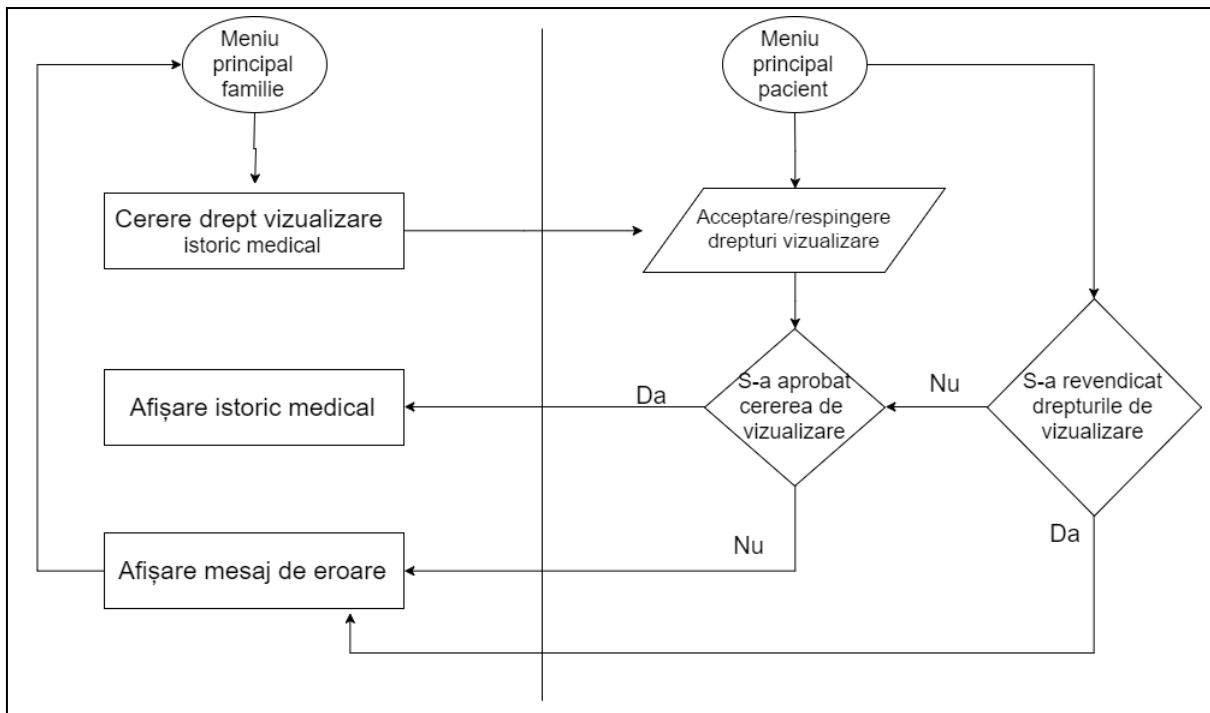


Fig. 4.14 – Organigramă acces familie

Accesul familiei/prietenilor la istoricul medical ar trebui să fie simplu și la puterea unei apăsări de buton și așa și este. După cum se poate vedea în figura 4.14, după ce se cere accesul la istoricul medical, acesta poate fi revendicat în orice moment printr-o simplă operațiune.

4.4 Testarea aplicației

Aplicația a fost testată fizic pe mai multe dispozitive cu succes:

- OnePlus X – Android API level 23
- LG G2 mini – Android API level 21
- LG Nexus 4 – Android API level 20
- Samsung S7 Edge – Android API level 23

Virtual, aplicația a fost testată cu ajutorul serviciului RoboTest de la Google Firebase. Acesta a fost testat cu succes pe următoarele dispozitive:

- Nexus 5X – Android API level 23
- Nexus 7 - Android API level 21

Din păcate, la testul virtual, am descoperit că aplicația nu este compatibilă cu unele elemente din interfața atașată de cei de la Motorola dar problema va fi cercetată și rezolvată.

În plus, aplicația este monitorizată în continuu de către serviciul Analytics de la Firebase. Acest serviciu raportează toate problemele apărute iar în ultimele 4 zile de la ultima versiune a aplicației nu a fost reperată nicio eroare având în vedere că au fost 10 utilizatori care au folosit aplicația și 30 în ultima lună (figura 4.15)

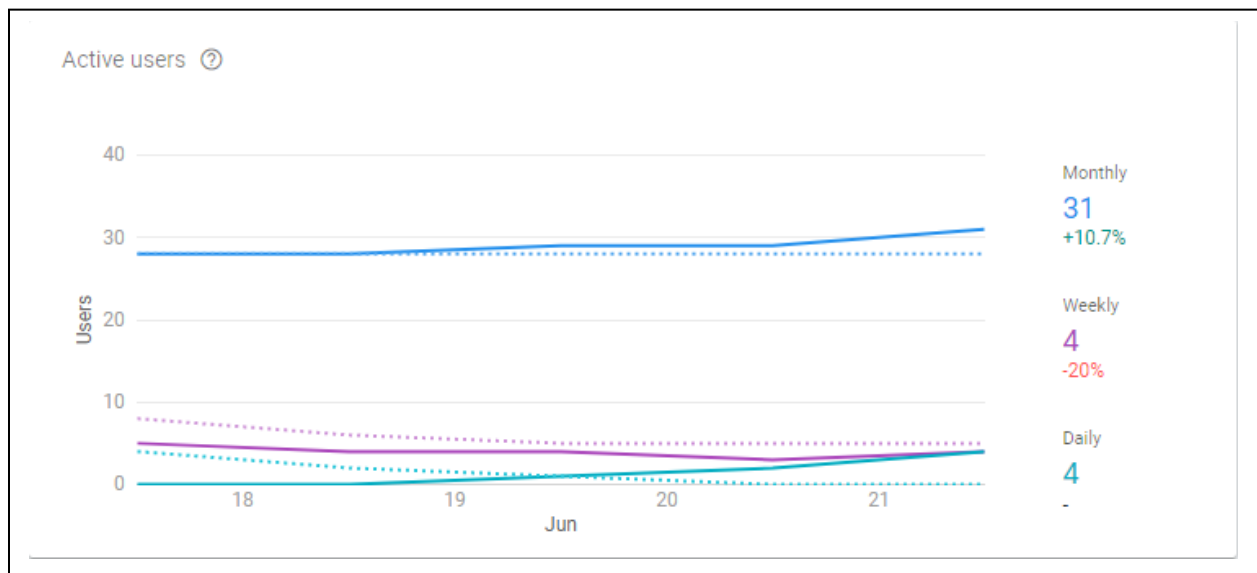


Figura 4.15 – Grafic număr utilizatori

5. Utilizarea aplicației

După cum s-a specificat și în capitolele anterioare, aplicația este destinată pentru trei tipuri de utilizatori. Aceasta are funcționalități diferite în funcție de fiecare. Desigur, interfața este foarte asemănătoare însă funcționalitățile sunt diferite. Vom parcurge interfața și posibilitățile fiecărui tip de utilizator pentru a observa fiecare ce posibilități are în momentul de față.

5.1 Părți comune

Deși există trei tipuri de utilizatori, unele părți sunt comune, cum ar fi, alegerea tipului de utilizator, autentificarea sau înregistrarea unui cont nou. În momentul deschiderii aplicației, dacă nu este niciun cont autentificat deja, aplicația îți va oferi posibilitatea de a alege tipul de utilizator (figura 5.1). După alegerea tipului, ecranul de înregistrare va apărea unde utilizatorul poate introduce un email și parola dacă acesta are cont deja (figura 5.2). În caz contrar, acesta se poate înregistra la înregistrarea inițială, se cer doar un email și introducerea de două ori a parolei (figura 5.3). În funcție de tipul de utilizator, acesta va trebui să își introducă datele personale sau nu la pasul următor.

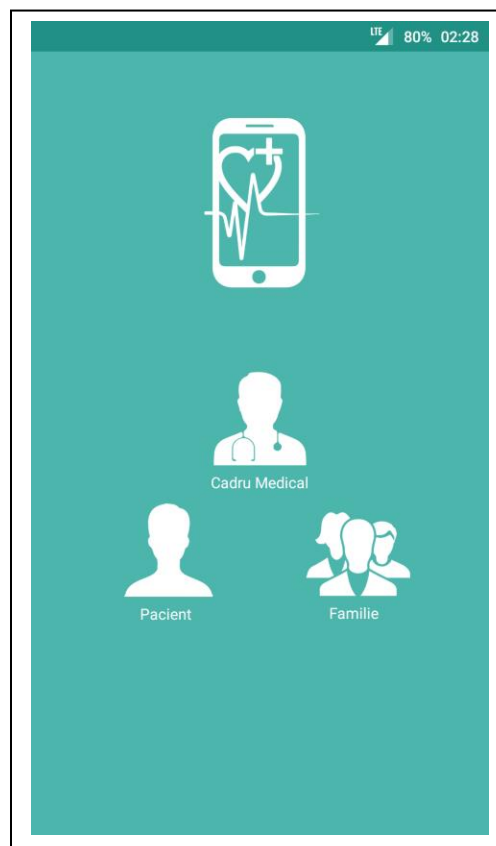


Fig. 5.1 – Alegerea tipului de

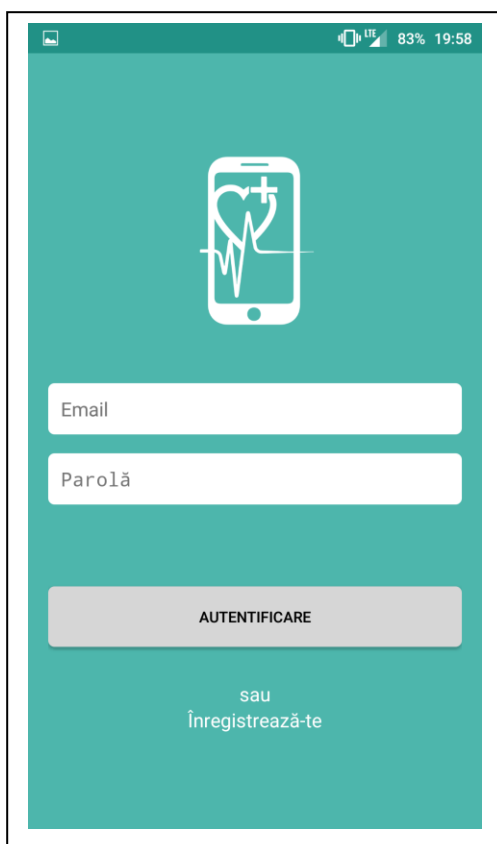


Fig. 5.2 – Interfața de autentificare

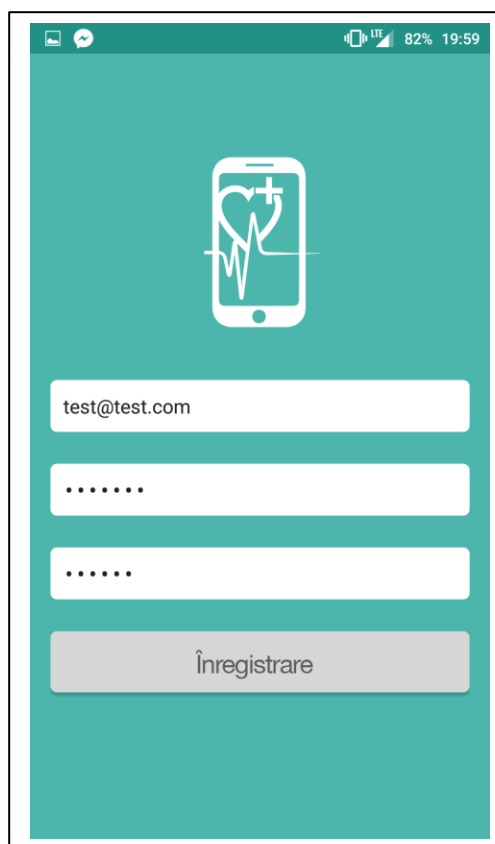


Fig. 5.3 – Interfața de înregistrare

Desigur, în cazul unor erori, se va afișa un mesaj corespunzător pecum în figurile 5.4 și 5.5 pentru partea de înregistrare și figura 5.6 pentru partea de autentificare.

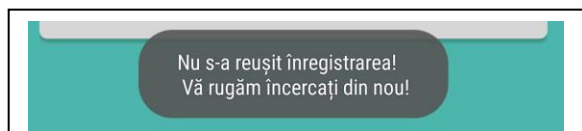


Fig. 5.4 – Eroare de înregistrare 1

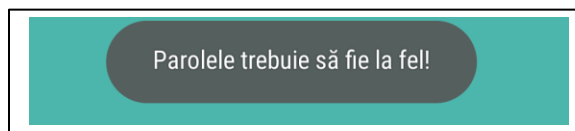


Fig. 5.5 – Eroare de înregistrare 2

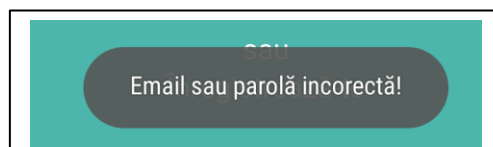


Fig. 5.6 – Eroare de autentificare

5.2 Familie

5.2.1 Înregistrare date personale

După introducerea adresei email și a parolei, un cont nou se va crea dar nu veți putea accesa aplicația decât dacă veți completa și datele personale. În figura 5.7 veți observa toate datele necesare. Atenție, toate datele sunt obligatorii. Cea mai important element este codul unic. Acela trebuie cerut de la pacientul care că este rudă sau membru al familiei. Chiar dacă după introducerea codului acesta va fi validat, veți avea acces în aplicație dar nu veți putea accesa istoricum medical decât după ce acesta va confirma accesul.

În cazul în care nu mai aveți sau nu ați avut niciodată acces la istoricul medical al unui pacient, veți primi un mesaj ca în figura 5.8.

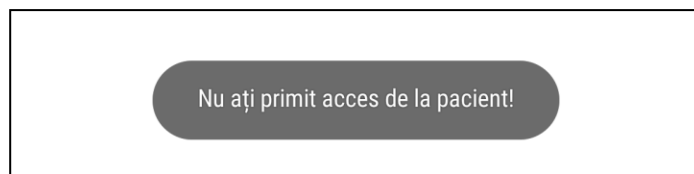


Fig. 5.8 – Eroare acces pacient

A screenshot of a mobile application registration screen. The background is teal. At the top, there is a status bar showing "LTE", "69%", and "21:03". Below the status bar is a white icon of a smartphone with a heart and a pulse line. There are four white input fields: "Prenume", "Nume", "Cod unic pacient", and a "SALVEAZA" button at the bottom. Between the "Nume" and "Cod unic pacient" fields, there are two options: "Fotografiează" with a camera icon and "Încarcă imagine" with a folder icon.

Fig. 5.7 – Introducerea datelor personale ale utilizatorului familie.

5.2.2 Meniul principal și facilități

Meniul principal (figura 5.8) oferă toate opțiunile posibile acestui tip de utilizator. Acesta poate să verifice datele personale care nu sunt multe deoarece nu este nevoie, să meargă pe pagina principală sau să se delogheze și să vadă istoricul medical al unei persoane dacă acesta are acce.

În figura 5.9 se regăsește contul utilizatorului cu datele personale și în figura 5.10, modul în care acesta va vizualiza istoricul medical al unui pacient.

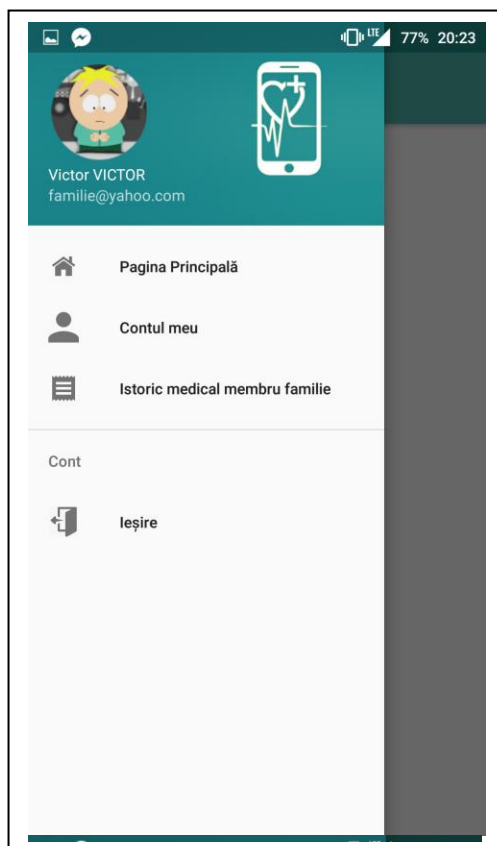


Fig. 5.8 – Meniu principal familie

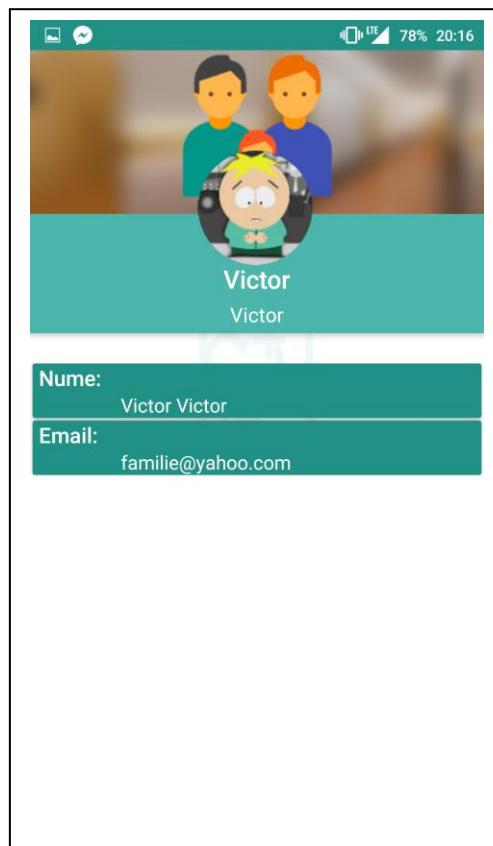


Fig. 5.9 – Detalii cont personal



Fig. 5.10 – Vizualizare istoric medical

5.3 Pacient

5.3.1 Meniul principal

Meniul principal (figura 5.11) al pacientului este asemănător cu restul ca și design însă oferă alte funcționalități. Mai exact din acest meniu puteți:

- Merge pe pagina principală
- Afișa date personale ale contului
- Vizualiza istoric medical
- Oferi/luat acces familie la istoricul medical
- Deloga

Pagina principală de această dată are mai multe de oferit (figura 5.12). Un pacient internat poate oricând să solicite asistență medicală pentru diferite motive (asistență fizică, consult, urgență). Orice cerere poate fi anulată (5.13). În momentul în care un cadru medical acceptă cererea unui pacient acesta este anunțat. La final, acesta va putea apăsa butonul de finalizare pentru a opri cererea (figura 5.14)

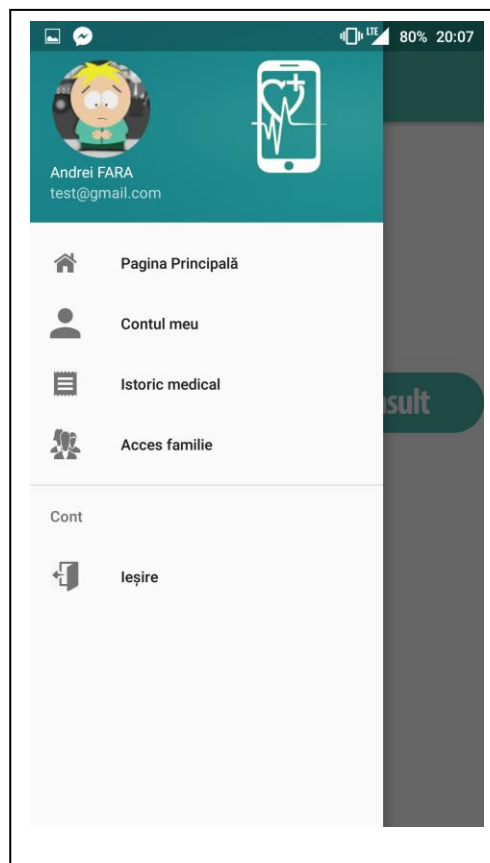


Fig. 5.11 – Meniul principal pacient



Fig. 5.12 – Pagina principal pacient



Fig. 5.13– Anularea cererii



Fig. 5.14 – Finalizarea cererii

5.3.2 Vizualizarea contului

La opțiunea de vizualizare a contului, pacientul va putea să își vadă detaliile despre cont (figura 5.15) însă nu va putea să vadă istoricul medical. Acela este disponibil la secțiunea următoare care arată exact ca cea din figura 5.10.

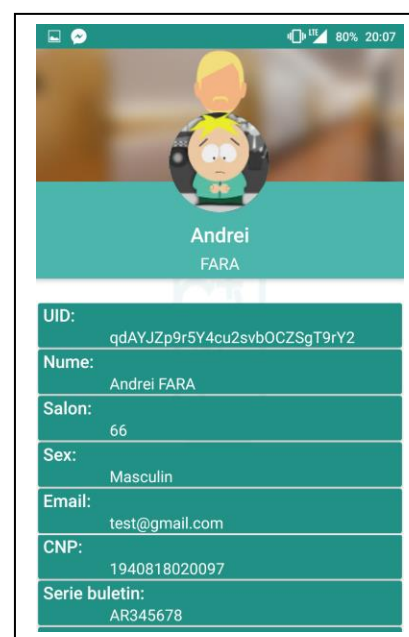


Fig. 5.15 – Datele personale

5.3.3 Acces membru al familiei

Puteți oricând vizualiza cine a cerut acces la istoricul medical. Inițial va trebui să îi oferiți acces apăsând butonul în forma de bifă sau să respingeți dând pe butonul în formă de X. Oricând veți putea reveni și reoferi acces sau să îl anulați. (figura 5.16)

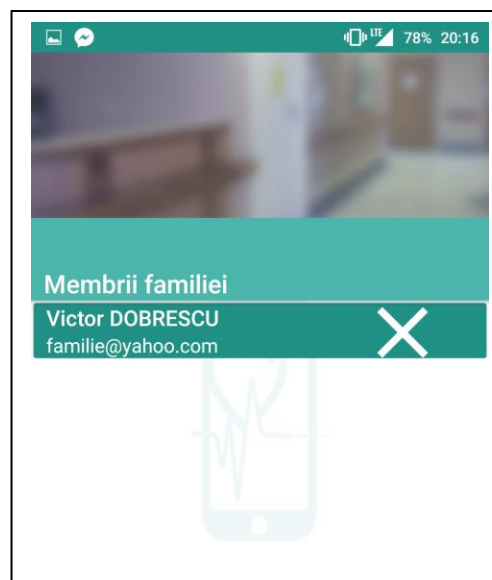


Fig. 5.16 – Acces familie

5.4 Cadru medical

5.4.1 Meniul principal

Meniul principal al cadrelor medicale este cel mai complex așa cum este și normal. Ca și funcții, un cadru medical poate din meniul principal să (figura 5.17):

- Vizualizeze pagina principală
- Vizualizeze contul personal (ca în figura 5.15 dar cu datele specifice unui cadru medical)
- Vizualizeze lista cu pacienți
- Vizualizeze saloanele
- Vizualizeze pacienții fără internare
- Să se delogheze

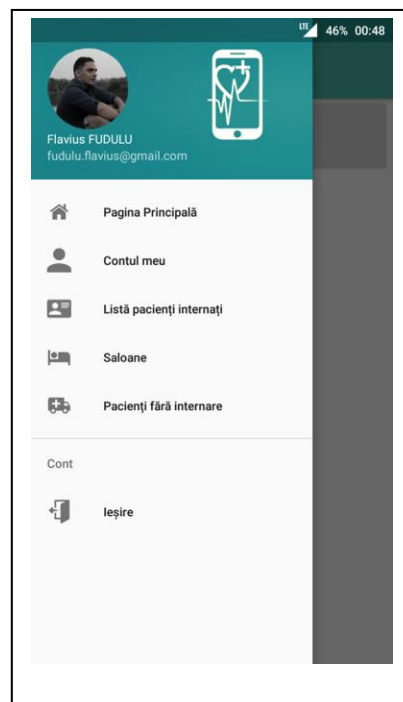


Fig 5.17 – Meniul principal

5.4.2 Lista cu pacienți

În lista cu pacienți, un cadru medical poate Oricând să caute un pacient din bara de căutare. Odată găsit pacientul dorit, cadrul medical are trei opțiuni:



- Afișarea datelor personale ale Pacientului



- Afișarea istoricului medical al pacientului



- Introducerea primului tratament

Dacă un pacient este nou internat dar nu are salvat un diagnostic, va trebui să se introducă un diagnostic și un tratament. În cazul în care acesta apare în baza de date cu cel puțin un diagnostic, cadrul medical va putea să vadă acele date și să adauge unele noi, sau doar să vadă profilul personal al pacientului.

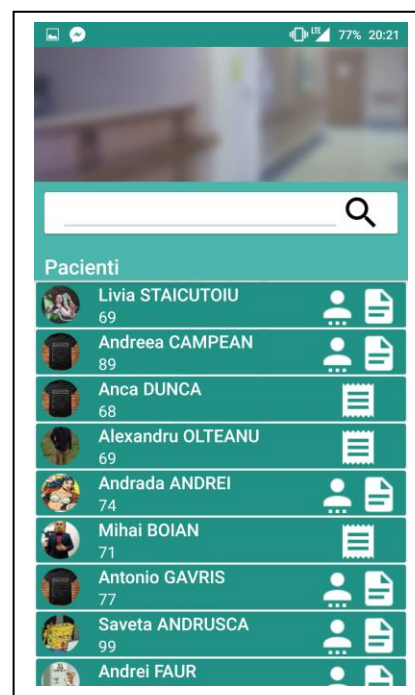


Fig. 5.18 – Listă pacienți

5.4.3 Vizualizare saloane

Pentru o mai ușoară gestionare a pacienților, aceștia pot fi sortați și în funcție de salonul în care sunt internați.(figura 5.22) O simplă apăsare pe salonul pe care se face vizita și toți pacienții internați în acea încăpere vor apărea precum în figura 5.18. Aceleași atribuții se vor aplica și în acest caz.

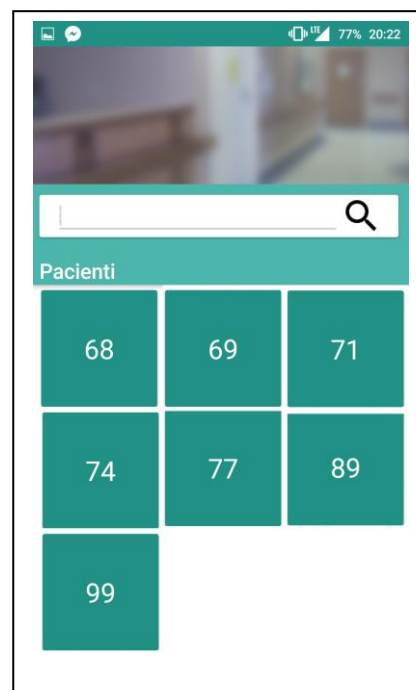


Fig. 5.22 – Afișarea saloanelor

5.4.4 Diagnosticul și tratamentul

Dacă pacientul are deja o internare cu un diagnostic pus și cu tratament, cadrul medical poate să adauge un diagnostic sau tratament nou folosind butonul de “Tratament nou”.(figura 5.19) În cazul în care se dorește externarea pacientului, se folosește butonul respectiv. Activitățile de adăugare tratament și de externare sunt similare în design. (figura 5.20 și 5.21). Datele existente sunt completate automat iar cadrul medical va introduce doar datele noi și va salva modificările.

The image shows a mobile app interface titled "Smart Med". At the top is a teal header with a white heart and pulse icon. Below the header is a teal background with a white medical icon. The main area is white and contains several input fields with placeholder text: "9Q2c2YcmzRfZN2dsaxlUyFVY1io2", "Andreea", "CAMPEAN", "06/23/17", "Vezi fisa pacient" (repeated four times), "Data Externarii", and a grey "SALVEAZA" button at the bottom.

Fig. 5.20 – Externare pacient

The image shows a mobile app interface for patient treatment details. At the top is a teal header with a white heart and pulse icon. Below the header is a teal background with a white medical icon. The main area is white and contains several input fields with placeholder text: "CAMPEAN", "06/23/17", "Diagnostic", "Medicatie", "Data Aplicarii Medicatiei", "Detalii", "Data externarii", and a grey "SALVEAZA" button at the bottom.

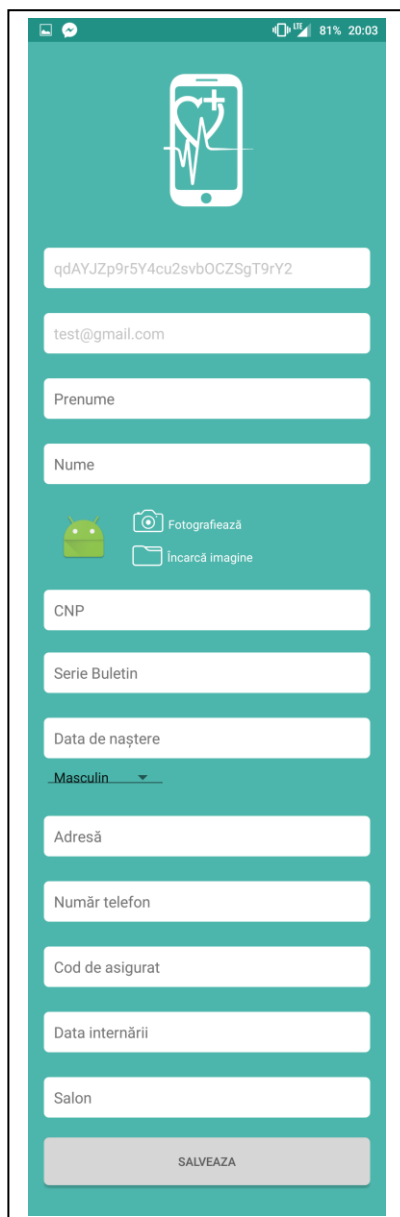
Fig. 5.19 – Vizualizare tratament pacient

The image shows a mobile app interface for patient treatment details. At the top is a teal header with a white heart and pulse icon. Below the header is a teal background with a white medical icon. The main area is white and contains several input fields with placeholder text: "CAMPEAN", "06/23/17", "Diagnostic", "Medicatie", "Data Aplicarii Medicatiei", "Detalii", "Data externarii", and a grey "SALVEAZA" button at the bottom.

Fig. 5.21 – Adăugare tratament și/sau diagnostic nou

5.4.5 Vizualizare pacienți fără internare

În momentul în care un pacient își crează un cont nou, acesta nu îl poate folosi decât după ce îi este făcută o internare și activat contul. Deși acesta apare în meniul principal de notificări ale cadrului medical, există și o secțiune separată în care cadrul medical poate căuta toți pacienții care nu au internarea făcută (figura 5.22)



The screenshot shows a vertical form for patient registration. At the top is a teal header with a white icon of a heart and a pulse line. Below the header are several input fields: a long alphanumeric code, an email address (test@gmail.com), first name (Prenume), last name (Nume), a section for profile picture with 'Fotografiează' and 'Încarcă imagine' options, a CNP field, a 'Serie Buletin' field, a 'Data de naștere' field with a dropdown menu set to 'Masculin', an 'Adresă' field, a 'Număr telefon' field, a 'Cod de asigurat' field, a 'Data internării' field, and a 'Salon' field. At the bottom is a grey button labeled 'SALVEAZA'.

Fig. 5.23 – Activitatea de internare

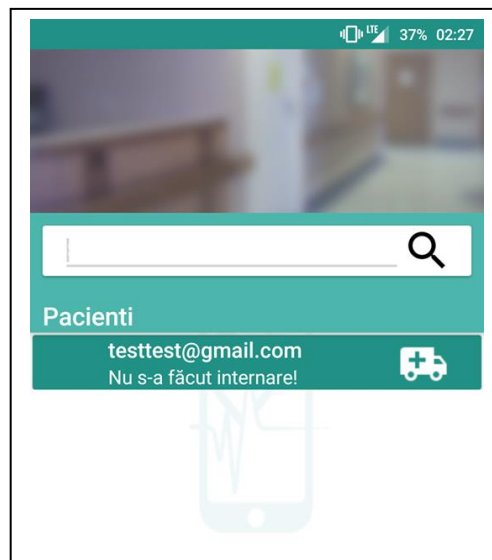


Fig. 5.22 – Secțiunea de pacienți fără internare

Activitatea de internare este simplă, completând datele personale ale pacientului și atribuind un salon, diagnosticul și tratamentul făcându-se după procesul de internare. Se poate observa activitatea de internare în figura 5.23

5.4.6 Pagina principală – Notificări

În pagina principală a cadrului medical (figura 5.24) se regăsesc notificări importante. Faptul că nu sunt notificări înseamnă că toate problemele au fost tratate. Tipurile de notificări ce pot apărea sunt:

- Cont nou se asistent
Se va putea valida direct de aici contul în cazul în care chiar exista un asistent nou.
- Internare nouă
Se va putea face internare direct de aici Contul în cazul în care chiar exista o Internare nouă.
- Alerte (marcate cu roșu):
 - Asistență fizică
Pacientul are nevoie de îngrijiri fizice (pentru cei imobilizați la pat)
 - Consult medical
Pacientul are schimbări în starea de sănătate sau simptome și are nevoie de un doctor pentru a stabili cauza.
 - S.O.S.
Pacientul are o stare critică și are nevoie urgentă de intervenție medicală

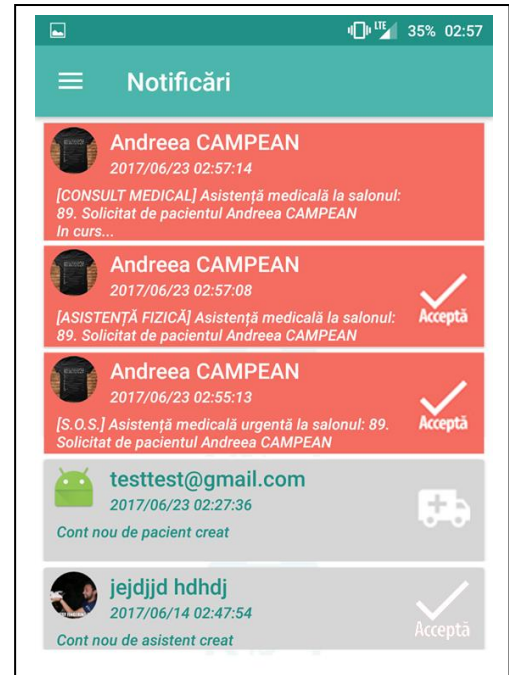


Fig. 5.24 – Pagina principal cadru medical. Notificări

6. Concluzii

Prin dezvoltarea acestui sistem de comunicare între pacienți și cadre medicale, păstrând funcționalitățile necesare cadrelor medicale dar și oferind pacienților acces la toate datele lor medicale instant, fără a mai fi nevoiți să ducă după ei o mini bibliotecă de foi consider ca am reușit să aduc o idee și o bază pentru un viitor mult mai bun în sistemul medical.

În momentul de față, pacienții sunt nevoiți să ducă mereu după ei o multitudine de documente, sau să își instaleze o multitudine de aplicații pentru a avea o evidență a istoricului medical. Cadrele medicale se confruntă cu aplicații desktop, create de cele mai multe ori parcă cu alte scopuri decât cel de a-i ajuta. În plus nu exista nicio conexiune directă între cele două categorii de aplicații sau între cadrele medicale și pacienți.

Pentru moment aplicația reprezintă o bază pe care se poate dezvolta mult mai mult, adăugând module pentru diverse secții medicale sau pentru diferite nevoi ale pacienților. Este un start pentru că s-a creat deja o conexiune care nu a existat până acum și în plus, s-au rezolvat probleme de aspect, funcționalitate și nevoi.

Procesul de internare se face foarte ușor și intuitiv, pacientul are acces direct la toate datele sale din momentul în care se introduc datele de către cadrele medicale. Cadrele medicale au luxul unei aplicații care îi leagă direct cu pacienții și cu datele de care au nevoie, chiar în buzunarul lor. Asistența medicală se face aproape instant și având în vedere mobilitatea, aceasta scutește timpul de acțiune, în unele momente fiind destul de critice încât ar putea chiar salva vieți.

Pe lângă toate aceste aspecte, într-o lume în care suntem obișnuiți să putem împărtăși informațiile la puterea unei apăsări de buton, acum avem această opțiune și în domeniul medical. Nu va mai trebui să trăim cu problema identificării tratamentului și a diagnosticelor trecute a copiilor noștri, a părinților ajunși la o vârstă înaintată sau a oricărei persoane dragi.

În concluzie, consider că am reușit să pun bazele unei aplicații care ar putea revoluționa tehnologia de management al spitalelor, și care rezolvă problema legăturii inexistente între pacienți și cadrele medicale. Sper că în continuare voi reuși să dezvolt aplicația încât să ajungă la un nivel în care să devină indispensabilă.

7. Referințe bibliografice

- [1] <http://www.eweek.com/mobile/google-open-sources-android-on-eve-of-g1-launch>
- [2] https://ocw.cs.pub.ro/courses/_media/eim/laboratoare/laborator01/android_architecture.png
- [3] <https://developer.android.com/about/index.html>
- [4] <https://www.engadget.com/2007/11/12/googles-android-os-early-look-sdk-now-available/>
- [5] <https://developer.android.com/studio/intro/index.html>
- [6] Ștefan Tanasa, Cristian Olaru, Ștefan Andrei, "Java de la 0 la expert", Polirom, 2003.
- [7] Adriana Olteanu, "Baze de date și utilizarea acestora", 2015
- [8] <http://www.scribub.com/stiinta/informatica/baze-de-date/MODELE-SI-TIPURI-DE-BAZE-DE-DA81161.php>
- [9] <https://www.upwork.com/hiring/data/sql-vs-nosql-databases-whats-the-difference/>
- [10] <https://firebase.googleblog.com/2016/05/firebase-expands-to-become-unified-app-platform.html>
- [11] <https://firebase.googleblog.com/2016/05/firebase-expands-to-become-unified-app-platform.html>