Universitatea Transilvania din Brașov
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Programul de studii:

Informatică aplicată

Lucrare de licență

RoNature

Aplicație mobilă social media

Autor: **Amália-Angéla Bacsó**

Coordonator ştiințific: **Lect. Dr. Răzvan Bocu**

Braşov, 2022

Cuprins

Lista acronimelor

- API - Application Programming Interface
- BD - Baze de Date
- FCM - Firebase Cloud Messaging
- GCM - Google Clouds Messaging
- IDEA - International Data Encryption Algorithm
- HTTP - HyperText Markup Language
- IDE - Integrated Development Environment
- IOS - iPhone Operating System
- LTS - Long-term Support
- SDK - Software Development Kit
- SO –Sistem de Operare
- UI - User Interface
- UML – Unified Modeling Language
- JSON - JavaScript Object Notation
- XML - Extensible Markup Language
- GDPR - The General Data Protection Regulation

Lista figurilor

Lista secvențelor de cod

Capitolul 1

INTRODUCERE

În acest capitol intenționez, pentru început, să descriu factorii care au influențat alegerea implementării acestui proiect, care dorește să ofere ajutor și suport iubitorilor de călătorii, drumeții și excurii în natură. După aceea, voi expune motivele pentru care consider că aplicația ar avea un impact benefic asupra publicului.

După această prezentare, introducerea va merge înspre prezentarea altor platforme sociale și în cadrul studiului comparativ asupra celorlalte aplicații voi prezenta de ce o astfel de aplicație este mai bună și mai benefică față de cele existente momentan. Se va pune în evidență printr-o descriere amplă, părțile care ajută această aplicație să se facă deosebită prin originalitate, inventivitate, la care are avea acces o sumedenie de clienți fară ca serviciul să fie contracost.

1.1 Scurtă prezentare a temei

Această aplicație, numită RoNature dorește să fie o platformă socială, ca Facebook, Instagram, Snapchat etc. Aplicația are la bază comunicarea între oameni, și partajarea pozelor. Aplicația dorește să vină în ajutor tuturor persoanelor care sunt iubitori de excursii, cărora le plac locurile noi și pentru cei pasionați de fotografie astfel având ocazia să împărtășească obiectivele noi văzute și pe urmă ajute pe ceilalți să își găsească noi atracții turistice.

Aplicația este simplă și foarte ușoară de folosit, chiar intuitivă. Toate postările sunt făcute publice, astfel toată lumea are acces la ele și pentru a oferi un ajutor suprem pentru utilizatori în găsirea locurilor vizualizate. Este obligatoriu introducerea locației iar aplicația pune la dispoziție și o hartă cu poze, astfel pe baza locației curente a utilizatorului, acesta își poate găsi foarte ușor cele mai aproipate artacții.

1.2 Motivația alegerii temei

Trăim într-o lume a vitezei și tehnologiei, în care telefonul mobil reprezintă una dintre cele mai puternice și versatile unelte de care dispune omul la o apăsare pe ecran distanță.

În contextul actual ne-am obișnuit să ne păstram pozele, clipurile video virtual și nu în format fizic, nu mai trimitem postcarduri din excursii prietenilor, ci le distribuim pe platfromele sociale și aplicația RoNature vine în a ne ușură viață și mai mult, dar o să vedem în cele ce urmează ce a fost motivul de baza, care a contribuit la dezvoltarea aplicației.

Pentru a explica mai ușor motivul pentru care am ales această temă, o să încep cu o scurtă prezentare legată de persoana mea. Iubesc natura, tot ceea ce înseamnă munți, păduri, lacuri, mare, râuri, cascade, stănci.. cei pasionați ca și mine sigur că o să mă înțeleagă. Obișnuiesc să profit de fiecare ocazie să admir natura, să descoper obiective noi și să explor fiecare loc, indiferent unde mă aflu, dacă locația respectiva are o popularitate, un renume sau este total anonim. Astfel am descoperit peisaje superbe cu o priveliște fascinantă și am ajuns la concluzia că avem o țara uluitor de frumoasă, dar mai puțin valorificată.

De multe ori în apropierea noastră sunt zone cu totul speciale, dar nu le cunoaștem. Dicutând cu oameni am obesrvat că trăim poate de mulți ani în locația respectivă și nu știm ce comori ascunde natura. Dacă le-am cunoaște și le-am valorifica, doar zonele din jurul nostru, în primul rând ne-am bucura mai mult și mai mulți de țara noastră. Iar ca efect secundar acest lucru ar avea un impact imens asupra turismului intern, ce drept consecință ar produce și ridicarea turismului internațional.

Cum reiese și din prezentarea de mai sus, motivul alagerii acestei teme pentru a dezvolta aplicația RoNature are la bază un motiv foarte simplu și anume, iubire față de natură și pasiunea pentru drumeții.

Consider că aplicația ar fi de folos tuturor persoanelor iubitori de natură, care își caută o destinație sau pur și simplu persoanelor cărora le place să adimire poze cu peisaje. Iubitori de natură, care își caută o destinație este și foarte utilă, deoarece bazată de locația aleasă, pot vedea zonele vizitate de ceilalți și pot decide pe baza imaginilor dacă le-ar place sau nu. Evident la rândul lor au posibilitatea să distribuie și poze, sau să între în conversație privata cu ceilalți utilizatori.

1.3 Studiu comparativ cu aplicațiile deja existente

La momentul actual există sute de rețele sociale, o să enumăr câteva dintre ele, care sunt mai aporiate de conceptul de baza a aplicației noastre și o să prezint prin câteva cuvinte pe fiecare.

Încep cu Facebook, este încă cea mai mare rețea de socializare din lume. Se spune că a avut aproximativ 2 miliarde de utilizatori lunari din decembrie 2017. (1) Este bine cunoscut faptul că utilizatorii îl folosesc pentru a partaja poze de toate categoriile și videoclipuri, a posta gânduri, a distribui postările altora. O alta platforma care este tot deținută de Facebook este Instagramul, iar acesta este o platformă de accesare a fotografiilor și videoclipurilor. Se estimează că în ianuarie 2018 va avea aproximativ 800 de milioane de utilizatori. (1) Pinterest este, de asemenea, un site de socializare unde se adaugă conținut sub formă de pin, iar în ianuarie 2018, are aproximativ 200 de milioane de utilizatori, iar conexiunea de bază este partajarea fotografiilor. (1). Tumblr este o rețea de peste 350 de milioane de bloguri și peste 500 de milioane de bloguri de utilizatori. Rețeaua socială acceptă web și mobil. (1) Photobucket este un site de stocare a fotografiilor și videoclipurilor cu peste zece miliarde de fotografii și peste 100 de milioane de membri. (1) Imgur (Nou) este un site de partajare a fotografiilor unde membrii pot vota (și pot clasa) fotografiile. Site-ul conține sute de milioane de imagini. (1)

Aceste aplicații sunt foarte bine dezvoltate, capabile de a suporta mii de utilizatori și de a stoca datele acestora. Au o arhitectură bine dezvoltată, un desing care atrage utilizatorii și folosirea este foarte intuitivă. Utilizatorii pot distribui sătari, gânduri, materiale foto-video, după preferință, pot intra în contact cu ceilalt, au șansă de a reacționa, a comenta la postările altora, a intra în discuții cu aceștia. Utilizatorii își pot crea cercul lor de interes și în funcție de acesta vizualizează postările, însă aceste platformele sunt foarte generice.

Punctul „mai slab” a acestor aplicații este genericitatea ceea ce crează confuzie și greutăți celor care caută postări doar legate de o arie de interes, deoarece printre miile de postări de alt gen se pierd. Aici vine aplicația RoNature în ajutorul acelora care doresc să aibă acces la postări legate de natură și acesta reprezintă și punctul forte a aplicației. Toate datele sunt adunate la un loc concret și aplicația se construiește în jurul unei teme specifice.

În concluzie, putem afirma că am explicat așa cum se cuvine aplicația noastră, prin introducere, în partea ce ține de motivația și obiectivul proiectului, prezentarea pe scurt a aplicației.

1.4 Structura lucrării

- Capitolul 1 - acesta este capitolul curent, are rol introductiv, se prezintă motivele alegerii acestui subiect împreună cu structura lucrării de față.
- Capitolul 2 - acesta este capitolul cu explicații și exemple pentru tehnologiile folosite în proiect.
- Capitolul 3 - acesta capitolul curpinde explicații asupra implementărilor
- Capitolul 4 - acesta este capitolul în care se explică funcționalitățile aplicației, pașii de utilizare
- Capitolul 5 - acesta este capitolul ce conține concluziile precum și direcții de dezvoltare/posibile ajustări.

Capitolul 2

TEHNOLOGII UTILIZATE ȘI NOȚIUNI TEORETICE

În realizarea proiectului am folosit următoarele tehnologii, pe care urmează să le discutăm mai pe larg în cea ce urmează:

- IDE: Android Studio
- Frontend: XML
- Backend: Java
- Baze de date: Firebase

2.1 Android Studio IDE

2.1.1 Android Studio

Android Studio este mediul oficial de dezvoltare pentru sistemul de operare Android (IDE) Google. JetBrains este construit pe software-ul IntelliJ IDEA și este conceput special pentru dezvoltarea Android. (2)

Structura proiectului:

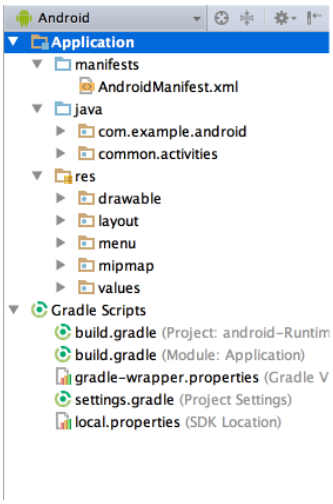


Figure 1 Proiect files în Android

Cum vedem și în figura 1 de mai sus, fiecare proiect are module:

- Adroid application
- Libaray
- Google App Engine

Toate fișierele de compilare sunt vizibile la nivelul superior:

- mainfests: contine fisierul AndroidMainfest.xml
- java: fisierele cu cod sursa și cele de testare Junit
- res: fisiere non-code, drawable-, layout-, mipmap-, values-files ce contin fisiere .XML

2.1.2 Componente Andorid

- Activities - O activitate reprezintă un ecran cu user interface (UI), deci Activitatea efectuează acțiuni pe ecran.

```
public class MainActivity extends Activity {
}
```

Code 1 Activity example

- Services - Un serviciu este o componentă care se ocupă de procesarea în fundal asociată cu o aplicație.

```
public class MyService extends Service {
}
```

Code 2 Service example

- Broadcast Receivers - raspunde la mesaje broadcast, se ocupă de comunicarea dintre SO Android și aplicații.

```
public class MyReceiver  extends  BroadcastReceiver {
   public void onReceive(context,intent){}
}
```

Code 3 Broadcast Receiver example

- Content Providers - se ocupă de problemele legate de gestionarea datelor și a bazelor de date.

```
public class MyContentProvider extends  ContentProvider {
   public void onCreate(){}
}
```

Code 4 Content Provider example

Mai există și alte componente, pe care le vom enumera mai jo, acestea sunt utilizate pentru construcția componenteleor enumerate mai sus, legare între ele și logica acestora:

- Fragments - Reprezintă o porțiune UI într-o activitate.
- Views - Elemente UI care sunt desenate pe ecran, inclusiv butoane, formulare de liste etc.
- Intents - Mesajele conectarea componentelor împreună.
- Layouts - sunt ierarhiile care controlează formatul ecranului și aspectul vizualizărilor.
- Resources - Elemente externe, cum ar fi șiruri, constante și imagini desenabile.
- Mainfest – cupleaza toate elementele enumerate mai sus și este cuprinsa în AndroidManifest.xml. Acesta descrie componentele aplicației și cum interacționează între ele.

2.1.3 Design Componente UI

- Layout-uri:
  - Linear
  - Frame
  - Constraint
  - Relative

- Elemente de view:
  - EditText
  - ImageView

- TextView
- Buttons

- Intent-uri:
  - Intent Filter
  - RecycleView
  - ListView
  - Explicit
  - Implicit

- Toast
- Bottom Sheet-uri
- Fragmente
- Tab-uri
- Navigation Drawer
- Dialog-uri

2.1.4 Stocare

Sunt patru tipuri de stocare oferite de Android.

- Stocare internă a fișierelor: stocați fișierele private din aplicație pe sistemul de fișiere al dispozitivului.
- Stocare externă a fișierelor: stocați fișiere pe sistemul de fișiere extern partajat. Acest lucru este de obicei pentru fișierele de utilizator partajate, cum ar fi fotografiile.
- Preferințe partajate: stocați datele primitive private în perechi cheie-valoare.
- Baze de date: stocați datele structurate într-o bază de date privată.
- FileProvide : dacă doriți să partajați fișiere cu alte aplicații

2.2 XML

Interfața de utilizator, așa prescurtată UI, a aplicației este tot ceea ce utilizatorul poate vedea și cu care poate interacționa. Cu alte cuvinte, interfața de utilizare a aplicației

este tot ceea ce contează pentru utilizator, acesta poate să atrage utilizatorii și poate să îi îndepărtează. În dezvoltarea aplicației noastre folosim XML - Extensible Markup Language – pentru interfață de utilizator.

XML este un limbaj de marcare, folosește reguli pentru codificare, într-un format care este citibil de om, cât și de mașină. Plus este scalabil și foarte ușor de utilizat de către dezvoltatori. În Android Studio, folosim XML pentru crearea de layout-uri, pentru că XML este un limbaj ușor ce nu produce greutăți celor care îl folosesc, dezvoltatorilor. (3)

XMLPullParser are rolul, cum sugerează și numele, de a traduce un format XML într-un cod Java funcțional. Este de menționat că Android oferă 3 tipuri de parsere XML: DOM, SAX și XMLPullParser, însă varianta recomandată de Android este XMLPullParser pentru că este eficient și ușor de utilizat.

Acest proces se desfășoară în următorul fel:

- citirea unui format XML.
- analizarea acestuia cu XMLPullParser
- crearea unui obiect Java view pentru a crea UI-ul din aspectul XML analizat.

Acesta este asa numitul „secret” din spatele codului implementat pe care îl găsim atunci când creăm o activitate:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The XML layout located in res>layout>activity_main.xml
        setContentView(R.layout.activity_main);
}
```

Figure 2 Activitate main exemplu

În Android sunt mai multe scopuri de a utiliza XML-uri și fiecare astfel de scop are nevoie de un anumit tip de fișiere xml, pe acestea le vom prezenta în cele ce urmează.

Va fi prezentat un simplu XML și Designul produs de acesta:

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorGreen">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="Hello Word!"
        android:textStyle="bold|italic"
        android:textSize="30dp"
        android:textColor="@color/colorWhite"/>

</RelativeLayout>
```

Code 5 „Hello word" xml exemplu

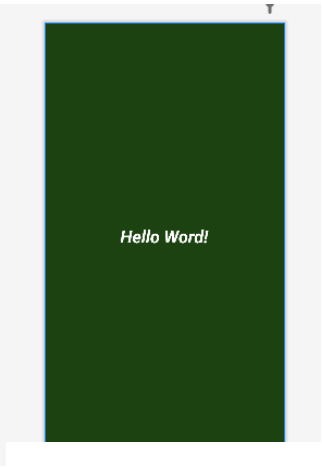


Figure 3 Desing "Hello word"

2.2.1 Interfața de utilizator

XML este folosit în definirea interfeței de utilizare, din motive diferite pe care le-am discutat mai sus, însă fișierele folosite pentru a face asta sunt fișierele XML Layout care cuprind toate elementele și tool-urile pe care dorim să le folosim. De exemplu cum ar fi ImageView, TextView, Button etc.... Acestea sun stocate și pot fi accesate din folderul res, layouts. Cum se poate vedea la Figura 1, mai multe se vor discuta în următorul capitol.

2.2.2 Definirea componentelor

Un XML este, de asemenea, folosit în definirea componentelor pe care aplicația le conține: activități, fragmente și starea, numele pachetelor aplicației, receptorilor, permisiunile și serviciile de care are nevoie aplicația. Toate acestea se gasesc în fișierul xml AndroidManifest, ce reprezinta unul dîntre cele mai importante fișiere dintr-un proiect android. La fel și acesta se poate vedea la Figura 1, iar în următorul capitol va fi prezentat și cel din aplicatia RoNature.

2.2.3 Șiruri de caractere

Șirurile hardcodate nu sunt foarte practice în dezvoltarea unui proiect, pentru a evita acest lucru putem înclocui aceste șiruri cu un singur string.

XML este de ajutor în acest sens, astfel se pot defini toate șirurile dintr-un fișier numit de obicei strings xml(strings.xml). Acesta permite accesarea stringurilor definite în toată aplicația atât în activitați cât și în fișierele layout XML, totodată acest mod aduce și o îmbunătățire la reutilizare. Acesta poate fi accesată din folderul res, values, string.xml.

2.2.4 Definirea stilului

Se definesc diferite stiluri și look-uri pentru UI a proiectului: În plus față de toate funcționalitățile anterioare, XML vă permite, de asemenea, să definiți temele și stilurile personalizate ale aplicației voastre în fișierul Styles xml (styles.xml).

2.2.5 Definirea culorilor

XML oferă opțiunea de a defini culorile personalizate pe care dorim să le folosim în aplicațiile noastre. Simplu definim culorile în fișierul Color xml (colors.xml) și le folosim în aplicația noastră din acest fișier. Acest mod ușurează și procesul în cazul în care dorim să schimbăm anumite culori la aplicație. Nu mai avem nevoie să parcurgem fiecare layout pentru a schimba manul fiecare loc unde dorim să aducem înlocuirea, schimbăm din colors.xml și schimbrea este adusă la nivelul proiectului.

2.2.6 Drawables

O altă componentă foarte importantă a aplicației este folderul drawables, care conține toate graficele pentru elemente sau view-uri de care se folosesc ca elemet de design foarte puternic în realizarea unei aplicați cu interfață personalizată, prietenoasă. Aceste elemente se pot crea după imaginația și gustul clientului, pe lângă acestea Android Studio pune la dispoziție o mare varietate de designuri predefinite, aparținând anumitor categorii. În cele ce urmează vom prezenta câteva astfel de elemente xml gata predefinite, care încă pot fi customizate.

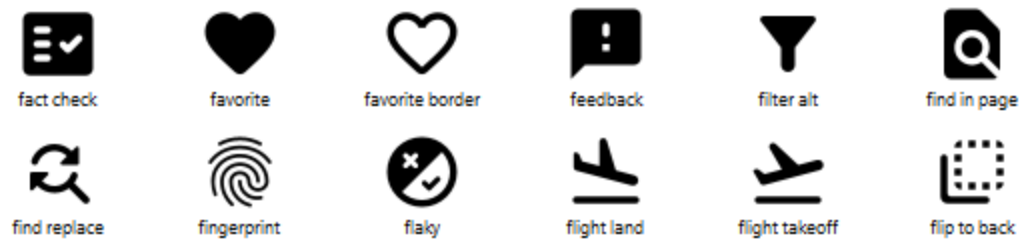


Figure 4 Drawables exemplu

2.3 Java

Java este un limbaj de programare de nivel înalt, obiect orientat, funcțional non-declarativ realizat de Sun Microsystems în 1995 (companie care a devenit filială a Oracle). (4) O mare mulțime dintre aplicațiile din ziua de astăzi sunt clădite pe limbajul de programare Java, un exemplu unde are o cotă foarte mare din piață îl reprezintă sistemul bancar.

Java se regăsește atât în aplicații pentru calculator cât și în aplicații mobile, deși cele mai populare sisteme de operare desktop, prin sistem de oprare desktop ne referim la un sistem de operare care rulează pe un calcultor, explicit nu pe un dispozitiv mobil, suportă aplicații create pe baza Java (ne referim la Windows OS, Linux OS și OS X), pe partea de aplicații mobile windows nu suportă decât formatele proprietare.

Din martie 2022, Java 18 este cea mai recentă versiune, în timp ce Java 17, 11 și 8 sunt versiunile actuale de suport pe termen lung. Oracle recomandă dezinstalarea versiunilor învechite și neacceptate, din cauza problemelor de securitate nerezolvate în versiunile anterioare și sfatul este ca utilizatorii să treacă la o versiune acceptată, cum ar fi una dintre versiunile LTS(8, 11, 17).

În dezvoltarea proiectului nostru vom folosi Java 18, pentru implementarea totală a aplicației, practic tot codul scris ce ține de backendul este realizat cu java.

2.4 Firebase

Firebase este o baza de date NoSql, pentru aplicații Android utilizează JSON pentru stocarea datelor.

Este o platformă web, de aplicații, ce vine în ajutorul dezvoltatorii la crearea aplicațiilor de calitate. Informațiile sunt stocate în format JSON, acesta nu folosește query-uri de inserarea, adăugare, actualizare și ștergere. Firebase reprezintă backend-ul, ce este folosit pentru stocarea de date, în bază de date, pentru sistem. (5)

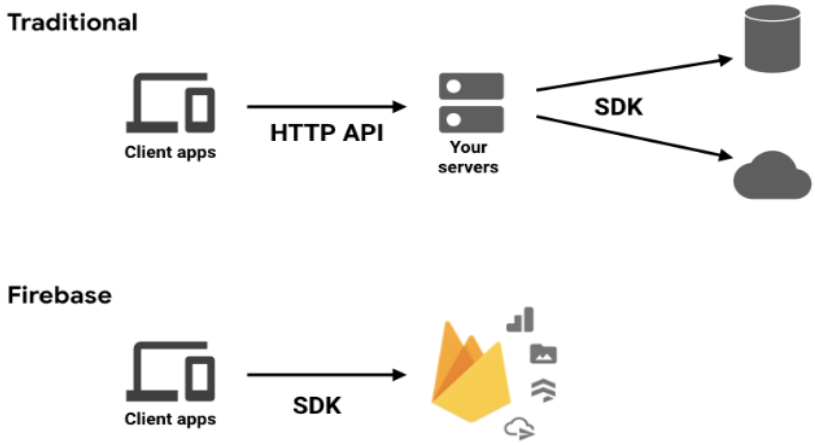


Figure 5 Traditional VS Firebase database

Pentru a înțelege Firebase, prima dată trebuie să definim ce înseamnă abordarea NoSql. Termenul vizează tipuri non-relaționale de BD, care stochează datele într-un format mai intuitiv această abordare diferă de varianta clasică reprezentată cu tabelele relaționale. Bazele de date NoSQL pot fi interogate folosind API-uri, limbaje de interogare declarative și prin exemplu, de aceea sunt denumite „nu doar SQL”.

O să verificăm și ce oferă baza de date NoSql, ce oferă în plus și ce nu pot altele. O diferență majoră ar fi stocarea nestructurată, astfel queriurile sunt mult mai simple și rapide, prin urmare programarea este mai ușoară. (6)

Încă o diferență semnificativă este așa numită scalare orizontală, ce face posibilă adăugarea de mai multe mașini pe mai multe servere pentru a gestiona datele. BD SQL clasice folosesc scalarea veritcală ceea ce necesită adăugarea mai multor memorii la mașina

inițială ceea ce după un timp devine nesustenabilă. În timp ce scalarea orizontală are capacitatea de a gestiona o cantitate mare de date într-un mod mult mai eficient.

Beneficiile utilizare ale bazei de date NoSql sunt pe măsură:

- Scalabilitate
- Flexibilitate
- Performanță ridicată
- Disponibilitate
- Extrem de funcțional

Scurt istoric al Firebase:

Firebase a fost inițial un furnizor de servicii de chat online pentru diverse site-uri web prin API și rula cu numele Envolve . A devenit popular, deoarece dezvoltatorii l-au folosit pentru a face schimb de date despre aplicații, precum starea unui joc, în timp real, între utilizatorii lor, mai mult decât chat-urile. Acest lucru a dus la separarea arhitecturii Envolve și a sistemului de chat. Arhitectura Envolve a fost evoluată în continuare de către fondatorii ei, James Tamplin și Andrew Lee, la ceea ce este Firebase în zilele noastre în anul 2012. (7)

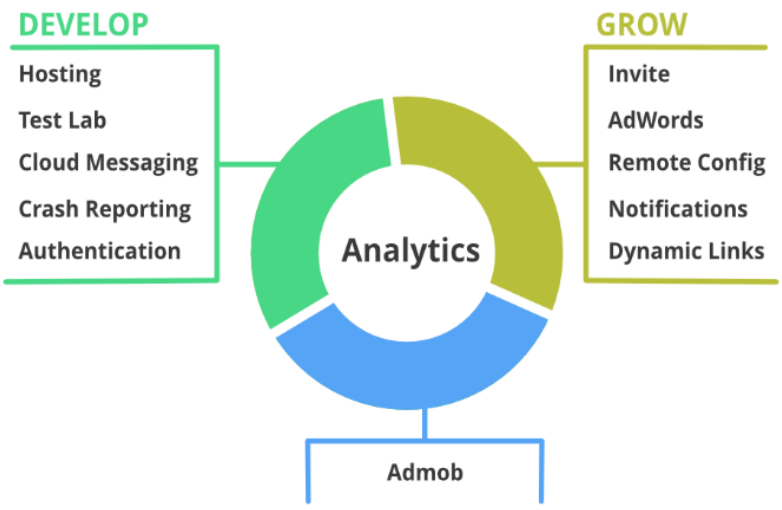


Figure 6 Caracteristici firebase

Caracteristicile principale se enumera în 3 categorii în care firebase își oferă serviciile, asa cum se vede mai jos la figura 6.

Această caracteristică include în principal servicii de backend care ajută dezvoltatorii să-și construiască și să-și gestioneze aplicațiile într-un mod mai bun.

Serviciile incluse în această funcție se pot vedea la figura 6.

2.4.1 Baza de date real-time

Firebase Realtime Database este o bază de date NoSQL bazată pe cloud, care vă gestionează datele la o viteză uluitoare de milisecunde. În cel mai simplu termen, poate fi considerat un fișier JSON mare. (7)

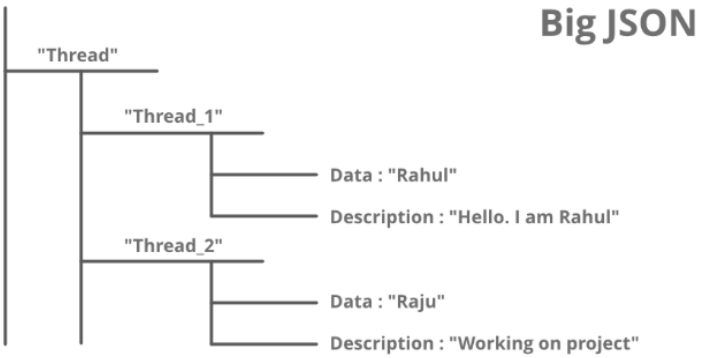


Figure 7 Big JSON

2.4.2 Cloud Firestore

Cloud Firestore este o bază de date de documente NoSQL care oferă servicii precum stocarea, sincronizarea și interogarea prin intermediul aplicației la scară globală. Stochează date sub formă de obiecte cunoscute și sub numele de Documente. Are o pereche cheie-valoare și poate stoca tot felul de date, cum ar fi șiruri de caractere, date binare și chiar arbori JSON. (7)

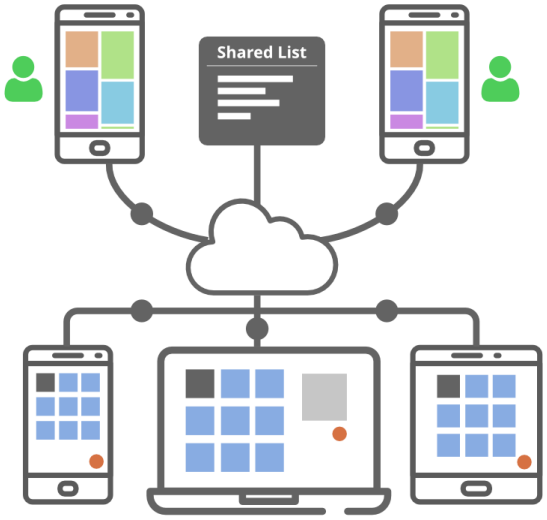


Figure 8 Cloud firestore

2.4.3 Autentificare

Autentificare serviciul Firebase Authentication oferă biblioteci UI și SDK-uri ușor de utilizat pentru a autentifica utilizatorii în aplicația dvs. Reduce forța de muncă și efortul necesar pentru dezvoltarea și menținerea serviciului de autentificare a utilizatorilor. Se ocupă chiar și de sarcini precum fuzionarea conturilor, care, dacă este făcută manual, poate fi agitată. (7)

Figure 9 Autentificare firestore

2.4.4 Remote Config

Serviciul de configurare la distanță ajută la publicarea imediată a actualizărilor pentru utilizator. Modificările pot varia de la modificarea componentelor interfeței de utilizator până la modificarea comportamentului aplicațiilor. Acestea sunt adesea folosite în timp ce se publică oferte sezoniere și conținut pentru aplicația care are o viață limitată. (7)

Figure 10 Remote config firestore

2.4.5 Găzduire

Firebase oferă găzduire de aplicații cu viteză și securitate. Poate fi folosit pentru a găzdui site-uri web și microservicii Stati sau Dynamic. Are capacitatea de a găzdui o aplicație cu o singură comandă. (7)

2.4.6 Firebase Cloud Messaging (FCM)

Serviciul FCM oferă o conexiune între server și utilizatorii finali ai aplicației, care poate fi folosită pentru a primi și trimite mesaje și notificări. Aceste conexiuni oferă siguranță în funcționare și sunt eficiente din punct de vedere al bateriei. (7)

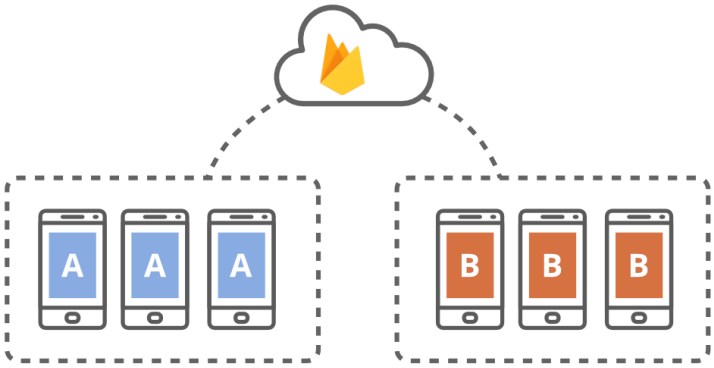


Figure 11 Firebase Cloud Messaging

2.4.7 Firebase Analytics

Oferă o perspectivă asupra utilizării aplicației, însă contracost. Această caracteristică permite dezvoltatorului să înțeleagă cum folosesc consumatorii produsul. SDK-ul are feauterul de a surprinde evenimente, proprietăți și permite de asemenea, obținerea de date personalizate. (5)

2.4.8 Firebase Storage

Facilitează transferul de fișiere ușor și sigur, indiferent de calitatea retelei pentru aplicațiile Firebase. Firebase Storage este susținut de stocarea Google Cloud, acesta este un serviciu de depozitare a datelor. Programatorul îl poate folosi pentru a stoca poze, audio, video sau altele conținuturi generate de utilizatori. (5)

2.4.9 Firebase Test Lab for Android

Test Lab oferă infrastructură bazată pe cloud pentru testarea aplicațiilor Android. Cu o singură operație, dezvoltatorii își pot testa aplicațiile pe o mare varietate de dispozitive și dispozitive configurate. Diverse rezultate ale testelor, cum ar fi capturi de ecran, videoclipuri și jurnale, sunt disponibile în consola Firebase. Un avantaj este că, chiar dacă un dezvoltator nu a scris un cod de testare pentru aplicația sa, TestLab poate rula aplicația automat, căutând blocări. (5)

2.4.10 Firebase Crash Reporting

Rapoartele detaliate de eroare sunt create în aplicație. Erorile sunt grupate după asemănare și gradul de severitate. O altă caracteristică este că dezvoltatorul poate înregistra evenimente personalizate pentru a ajuta la capturarea pașilor care duc la blocarea aplicației. (5)

2.4.11 Firebase Notifications

Permite notificări specifice utilizatorilor pentru dezvoltatorii aplicatiei mobile și serviciile sunt disponibile gratuit. (5)

2.3.12 Utilizarea feature-urilor Firebase în aplicație

Utilizarea feature-urilor firebase în aplicatii andorid este foarte ușoară și nu necesită decât câteva linii de cod. (5)

❖ *Autentificare*

```
FirebaseAuth auth=FirebaseAuth.getInstance();
auth.signInWithEmailAndPassword(email, password)
 .addOnCompleteListener(new OnCompleteListener())
{
        @Override
        public void onComplete (Task task){
                if(task.isSuccessful()){
                        FirebaseUser user=task.getResult().getUser();
                        String email=user.getEmail();
                        //...
                }
        }
});
```

Code 6 Firebase autentificare exemplu

După ce toate dependențle firebase sunt adăugate utilizarea este foarte simplă cum sugerează și exemplul de autentificare de la code 6.

❖ *Baza de date*

Baza de date real-time este folosită, după ce dependințele Firebase pentru baza de date sunt adăugate în aplicație datele, nestructurate pot fi adăugate în baza de date. Următorul exemplu justifică acest fapt.

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Code 7 Firebase message examplu

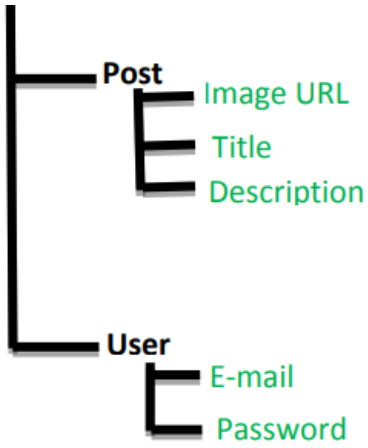


Figure 12 Structura unei baze de date Firebase exemplu

❖ *Stocarea datelor*

Fișierele de poze, video clipuri și audio se pot stoca în BD. Aceste date sunt securizate și solide, deci la apariția unor erori, se reia ultimul punct din rețea. Pentru stocare trebuie respectați anumite pași:

- ❖ Crearea unei instanțe *FirebaseStorage storageobject = FirebaseStorage.getInstance();*
- ❖ Crearea referinței la locația respectivă *StorageReference FileRef = storageRef.child("path");*
- ❖ Încărcarea fișierelor se face cu *metodele putFile(), putData()* sau *putStream()* ce returnează un UploadTask

❖ *Inserarea datelor*

Pentru inserarea de date se poate folosi un object sau un map. După ce obiectul sau mapa este creată, urmează să adaugăm referința firebase la poziția unde urmează ca „copilul” să fie inserat.

Metode folosite pot fi: *ref.push().setValue(xobiect) or ref.setValue(xobiectt)* (5)

❖ *Actualizarea datelor*

Pentru acatualizare trebuie adăugată referința firebase a elemetului „părinte” la care este dorită actualizarea, mai apoi se crează o mapă care conține valorile cu care se vor înlocui cele actuale.

Metode folosite pot fi: *ref.updateChildren(mapă)* (5)

❖ *Ștergerea datelor*

Pentru ștergera de date trebuie adăugat refeința firebase elementului și se cheamă urmatoarea metoda: *ref.removeValue(xobiect)*

2.5. Jira

Jira este o platformă, dezvoltată de compania australiană Atlassian, care permite managementul proiectelor în toate fazele acestora – planificare, urmărire, lansare a software-ului.

Figure 13 Jira

DE CE AM ALES JIRA

Jira este unul dintre cele mai populare instrumente de management de proiecte din lume, el este folosit pe larg în companii renumite. Există șanse mari că noul angajat într-o companie să cunoască deja Jira – economisind astfel timpul pentru instruirea lui în nouă companie.

Jira ne oferă posibilitatea de a crea și configura proiecte, boarduri de lucru, sarcini, aici putem planifică, prioritiza și distribui activitățile, estima efortul depus pentru îndeplinirea sarcinilor, de a crea și vizualiza diferite tipuri de rapoarte, și multe altele datorită instrumentelor și aplicațiilor adiționale.

Flexibilitate și Personalizare - Putem utiliza Jira pentru a urmări orice tip de sarcini, putem personaliza saricinile pe care le folosim, configura statuturile sau workflow prin care va trece fiecare tip de sarcină; putem configura diverse tipuri de boarduri (Scrum, Kanban, Roadmap), utiliza diverse tehnici de estimare a efortului (story points, timp, chiar și unitatea de măsură proprie), de a integra instrumente și aplicații adiționale (Confluence, Trello, Bitbucket, Slack, Zoom, Teams, Jenkins etc).

Confidențialitate și Securitate – Jira investește în GDPR (The General Dată Protection Regulation 2016/679 is a regulation în EU law on dată protection and privacy în the European Union and the European Economic Area), protecția datelor personale, criptarea datelor, diverse nivele de permisiune etc.

Utilizare - Deși Jira este un produs complex, el este intuitiv și ușor de utilizat.

Transparență - Fiecare sarcina este atribuită în mod clar fiecărei persoane, fiecare membru al echipei știe cine și la ce lucrează la moment, oricine poate lasă un comentariu sau verifică în istoric ajustările efectuate asupra unei sarcini – transparență și aflarea tuturor

“pe aceeași pagină” a avut mereu valoare în echipele repartizate global, acum în perioada pandemiei și WFH devenind indispenabilă.

CUM SE UTILIZEAZĂ JIRA

- PAGINA PROTECULUI

Pentru a crea sau accesa un issue în cadrul proiectul în care activăm, deschidem pagină proiectului. O putem face din Meniul principal (bară de meniu din partea superioară a paginii) > Projects.

- CUM NAVIGĂM ÎN JIRA

Din meniul din partea de sus putem naviga în interiorul Jira. Incluzând diferite proiecte, boarduri, issues care nu sunt neapărat legate de un board

Din meniul vertical din partea stânga putem naviga în interiorul unui proiect – în backlog, sprint active, rapoarte pentru diferiți metrici dacă au fost definiți

- BACKLOG

În stânga paginii avem un meniu lateral al proiectului selectat. De aici putem naviga către Backlog, Sprinturile Curente, Rapoarte, Issues și altele.

Pe pagină Backlog putem vizualiza lista tuturor lucrărilor (stories, tasks, bugs, improvements) care trebuie îndeplinite în cadrul proiectului de-a lungul duratei acestuia. Odată ce un issue trebuie luat în lucru conform planului, acesta este transferat din Backlog în Sprint de către Scrum Master.

- SPRINTUL CURENT

Un Sprint este o perioadă scurtă (în mod ideal între două și patru săptămâni) în care echipa de dezvoltare implementează și furnizează un increment pentru produs, de exemplu o versiune a aplicației.

Itemii sunt repartizați pe coloane în dependență de statutul lor curent. (8)

Capitolul 3

DETALII IMPLEMENTARE

Capitolul de față se construiește pe ideea unei prezentări clare a tuturor aspectelor legate de aplicație. Vom pune în evidență prin secțiuni diferite cele mai importante componente/module ale aplicației RoNature. Vom descrie sistemul, componentele lui, vom furniza fragmente de cod mai importante și diagra UML a aplicației.

Pentru arhitectura aplicației, vom avea în vedere și modalitatea prin care au fost integrate în aplicație tehnologiile enumerate mai devreme în capitolul 2 și anume: cum a fost construită baza de date folosită pentru stocarea informațiilor și tehnologiile folosite pentru realizarea UI și backend al aplicației mobile RoNature.

3.1 Autentificare și ecranul de pornire

Ecranul de pornire, reprezintă acel ecran care apare de fiecare data la deschiderea aplicației pentru câteva secunde, acesta a fost proiectat cu ajutorul unui xml și îl vom vedea în următorul a capitol, la prezentarea aplicației.

Când deschidem aplicatia pentru cateva secunde va aparea „splash screenul” în acest timp cat utilizatorul vede acest screen în spate se verifica da suntem conectati și astfel se deschide pagina principală a aplicației, dacă nu se deschide pagina de autentificare.

Autentificarea se realizează folosind Firebase; primul pas este conectarea aplicației cu baza de date Firebase. Ca mod de autentificare este folosit metodă de email și parolă. La accesarea aplicației se verifică dacă de pe dizpozitivul respectiv a fost autentificat un utilizator mai devreme și dacă a rămas logat, se vă deschide Home page-ul aplicației, altfel se va deschide pagina Login/Registration.

```
FirebaseUser user = mAuth.getCurrentUser();
if (user == null) {
    Intent intent = new Intent( packageContext: SplashScreen.this, LoginActivity.class);
    startActivity(intent);
    finish();
} else {
    Intent mainIntent = new Intent( packageContext: SplashScreen.this, DashboardActivity.class);
    mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(mainIntent);
    finish();
}
```

Code 8 Navigare splash screen

Dacă utiliztorul și-a uitat parola, poate să sugereze recuperarea acesteia și i se va trimite un mesaj automat la adresa de email cu care s-a înregistrat. Aplicația pune la dispoziție și autentificarea cu alte conturi deja existente pe platforma Facebook.

3.2 Funcționalități

Vom enumera pe scurt și vom adaugă o mică descriere funcționaliaților principale, mai apoi uremază să le discutăm mai în detaliu pe fiecare în parte. Aplicația are la dispoziție 6 funcționalități principale, printre care se numără următoarele:

- Profil, aici va aparea timeline-ul utilizatorului și datele personale a contului.
- Users, aici avem lista cu toți utilizatorii din aplicație.
- Map, aici vor apărea pozele adăugate în jurul unei locații alese.
- Add Post, cum sugerează și numele se creează postările de către utilizatori.
- Chat, utilizatorii pot începe conversații cu ceilalți utilizatori.
- Home, aici vor fi afișate toate postările utilizatorilor.

3.2.1 Profil

Pagina de profil conține toate datele personale a utilizatorului, cu opțiunea de a le edita. Utilizatorul apăsând pe butonul de editare poate să își schimbe parola, numele de utlizator și poza de profil, care se vor afișa pe pagina, alături de timelinul cu toate postările lui.

Un aspect foarte important la pagina de profil este opțiunea de al edita: se poate schimba numele de utilizator; se poate schimba parola, această opțiune este iar important din punct de vedere a securității; se întâmplă să ne uităm parola, sau cineva să îl afle și astfel utilizatorul este nevoit să îl schimbe; se poate adaugă o poză de profil și de fundal după alegere prin accesarea pozelor stocate în folderul telefonului sau prin intermediul camerei dizpozitivului.

Mai jos vă fi prezentata o secvență de cod pentru actualizarea parolei și a numelui, actualizarea pozei de profil se face similar cu adăugarea postărilor ce vom discuta mai târziu. Pentru editare se creează o nouă activitate care se deschide prin apăsarea butonului de editare din fragmentul profilului.

Prima data se verifica dacă parola introdusă este cea veche a contului, dacă coincide cu cea stocata în baza de date, iar dacă această condiție este îndeplinită atunci se vă schimba parola, altfel va apărea un mesaj de eroare. Așa cum am menționat anterior utlizatorul mai are posibilitatea de a-și schimba parola, la etapa de autentificare, în caz că nu își amintește parola va fi trimis un mesaj către mailul cu care este înregistrat și așa poate să își seteze o alta parolă.

```
private void updatePassword(String oldp, final String newp) {
    progressDialog.show();
    final FirebaseUser user = firebaseAuth.getCurrentUser();
    AuthCredential authCredential = EmailAuthProvider.getCredential(user.getEmail(), oldp);
    user.reauthenticate(authCredential)
            .addOnSuccessListener(aVoid -> user.updatePassword(newp)
                    .addOnSuccessListener(aVoid1 -> {
                        progressDialog.dismiss();
                        Toast.makeText( context: EditProfilePageActivity.this,  text: "Changed Password",
                                Toast.LENGTH_LONG).show();
                    }).addOnFailureListener(e -> {
                        progressDialog.dismiss();
                        Toast.makeText( context: EditProfilePageActivity.this,  text: "Failed password update",
                                Toast.LENGTH_LONG).show();
                    })).addOnFailureListener(e -> {
        progressDialog.dismiss();
        Toast.makeText( context: EditProfilePageActivity.this,  text: "Failed password update",
                Toast.LENGTH_LONG).show();
    });
}
```

Code 9 Update password

Pentru adăugarea, actualizarea numelui de utilizator se creează o conexiune cu baza de dată, real time database și cu ajutorul unui hashmap, construit din cheie-valoare se face update-ul. Se crează un layout ca userul să introducă numele dorit, se preia textul introdus,

care reperzintă valoarea hasmapului, și pe baza cheii se face actualizarea la câmpul respectiv.

După adăugarea datelor în Firebase, mai urmează să fie preluat aceste date și afișate în fragmentul de profil. Informațiile se preiau de la nodul users din baza de date, în așa fel încât id-ul utilizatorului curent să coincidă cu id-ul utilizatorului de la nodul users.

```
HashMap<String, Object> result = new HashMap<>();
result.put(key, value);
databaseReference.child(firebaseUser.getUid()).updateChildren(result).addOnSuccessListener(aVoid -> {
    progressDialog.dismiss();

    Toast.makeText( context: EditProfilePageActivity.this,  text: "Updated name",
            Toast.LENGTH_LONG).show();
}).addOnFailureListener(e -> {
    progressDialog.dismiss();
    Toast.makeText( context: EditProfilePageActivity.this,  text: "Unable to update",
            Toast.LENGTH_LONG).show();
});
```

Code 10 Update name 1

Încă un aspect important ce trebuie tratat: Când se schimbă numele de utlizator este nevoie să verificăm toate postările, comentariile, conversațiile, reacțiile.. toate acțiunile în care acesta a fost implicat și trebuie să schimbăm și acolo numele cu cel nou setat. Mai jos voi oferi ca un exemplu pentru schimbarea numelui la postari deja înregistrate:

```
if (key.equals("name")) {
    final DatabaseReference databaser = FirebaseDatabase.getInstance().getReference( path: "Posts");
    Query query = databaser.orderByChild("uId").equalTo(uId);
    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
                String child = databaser.getKey();
                dataSnapshot1.getRef().child("uname").setValue(value);
            }
        }
```

Code 11 Update name 2

Tot pe pagina de profil se creează un time-line, unde se afișează în ordine cronologică, de la ce mai nouă la cea mai vechi, postările utilizatorului curent. Implementarea nu este una prea complicată, în xmlul fragmentului de profil se mai adaugă

un scrol și recycle view, mai apoi în fișierul java se creează o nouă funcție care preia din nodul postări, valorile în așa fel că id-ul ușerului curent să coincidă cu id-ul ușerului de la postare, afișarea postărilor pe pagina de profil se face cu ajutorul calaselor model și adapter post. Tot pe pagina de profil se creează un time-line, unde se afișează în ordine cronologică, de la ce mai nouă la cea mai vechi, postările utilizatorului curent. Implementarea nu este una prea complicată, în xmlul fragmentului de profil se mai adaugă un scrol și recycle view, mai apoi în fișierul java se creează o nouă funcție care preia din nodul postări, valorile în așa fel că id-ul ușerului curent să coincidă cu id-ul ușerului de la postare.

```
for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
    ModelPost modelPost = dataSnapshot1.getValue(ModelPost.class);
    posts.add(modelPost);
    adapterPosts = new AdapterPosts(getActivity(), posts);
    postRecycleV.setAdapter(adapterPosts);
}
```

Code 12 Photo timeline

3.2.2 Users

Pagina numită users, permite vizualizarea tuturor utilizatorilor înregistrați în apicație, punând la dispoziție o listă cu acestea. Dacă în aplicație sunt înrregistrate mai puține persoane, câteva zeci, este ușor să glișăm până când le găsim, însă la un număr mai mare de persoane acest lucru devine mult mai greu, de aceea s-a implementat funcționalitatea de a căuta și de a afișa o listă doar cu persoanele căutate, după cele două criteri, dupa numele de utilizator sau adresa de email. În lista utlizatorilor le apare imaginea de profil, numele, adresa de email cu care s-au înregistrat și statusul lor de activitate: online/offline.

```
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        usersList.clear();
        for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
            ModelUsers modelUsers = dataSnapshot1.getValue(ModelUsers.class);
            usersList.add(modelUsers);
            adapterUsers = new AdapterUsers(getActivity(), usersList);
            recyclerView.setAdapter(adapterUsers);
        }
    }
```

Code 13 Users list

Se creează o clasă de model pentru utlizatori cu valorile necesare și se generează setteri și getteri pentru acestea și o clasă de adapter, care conține toate funcțiile necesare. Se creează un nou layout model pentru cum vor fi afisați utlizatorii și fragmentul principal. Secventa de cod prezentată de mai jos este o bucată din funcția ”loadUsers”.

3.2.3 Map

În proiect am integrat și o funcționalitate cu hartă, pentru a vă imagina mai ușor, menționez faptul că seamănă cu google photos map. Apăsând pe butonul map va apărea o hartă pe care va fi marcată locația curentă a utilizatorului, evident dacă este dată permisiunea de a folosi locația. Iar în jurul locului specificat la postare sub forma unei marcaj cu poză vor fi afișate pozele pe mapă.

```
private GoogleMap mMap;
private ActivityMapsBinding binding;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    binding = ActivityMapsBinding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());

    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync( callback: this);
}
```

3.2.4 Add Post

La navigarea paginii „Add Post” utilizatorul va vedea un layout (va fi prezentată în capitolul 4) unde este nevoie de un camp pentru locație, descriere, un image view și un buton. Aceste elemente vizuale sunt realizate în xmlul de add post frament. Implementarea functionalității din spate este mai complicată și se va insista mai mult în cele ce urmează pe adăugarea de imagine așa cum am mentionat anterior la subtitul „Profil”.

Se crewaza view-ul în xml și logica din spate se face în clasa „Add Post” fragment. În primul pas se fac acțiuni pentru a lua informațiile legate de userul curent cu ajutorul unui query din baza de date. Va fi nevoie de aceste infromații, deoarece la crearea unei postări, trebuie să afișăm userul, alături de poza lui de profil, din partea căruia provine postarea.

```
for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
    name = dataSnapshot1.child("name").getValue().toString();
    email = "" + dataSnapshot1.child("email").getValue();
    dp = "" + dataSnapshot1.child("image").getValue().toString();
}
```

Code 14 Take user data from firebase

Câmpurile destinate locației, descrierii și imaginii nu pot să rămână necompletate la crearea unei postări, în caz contrar se va afișa un mesaj de eroare, lânga câmpul nerespectiv, atragand atentia userului să competeze campul.

Când se detecteaza apasarea imaginii, se afiseaza un dialog cu doua optiuni: de a alege pozele din galeria foto a dispozitivului mobil, sau a face o poză cu ajutorul camerei care este disponibil de către dispozitiv.

```
builder.setItems(options, (dialog, which) -> {
    requestCameraPermission();
    requestStoragePermission();
    if (which == 0) {
        pickFromCamera();
    } else if (which == 1) {
        pickFromGallery();
    }
```

Code 15 Option dialog

Este nevoie evident ca aplicația să aibă accesul de a folosi camera și spațiul de stocare, altfel se cere permisiunile și dacă sunt date se pot folosi datele, dacă nu se va afișa un mesaj corespunzător. Dacă se obține accesul pentru folosirea galeriei foto, programul are acces doar la fotografii și astfel putem realiza o postare cu succes.

```
case STORAGE_REQUEST: {
    if (grantResults.length > 0) {
        boolean writeStorageaccepted = grantResults[0] == PackageManager.PERMISSION_GRANTED;
        if (writeStorageaccepted) {
            pickFromGallery();
        } else {
            Toast.makeText( context: this,  text: "Please Enable Storage Permissions",
                    Toast.LENGTH_LONG).show();
        }
    }
}
break;
```

Code 16 Storage request

Dacă se obține acces doar pentru a scrie în memoria dispozitivului, prin folosirea camerei, acest lucru nu înseamnă că avem acces și la fotografia salvată în galeria, facută prin intermediul aplicației, deci trebuie să dăm acces și la galerie în caz contrar nu se poate efectua o postare cu succes și se va afișa un mesaj informativ corespunzător.

```
switch (requestCode) {
    case CAMERA_REQUEST: {
        if (grantResults.length > 0) {
            boolean camera_accepted = grantResults[0] == PackageManager.PERMISSION_GRANTED;
            boolean writeStorageaccepted = grantResults[1] == PackageManager.PERMISSION_GRANTED;
            if (camera_accepted && writeStorageaccepted) {
                pickFromCamera();
            } else {
                Toast.makeText( context: this,  text: "Please Enable Camera and Storage Permissions",
                        Toast.LENGTH_LONG).show();
            }
        }
    }
    break;
```

Code 17 Camera request

Dupa ce se trece cu succes de aceste etape, aplicația a primit accesul cerut și toate câmpurile sunt completate prin apăsarea butonului de upload, programul trece într-o altă etapă. Aici mă voi opri un pic și vreau să prezint o bucată mai mare de cod referitoare la adăugarea pozelor, deoarece o întâlnim la chat, la profil și am decis să tratez mai în amănunte aici la adăugarea pozelor.

```
private void pickFromCamera() {
    ContentValues contentValues = new ContentValues();
    contentValues.put(MediaStore.Images.Media.TITLE, "Temp_pic");
    contentValues.put(MediaStore.Images.Media.DESCRIPTION, "Temp Description");
    imageuri = this.getContentResolver().insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
            contentValues);
    Intent camerIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    camerIntent.putExtra(MediaStore.EXTRA_OUTPUT, imageuri);
    galleryActivityResultLauncher.launch(camerIntent);
}

    private void pickFromGallery() {
    Intent galleryIntent = new Intent(Intent.ACTION_PICK);
    galleryIntent.setType("image/*");
    galleryActivityResultLauncher.launch(galleryIntent);
}
```

Code 18 Pick photo from Camera and Gallery

Codul de mai sus prezintă cele două funcții care se ocupă de extragera pozelor fie din galerie fie prin camera, dar mai întâi trebuie să avem acces la aceste să facem acest lucru. Bucata de cod prezentată de mai jos, ne va prezenta cererea și verificarea de acces.

In etapa următoare la adăugarea pozei se verifică dacă se încărcă datele postării cu succes în baza de date iar abia după aceasta se afiseaza mesajul cu verificarea ca postarea a fost publicată cu succes. La această etapă pot apărea diferite erori, nu am dat image uri-ul corect, nu avem referința corectă la baza de date, nu există nodul la care dorim să încărcam datele... etc. Când datele sunt încărcate cu succes în baza de date firebase se va vedea ceva asemănător la 3.3 Baza de date, am ținut să ilustrez acest exemplu, pentru a fi observată și modul în care se salveaza datele în firebase, real time database.

3.2.5 Home

Pagina de „Home”, este primă pagină care apare când se deschide aplicația, are și un nume sugestiv, home page. Pe această pagină, apar toate postările utilizatorilor, inclusiv

a utilizatorului curent. Postările apar în ordine cronologica de la cea mai nouă până la cea mai veche postare. Aici este implementată și funcționalitatea de căutare(search) și tot pe această pagină avem și funționalitatea de logout.

O să începem să discutăm în amănunte despre fiecare funcționalitate. Doresc prin a începe cu funcționalitatea de logout. Dacă ne putem autentifica în aplicatie este firesc ca dorim și să ne deconectăm și cel mai la îndemână este ca aceasta fucționalitatea să fie pe pagina cu care se deschide aplicația.

```
if (item.getItemId() == R.id.logout) {
    firebaseAuth.signOut();
    startActivity(new Intent(getContext(), SplashScreen.class));
    getActivity().finish();
}
```

Code 19 Logout

Pe partea de sus a paginii avem un meniu, când se detectează apăsarea acestuia aici va apărea și funcționalitatea de a ne deconecta. Deconectarea se face cu metoda de signout, pusa la dispozitie în firebase auth, dupa aceasta operatie aplicatie se intoarce la pagina de start pentru cateva secunde, pagina numita „splash screen”, iar dupa la login de unde putem naviga la register.

Tot în partea de sus avem și functionalitatea de a căuta. Această metodă este una necesară, în caz ca dorim să căutam o postarea și sunt sute de postări înregistrate deja. Căutarea se poate face după locație sau în descrierea postării. În functie de ce tastează utilizatorul, cu ajutorul unui query se face o căutare în baza de date, dacă se găsește un titlu sau descriere ce contine textul introdus se vor afișa toate postările.

```
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

    if (dataSnapshot.child(postId).hasChild(myuid)) {
        likeBtn.setCompoundDrawablesWithIntrinsicBounds(R.drawable.ic_hearted,  top:
        likeBtn.setText("Liked");
    } else {
        likeBtn.setCompoundDrawablesWithIntrinsicBounds(R.drawable.ic_heart,  top: 0
        likeBtn.setText("Like");
    }
}
```

Code 20 Reacție la postări

```
private void searchPosts(final String search) {
    DatabaseReference databaseReference = FirebaseDatabase.getInstance()
            .getReference( path: "Posts");
    databaseReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            posts.clear();
            for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
                ModelPost modelPost = dataSnapshot1.getValue(ModelPost.class);
                assert modelPost != null;
                if (modelPost.getTitle().toLowerCase().contains(search.toLowerCase()) ||
                        modelPost.getDescription().toLowerCase()
                                .contains(search.toLowerCase())) {
                    posts.add(modelPost);
                }
                adapterPosts = new AdapterPosts(getActivity(), posts);
                recyclerView.setAdapter(adapterPosts);

            }
        }
```

Code 21 Search posts

Pe pagina de home sunt afisate postarirel utilizatorilor în ordine cronologică. Fiecare postare are locația, descrierea în fotografia alături de numele și fotografia utilizatorului care a creat postarea respectivă. Totodată la fiecare postarea este marcată data și ora la care a fost creată. Postările pot primi reacții de love și comentarii, iar numarul de aprecieri și comentarii sunt afișate dedesuptul pozei și se apdatează în timp real. Fragmentul de cod care tratează reacțiile utilizatorilor se poate vedea la Code 21. Autorii postării au dreptul de a edita și a șterge o anumită postare, care le aparține apăsând de partea dreaptă a postării pe butonul more.

In captura de ecran ce suprinde un fragment de cod, se poate vedea o postarea cu toate detaliile aferente.

```
for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
    String ptitle = dataSnapshot1.child("titleETxt").getValue().toString();
    String descriptions = dataSnapshot1.child("description").getValue().toString();
    uimage = dataSnapshot1.child("uimage").getValue().toString();
    hisdp = dataSnapshot1.child("udp").getValue().toString();
    hisuid = dataSnapshot1.child("uId").getValue().toString();
    String uemail = dataSnapshot1.child("uemail").getValue().toString();
    hisname = dataSnapshot1.child("uname").getValue().toString();
    ptime = dataSnapshot1.child("ptime").getValue().toString();
    plike = dataSnapshot1.child("plike").getValue().toString();
    String commentcount = dataSnapshot1.child("pcomments").getValue().toString();
    Calendar calendar = Calendar.getInstance(Locale.ENGLISH);
    calendar.setTimeInMillis(Long.parseLong(ptime));
    String timedate = DateFormat.format( inFormat: "dd/MM/yyyy hh:mm aa", calendar).toString();
    nameTxt.setText(hisname);
    titleTxt.setText(ptitle);
    descriptionTxt.setText(descriptions);
    likeTxt.setText(plike + " Likes");
    timeTxt.setText(timedate);
    commentTxt.setText(commentcount + " Comments");
    if (uimage.equals("noImage")) {
        imageIV.setVisibility(View.GONE);
    } else {
        imageIV.setVisibility(View.VISIBLE);
        try {
            Glide.with( activity: PostDetailsActivity.this).load(uimage).into(imageIV);
        } catch (Exception e) {
```

Code 22 Post details

3.2.6 Chat

Când navigăm la pagina „Chat” apare o listă cu persoanele cu care au intrat deja în contact, sau pot crea contacte noi. Persoanelor de contact le apare numele, poza de profil și ultimul mesaj din conversație. Intrând în conversație, userul poate trimite mesaje text sau imaginii. Mesajele trimise pot fi șterse dacă userul se râsgandește.

Se crează layouturi pentru mesaje în xml-uri în urmatorul fel, vor fi două așa numite coloane în conversație un xml corespunzator pentru fiecare, o coloană pentu mesajele userului curent și în cealaltă parte mesajele primite de la o altă persoană. Avem o activitate

care combină cele două fragmente cu colanele și are un edit text unde utilizatorul poate introduce textul, apasand be image button-ul send, mesajul va fi trimis. Lângă edit textul pentru inroducerea mesajului, avem încă un image button, apăsând pe ea putem atașa imagini.

Partea de backend se creeaza două calse: clasa model și adapter pentru chat, în model sunt definite toate atributele alături de getteri și setteri și în adapter sunt implementate toate funcțiile care vor opera asupra atributelor. Am creat o funcție în clasa adapter pentru ștergerea mesajelor. Când detectează ca utilizatorul apasă mai lung pe un mesaj, care poate să fie poză sau text, va apărea un allert box cu două opțiuni. Mesajul se poate șterge, adică se va înlocui mesajul actual cu un mesaj cu textul „This message was deleted" sau se poate da unsend, la apăsara acestui buton se mesajul trimis va dispărea. La prima opțiune se suprascrie data salvată din baza de date cu mesajul „This message was deleted". La a doua opțiune se folosește funcția „datasnapshot.getChildren(„..")" și „removeValue()", pentru a înțelege mai bine vom atașa bucată de cod mai jos.

```
if (dataSnapshot1.child("sender").getValue().equals(myuid)) {
    // unsend
    dataSnapshot1.getRef().removeValue();
```

Code 23 Undesnd message

```
//delete
 HashMap<String, Object> hashMap = new HashMap<>();
hashMap.put("message", "This Message Was Deleted");
dataSnapshot1.getRef().updateChildren(hashMap);
Toast.makeText(context, text: "Message Deleted.....",Toast.LENGTH_LONG).show();
```

Code 24 Delete message

Ștergerea în ambele moduri se va aplica în ambele conversatii și la destinatar și la expediator. Dar trebuie să menționăm ca am tratat excepția când destinatarul încearcă să șteargă, în orice mod, mesajul trimis de expediator să nu o poată finaliza și să apară o notificare corespunztoare.

Când trimitem un mesaj text, datele acestuia sunt luate, se creeaaza o referință către baza de date și se încărca mesajul.

```
DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference();
String timestamp = String.valueOf(System.currentTimeMillis());
HashMap<String, Object> hashMap = new HashMap<>();
hashMap.put("sender", myuid);
hashMap.put("receiver", uid);
hashMap.put("message", message);
hashMap.put("timestamp", timestamp);
hashMap.put("dilihat", false);
hashMap.put("type", "text");
databaseReference.child("Chats").push().setValue(hashMap);
final DatabaseReference ref1 = FirebaseDatabase.getInstance().getReference( path: "ChatList")
        .child(uid).child(myuid);
ref1.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

        if (!dataSnapshot.exists()) {
            ref1.child("id").setValue(myuid);
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
```

Code 25 Message and details

Mai apoi pe cealaltă parte acest mesaj în funcție de către cine arată referința id-lui, acest mesaj este extras din firebase și este afișata în chatul destinatarului, alături de adresa și informațiile expediatorului.

Când se apasă buton „attachement” va apărea un dialog cu două opțiuni, să atașăm o poza din galerie, sau să facem prin aplicație o poza pe care să o trimitem. La alegerea unei opțiuni dacă nu este permisiunea dată pentru galerie și stocarea datelor, v-a apărea un nou dialog unde putem să dăm acces, mai apoi putem contiuna să navigam printre poze și să alegem pe care dorim să îl trimitem. La alegera unei poze, se caută referința pozei, image uri-ul, și acesta v-a fi stocta în firebase mai multe detalii legate de tratarea pozelor au fost tratate mai inainte la 3.2.4 App Photos, se procedează la fel. Singura diferență este că la selectarea unei poze acesta este trimisa și nu postată.

```
private void readMessages() {

    chatList = new ArrayList<>();
    DatabaseReference dbref = FirebaseDatabase.getInstance().getReference().child("Chats");
    dbref.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            chatList.clear();
            for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
                ModelChat modelChat = dataSnapshot1.getValue(ModelChat.class);
                if (modelChat.getSender().equals(myuid) &&
                        modelChat.getReceiver().equals(uid) ||
                        modelChat.getReceiver().equals(myuid)
                                && modelChat.getSender().equals(uid)) {
                    chatList.add(modelChat);
                }
                adapterChat = new AdapterChat( context: ChatActivity.this, chatList, image);
                adapterChat.notifyDataSetChanged();
                recyclerView.setAdapter(adapterChat);

            }
```

Code 26 Read messages

La trimiterea fiecarui mesaj, de tip text sau poza, se reține și stochează și data și ora exacta când a fost trimisă. Dedesuptul mesajelor va apărea textul „delivered” și dacă destinatarul a intrat în conversație va aparea mesajul „seen”, adică mesajul a fost citit.

```
private void checkTypingStatus(String typing){

    DatabaseReference dbref = FirebaseDatabase.getInstance()
            .getReference( path: "Users").child(myuid);
    HashMap<String,Object> hashMap=new HashMap<>();
    hashMap.put("typingTo", typing);
    dbref.updateChildren(hashMap);
}
```

Code 27 Typing status

În cazul în care un utilizator ne scrie, se va afișa mesajul „Typing...”. Orice tip de modificare facută de către utilizatori se schimbă, se updatează în timp real în baza de date.

```
private void checkOnlineStatus(String status){

    DatabaseReference dbref = FirebaseDatabase.getInstance()
            .getReference( path: "Users").child(myuid);
    HashMap<String,Object> hashMap=new HashMap<>();
    hashMap.put("onlineStatus", status);
    dbref.updateChildren(hashMap);
}
```

Code 28 Online status

Când ieșim din conversație și ajungem în lista de mesaje, aici va apărea starea persoanelor, online sau data și ora când au fost ultima data online. În lista de mesaje, în dreptul fiecărui chat în parte, se poate vedea profilul și numele persoanelor, respectiv ultimul mesaj din conversație. Dacă ultimul mesaj din conversație este o poză, aceasta nu poate fi afișată și astfel utilizatorul va vedea mesajul „Photo”.

Toate datele sunt salvate în firebase sub nodul „Chat”.

3.3 Baza de date

Datele aplicației sunt stocate în bd Firebase, aici la dispoziție doua baze de date, baza de date real time și firesore database. Dupa un studiu comparativ între cele două, am decis ca voi alege baza de date real time, deoarece acesta este mai util privind nevoile aplicației.

Figure 14 Real timpe database

In cele ce urmează vom enumera pașii pe care le-am urmat pentru a conecta aplicația din android studio cu baza de date firbase. Înaninte de toate am avut nevoie să instalez

ultima versiune de android studio și să setez ultima versiune la dependintel, deoarece firebase este constant updatat astfel nu ar fi fost compatibil cu o versiune mai veche.

Am creat un proiect cu același nume ca a aplicației în firebase, la crearea unui proiect se generează automat un fișier cu configurații, denumita „google-services.json" pe care trebuie să îl includem în proiectul nostru. Pentru ca acest fișier să poată face conexiune cu baza de date avem nevoie să adăugăm alte dependinte în gradle fisier, voi atașa mai jos o imagine cu aceste dependinte.

În proiectul firebase am creat și baza de date real time în care vom avea salvate datele într-un format Json. Big Json-ul aplicației noastre arată ca în imaginea de mai jos. Cum observăm există nodrui care la rândul lor au noduri separate cu seturi de informații. în nodul Chat avem toate conversațiile salvate, în nodul Post sunt salvate toate postările utilizatorilor, în nodul Likes sunt salvate reacțiile la o anumită postare și așa mai departe...

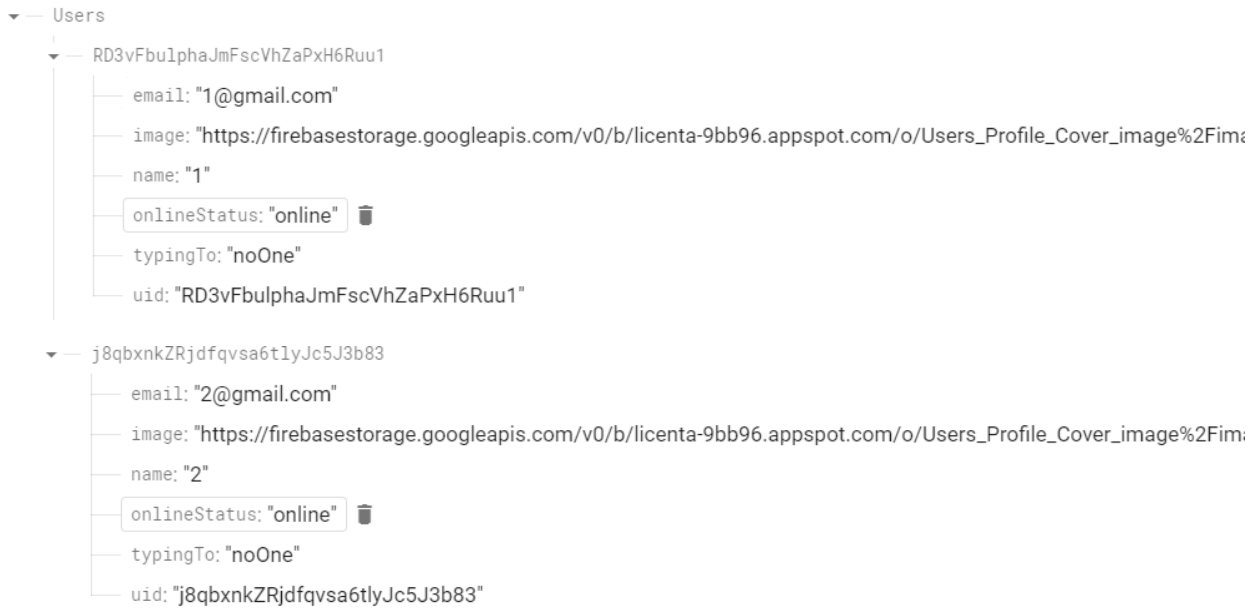


Figure 15 Json Users

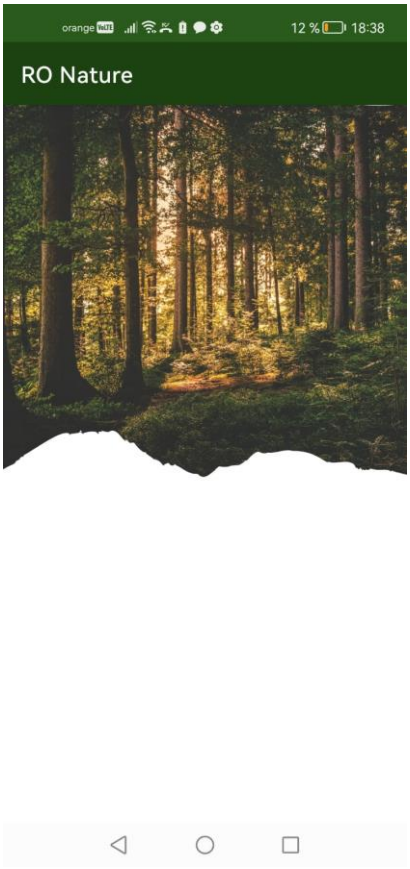
Baza de date are și un fișier cu configurări, aici putem seta accesul la baza de date. Este important ca setările default să fie schimbate, deoarece baza de date este publica și oricine poate ajunge la date din aplicație. Pentru a reduce vulnerabilitatea bazei de date au acces pentru a scrie și a citi datele în primul rand doar utilizatorii care sunt înregistrări în aplicație, în al doilea rând au acces doar la date care le aparțin și aici pot aduce modificări.

Figure 16 Json Post

Capitolul 4

PREZENTAREA APLICAȚIEI

4.1 Autentificare



Prima pagină cu care utilizatorul intră în contact este splash screenul mai apoi pagină de autentificare sau home page-ul. Din această pagină putem fie să ne autentificăm în aplicație, iar dacă credidențialele sunt corecte să fim redirecționați către pagina principală, fie să navigăm la pagina de înregistrare de unde putem crea un nou cont în aplicație.

Figure 17 Splash screen

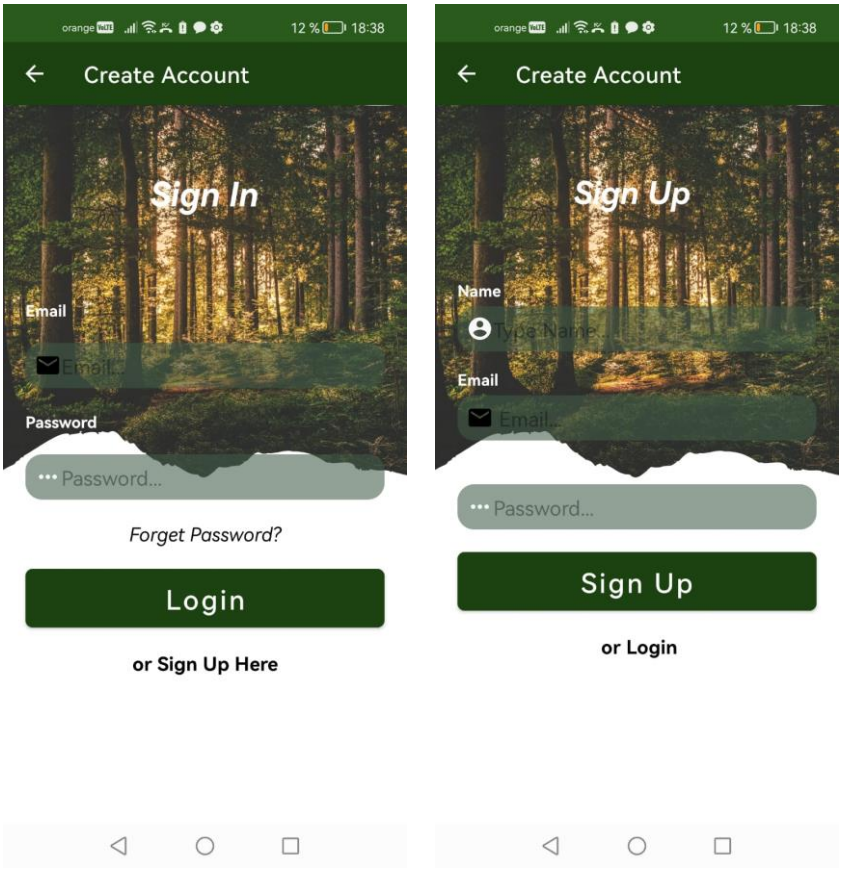


Figure 18 Sign up/in

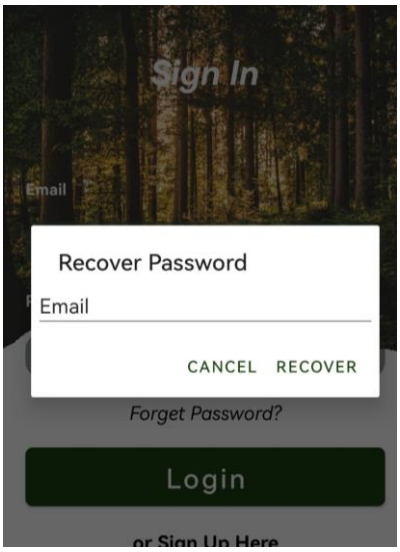


Figure 19 Recover password

În cazul în care utlizatorul incearcă să se logeze, dar nu își amintește de parolă, poate să o recupereze. Se trimite un mesaj la adresa de email cu care utlizatorul a fost înregistrat și aici poaste să își seteze o parolă nouă cu care poate să se logeze.

4.2 Home Page

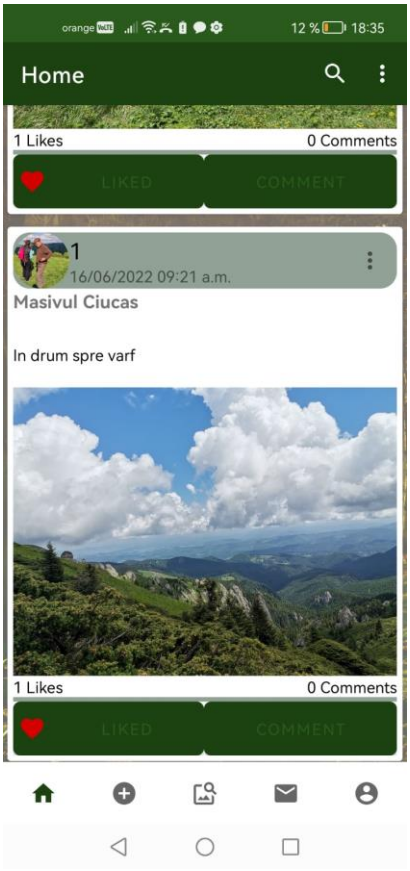


Figure 20 Home page

Home pageul este următoarea pagina cu care vin în contact utilizatorul, după autentificare. În cazul în care nu s-a deconectat aceasta este prima pagina pe care o întâlnește. Aici apar postările tuturor utilizatorilor în ordine cronoligcă după dată și glisând în jos vor apărea și cele postate anterior.

În pareta de sus a aplicației utilizatorul poate să se delogeze și să se folosească de funcționalitatea de cătuare. Căutarea se face dupa locația sau descrierea postării și se vor afișa doar postările corespunzătoare. În partea de jos a paginii se găsesc toate butoanele din lista orizontală, prin intermediul cărora utilizatorul poate să treacă de la o pagină în alta.

4.3 Add Post

Utilizatorul navigând la paginii „Add Post” va vedea un layout cu un câmp pentru locație, descriere, un image view și un buton. Câmpurile trebuie completate, altfel se va afișa un mesaj pentru a atrage atenția utilizatoului să completeze cămpul respectiv.

Pentru atașarea pozei utlizatorul poate alege între două opțiuni: să încarce din galerie sau să facă o poză cu ajutorul camerei, dar înainte de toate, trebuie să dăm acces aplicației la stocarea de memorie și folosirea camerei.

După ce toate câmpurile au fost completate și se apasă pe butonul de POST, toate informațiile sunt încărcate și stocate în baza de date firebase, în timpul încărcări ce durează

câteva secunde, utilizatorul va primi un mesaj cu textul că este în curs de postare. Când toate datele sunt salvate în real time database de aici urmează să fie extrase pentru a fi afișate pe pagina de home.

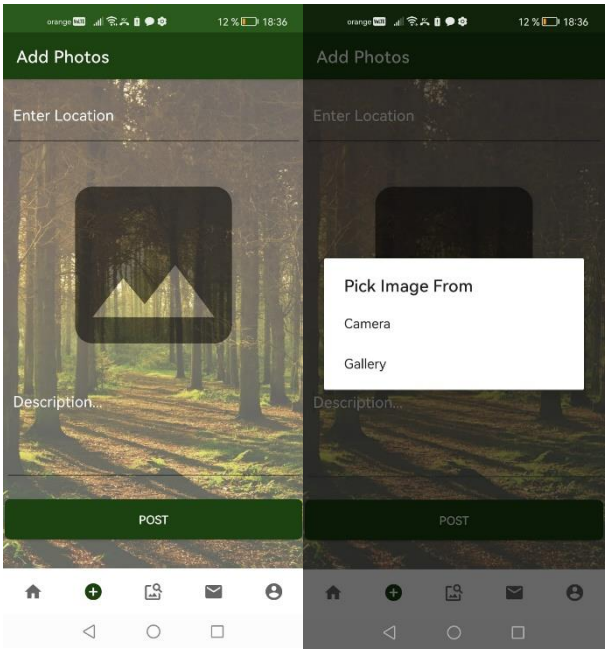


Figure 21 Add post

4.4 Users and Chat

Utilizatorul apăsând de butonul users din lista orizontală, se deschide o pagina cu lista care conține utilizatorilor înregistrați în aplicație. Fiecare utilizator are imaginea de profil, numele și adresa de mail afișată. Lângă căsuța fiecaruia este un emoticon cu mesaj, apăsțnd pe acesta se începe o conversație cu persoana aleasă.

Pentru că aplicația poate să aibă sute de utilizatori și atunci căutatrea nu mai este așa de simplă, în partea de sus a paginii este funcția de căutare. Se tastează un text, numele sau adresa de email dorita și se va afișa o listă cu utilizatori care țin de aceasta.

Navigând mai departe prîntre butoanele orizontale se găsește butonul chat. La apăsarea butonului se deschide o pagină cu lista persoanelor cu care intrasem deja intr-o conversație. Este afișat poza lor de profil, numele și statusul lor, dacă sunt online sau ultima dată când au fost online. Totodată este afișat ultimul mesaj din conversație. În cazul în care

ultimul mesaj este un mesaj imagine se afișează textul ,,Poză,,. Dacă se detectează că utlizatorul a dat click pe una dintre conversații, acesta se deschide.

În partea de sus apare numele persoanei cu care am intrat în conversație, dedesupt statusul online sau data când ultima oară a fost online. În dreapta apar mesajele primite de la destinatar, în stânga apar mesajele trimise de către utilizatorul curent. Mesajele de la destinatar au coloare albă, iar mesajele de la expediator au culoare verde.

În partea de jos sunt doua butoane și un spațiu pentru introducerea mesajului. Apăsând pe butonul de atașare se accesează pozele din dispozitiv, fie prin galerie sau prin cameră. Apăsând butonul de send, se trimite mesajul, evident dacă se găsește introdus un text. La trimiterea fiecărei mesaj va apărea sub ea textul ,,delivered,, pănа când destinatarul îl citește. În cazul în care destinatarul este în curs de a scrie ceva, se va afișa textul ,,is typing,,.

Mesajele se pot șterge dacă utlizatorul apasă lung pe mesajul respectiv și dacă acest mesaj îi aparține. Sunt două optiuni de a șterge sau de a da unsend la mesaj. În cazul în care se șterge, practic în locul mesajului va apărea textul ,,message was deleted,, iar în cazul în care se dă unsend, mesajul va dispărea fără urmă. Trebuie precizat ca ștergera se întâmplă în ambele părți, atât la expediator cât și la destinatar.

Figure 22Profil page

4.5 Profil Page

Pagina de profil conține toate datele legate de utilizatorul curent. În partea de sus, sub forma de cerc este afișată poza de profil și în stânga lui numele de utilizator și adresa de email. Mai în jos în timeline sunt afișate toate postările în ordine cronologică de la cel mai nou la cel mai vechi.

Este un buton rotund edit, apăsând pa acesta utilizatorul are posibilitatea de a-și edita datele personale. Poate să își schimbe poza de profil. Din nou pentru atașarea pozei utlizatorul poate alege între două opțiuni: să încarce din galerie sau să facă o poză cu ajutorul camerei, dar înainte de toate, trebuie să dăm acces aplicației la stocarea datelor în memorie și folosirea camerei.

Numele de utilizator nou trebuie introdusă în dialogul pentru schimbarea numelui și se finalizează cu apăsarea butonului de update. La apăsarea butonului se afișează un mesaj care informează utlizatorul că numele este în curs de resetare, mesajul dispare după ce datele vechi au fost înlocuite cu datele noi în firebase.

Pentru schimabrea parolei, trebuie să intrudcem parola veche, astfel se poate verifica că utilzatorul care se înregistrează este cel care schimba parola și nu altcineva. Dacă cea veche este corecta atunci se reseteaza parola, altfel utilizatorul este înștiințat ca parola veche nu este corectă. La apăsarea butonului se afișează un mesaj care informează utlizatorul că parola este în curs de resetare, mesajul dispare după ce datele vechi au fost înlocuite cu datele noi în firebase.

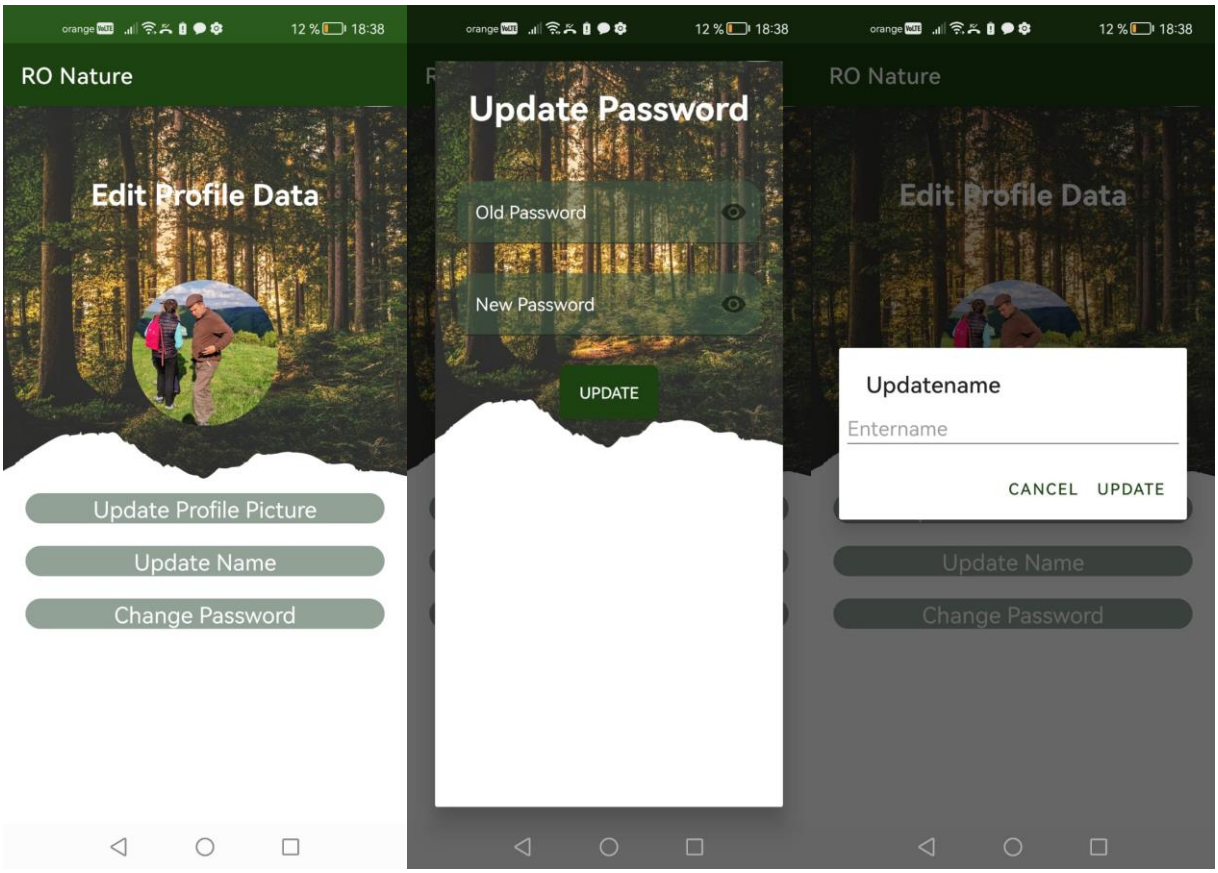


Figure 23 Edit profil

4.6 Map

Funcționează ca goole photos map, se afișează pozele cu pin-uri pe mapă în jurul locației curente pe care o are utilizatorul.

Capitolul 5

CONCLUZIE

Indiferent de complexitatea explicațiilor cu privire la implementarea aplicației, a descrierii modului în care au fost puse în practică ideile specifice, și a noțiunilor teoretice prezentate până acum cu privire la ceea ce reprezintă această aplicație, atât pentru dezvoltator, cât și pentru utilizator, este important de menționat în continuare câteva concluzii bine definite din punct de vedere a aspectelor puse în evidență pe parcursul documentației.

Aceste concluzii au ca scop clarificarea într-un mod recapitulativ, a unor trăsături legate de componentele originale întâlnite în implementare, care ar putea fi interpretate ca fiind puncte forte ale dezvoltării aplicației. Elementele de specialitate despre care dorim să amintim în cele ce urmează, pot fi legate și de părerea oferită în mod constructiv, de către utilizatorul care a folosit aplicația, nefiind luate în calcul doar elementele care nu au precedent în celelalte aplicații deja existente pe piață.

În același timp, dar poate nu în aceeași măsură, vom prevedea și posibilele neajunsuri ale aplicației, deoarece din anumite puncte de vedere se poate vorbi și despre acest considerent. Neajunsurile sunt idei de dezvoltare care probabil nu au fost gândite de programator spre a fi implementate în varianta curentă a aplicației, dar pot fi și aspecte ale perspectivei unui client mai puțin mulțumit.

În plus, neajunsurile pot fi interpretate ca fiind dezavantajele aplicației, de aceea este necesară explicarea în detaliu a elementelor legate de acest subiect. Aceste dezavantaje nu sunt resimțite în mod clar în aplicație, dar un studiu detaliat al implementării ne poate duce cu siguranță, spre elaborarea unei liste din acest punct de vedere.

În mod cert, ne-am focalizat atenția atunci când ne-am îndreptat pașii către dezvoltarea jocului, spre a aduce cât mai multe avantaje utilizatorului în momentul interacțiunii acestuia cu aplicația. Atunci când spunem avantaj ne referim, în principiu, la scoaterea în evidență a acelor elemente pentru care acestă aplicație este socotită ca fiind diferită, de îndată ce se compară cu celelalte existente.

Considerăm ca avantaj, și momentul în care beneficiarii acordă încredere, oferind sugestii de perfecționare sau promovându-l pentru a avea un număr cât mai mare de vizualizări.

Un beneficiu al sesizărilor insuficiențelor pe care le are, ar fi posibilitatea de a aduce în discuție noi idei de dezvoltare, de optimizare sau chiar de îmbunătățire a elementelor deja existente.

Scopul acestui proiect a fost să vină în ajutor tuturor persoanelor care sunt iubitori de excursii, le plac locurile noi, sunt pasionați de fotografie astfel având ocazia să împărtășească obiectivele noi văzute, pe o platformă destinată total pentru această arie de interes. Consider că acest scop a fost atins, utilizatorii pot împărtăși poze alăturând locația, pot să comunice prin intermediul chatului, sau a comentariilor și reacțiilor dipuse sub fiecarea postare. Cred că varianta curentă a aplicației ar putea fi dispusă ca prima varianta care evident va urma să mai fie dezvoltată și noi variante să fie dipuse. Viitoarele propuneri de dezvoltare pentru aplicația din această lucrare sunt:

- afișarea în album a postărilor pe profil, momentan este disponibilă varianta time-line
- accesarea paginii de profil a utilizatorilor
- salvarea postărilor favorite, care să poată fi accesate de pe pagina noastră de profil
- editarea postărilor

Nejunsturile care vor mai apărea acestea pot fi noi idei pentru lansarea unei variante noi a aplicației. Așadar i se pot adăuga diverse funcționalități, care să contribuie la dezvoltarea sa.

Acest proiect m-a ajutat să înțeleg mai bine cum se structurează un proiect adroid, care sunt provocările când dorim să interconectăm mai multe aplicații, în cazul aplicației propuse în lucrare: aplicația mobilă, serverele Google, baza de date Firebase, am învățat să creez interfețe piretenoase cu xml, am acumulat mai multă cunoștință în Java, am învățat cum să îmi calculez mai bine timpul de lucru și nu în ultimul rând cum se să dezvolt o aplicație mobilă.

Bibliografie selectivă

1. **Rețele de Socializare. [Interactiv] [Citat: 26 mai 2022.] https://socialmedialist.org/retele-de-socializare.html.**

2. Ciobana, Marian. Ce este Android Studio? [Interactiv] 26 octombrie 2021. [Citat: 2 mai 2022.] https://geeki.ro/ce-este-android-studio-iata-caracteristicile/.

3. Gattal, Abderraouf. Understand Android Basics. [Interactiv] 28 august 2018. [Citat: 4 mai 2022.] https://medium.com/@Abderraouf/understand-android-basics-part-2-ui-is-all-what-matters-to-the-user-xml-android-6492c434a850.

4. Java. [Interactiv] Wikipedia, 11 mai 2022. [Citat: 29 mai 2022.] https://ro.wikipedia.org/wiki/Java_(limbaj_de_programare).

5. Khawas, Chunnu. Application of Firebase in Android App Development-A Study. [Interactiv] Sikkim University, iunie 2018. [Citat: 29 mai 2022.] https://www.researchgate.net/publication/325791990_Application_of_Firebase_in_Android_App_Development-A_Study.

6. Ce este NoSQL? [Interactiv] Oracle, 2022. [Citat: 13 mai 2022.] https://www.oracle.com/ro/database/nosql/what-is-nosql/.

7. Firebase. [Interactiv] Geeks, 15 iulie 2021. [Citat: 13 mai 2022.] https://www.geeksforgeeks.org/firebase-introduction/.

8. Endava. *JIRA & ZEPHYR FOR JIRA.* Brasov : s.n., octombrie 2021.

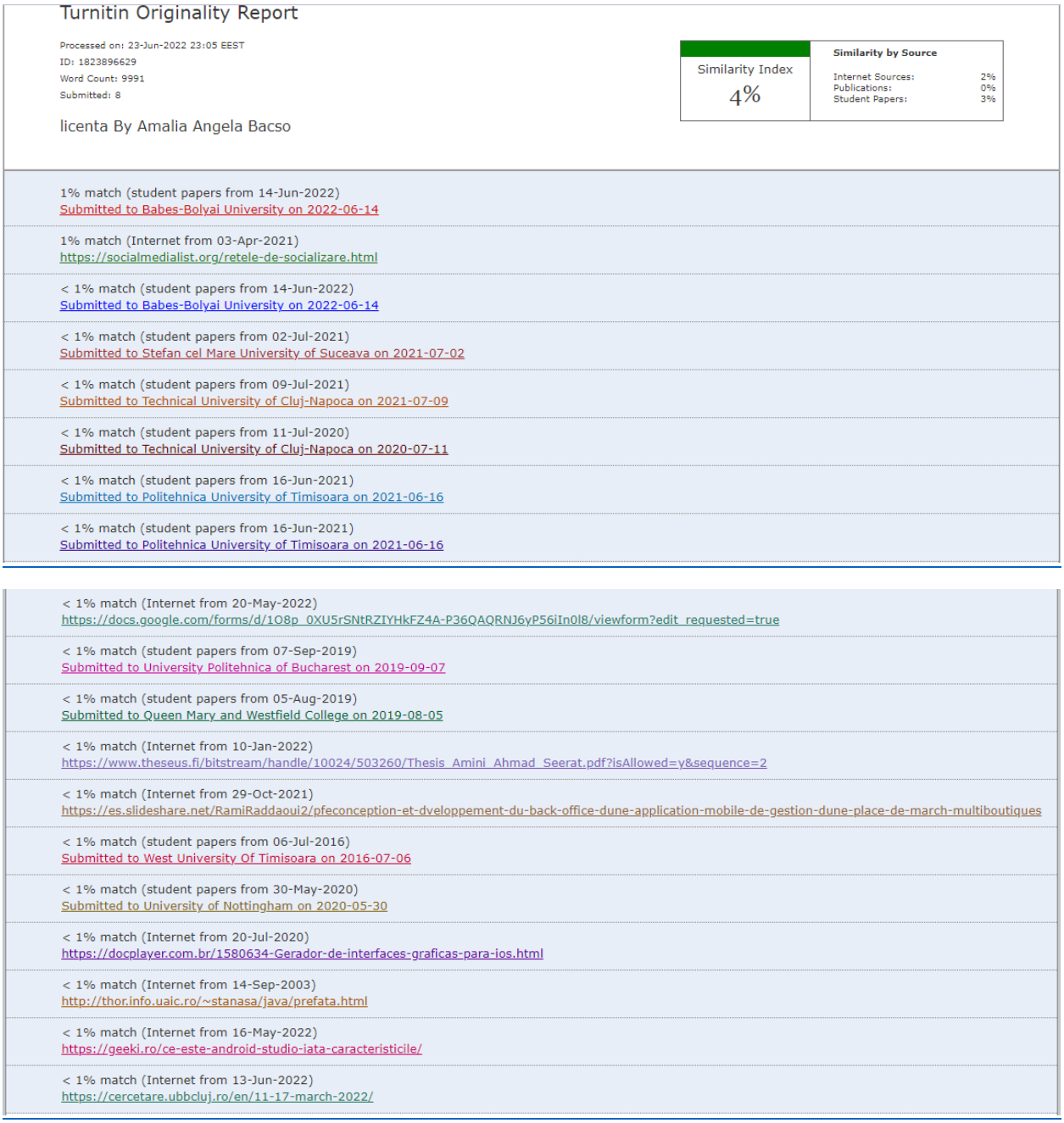
10. S, Ashok Kumar. *Mastering Firebase for Android Development.* iunie 2018.

11. Sheusi, James C. *Android application development for Java Programmers.*

12. *Development of Social Media Strategies in Tourism Destination.* s.l. : Elsevier Ltd.

13. Sahatqija, Kosovare, Ajdari, Jaumin și Zenuni, Xhemal. *Comparison between relational and NOSQL databases.* 2018.

14. *Analyzing the user interface of Android apps.* Konstantin Kuznetsov și Vitalii Avdiienko. 2018.



Turnitin Originality Report

Processed on: 23-Jun-2022 23:05 EEST
ID: 1823896629
Word Count: 9991
Submitted: 8

licenta By Amalia Angela Bacso

Similarity Index
4%

Similarity by Source
Internet Sources: 2%
Publications: 0%
Student Papers: 3%

1% match (student papers from 14-Jun-2022)
Submitted to Babes-Bolyai University on 2022-06-14

1% match (Internet from 03-Apr-2021)
https://socialmedialist.org/retele-de-socializare.html

< 1% match (student papers from 14-Jun-2022)
Submitted to Babes-Bolyai University on 2022-06-14

< 1% match (student papers from 02-Jul-2021)
Submitted to Stefan cel Mare University of Suceava on 2021-07-02

< 1% match (student papers from 09-Jul-2021)
Submitted to Technical University of Cluj-Napoca on 2021-07-09

< 1% match (student papers from 11-Jul-2020)
Submitted to Technical University of Cluj-Napoca on 2020-07-11

< 1% match (student papers from 16-Jun-2021)
Submitted to Politehnica University of Timisoara on 2021-06-16

< 1% match (student papers from 16-Jun-2021)
Submitted to Politehnica University of Timisoara on 2021-06-16

< 1% match (Internet from 20-May-2022)
https://docs.google.com/forms/d/1O8p_0XU5rSNtRZIYHkFZ4A-P36QAQRNJ6yP56iIn0l8/viewform?edit_requested=true

< 1% match (student papers from 07-Sep-2019)
Submitted to University Politehnica of Bucharest on 2019-09-07

< 1% match (student papers from 05-Aug-2019)
Submitted to Queen Mary and Westfield College on 2019-08-05

< 1% match (Internet from 10-Jan-2022)
https://www.theseus.fi/bitstream/handle/10024/503260/Thesis_Amini_Ahmad_Seerat.pdf?isAllowed=y&sequence=2

< 1% match (Internet from 29-Oct-2021)
https://es.slideshare.net/RamiRaddaoui2/pfeconception-et-dveloppement-du-back-office-dune-application-mobile-de-gestion-dune-place-de-march-multiboutiques

< 1% match (student papers from 06-Jul-2016)
Submitted to West University Of Timisoara on 2016-07-06

< 1% match (student papers from 30-May-2020)
Submitted to University of Nottingham on 2020-05-30

< 1% match (Internet from 20-Jul-2020)
https://docplayer.com.br/1580634-Gerador-de-interfaces-graficas-para-ios.html

< 1% match (Internet from 14-Sep-2003)
http://thor.info.uaic.ro/~stanasa/java/prefata.html

< 1% match (Internet from 16-May-2022)
https://geeki.ro/ce-este-android-studio-iata-caracteristicile/

< 1% match (Internet from 13-Jun-2022)
https://cercetare.ubbcluj.ro/en/11-17-march-2022/

Figure 24 Rezultat originalitate de la Turnitin