**UNIVERSITATEA DIN BUCUREȘTI**

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**SPECIALIZAREA INFORMATICĂ**

**Lucrare de licență**

**Social Platform pentru călătorit**

**Absolvent Bădescu George-Gabriel**

**Coordonator științific**

**Prof Dr. Ștefănescu Alin**

**București, septembrie 2022**

**Abstract**

Conceptul de platformă de socializare a apărut relativ recent în istoria tehnologică a omenirii, prima platformă de “social media” apărând în urmă cu mai bine de 20 de ani în 1997 aceasta numindu-se “SixDegrees.com” [1], unde utilizatorii puteau să se asocieze între ei și puteau să își trimită mesaje. De atunci numărul platformelor de socializare a fost într-o continuă creștere, acestea având mai mult sau mai puțin succes, până în anul 2004 când s-a lansat Facebook [1], platforma care este utilizată de aproape 3 miliarde de utilizatori activi [2] și în zilele noastre, fapt ce ne arată că platformele de “social media” sunt prezente și vor face parte din viețile noastre de aici înainte.

Tema acestei lucrări este reprezentată de aplicația numită “Travel-IT – Aplicație Android pentru oamenii pasionați de călătorii”, o aplicație menită să unească oamenii ale căror pasiuni sunt călătoritul, cunoașterea de locuri noi, exprimarea opiniilor și dorința de a-și împărtăși experiențele proprii cu cei din jur. Am ales ca platforma să fie dezvoltată pe mobile, deoarece numărul de utilizatori care dispun de un smartphone este în continuă creștere iar, aplicațiile de tip “social media” sunt utilizate în mod frecvent prin intermediul acestor dispozitive.

În cadrul dezvoltării aplicației am ales să folosesc următoarele tehnologii: Android având ca bază limbajul Kotlin, o alternativă modernă pentru Java, împreună cu diferite servicii oferite de Firebase precum: Authentication pentru logare, Firebase Realtime Database pentru stocarea în timp real a informațiilor, fiind o bază de date de tip No-SQL, Cloud Messaging și Cloud Storage pentru trimiterea de notificări pentru useri. De asemenea am folosit și diferite librării din sistemul Android cum ar fi Glide sau Picasso pentru încărcarea imaginilor din Firebase și Volley pentru trimiterea de request-uri HTTP către API-urile folosite. La nivel de arhitectură al aplicației am ales să folosesc MVVM pentru o gestiune mai bună a utilizării codului, separând backend-ul de frontend.

În continuare, se vor prezenta pașii care au dus la dezvoltarea aplicației, atât la nivel tehnologic cât și la nivel conceptual.

**Abstract**

The social media platform concept it appeared relative recently in the technological history of mankind, the first social media platform appeared more than 20 years ago in 1997 on a platform called “SixDegrees.com” [1], where users could connect and send messages to each other. Since then, the number of social media platforms has been in continuous growth, with more or less success until in 2004 when Facebook was launched [1] and became the platform that is used by over a billion users even today [2], a fact that show us that social media platforms are present and will be part from our lives from now on.

The subject of this paper is represented by the application called "Travel-IT - Android application for people who are passionate about traveling", an application intended to unite people whose passions are traveling, getting to know new places, expressing their opinions and wanting to share their own experiences with their friends. I chose for the platform to be developed on mobile, because the number of users who have a smartphone is constantly increasing and social media applications are frequently used through these devices.

During the development of the application I chose to use the following technologies: Android based on the Kotlin language, a modern alternative to Java with various services offered by Firebase such as: Authentication for authentication process, Firebase Real Time Database for real-time storage of information, being a No-SQL database, Cloud Messaging and Cloud Storage for sending user notifications. I also used different libraries from the Android system such as Glide or Picasso for loading images from Firebase and Volley for sending HTTP requests to the APIs that I used. At the application architecture level I chose to use MVVM for better management of code usage, separating the backend from the frontend.

In the following I will present the steps that led to the development of the application both at the technological and conceptual level.

**Lista figurilor**

https://datareportal.com/reports/digital-2022-social-media-referrals?rq=social%20media

**Introducere**

**Context și motivație**

Ne aflăm în secolul XXI supranumit și secolul vitezei în care omenirea a căutat și încă se află în proces de căutare de soluții care să le facă viața mai ușoară prin intermediul tehnologiei. Fie că vorbim despre completarea unor acte în format digital, fie că vorbim despre cumpărături online sau fie că vorbim despre învățământ, se poate observa în mod cert că oamenii caută tot mai des să își rezolve problemele în mediul online deoarece acesta facilitează toate serviciile enumerate, împreună cu multe altele, într-un mod mult mai eficient, nevoile oamenilor reușind să fie rezolvate prin doar câteva click-uri fără a mai fi nevoie să aștepte la cozi interminabile.

Pe lângă aceste servicii care au economisit semnificativ durata anumitor activități, în ultimele două decenii s-a constatat o creștere a interacțiunilor între oamenii în mediul online, deoarece în ziua de azi, cu ajutorul unui telefon ce are o conexiune la internet, oricine poate intra pe o platformă de socializare prin intermediul căreia poate comunica cu prietenii, familia sau colegii de muncă, astfel oamenii pot să țină legătura cu cei din jur mult mai ușor, distanța nemaifiind o piedică în acest sens.

De asemenea, pe langă un mijloc de comunicare cu persoanele apropiate, platformele de social media reprezintă pentru oameni un loc în care își pot exprima liber opiniile, își pot povesti experiențele din diferite întâmplări din viața lor sau pot afla informații noi, astfel cei care au o dorință de afirmare și-o pot satisface printr-o simplă postare. Conceptul de social media reprezintă un avantaj atât pentru întreținerea relațiilor interumane cât și pentru colectarea datelor, pentru aflarea preferințelor acestora referitoare la diferite subiecte cu scopul îmbunătățirii conținutului si al reclamelor oferite de către platformă.

Motivul principal care m-a inspirat în alegerea acestei teme este dat de pasiunea pentru călătorit, fiind atras să cunosc locuri și culturi noi, cât si dorința de a creea de la zero o aplicație prin intermediul căreia toții oamenii care au această pasiune se pot aduna și își pot împărtăși experințele și întâmplările petrecute din timpul călătoriilor.

**Obiective și soluții**

Obiectivul principal al aplicației este acela de a oferi oamenilor pasionați de călătorit un mediu virtual plăcut în care să se simtă liberi să își împărtașeacă experiențele din timpul vacanțelor cu cei din jur și să își poată păstra amintirile din locurile vizitate, cât si viitoarele locuri pe care vor dori să le viziteze.

Soluția pe care am ales să o dezvolt pentru atingerea obiectivului propus este punerea la dispozitia utilizatorului o platformă de social media creată special pentru călătorit. Platforma numită Travel-IT este o aplicație mobile, pentru Android, de actualitate, ce utilizează tehnologii de ultimă generație, care printr-un design modern și o arhitectură riguroasă, împachetează nevoile de bază pe care le are un utilizator când accesează o platformă de acest tip.

Prin intermediul platformei Travel-IT, fiecare utilizator se poate înregistra în modul clasic prin crearea unui cont folosindu-se de email și parolă, sau se poate conecta direct în aplicație dacă deține un cont de Facebook sau de Gmail activ.

În pagina de profil, utilizatorul își poate vizualiza profilul, datele de contact, fotografia de profil, atât utilizatorii pe care îi urmărește cât și pe cei care îl urmăresc pe el, postările proprii, istoricul destinațiilor cât și cele pe care utilizatorul urmează să le viziteze. De asemenea, utilizatorul poate să își editeze profilul după ce l-a creat.

În pagina de home, utilizatorul poate vedea postările utilizatorilor pe care îi urmărește, le poate aprecia aceste postări, poate adăuga comentarii la ele sau le poate raporta la administrator dacă utilizatorul consideră că postarea nu corespunde cu scopul platformei. Tot în cadrul acestei pagini se regăsește o bară de căutare, unde utilizatorul poate căuta persoane din aplicație și mai există o secțiune de unde utilizatorul poate să adauge o postare nouă. Fiecare utilizator își poate edita descrierile la propriile postări după ce le-a publicat și poate să șteargă comentariile asociate unei postării proprii, daca le consideră inadecvate.

Pentru a asigura calitatea conținutului din aplicație am creat și rolul de administrator, care primește postările raportate de utilizatori și poate să decidă dacă păstrează postarea sau, dacă o consideră inadecvată, poate să o șteargă. În cazul în care un utilizator încalcă principiile platformei în mod repetat, administratorul are dreptul să șteargă contul acelui utilizator.

**Structura lucrării**

În paginile ce vor urma am să abordez aspectele din lucrarea de licență pe capitole având următoarea structură:

**Capitolul 1** - se vor prezenta tehnologiile folosite precum mediul de dezvoltare, limbajele, librăriile și API-urile folosite cât și modul în care au fost utilizate acestea în cadrul aplicației .

**Capitolul 2** - se va prezenta analiza aplicației prin stabilirea unui grup țintă, crearea unui grafic pentru a vizualiza prioritizarea cerințelor, compararea aplicației cu cele deja existente pe piață și o analiză SWOT a acesteia.

**Capitolul 3** - se va prezenta pe larg structura bazei de date și rolul fiecărui tabel în aplicație.

**Capitolul 4** – se va prezenta arhitectura aplicației, explicând cum funcționează aceasta și cum se reflectă la nivel tehnic în aplicație.

**Capitolul 5** – se vor prezenta în detaliu funcționalitățile aplicației care sunt puse la dispoziția utilizatorilor, cât și modul în care acestea au fost implementate.

# 1.Tehnologii folosite

În următrul capitol se vor prezenta limbajele de programare alese, mediul de lucru, API-urile folosite, structura codului și mijloacele folosite pentru gestionarea versionării acestuia.

## 1.1 Limbaje de programare folosite

În urma deciziei mele de a dezvolta o aplicație destinată pentru telefoanele cu Android, am avut de ales ca limbaj de bază între Java sau Kotlin. În urma căutărilor de pe internet dar și a dorinței mele de a învăța ceva nou, am ales să folosesc limbajul **Kotlin** deoarece deși în primă fază este mai greu de învățat decât Java [3], limbajul Kotlin îmbină principalele paradigme utilizate în progamare precum: programarea orientată pe obiecte (abstractizare, încapsulare, moștenire și polimorfism) cât și orientarea pe funcții (imutabilitate, funcții care sunt utilizate la același nivel cu clasele și funcțiile pure) [4].

Abstractizarea și încapsularea au fost folosite pentru a defini modificatorii de acces din cadrul claselor implementate, cât și pentru metodele acestora, astfel încât codul să fie modularizat.

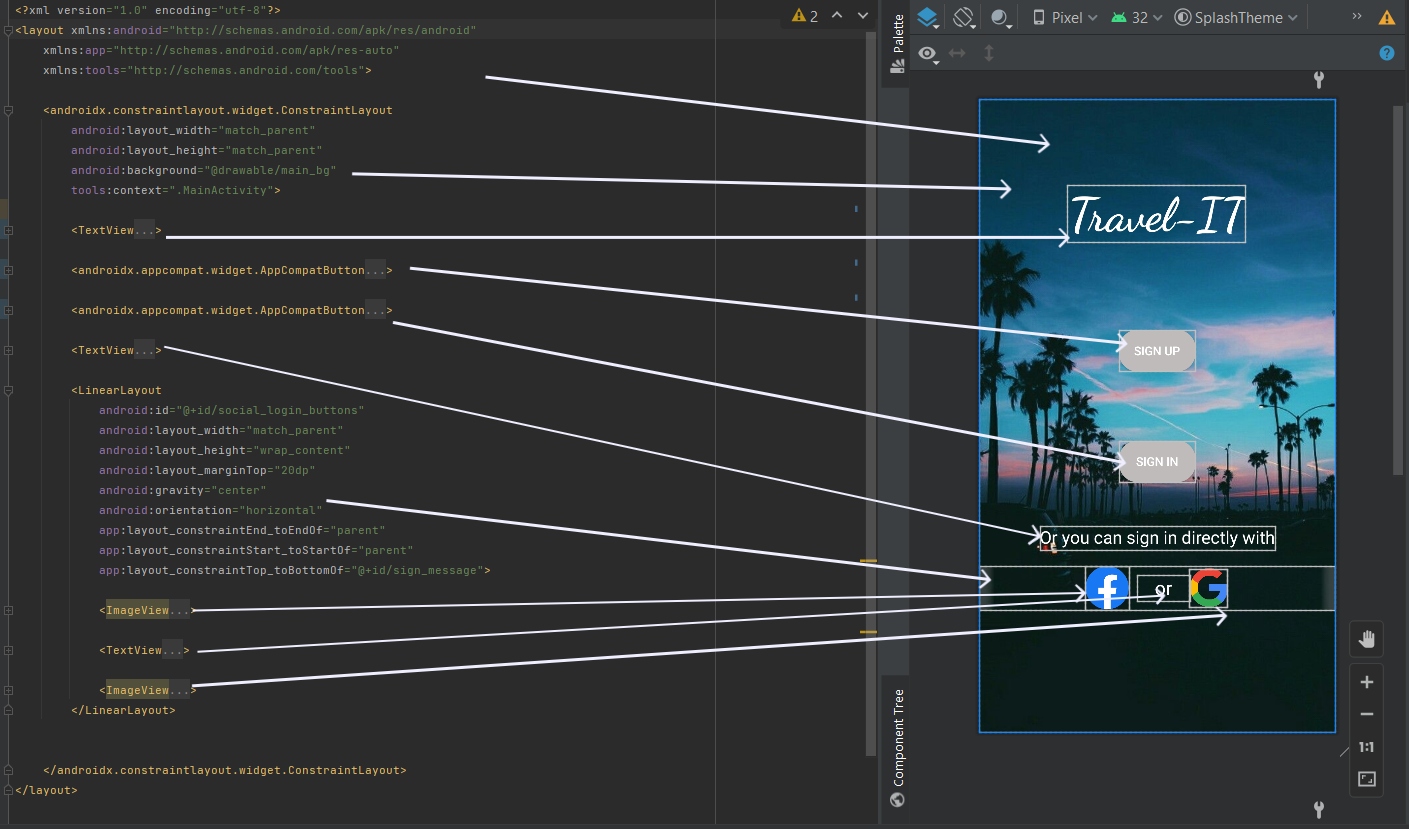
Moștenirea am folosit-o pentru a îmi defini propriile activități sau fragmente derivate din cele de bază din cadrul sistemului Android (Fragment, Activity).

Polimorfismul a fost folosit deoarece am avut nevoie să suprascriu mai multe metode definite în clasele de bază precum: *OnCreate* pentru crearea unei activități, *OnCreateView* pentru crearea unui fragment, *OnBackPressed* pentru definirea comportamentului în momentul în care un user apasă pe *back key-ul* de pe telefon.

De asemenea, un plus pe care îl are Kotlin față de limbajul Java este dat de faptul că în Kotlin se elimină problema legată de *NullPointerException,* prin *Null Safety*, fiecare referință fiind nevoită să fie declarată explicit de cătreprogramator dacă se dorește să poată fi *nullable* [5].

Un alt avantaj al acestui limbaj de programare este programarea asincronă care se realizează cu ajutorul *design pattern-ului* pentru concurență numit *Courutine* [6]prin care se poate suspenda execuția unor *blocking* *thread-uri* la un moment dat, spre deosebire de Java unde gestionarea *multithread-urilor* este mai dificilă [7].

**XML** repreznită un limbaj de marcare ce are ca scop crearea interfeței aplicației de Android prin scrierea *layout-urilor.* Acesta este asemanător cu HTML, care este utilizat pentru aplicațiile web. În cadrul unui *layout* se va defini un element de bază care va fi de tipul View sau ViewGroup, în care se va folosi XML pentru adaugarea obiectelor și proprietăților acestora, astfel, în momentul compilării, fișierul XML va fi văzut ca un obiect de tip View prin suprescrierea metodei *OnCreate()*  din cadrul fragmentelor [8]*.* Pe lângă generarea de *layout-uri* propriu zisă, XML mai este folosit și pentru definirea fișierului *AndroidManifest.xml*, scrierea paletei de culori, pentru stocarea textului din aplicație sau pentru a stiliza icon-urile.



**JSON** [9]este un format de reprezentare și interschimbare de date în cadrul unei aplicații sau între mai multe aplicații find un subset al limbajului JavaScript. Acest format este ușor de înțeles de către oameni, este un limbaj independent, dar folosește convenții asemănătoare precum cele din C/C++, Java, Python etc. În general JSON se găsește de 2 tipuri:

- sub forma unor perechi de tipul cheie/valoare asemenea unor dicționare

- sub forma unei liste ordonate asemenea unui vector

Am utilizat JSON-uri pentru baza de date folosită, Firebase Realtime Database stocându-mi obiectele sub această formă.

## 1.2 Sistemul de operare Android

### 1.2.1 Introducere

Android este un sistem de operare open­­-source ce are la bază o versiune modificată de kernel de Linux. Acesta a fost dezvoltat de Google încă din 2007, este bine documentat, având o popularitate crescută în rândul programatorilor. Pe lângă sistemul de operare dedicat telefoanelor mobile, Android are variații de sistem și pentru ceasuri inteligente (Wear OS), televizoare inteligente Android TV) sau chiar mașini (Android Automotive) [10].

### 1.2.2 Mediul de dezvoltare

Pentru dezvoltarea aplicației în Android am folosit ca și IDE Android Studio [11], fiind IDE-ul oficial pentru acest gen de aplicații. Android Studio este oferit de către Google în parteneriat cu JetBrains și permite dezvoltarea aplicațiilor mobile cât și ale celor dedicate pentru ceasuri, televizoare sau mașini care dispun de sistem de operare Android. Acesta vine cu o interfață modernă, intuitivă, ușor de personalizat, care evidențiază clar structura de bază a proiectului.

IDE-ul vine împreună cu o serie de dispozitive virtuale pe care dezvoltatorul poate alege să își testeze aplicația, sau acesta poate testa direct pe un dispozitiv fizic dacă are Android și un cablu USB. De asemenea Android Studio dispune atât de o gamă largă de librării, cât și de șabloane de cod pentru activități predefinive sau de icon-uri deja existente pentru a facilita dezvoltarea mai ușoară a aplicațiilor. Pe lângă acestea, IDE-ul prezintă și integrare cu sistemul de versionare GitHub, fapt ce ajută la o gestionare mai bună și mai accesibilă a versionării proiectului.

### 1.2.3 Structura unei aplicații Android

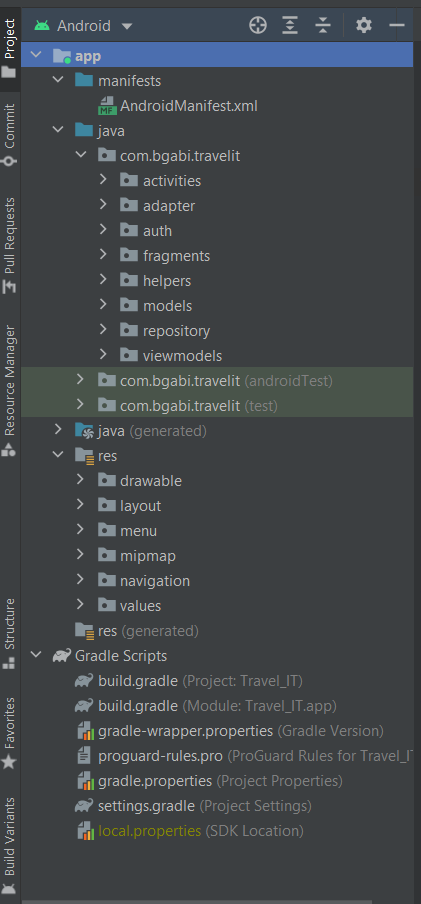
Proiectele dinAndroid Studio au o structură de bază comună [12], fiecare proiect conține unul sau mai multe module ce conțin fișiere cu cod sau resurse. Anatomia de bază a unei aplicații Android este alcătuită din următoarele fișiere:

**-** Manifests: ce conține fișierul AndroidManifest.xml în care se află informații esențiale despre sistemul de operare, Google Play, și informațiile necesare pentru tool-urile care construiesc Android-ul. Acest fișier este unic în cadrul fiecărui proiect și este obligatoriu să conțină informații despre componentele aplicației precum: activități sau servicii, trebuie să fie incluse toate permisiunile pe care trebuie să le aibă utilizatorul pentru ca aplicația să funcționeze corect și este nevoie să fie incluse și funcționalitățile hardware și software de care are nevoie aplicația, pentru stabili compatibilitatea dispozitivelor cu aplicația [13].

- Java: care este alcătuit din fișiere cu cod sursă Kotlin în care se păstrează partea de backend a aplicației, fiind alcătuită din activități, fragmente și alte fișiere ajutătoare ce ajută la definirea funcționalităților aplicației (helpere, repository-uri, viewmodele), precum și fișiere pentru teste unitare.

- Res: care conține resursele aplicației (layout-uri XML, imagini, fișiere audio respectiv video, teme, culori, text), ele alcătuind partea de frontend a aplicației, reprezentând ceea ce o să vadă utilizatorul final.

- Gradle Scripts: ce conțin scripturile și setările legate de Gradle. Acesta este un mecanism de automatizare pentru build-ul aplicației, controlând ciclul construiriii aplicației (compilează fișierele sursă de Kotlin, rulează teste, instalează aplicația pe dispozitiv), determină ordinea optimă de rulare a sarcinilor și se ocupă cu gestionarea dependințelor dintre proiecte si librăriile terțe. Fiecare aplicație conține două fișiere Gradle: unul la nivel de proiect și celălalt la nivel de modul. Gradle-ul de la nivel de proiect are ca scop definirea configurației ce se va aplica pentru toate modulele din proiect. Gradle-ul de la nivel de modul este locul în care se definesc dependințele aplicației și unde se declară versiunile de SDK (acestea se referă la versiunea minimă pe care aplicația poate fi instalată, versiunea de test și versiunea pentru care dezvoltăm aplicația)[14] .



### 1.2.4 Android Jetpack

Android Jetpack reprezintă o suită de librării care ajută la dezvoltarea aplicațiilor ajutând programatorii să folosească cele mai bune practici, reducând dimensiunea codului și scriind cod care poate fi rulat pe majoritatea versiunilor de android, astfel dezvolatorii pot să scrie cod fără să își pună problema dacă ceea ce scriu este compatibil pentru versiunile de Android dorite [15]. Namespace-ul folosit de către librărie este *androidx,* acesta fiind o îmbunătățire a librăriei originale *Android Support Library,* care este scoasă din uz, ea fiind înlocuită în totalitate de *androidx* [16]*.*

Pe parcursul dezvoltării aplicației am folosit diferite librării din cadrul Android Jetpack precum **Navigation Component**, pentru crearea barei de navigație dintre ecrane, **Lifecycle**, pentru a folosi obiectele *LiveData* și *ViewModel* pentru prelucrarea datelor din Firebase Realtime Database și pentru a folosi arhitectura MVVM. Despre librăriile folosite urmează să detaliez în subcapitolul 1.4.

## 1.3 Firebase

Este un software pentru dezvoltarea aplicațiilor atât web cât și mobile pentru partea de backend, punând diferite sericii la dispoziția programatorului, de la stocarea datelor într-o bază de date de tip No-SQL, până la date analitice sau algoritmi de machine learning [17]. Motivele principale care m-au făcut să aleg să folosesc aceasta platformă de servicii în detrimentul altor platforme precum AWS sunt următoarele [18]:

- Eliminarea neccesității de a stoca baza de date local sau pe un server fizic, Firebase menținându-și datele in Cloud, într-o bază de date de tip real-time, astfel datele sunt stocate și sincronizate cu cele din aplicație, ele fiind disponibile chiar și când suntem offline.

- Este un serviciu care inițial este gratis, programatorii pot să stocheze până la 50 de useri și se pot folosi de o gamă largă de servicii fără a fi nevoiți să plătească pentru acestea, astfel, programatorii pot să se familiarizeze cu Firebase și își pot da seama dacă serviciile oferite de acesta sunt potrivite pentru aplicațiile lor.

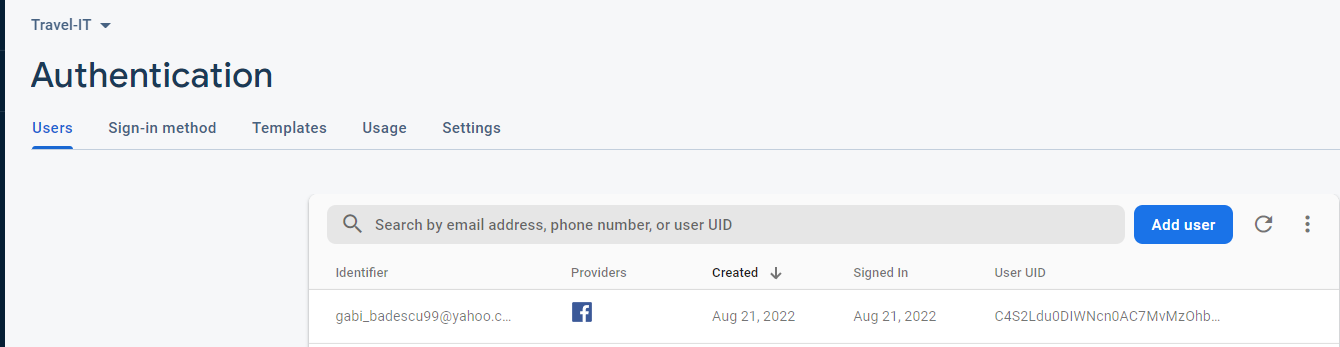
- Pentru stocarea datelor, Firebase folosește baze de date de tip NoSQL, acestea fiind mai rapide decât cele relaționale, astfel se eficientizează fluxul aplicației mai ales când aceasta conține foarte multe date.

- Diversitatea serviciilor oferite, pe langă cele menționate anterior, se mai adaugă si serviciul de autentificare, unde Firebase vine cu posibilitatea de a implementa procesul de logare cu diferite aplicații precum: Facebook, Google, Github, Twitter, oferă serviciu de notificări pentru utilizatorii aplicației, acestea pot să fie programate sau să se trimită în urma unei acțiuni și oferă un serviciu de analiză care captează diferite evenimente din cadrul aplicației ce ajută la îmbunătățirea aplicației.

– Accesibilitatea integrării serviciilor oferite de către Firebase în cadrul aplicației, programatorul având la dispoziție documentația serviciului în cadrul IDE-ului Android Studio, fiind explicați toți pașii necesari adăugării unui serviciu plus posibilitatea de a verifica ce pași au fost implementați cu succes și care nu [19].

În continuare voi prezenta fiecare serviciu oferite de către Firebase și cum a fost folosit acesta în cadrul dezvoltării aplicației.

**Firebase Authentication** este un serviciu de autentificare sigur și ușor de implementat, oferind o soluție de tipul „end-to-end identity solution” [20], permite implementarea diferitor sisteme de autentificare: pe bază de email și parolă, prin intermediul unui cod pe care utilizatorul îl primește pe telefonul personal sau prin aplicații terțe precum Facebook, Google, GitHub etc.



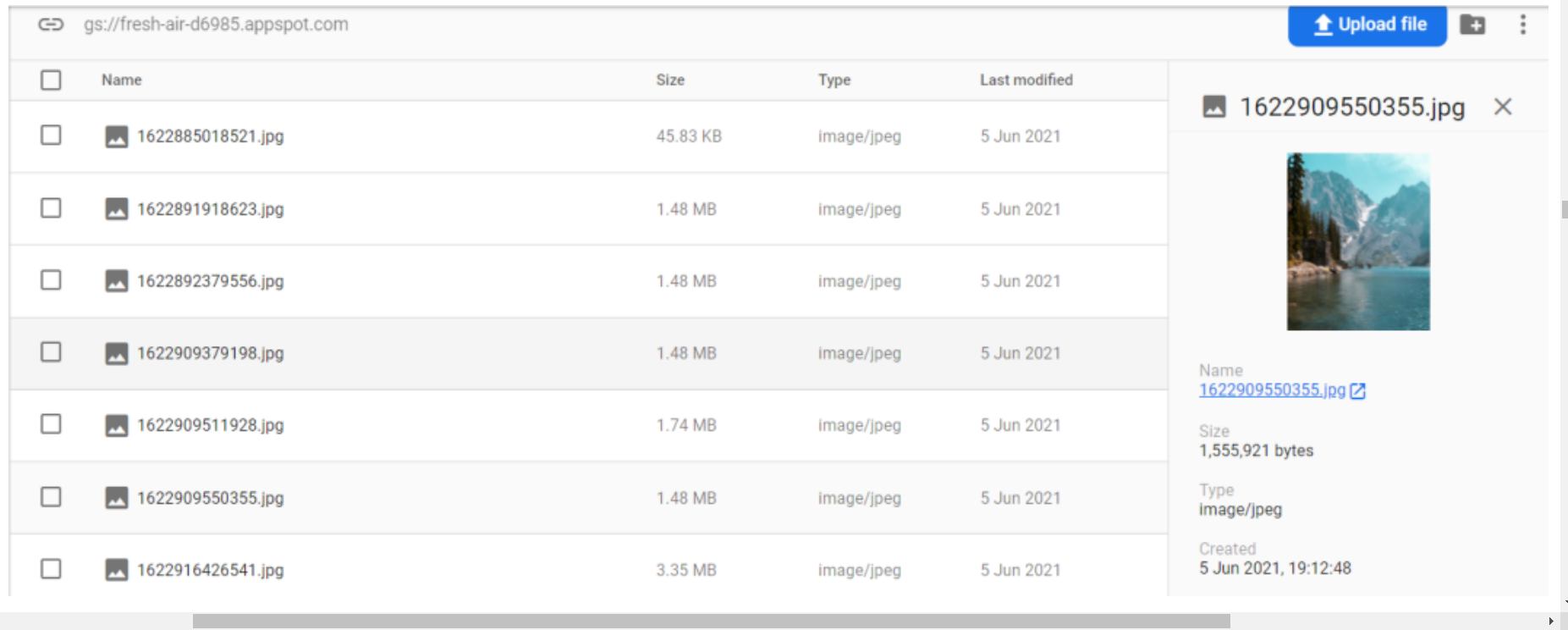
Pentru dezvoltarea aplicației s-a folosit acest serviciu fiind disponibile mai multe metode de autentificare. Prima metodă implementată a fost înregistrarea în aplicațtie cu un email valid și parolă, astfel când un utilizator dorește să se înregistreze în aplicație, el va fi nevoit să introducă un email valid, împreună cu o parolă, aceasta va fi concatenată cu o valoare arbitrară și i se va aplica o funcție hash, valoarea finală fiind stocată în Firebase [21], astfel după cum se poate observa și în poză, programatorul poate să vadă anumite informații despre user cum ar fi email-ul, metoda de autentificare, data la care a fost creat, ultima accesare a aplicației, însă parola este afișată sub forma unui hash, fără a mai fi posibilă extragerea parolei inițiale, respectandu-se astfel confidențialitatea utilizatorilor. Pe langă această metodă de autentificare, au mai fost implementate, atât logarea cu Facebook, cât si cea cu un cont Google asociat dispozitivului. Atunci când user-ul alege una din cele două metode, Firebase va adăuga automat utilizatorul în Authentication, urmând ca la următoarea logare să se logheze direct fără a mai fi nevoie să își mai introducă credențialele în aplicația terță.

**Firebase Realtime Database** reprezintă un tip de baze de date oferit de Firebase, este de tipul No-SQL și este încărcată pe cloud. Datele sunt sincronizate pentru toții userii, fiind actualizate în timp real, sunt păstrate în memoria cache a telefonului, astfel rămân accesibile chiar și când utilizatorii sunt offline, acestea reactualizându-se în momentul în care utilizatorul are din nou conexiune la internet rezolvând în mod automat un eventual conflict între date. Serviciul prezintă un limbaj bazat pe expresii numit **Firebase Realtime Database Security Rules**, unde se definește cum vor fi structurate datele si când se pot face operații de scriere și de citire asupra lor. Acesta se poate integra cu Firebase Authentication, astfel programatorul poate se decidă ce user poate să aibă acess la un aumit set de date din aplicație. Fiind o bază de date non-relațională aceasta este mai optimizată decât una relațională iar API-ul acesteia este construit să accepte doar operații ce se pot executa rapid pentru a putea actualiza datele în timp real, astfel programatorul este nevoit să își organizeze baza de date într-un mod cât mai eficient [22].

O poza sau 2

**Firebase Cloud Storage** este un serviciu dezvoltatorii care vor să stocheze și să afișeze conținut generat de către utilizator precum imagini sau videoclipuri. Acest serviciu oferă securitate de la Google pe partea de încărcare și descărcare de fișiere.

Avantajul folosirii acestui serviciu este dat de diferiți factori precum : permiterea execuției unor operații robuste pentru partea de încărcare și descărcare de fișiere, astfel dacă un utilizator are o conexiune slabă la internet sau în timpul execuției uneia din cele două acesta își pierde conexiunea la internet, în momentul când se va conecta, procesul se va relua de unde a rămas. Acest serviciu este sigur, punând la dispoziția programatorilor o autentificare intuitivă, Cloud Storage-ul venind cu un model de securitate pentru a controla accesul utilizatorilor la fișiere în funcție de numele fișierului, mărimea, conținutul acestuia sau alte metadate [23].



## 1.4 Librării și framework-uri folosite

### 1.4.1 Lifecycle

În Android, fiecare activitate sau fragment are un lifecycle, ce are rolul de a ajuta programatorii să înțeleagă prin ce stadii trec acestea pe măsura ce utilizatorul navighează prin aplicație. Mai jos voi explica procesul prin care trece o activitate, cel pentru fragment fiind unul asemănător.

O activitate are 6 stadii prin care poate trece: *onCreate(), onStart(), onResume, onPause()* și *onDestroy()*, sistemul invocând aceste metode în funcție de acțiunile utilizatorului din aplicație. Programatorul nu este nevoit să implementeze toate aceste funcții, fiind nevoie să le implementeze doar atunci când își dorește ca aplicația să aibă un comportament personalizat pentru un anumit stadiu.

*OnCreate()* este prima metodă ce se apelează când se instanțiază o activitate din ecranul principal sau dintr-un intent. În cadrul acestei metode se inițializează majoritatea componentelor necesare activității, fiind apelată o singură dată.

*OnStart()* are rolul de a face activitatea vizibilă pentru utilizator. Aceasta poate fi apelată de mai multe ori, un caz fiind acela când un utilizator deschide o altă activitate, iar apoi se întoarce la cea inițială.

*OnResume()* este invocată imediat după *onStart(),* aceasta aducând fiecare component asociat cu activitatea curentă în foreground (în ecranul utilizatorului), activitatea fiind interactivă pentru user. Activitatea rămâne în acest stadiu până când utilizatorul o va părăsi.

*OnPause()* este apelată în momentul în care un utilizator navighează către altă activitate sau către altă aplicație. În acel moment, focusul nu va mai fi pe activitate, aceasta trecând in background, eliberând din resurse și oprind operațiile care nu sunt necesare. Programatorul trebuie să se asigure că toate resursele pe care le va elibera când va apela această metodă, vor fi realocate în metoda *onResume()*.

*OnStop()* se apelează de regulă după *onPause()* când majoritatea proceselor din activitate au fost oprite. Aceasta are loc când userul navigează către altă activitate sau când apasă butonul de home, eliberând majoritatea resurselor și operațiilor care nu sunt necesare când activitatea nu este vizibilă. În acest punct, activitatea încă există, view-urile rămânând încă active, iar când userul va redeschide activitatea, componentele nu se vor reîncărca, ci vor reveni dintr-un *backstack.*

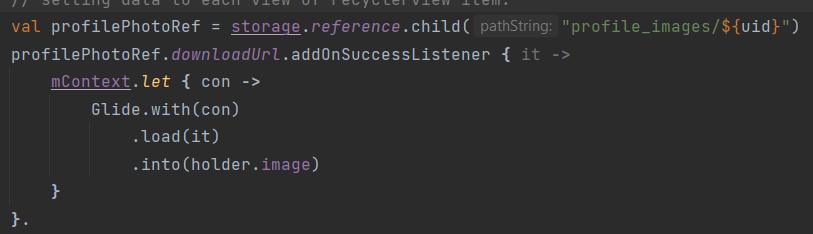
*OnRestart()* se execută în momentul în care un user resetează activitatea, acesta putând să interacționeze din nou cu componentele din cadrul activității. Când un user revine în activitate, deoarece metoda de creare a activității se va apela o singură data, se va apela metoda de start, apoi cea de resume.

*OnDestroy()* se apelează cand activitatea s-a oprit din execuție. Această metodă se poate apela și în cazurile în care userul nu parăsește activitatea, precum rotirea ecranului sau schimbarea limbii din aplicație, sistemul distrugând activitatea si recreând-o cu configurațiile necesare [24].

În cadrul aplicației au fost folosite diferite componente din această librărie precum LiveData care trimite notificări mai departe către observatori. Deoarece este de tipul *lifecycle-aware,* acesta va ține cont de stările celorlalte componente, asigurându-se că le va actualiza doar pe cele care sunt active în acel moment [25]. Pentru stocarea si managerierea datelor din interfață în contextul lifecycle-ului am folosit ViewModel, ajutând la păstrarea datelor chiar și după o rotire a ecranului. Aceste componente au avut un rol esențial în colectarea și trimiterea datelor mai departe către utilizator, asigurând un proces consecvent al experienței utilizatorului cu aplicația, aceasta așteptând ca datele să fie primite din Firebase și după aceea le va afișa în interfață.

### 1.4.2 Glide

Este un framework pentru gestionarea și procesarea rapidă și eficientă a încărcării imaginilor, fiind o librărie open source, acreditată de către Google. Acesta permite preluarea, decodarea și afișarea datelor imaginilor, sau videoclipurilor. Glide utilizează un API robust care permite programatorului să își acceseze imaginile din aproape orice loc.



În aplicația Travel-IT am folosit Glide pentru a aduce imaginile din Firebase Storage și a le încărca în interfață, acestea fiind, fie de tipul *CircleImgeView* pentru fotografia de profil a uitilizatorului, fie *ImageView* pentru postări [26].

În figura de mai sus este prezentat un exemplu de folosire al framework-ului Glide prin următorii pași:

- Se verifică dacă există o fotografie de profil în fișierul *profile\_images* pentru un utilizator, folosindu-ne de uid este este o cheie unică generată de Firebase Authentication.

- În caz afirmativ, vom folosi contextul curent și vom încărca imaginea din URL-ul găsit în câmpul corespunzător.

- Dacă utilizatorul nu deține o fotografie de profil, atunci acestuia îi se va atribui din oficiu o fotografie defult.

### 1.4.3 MDC (Material Compenents for Android)

Este o librărie ce conține un set de componente pentru a construi o interfață modernă și compactă a aplicației folosind *Material Design,* pentru a face experiența utilizatorului cât mai plăcută și mai intuitivă [27].

În cadrul dezvoltării aplicației am ales să folosesc diferite componente precum CardView pentru postări sau notificări, Buton, TextInputLayout împreună cu EditText pentru a oferi field-urilor care trebuie completate de către utilizator un design modern, sau BottomNavigation, fiind o alternativă nouă pentru Navigation Drawer, deoarece în ziua de azi, cele mai multe modele de telefoane au ecranul mare, astfel BottomNavigation pune la dispoziția utilizatorului un mod mai accesibil de a naviga în cadrul unei aplicații.

De asemenea am folosit și componente predefininte din cadrul temelor, utilizându-le în fișierul themes.xml precum Theme.MaterialComponents.Light.NoActionBar, pentru a ascunde bara de acțiuni din Splash Screen și din pagina de intrare a aplicației.







### 1.4.4 Maps SDK for Android

Este o librărie folosită pentru integrarea hărților personalizate în cadrul sistemului Android, având la baza Google Maps. Această librărie a fost folosită în cadrul aplicației penru generarea unei hărți care să conțină istoricul locațiilor vizitate.

Astfel, fiecare utilizator poate să își acceseze harta proprie, fiecare locație fiind preluată din postările pe care utilizatorul le-a publicat, acestea fiind marcate cu câte un pin roșu. În cadrul hărții, utilizatorul poate să ajusteze scara la care se uită pe hartă și mai are la dispoziție un buton ce îl duce la locația curentă.

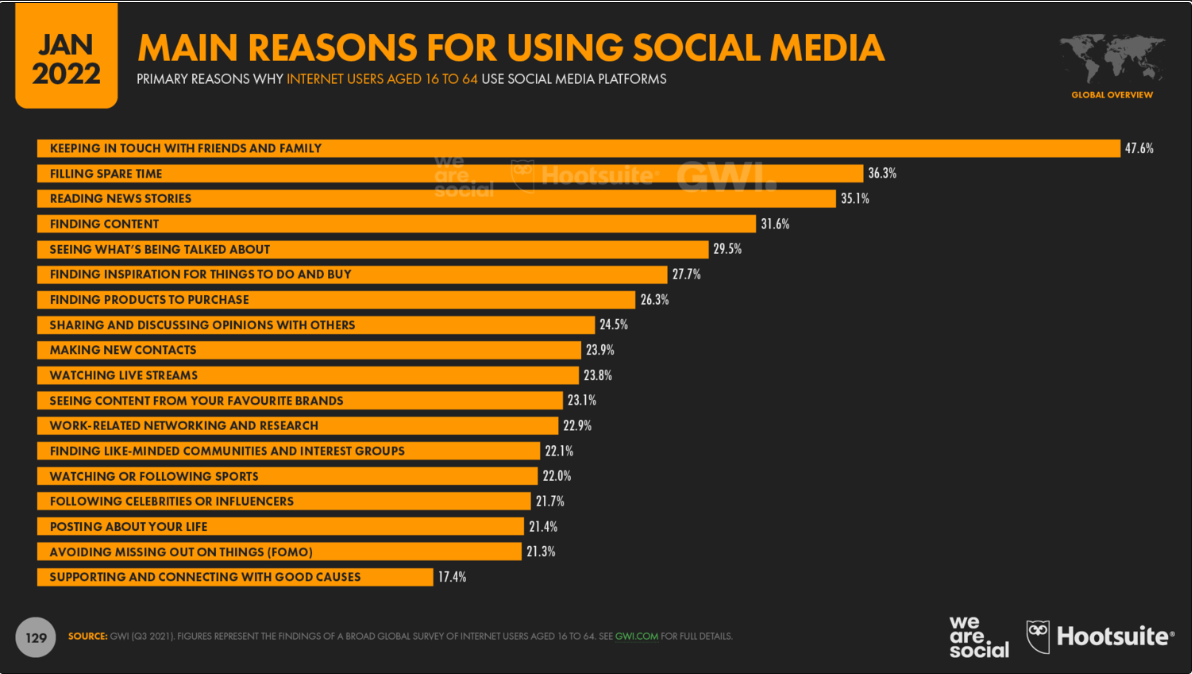
# 2. Raport de analiză al aplicației

Pentru a începe un proces de dezvoltare al unei aplicații este nevoie mai întâi să se întocmească un raport de analiză al acesteia pentru a stabili fezabilitatea acesteia. În cadrul raportului ar trebui să se regăsească nevoia care a dus la necesitatea unei astfel de aplicații, asigurarea existenței unui grup țintă ale cărui nevoi să fie îndeplinite de către aceasta. După ce s-a constatat că există o nevoie în piață pentru aplicația respectivă, va trebui să se analizeze ce există pe piață în prezent, mai precis aplicațiile concurete ce au o temă asemănătoare și funcționalitățile cu care vin acestea. După ce a fost analizată piața, trebuie să se întocmească o listă cu funcționalitățile pe care le oferă aplicația și prioritizându-le în urma unui raport între importanța și dificultatea implementării unei funcționalități.

Astfel, în acest capitol urmează o prezentare a unui raport de analiză pentru aplicația Travel-IT, urmând pașii enumerați anterior.

## 2.1 Aria de interes și nevoile pe care aplicația le satisface

În ziua de azi, a deține un cont pe o platformă de social media nu mai este o noutate pentru majoritatea (peste 50% din populație) oamenilor. Acest tip de aplicație a cunoscut o creștere considerabilă în ultimii ani, fiind unul dintre cele mai accesate tipuri de aplicație.



În urma unei analize a statisticii celor de la Hootsuite [28], s-a constatat că motivele pentru care oamenii folosesc social media s-au diversificat în ultimii ani, datorită evoluției tehnologiei, aplicațiile de social media având tot mai multe funcționalități, precum și obișnuirii oamenilor cu acest mediu virtual. Astfel se poate observa că motivul inițial pe care l-a avut crearea unei astfel de platforme, și anume păstrarea unei legături cu oamenii apropiați din viața noastră, reprezintă doar 47.6% din totalul motivelor, rețelele sociale cuprinzând și funcționalități din aplicații de tip blog, streaming (datorită adăugării funcționalității de videoclipuri live), cumpărături prin existența grupurilor de vânzare-cumpărare, sau chiar de informare prin existența paginilor de știri care își publică articolele prin intermediul acestor rețele.

Aplicația Travel-IT este o aplicația de social media ce își propune să se îndrepte dinspre general spre particular țintind un grup de oameni ce au o pasiune de a descoperi și împărtăși experiențele dobândite pe parcusul călătoriilor lor. Deși la prima vedere ar putea părea că aplicația se adresează unui grup relativ mic de oameni, în realitate aceasta este o platformă ce întrunește principalele arii de interes ale oamenilor care aleg să folosească o platformă de tip social media precum:

- păstrarea legăturii cu prietenii și cunoștiințele, fiind principalul motiv pentru care lumea folosește acest gen de platformă

- relaxarea în timpul liber

- descoperirea de conținut nou și inspirația din acesta pentru noi călătorii

- împărtășirea opiniilor referitoare la experiențe personale cu ceilalți utilizatori ai aplicației

- cunoașterea de oameni noi

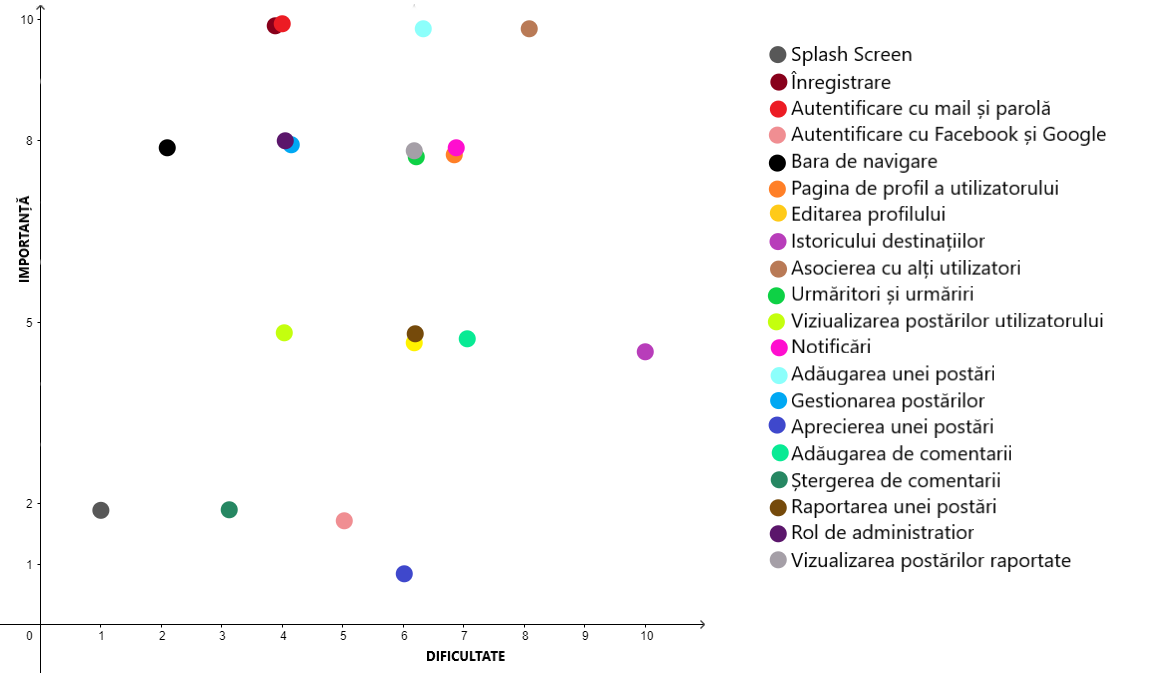
- folosirea platformei pe post de blog de călătorit

## 2.2 Analiza funcționalităților și prioritizarea acestora

În urma stabilirii grupului țintă și al obiectivelor aplicației, stabilind ce nevoi încearcă să satisfacă, au fost stabilite principalele funcționalități ale aplicației. Pentru organizarea procesului de dezvoltare al aplicației, a fost nevoie de o gestionare atentă a cerințelor, acest lucru realizându-se printr-o prioritizare a acestora după un raport de necesitate-complexitate, care să reflecte într-un mod clar fezabilitatea funcționalităților aplicației.

Pentru a întocmi și a viziualiza acest raport, a fost realizat un grafic al funcționalităților ce conține 4 categorii în care se vor încadra funcționalitățile ce vor fi implementate în aplicație, oferind o imagine de ansamblu asupra fezabilității acestora. Aceste categorii sunt: ușoare-valoroase, dificile-valoroase, ușoare-nevaloroase, dificile-nevaloroase.

În urma analizării graficului, am împărțit funcționalitățile după categorii în următorul mod: 1. Ușoare și valoroase:



- Înregistrarea în aplicație

- Autentificarea cu mail și parolă

- Bara de navigare

- Gestionarea postărilor

- Vizulizarea postărilor utilizatorului curent

- Rol de administrator pentru gestionarea raporturilor postărilor de către utilizatori

2. Dificile și valoroase:

- Pagina de profil a utilizatorului (prin crearea unui UI modern)

- Editarea profilului

- Istoricul destinațiilor

- Asocierea cu alți utilizatori

- Urmăritori și urmăriri

- Notificări

- Adăugarera unei postări

- Adăugarea de comentarii

- Raportarea unei postări

- Vizualizarea postărilor raportate si gestionarea acestora

3. Ușoare și nevaloroase:

- Splash Screen

- Ștergerea de comentarii

4. Dificiile și nevaloroase:

- Autentificarea cu Facebook și Google

- Aprecierea unei postări

Este de menționat faptul că aceasta este o listă inițială de repartizare a funcționalitățiilor, ea suferind ușoare modificări pe parcursul dezvoltării, cum ar fi schimbarea priorității anumitor funcționalități, însă lista a fost respectată în esență, reflectând cerințele de dezvoltare necesare.

## 2.3 Aplicații similare

În urma unei analize a pieții [29] am descoperit că există aplicații mai mult sau mai puțin asemănătoare cu Travel-IT, majoritatea aplicațiilor de călătorit fiind mai mult orientate spre partea de planificare a unei călătorii și mai puțin spre interacțiunea dintre utilizatori.

**Foursquare** este o aplicație care se află în piață de mai mulți ani, este orientată mai mult spre partea de planificare a unei călătorii, ce permite utilizatorului să caute atracții în apropierea sa, în funcție de distanța la care se află față de acestea, rating sau chiar după recomandările personalizate oferite de această aplicație.

De asemenea, utilizatorul își poate creea și reține informațiile în profil, salvând locațiile favorite, poate să urmărească alți utilizatori și poate să le citească recenziile.

**Trover** este o aplicație orientată mai mult către experiența călătoriilor utilizatorilor decât pe recenziile sau sfaturile acestora. Aceasta vine cu o pagină pentru cele mai apreciate fotografii, una pentru cele mai noi fotografii, una pentru fotografii din locuri căutate de către utilizator și una pentru căutarea dupa hashtag. Această aplicație este ideală pentru utilizatorii care doresc să știe la ce să se aștepte în momentul în care plănuiesc să plece în călătorie într-o anumită destinație.

**Travello** este o aplicație de tip social media, având un scop asemănător cu Travel-IT, aplicația din această lucrare. Aceasta conține o pagină de noutăți, unde utilizatorul poate să vadă postări despre călătorit de la diferiți useri din jurul lumii, se poate conecta cu ei scriind comentarii la pozele acestora sau trimițându-le un mesaj privat.

Aplicația vine cu o funcționalitate de grupuri pentru diferite tipuri de călători cum ar fi călătorii singuri sau cei care călătoresc într-o anumită parte a globului, astfel prin existența acestor grupuri, utilizatorii pot găsi mai ușor informațiile de care au nevoie, specifice tipului de călătorie pe care ș-il doresc. De asemenea, utilizatorul poate posta propriile poze și contoriza călătoriile.

**TravelBuddy** este o aplicație de tip social media dedicată utilizatorilor care doresc să își găsească un partener de călătorit. Aceasta are o pagină de Home de unde utilizatorul poate vedea postări filtrate în funcție de locația pe care acesta o selectează. Pentru a descoperi potențiali parteneri de călătorit, utilizatorul trebuie să introducă în aplicație destinația, genul partenerului și datele călătoriei. De asemenea, este oferită opțiunea de a vorbi cu localnicii despre detalii legate de destinația aleasă.

În urma analizei aplicațiilor similare, trăgând o concluzie, se poate observa că există un număr considerabil de aplicații în aria de călătorit, însă majoritatea sunt orientate mai mult spre o funcționalitate particulară și mai puțin către conceptul propriu zis de social media . Astfel aplicația Travel-IT înglobează funcționalități care se găsesc în anumite aplicații dintre cele enumerate, cea mai apropiată fiind Travello, însă aplicația Travel-IT este orientată mai mult spre utilizatorii care deja cunosc oamenii pe care vor să îi urmărească, în același timp încurajând desigur și cunoașterea de oameni noi și de eventuali parteneri de călătorit. De asemenea, un alt punct forte al aplicației Travel-IT este dat de opțiunea de a raporta administratorului o postare dacă aceasta este inadecvată, astfel fiind controlate anumite derapaje de la ceea ce își propune aplicația să ofere utilizatorilor săi.

## 2.4 Analiza SWOT

Analiza SWOT [30] ( Strengths - Puncte forte, Weaknesses - Puncte slabe, Oportunites – Oportunități, Threats – Amenințări) are drept scop privirea în ansamblu asupra fezabilității și scalabilității produsului dezvoltat prin punctele tari și prin oportunitățile aduse de către acesta, iar punctele slabe și amenințările au rolul de a veni cu o viziune de ansamblu pentru programator, legată de posibilele viitoare probleme ce pot apărea în cadrul aplicației.

Astfel, după cum se poate vedea din figura de mai jos, analiza SWOT an contribuiut la formarea unei viziuni de ansamblu asupra procesului de dezvoltare și al direcției spre care aplicația aleasă să se îndrepte, luând în calcul atât punctele tari ale acesteia cât și punctele slabe.

|  |  |
| --- | --- |
| **Puncte tari (S)**   * proces de înregistrare simplu și rapid * opțiunea de actualizare a profilului * designul modern al aplicației * asocierea cu alți useri * crearea de conținut * vizulizara locațiilor vizitate * împărtășirea impresiilor prin intermediul comentariilor | **Puncte slabe (W)**   * + lipsa unor recomandări de conținut * lipsa unui chat privat între uilizatorii aplicației |
| **Oportunități (O)**   * crearea unei comunității de oameni pasionați de călătorit * asigurarea unui content de calitate prin opțiunea de raportare a unei postări inadecvate | **Amenințari (T)**   * existența unui număr considerabil de aplicații ce țintesc acest grup țintă de persoane * existența unor grupuri de călătorit pe alte rețele de social media mai populare dar care nu sunt destinate călătoriilor |

# 3. Baza de date

Pentru stocarea datelor din aplicație am folosit o bază de date non-relațională (NoSQL) oferită de către Firebase numită Firebase Realtime Database. Principalul avantaj al acestui tip de bază este dat de către rapiditatea cu care se realizează transmiterea de informații de la Firebase către aplicație și invers, datele fiind structurate sub forma unui fișier JSON.

## 3.1 Structura bazei de date

În cadrul acestui capitol va fi prezentată în detaliu structura propiu-zisă a bazei de date, ajutând la crearea unei viziuni mai clare asupra aplicației. A fost folosită notarea cu \* pentru câmpurile care sunt obligatorii.

**1.** **Users**: în care se salvează informațiile referitoare la utilizatorii aplicației, necesare pentru interacțiunea între aceștia, având în componență următoarele câmpuri:

* uid – fiind un câmp unic generat de către Firebase Autentication pentru fiecare utilizator indiferent de metoda cu care acesta se înregistrează în aplicație;
* email\* - este un câmp unic, înregistrarea în aplicație nefiind posibilă fară utilizarea unui email valid;
* username\* - este un câmp unic, reprezentând numele cu care utilizatorul va fi asociat și cel care va fi vizibil pentru ceilalți utilizatori ai aplicației;
* description – acesta poate fi o scurtă descriere a utilizatorului sau un citat;
* followers – reprezintă lista de utilizatori care urmăresc o anumită persoană;
* following – reprezintă o listă de utilizatori pe care o persoană îi urmărește;
* travelHistory – reprezintă o listă cu istoricul locațiilor pe care un utilizator le-a vizitat și despre care a scris cel puțin o postare;
* userPosts – reprezintă postările utilizatorului;
* notifications – conține notificările utilizatorului salvate sub forma unei liste de string-uri, în momentul în care utilizatorul primește o apreciere ori un comentariu la o postare, sau poate fi o raportare dacă utilizatorul curent este admnistrator;
* admin – conține o valoare ce definește dacă utilizatorul curent este sau nu administrator;

**2. Posts**: în care se vor salva informațiile referitoare la detaliile despre postarea unui utilizator, având următoarele câmpuri:

* postId\* - acesta va fi unic, generat prin concatenarea numelui utilizatorului împreună cu data și ora postării;
* postUser – fiind utilizatorul care a creat postarea
* postDate\* - conține data la care a fost publicată postarea;
* postLocațion\* - conține locația postării;
* postDescription - conține o descriere a postării sub forma unei scurte povestiri a experienței utilizatorului dintr-o călătorie;
* postLikes – conține o listă cu utilizatorii care au apreciat postarea;
* postComments – conține o listă în care se găsesc comentariile de la postare
* postReported – conține o listă cu userii care au raportat postarea

**3. Comments**: în care se vor salva comentariile postărilor, având următoarele câmpuri:

* commentPost: conține postarea căreia îi aparține comentariul
* commentUser: conține utilizatorul care a publicat comentariul
* commentDate: conține data la care a fost postat comentariul
* comment: reprezintă conținutul comentariului propriu-zis

# 4. Arhitectura aplicației

În acest capitol va fi prezentată structura aplicației din punct de vedere arhitectural, descriind cum funcționează aceasta și motivele care au dus la dezvoltarea ei în în acest mod.

## 4.1 Arhitecturi populare utilizate în Android

Aplicațiile mobile din ziua de astăzi au o natură foarte dinamică, astfel devine imposibil pentru un programator să păstreze toată logica aplicației doar în activități sau fragmente. Astfel a apărut nevoia de modularizare a codului și odată cu această nevoie au apărut și șabloanele de proiectare pentru Android care asigură separarea diferitor concepte în fișiere diferite. Cele mai utilizate șabloane de proiectare utilizate în Android sunt următoarele [31]:

**Model-View-Controller (MVC)** este cel mai vechi șablon pentru Android și are ca scop separarea codului în 3 zone:

* Model: aici se vor stoca informațiile, structura obiectelor și se va asigura comunicarea cu baza de date;
* View: reprezintă interfața aplicației în care se vor aduce datele din Model;
* Controller: conține logica principală a aplicației, acesta se ocupă de preluarea acțiunilor pe care utilizatorul le face în aplicație și le trimite mai departe către Model pentru a face eventualele actualizări;

**Model-View-Presenter (MVP)** reprezintă următorul pas în evoluția arhitecturilor din Android, fiind mai nou decât MVC. Folosirea acestuia încă este recomandată pentru programatori. MVP este alcătuit din următoarele componente:

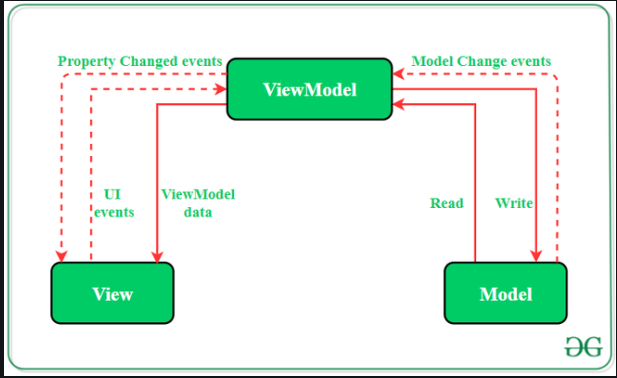
* Model: are același rol ca și la MVC;
* View: se ocupă cu vizualizarea datelor din Model în interfața aplicației și urmărește acțiunile utilizatorului pentru a informa Presenter-ul;
* Presenter: are rol de a obține datele din Model și de a gestiona logica ce se aplică la nivel de interfață pentru a ști ce este nevoie să afișeze. Acesta se ocupă cu gestiunea stărilor în care se află un view și hotărăște pașii ce ar trebui executați în funcție de acțiunile utilizatorului în aplicație;

**Model-View-ViewModel (MVVM)** este arhitectura cea mai recomandată în prezent de către Android. Aceasta cuprinde următoarele componente:

* Model: care este responsabil pentru abstractizarea datelor. Acesta se folosește în combinație cu ViewModel pentru a prelua și a salva datele;
* View: are rolul de a informa ViewModel despre acțiunile pe care utilizatorul le-a efectuat în aplicație;
* ViewModel: se ocupă cu transmiterea datelor relevante pentru View;

## 4.2 De ce MVVM?

Am ales să utilizez această arhitectură deoarece este recomandată de către Google, rezolvând diverse probleme prezente în MVC precum interdependența *layerelor* chiar dacă arhitectura este implementată corect. De asemenea, în MVC nu există un parametru specific pentru a gestiona logica interfeței. Pe langă MVC, și MVP vine cu o problemă și anume faptul că View și Presenter interacționează mult una cu cealaltă, astfel existând componente strâns legate între ele.

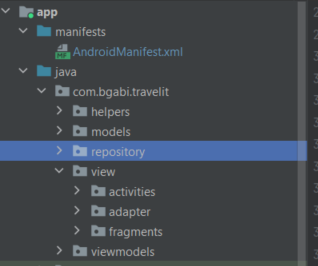


[31]

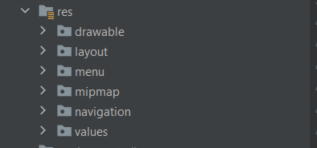
MVVM rezolvă atât problema din MVC cât și pe cea din MVP, separând logica interfeței de partea de backend a aplicației. Prin utilizarea acestui tip de arhitectură, rezolvarea problemelor legate de ciclul de viață al componentelor devine mai ușoară cu ajutorul ViewModel care împreună cu LiveData ajută la rezolvarea excepției *NullPointerExcepton* prin actualizarea view-urilor distruse. ViewModel este de asemnea o componentă de bază ce ajută la separarea activităților sau fragmentelor de logica de business, acesta ocupându-se cu obținerea de date [32].

## 4.3 MVVM în Travel-IT și navigarea în aplicație

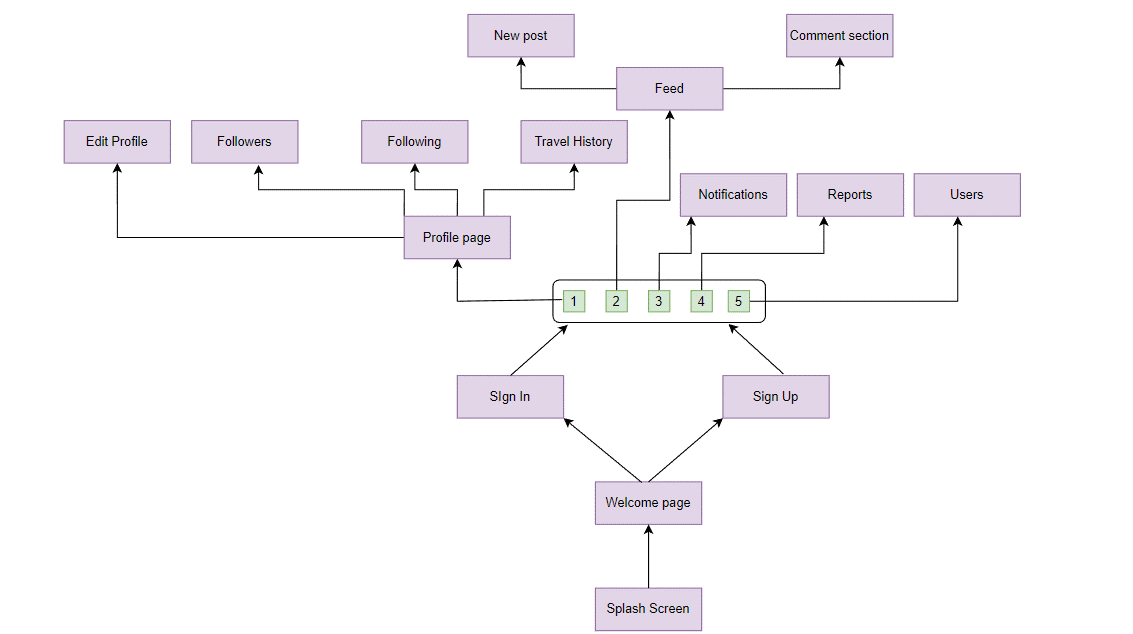
Pentru fiecare model din cadrul aplicației, am creat câte un ViewModel în care am folosit LiveData, având rolul de observator pentru starea curentă a componentelor, iar în funcție de aceasta, le va actualiza dacă este necesar.



Pentru afișarea listelor din cadrul aplicației (de postări, useri, comentarii), am folosit *recyclerview,* iar pentru fiecare am implementat câte un adaptor care a primit un obiect de view model al unei clase + poziția elementului.



De asemenea, pentru fiecare model am creat o clasă de repository unde am încărcat datele. Pentru funcția ce aduce datele au fost folosite corutine, pentru a asigura încarcarea datelor înainte de a se încărca pagina în interfața aplicației. În cadrul view model am apelat funcția definită în repository, apoi am apelat în adaptor *notifydatasetchanged()* pentru a actualiza conținutul din *recyclerview*.



Figura

# 5. Funcționalitățile aplicațiile

În cadrul acestui capitol vor fi prezentate funcționalitățile aplicației, paginile principale, procesul de dezvoltare al acestora din punct de vedere tehnic dar și imagini cu ecranele aplicației unde sunt prezente aceste funcționalități.

## 5.1 Splash Screen

Splash Screen este un ecran ce conține în general o animație sau o imagine reprezentativă pentru aplicație. Acesta este primul ecran al aplicației și are ca scop crearea unei experiențe plăcute a utilizatorului, Splash Screen-ul fiind prezent pentru câteva momente, cât timp aplicația își încarcă resursele necesare pentru funcționare, astfel utilizatorul nu mai este nevoit să vadă cum se încarca pagina și apar datele pe rând (cum se întâmplă si cu anumite site-uri web).

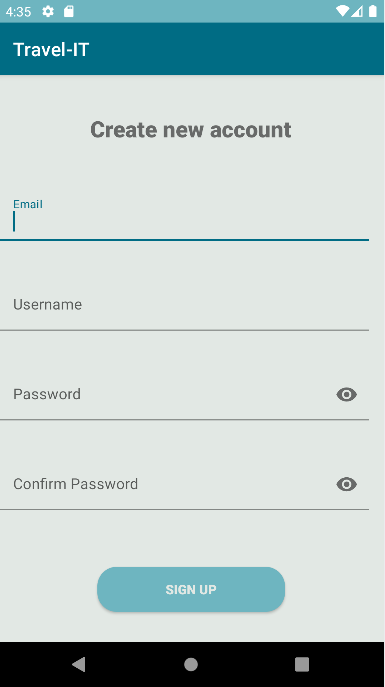


Figura 5.1 -Splash screen

Pentru implementarea acestei funcționalități, a fost creată o temă în care a fost moștenită proprietatea *“Theme.MaterialComponents.Light.NoActionBar”* pentru a ascunde *toolbar-ul* și a face bara de stare transparentă pentru a evidenția *splash screen-ul* ce conține doar o imagine care a fost setată ca și fundal*.* Această temă va fi activată în momentul în care aplicația va fi lansată, iar Splash Screen-ul va fi prezent până când se va inițializa *MainActivity*, care va avea asociată altă temă. Această tranziție de teme se va efectua prni apelul funcției *setTheme()* care primește ca parametru noua temă pentru aplicație.

## 5.2 Înregistrarea în aplicație

Înregistrarea în aplicație se face din ecranul principal al aplicației, formularul de înregistrare apare atunci când un utilizator apasă pe butonul *Sign Up*.



5.3 Formular de înregistrare

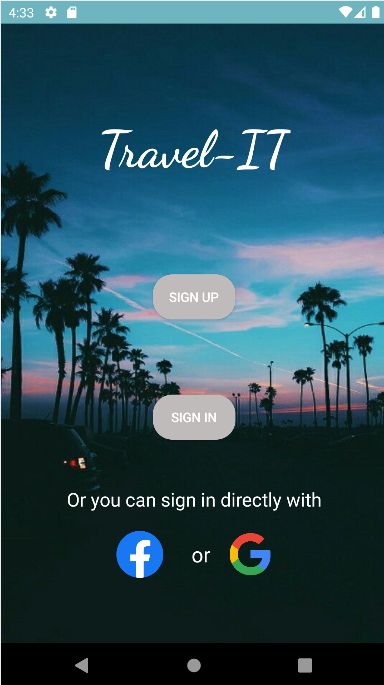


Figura 5.2 Ecranul principal

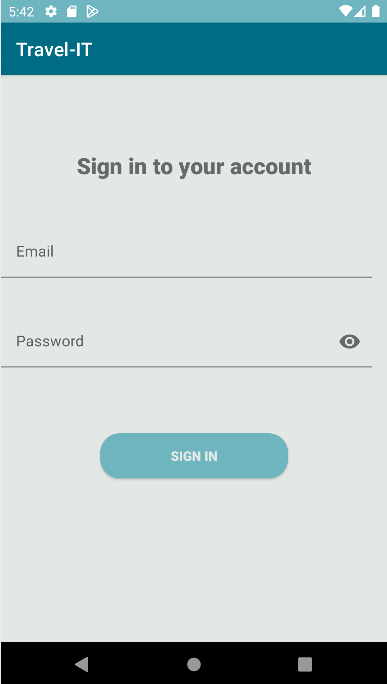
După ce a dispărut *splash screen-ul* și utilizatorul a apăsat pe butonul *Sing Up*, se va deschide un formular de înregistrare în aplicație în care va trebui să își introducă un email valid, care nu există deja, un nume de utilizator cu care va fi asociat în aplicație, o parolă de cel puțin 8 caractere care trebuie să conțină cel puțin o literă mare, una mică, o cifră și un semn de punctuație. Utilizatorul va trebui să reintroducă parola pentru a valida corectitudinea acesteia și a evita situații neplăcute precum pierderea accesului la noul cont creat. În cazul în care un utilizator introduce o adresă de email invalidă sau una deja existentă, o parolă care nu respectă criteriile descrise, sau o parolă care nu coincide în cele două câmpuri, se va afișa un mesaj pentru câmpul care este invalid, utilizatorul fiind nevoit să îl modifice pentru a finaliza cu succes procesul de înregistrare.

Pentru procesul de înregistrare am utilizat Firebase Authentication pentru gestiunea utilizatorilor, iar pentru stocarea datelor necesare în aplicație a fost folosit Firebase Realtime Database.

După ce utilizatorul se înregistrează cu succes, acesta va fi redirecționat către pagina sa de profil.

## 5.3 Logarea în aplicație

Pentru a se autentifica cu succes în aplicație, utilizatorul trebuie să completeze un formular ce conține adresa de email cu care acesta s-a înregistrat, împreună cu parola aferentă contului, iar în caz că utilizatorul va introduce un mail care nu există sau o parolă greșită, acesta va primi un mesaj de atenționare.

Figura 5.4 Formular de logare

Pe lângă autentificarea clasică cu email și parolă, acesta are posibilitatea să se înregistreze direct în aplicație dacă deține un cont de Facebook sau de Gmail (vezi figura 5.2) . Dacă un utilizator alege una din cele două opțiuni pentru prima dată, acestuia i se va crea automat un cont, urmând ca la următoarea autentificare în aplicație să se logheze direct în aplicație printr-un simplu click pe metod aleasă. Această funcționalitate a fost realizată tot prin intermediul Firebase Authentication, acesta punând la dispoziția utilizatorului posibilitatea de a crea un proces de autentificare folosind aplicații externe.

## 5.4 Bara de navigare

După ce se va autentifica în aplicație, utilizatorul va fi redirecționat automat către pagina de profil dacă este un utilizator normal sau către pagina de rapoarte dacă este administrator.



Figura 5.6 Bara de navigare administrator



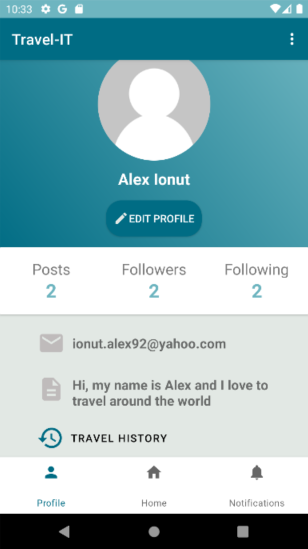
Figura 5.5 Bara de navigare user

Pentru ca navigarea prin aplicație să fie una ușoară și intuitivă, a fost folosit **Bottom Navigation** din cadrul MDC Android, fiind ideală mai ales pentru telefoanele care au un ecran mare.

Bara de navigare este afișată în partea de jos a ecranului și este inclusă în *HomeActivity,* fiind singura activitate folosită după ce utilizatorul se loghează în aplicație, fiecare tab din cadrul barei de navigare fiind implementat în interior unui fragment și al unui *layout* corespunzător.

## 5.5 Pagina de profil

Este pagina pe care este redirecționat utilizatorul după ce se autentifică în aplicație.

Figura 5.7 Pagina de profil

În cadrul acestei pagini se află poza de profil a utilizatorului, care după prima logare în aplicație arată ca cea din imagine, fiind o fotografie default pentru toți utilizatorii noi. Aceasta poate fi schimbată din pagina de editare a profilului, iar noua fotografie va fi stocată în Cloud Storage.

În pagina de profil se află, de asemenea, următoarele informații: username-ul utilizatorului, o secțiune în care sunt prezentate numărul de postări, urmăritori și urmăriri pe care utilizatorul le are. Fiecare din cele 3 acționează ca niște butoane, care în momentul în care sunt apăsate vor redirecționa utilizatorul spre pagina corespondentă. Tot în cadrul profilului apar și email-ul utilizatorului, descrierea și un buton de istoric al călătoriilor care redirecționează utilizatorul către o hartă cu locurile vizitate din postări.

Pe parcursul dezvoltării acestei pagini am întâmpinat diverse dificultăți. În ceea ce privește interfața, construirea design-ului a fost o provocare deoarece aceasta a constat în compunerea mai multor *ViewGroup-uri.* Pe de altă parte, pentru funcționalitățile de backend al aplicației, obținerea informațiilor necesare din baza de date a fost un proces complex, din cauza faptului că a fost o neconcordanță între finalizarea acestui proces și *flow-ul* normal al aplicației, astfel a fost necesară folosirea corutinelor pentru așteptarea preluării datelor din Firebase.

## 5.6 Editarea profilului

În această pagină utilizatorul își poate edita profilul, acesta poate să își actualizeze fotografia de profil prin apăsarea iconiței de lângă fotografia curentă (vezi Figura 5.8). În momentul în care acesta apasă pe iconiță, se va deschide automat galeria de poze a telefonului de unde va alege o nouă poză, aceasta actualizându-se în Cloud Storage.

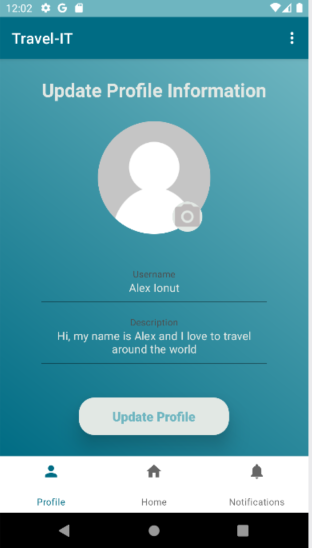


Figura 5.8 Editarea profilului

Pe lângă fotografia de profil, utilizatorul își poate actualiza și username-ul, cu condiția ca acesta să nu fie folosit deja de către un alt utilizator dar și descriera.

Aceste schimbări vor fi salvate în momentul în care utilizatorul apasă pe butonul de actualizare a profilului, acesta fiind redirecționat către pagina sa de profil unde vor fi afișate noile informații conform schimbărilor efectuate de utilizator.

## 5.7 Urmărirea altor utiliaztori

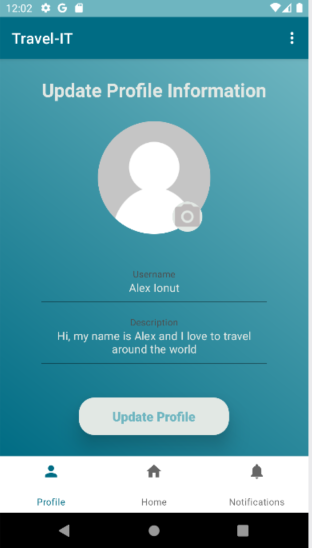


Figura 5.9 Câutare utilizatori

În cadrul paginii *Home* a aplicației, se regăsesc postările oamenilor pe care utilizatorul îi urmărește. De asemenea, sub bara de acțiuni se află un buton care, la apăsare, afișează o bară de căutare. Când aceasta este activă, lista de postări este ascunsă și se afișează o listă cu utilizatorii aplicației bazată pe *input-ul* introdus și filtrată după username pe măsură ce utilizatorul tastează în bara de căutare. Utilizatorii cu care un user nu este asociat, vor avea în partea dreaptă un buton de urmărire, care în momentul când este apăsat va dispărea. Acest eveniment va cauza creșterea numărului de urmăritori ai utilizatorului curent, precum și numărul de urmăriri ale utilizatorului care a fost urmărit. În momentul în care un utilizator îl urmărește pe cineva, acesta va putea să îi vizualizeze atât profilul cât și postările cu care poate interacționa în diferite moduri.

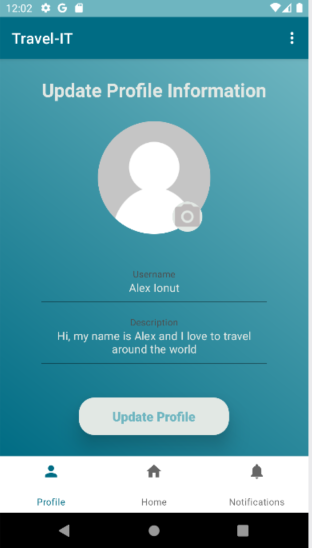


Figura 5.12 Profil utilizator

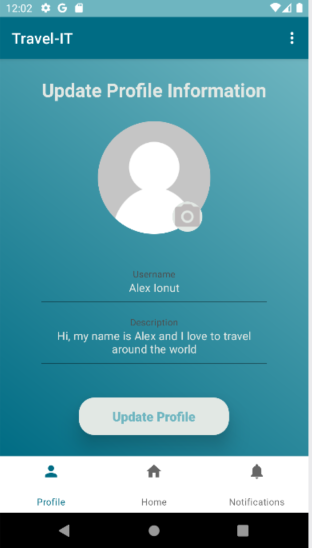


Figura 5.11 Pagină urmăritori

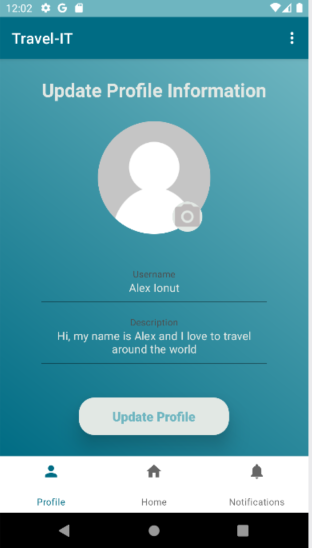


Figura 5.10 Pagină urmăriri

După cum se poate observa în figura 5.7, în cadrul paginii *Profile* un utilizator are o secțiune unde sunt afișate numărul de postări, de urmăriri și de urmăritori pe care acesta îi are în aplicație. Fiecare din cele 3 acționează ca niște butoane. Cel de *Posts* va redirecționa utilizatorul spre o pagină cu propriile postări (figura 5.10) de unde va avea opțiunea de ștergere. În pagina *Followers* (figura 5.11), se va găsi o listă cu oamenii care urmăresc și văd activitatea utilizatorului curent, acesta având dreptul de a șterge pe cineva dacâ nu mai vrea ca o persoană să îi mai vadă postările. În pagina *Following*  (figura 5.12), se află o listă cu oamenii pe care utilizatorul curent îi urmărește. Acesta are opțiunea de a nu îi mai urmări prin apăsarea butonului *Unfollow*, lista de utilizatori actualizându-se în timp real.

## 5.8 Postările și comentariile

Acestea reprezintă nucleul aplicației fiind modalitatea prin care are loc transferul de informații între utilizatori.

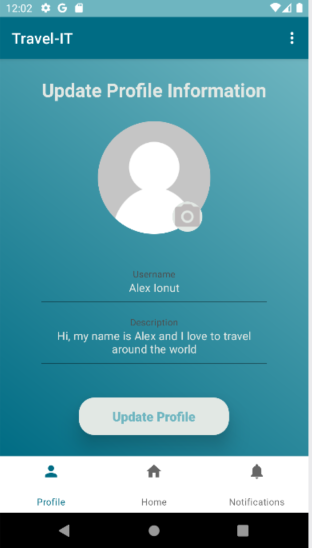


Figura 5.13 Pagina Home

În cadrul paginii *Home* se află postările oamenilor pe care un utilizator îi urmărește, fiind ordonate după data în care acestea au fost publicate. Un utilizator poate să interacționeze cu o postare în 3 moduri: prin aprecierea acesteia, prin lăsarea unui comentariu sau prin raportarea acesteia în cazul în care utilizatorul consideră că aceasta este inadecvată.

Sub bara de acțiuni (figura 5.13) se află un buton de adăugare a unei postări noi, care va redirecționa utilizatorul spre ecranul din figura 5.14.

Pentru a adăuga o postare nouă, utilizatorul va trebui să adauge o descriere sugestivă, o fotografie și o locație pentru aceasta, data fiind inițializată automat cu ora la care se efectuează adăugarea postării.

Pentru salvarea pozelor a fost nevoie de generarea unui identificator unic pentru a putea gestiona pozele de la postări din cadrul Firebase Storage, deoarece spre deosebire de fotografiile de profil ale utilizatorilor, unde pentru fiecare poză a unui utilizatror aveam un uid generat de Firebase Authenticator asociat pentru fiecare, aici am folosit un identificator care a avut o formulă de tipul *username+data și ora* postării, unde fiecare username este unic asociat unui *user*. După ce va apăsa butonul de *Add New Post,* utilizatorul va fi redirecționat către profilul său de unde își va putea vizualiza sau șterge postarea accesând secțiunea de *Posts,* ea fiind vizibilă în timp real și pentru ceilalți oameni care urmăresc utilizatorul și fac o reîmprospătare a paginii de *Home*.

Fiecare postare conține o pagină de comentarii (figura 5.15), fiecare utilizator având opțiunea să adauge un comentariu sau să șteargă unul deja adăugat de el. De asemenea, autorul posării are posibilitatea să șteargă orice comentariu de la postarea respectivă.

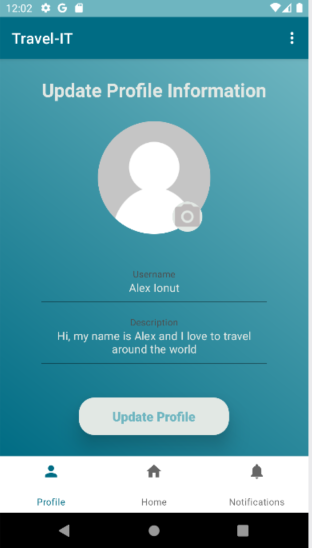


Figura 5.15 Pagina de comentarii

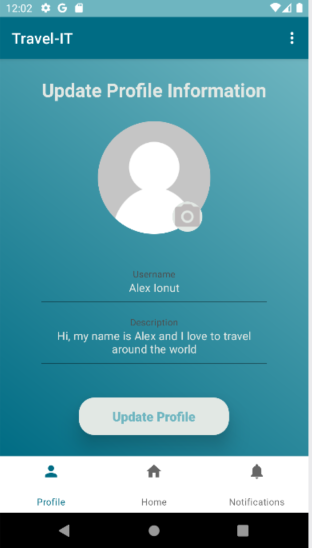


Figura 5.16 Istoric călătorii

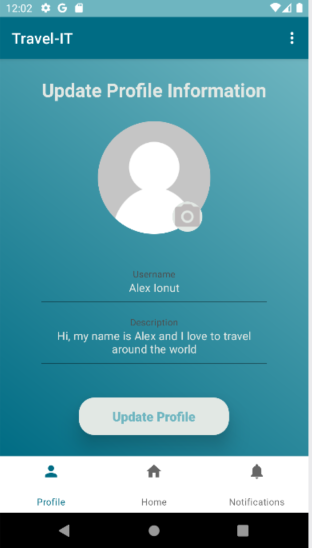


Figura 5.14 Adăugare postare nouă

## 5.9 Istoricul călătoriilor

În cadrul paginii de profil (figura 5.7), se alfă un buton *Travel History*, care deschide o nouă pagină (figura 5.15) în care utilizatorul iși poate vedea locațiile pe care acesta le-a vizitat prin intermediul unei hărți, unde locațiile vizitate vor fi preluate în mod automat din cadrul postărilor acestuia, fiacare dintre acestea fiind marcate cu câte un pin.

## 5.10 Notificările

Această funcționalitate a fost dezvoltată cu scopul înștințării utilizatorilor în momentul în care au loc acțiuni ce au un impact asupra contului lor.

Pagina de notificări se găsește in bara de navigare a oricărui utilizator care nu este administrator. Aceasta (figura 5.16) conține o listă în care utilizatorul poate primi sau șterge următoarele tipuri de notificări:

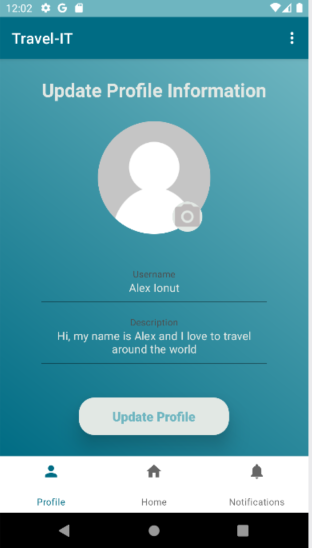


Figura 5.16 Pagina de notificări

* Când primește o apreciere la o postare care, la click, aceasta îl va redirecționa către acea postare.
* Când primește un comentariu la o postare care, la click, aceasta îl va redirecționa către acea postare.
* Când un alt utilizator a început să îi urmărească profilul.

Pe lângă notificările care sunt trimise în aplicație, mai există și cele folosite din Firebase Cloud Messaging ffind personalizate în cadrul unei campanii, care se trimit în momentul în care un utilizator nu a mai intrat în aplicație de o perioadă mai lungă de 3 zile.

## 5.11 Rolul de administrator

Pentru asigurarea unui conținut de calitate al aplicației Travel-IT, a fost introdus rolul de administrator. Acesta se autentifică în aplicație ca și un utilizator normal, iar în bara de navigare va avea doar două opțiuni: *Reports* și *Users* (figurile 5.17 și 5.18),care nu sunt disponibile pentru un utilizator normal.

În momentul în care un utilizator decide să semnaleze o postare ca fiind inadecvată, aceasta va ajunge în pagina de rapoarte unde se află o listă cu toate postările la care s-au semnalat nereguli. Un raport conține numele utilizatorului care a semnalat postarea și doua butoane: de acceptare sau ștergere al raportării. Administratorul poate să vizualizeze postarea printr-un click pe raportul respectiv. În cazul în care administratorul acceptă raportarea, atunci postarea va fi ștearsă. Dacă postările unui utilizator sunt repetate frecvent, administratorul poate să acceseze pagina *Users* unde sunt afișați toți utilizatorii aplicației care nu sunt admini și poate șterge un utilizator, iar odată cu ștergerea acestuia se vor șterge toate postările, comentariile sau urmăririle cu care acesta este asociat.

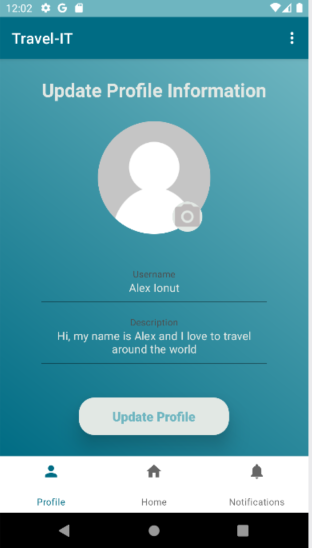


Figura 5.17 Pagina de rapoarte

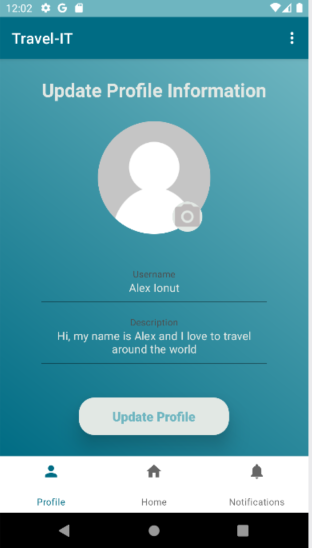


Figura 5.18 Users

# Concluzii

Odată apărut, conceptul de social media va rămâne prezent în viața oamenilor, aceștia interconectându-se tot mai mult odată cu avansul tehnologic. În ceea ce privește călătoritul, acesta se află pe un trend ascendent, fiind o activitate mult mai accesibilă ca acum câteva zeci de ani. Odată cu creșterea interesului față de această activitate, au apărut tot mai multe bloguri și aplicații în care oamenii își pot împărtăși aceaste experiențe cu cei din jur, astfel crescând într-un mod direct proporțional și calitatea călătoriilor prin învățarea din experințele oamenilor.

Aplicația Travel-IT este o aplicație modernă care poate fi utilizată de oricine are o pasiune pentru călătorit și care dorește să își împărtășească această experiență cu cei din jur sau vrea să afle experiențele altora. Aceasta reprezintă un mediu sigur, conținutul neadecvat fiind îndepărtat de către administratori, păstrându-se o atmosferă plăcută în cadrul aplicației.

Scopul aplicației poate fi considerat ca și îndeplinit, aplicația fiind una completă datorită funcționalităților integrate datorită cărora utilizatorului final beneficiază de un *flow* complet în cadrul acesteia. De asemenea, din punct de vedere tehnologic, aplicația Travel-IT este una de actualitate, folosind limbaje recent apărute precum Koltlin sau o arhitectura modernă precum MVVM, acestea fiind recomandate de către majoritatea dezvoltatorilor.

Deși aplicația își atinge scopul inițial, acesteia îi pot fi aduse numeroase îmbunătățirii la nivelul funcționalităților, cum ar fi introducerea unui chat între utilizatori pentru a facilita comunicarea între aceștia care în prezent se face în cadrul secțiunii de comentarii, sau prin contactarea pe email. Un alt beneficiu considerabil ce ar putea fi adus aplicației ar fi introducerea funcționalității de grup, astfel creându-se grupuri speciale pentru călătorii singuri sau pentru o anumită destinație. De asemenea, o funcționalitate benefică aplicației ar fi introducerea unor recomandări pe baza cărora utilizatorii aplicației să primească sugestii de postări bazate pe istoricul lor de călătorit sau pe activitățile utilizatorilor pe care acesta îi urmărește.

Nu un ultimul rând, un avantaj al aplicației Travel-IT este dat de posibilitatea de a se personaliza destul de ușor și pentru alte domenii sau pasiuni ale oamenilor.

În concluzie, aplicația Travel-IT incorporează o gamă variată de funcționalități, având un scop bine definit ce întrunește cerințele necesar pentru o aplicație mobile de succes.

**Bibliografie**

[1] Alexandra Samur (22 Noiembrie 2018) „The history of social media: 29+ Key Moments” <https://blog.hootsuite.com/history-social-media/> Accesat la 1 August 2022 15:30

[2] Simon Kemp (15 Iulie 2022) „Facebook statistics and trends” <https://datareportal.com/essential-facebook-stats> Accesat la 1 August 2022 16:37

[3] Luis Gillman „Is Kotlin easy to learn?” <https://www.compsuccess.com/is-kotlin-easy-to-learn/> Accesat la 2 August 2022 12:21

[4] J Steven Perry (7 Aprilie 2019) „Learn Kotlin, Unit 3: Object-and-function-oriented programming concepts and principles” <https://developer.ibm.com/tutorials/learn-kotlin-3/> Accesat la 2 August 2022 12:57

[5] Jet Brains (8 Iulie 2021) „Null Safety” <https://kotlinlang.org/docs/null-safety.html> Accesat la 2 August 2022 14:05

[6] <https://developer.android.com/kotlin/coroutines> Accesat la 2 August 2022 14:25

[7] Anshul Bashal (9 Noiembrie 2021) „Java vs Kotlin” <https://www.baeldung.com/kotlin/java-vs-kotlin> Accesat la 2 August 2022 15:03

[8] <https://www.learnhowtoprogram.com/android/introduction-to-android/introduction-to-xml-and-android-layouts> Accesat la 2 August 2022 15:50

[9] <https://www.json.org/json-en.html> Accesat la 2 August 18:25

[10] <https://en.wikipedia.org/wiki/Android_(operating_system)#History> Accesat la 2 August 18:50

[11] <https://developer.android.com/studio/intro> Accesat la 3 August 10:45

[12] <https://developer.android.com/studio/intro#project_structure> Accesat la 3 August 11:34

[13] <https://developer.android.com/guide/topics/manifest/manifest-intro> Accesat la 3 August 11:55

[14] https://www.geeksforgeeks.org/android-project-folder-structure Accesat la 3 August 14:20

[15] <https://developer.android.com/jetpack> Accesat la 3 August 15:02

[16] <https://developer.android.com/jetpack/androidx> Accesat la 3 August 15:08

[17] Linda Rosencrance „Google Firebase” <https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase> Accesat la 3 August 15:45

[18] SF APPWORKS (12 Ianuarie 2021) „Comparing Firebase vs. AWS | SF APPWORKS” [https://www.sfappworks.com/blogs/firebase-vs-aws-comparision Accesat la 15:55](https://www.sfappworks.com/blogs/firebase-vs-aws-comparision%20Accesat%20la%2015:55) Accesat la 3 August 15:58

[19] <https://firebase.google.com/docs> Accesat la 4 August la 11:15

[20] <https://firebase.google.com/products/auth> Accesat la 4 August la 11:53

[21] <https://firebaseopensource.com/projects/firebase/scrypt/> Accesat la 4 August la 12:40

[22] <https://firebase.google.com/docs/database> Accesat la 4 August 15:05

[23] <https://firebase.google.com/docs/storage> Accesat la 4 August 15:32

[24] Antony Gitau (26 Octombrie 2021) „How to implement the Android LifeCycle Callback Methods” <https://www.section.io/engineering-education/understanding-and-implementing-the-android-lifecycle/> Accesat la 5 August 10:30

[25] <https://developer.android.com/topic/libraries/architecture/livedata> Accesat la 5 August 11:17

[26] <https://github.com/bumptech/glide> Accesat la 5 August 17:44

[27] Suleiman (31 Decembrie 2019) ) „Mega guide to Android’s material design components” <https://blog.iamsuleiman.com/mega-guide-android-material-design-components/> Accesat la 5 August 18:34

[28] Simon Kemp (26 Ianuarie 2022) „Digital 2022: Social media referrals” <https://datareportal.com/reports/digital-2022-social-media-referrals?rq=social%20media> Accesat la 6 August 12:45

[29] Emma Roth (2 Martie 2020) „The 7 best social media apps for travelers” <https://www.makeuseof.com/tag/6-best-social-media-apps-travelers/> Accesat la 6 August 14:43

[30] Natalia Zhukova (18 Octombrie 2021) „How to do a SWOT analysis (with examples)” <https://www.semrush.com/blog/swot-analysis-examples> Accesat la 6 August 19:54

[31] Rishu Mishra (24 Iunie 2021) „Android Arhitecture Patterns” <https://www.geeksforgeeks.org/android-architecture-patterns/> Accesat la 8 August 16:50

[32] DhiWise (18 Iunie 2021) „Google recommendation for Android application development arhitecture” Accesat la 8 August 18:09