

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” IAȘI**  
**FACULTATEA DE INFORMATICĂ**



**LUCRARE DE LICENȚĂ**

**ServiceReq**

propusă de  
**Iftimie Gabriel Codruț**

**Sesiunea: februarie, 2020**

Coordonator științific  
**Prof. Colab. Olariu Florin**

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” IAȘI**  
**FACULTATEA DE INFORMATICĂ**

**LUCRARE DE LICENȚĂ**

**ServiceReq**

propusă de  
**Iftimie Gabriel Codruț**

**Sesiunea: februarie, 2020**

Coordonator științific  
**Prof. Colab. Olariu Florin**

## DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul “**ServiceReq**”, codul sursă al programelor și celelalte conținuturi care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică. De asemenea, sunt de acord ca Facultatea de Informatică din cadrul Universității “Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, 06.02.2020

Absolvent, Iftimie Gabriel Codruț

---

# Cuprins

<b>Introducere</b>	<b>4</b>
Motivație	4
Obiective generale	4
Scurtă descriere a soluției	5
<b>Abordare tehnică</b>	<b>5</b>
Instrumente software	5
Limbaje și medii de programare folosite	6
<b>Contribuții</b>	<b>7</b>
<b>1 Descrierea problemei</b>	<b>9</b>
<b>2 Abordări anterioare</b>	<b>10</b>
<b>3 Descrierea soluției</b>	<b>11</b>
3.1 Principalele funcționalități	11
<b>4 Detalii de implementare</b>	<b>19</b>
4.1 Aplicația server	19
4.2 Aplicația client	21
4.3 Diagrame	23
4.3.1 Diagrame use-case	23
4.3.2 Diagrame arhitecturale	24
<b>Concluziile lucrării</b>	<b>25</b>
<b>Bibliografie</b>	<b>27</b>

## **Introducere**

### **Motivație**

Cu toții am dat de momentul în care s-a defectat un lucru în casă sau avem nevoie de un serviciu și nu știm la cine să apelăm. Apelezi la un prieten sau altul în speranța că îți poate recomanda pe cineva, însă nici ei nu te pot ajuta.

Următorul lucru pe care te gândești să-l faci este să intri pe rețeaua de socializare Facebook, te înscrii într-un grup și întrebi cine te-ar putea ajuta cu problema ta. Toată lumea are păreri diferite și din nou, nu știi ce să faci. În plus tot procesul de înscriere într-un grup pe Facebook poate dura destul de mult, în funcție de rangul persoanei căreia îi este permisă acceptarea noilor membri în grup, din cauza disponibilității persoanei respective. O altă problemă o constituie lipsa unui astfel de grup.

Dintr-un alt punct de vedere sunt persoane care activează recent într-un domeniu ce constituie oferirea unui serviciu (ex. instalator, electrician, mecanic, etc) și din această cauză nu există un grup larg de persoane care pot recomanda persoana respectivă.

Datorită problemelor și motivelor enumerate mai sus, am decis să creez o aplicație Android în care utilizatorii solicită și răspund solicitărilor de servicii.

### **Obiective generale**

Un prim obiectiv este ca persoanele care utilizează această aplicație să fie diferențiate între ele prin modul în care acestea o folosesc. Diferențierea se face prin opțiunea “Helper” care se face prin editarea profilului creat. Prin această opțiune, persoanele care au selectat-o au o gamă de categorii ce constituie domenii în care pot ajuta alți utilizatori.

Utilizatorii care au selectat opțiunea “Helper” vor primi notificări atunci când alt utilizator solicită un serviciu dintr-o categorie pe care au selectat-o și care se află în zona pe care au ales-o.

Un alt obiectiv este capabilitatea interschimbării de mesaje între utilizatorii care solicită și oferă serviciile. Această opțiune este una obligatorie pentru o aplicație de acest gen pentru că odată ce ai văzut o persoană căreia îi poți oferi un serviciu, fluidizezi interacțiunea cu aceasta. Prin această funcție, după ce ai trimis un mesaj, acel mesaj rămâne salvat iar în cazul în care dorești să mai trimiți un mesaj persoanei respective, având persoana căreia i-ai

trimis mesajul salvată, găsirea persoanei respective este mult mai rapidă și nu trebuie căutată din nou în rândul postărilor.

O interfață intuitivă și ușor de înțeles este din nou o prioritate pentru ca aplicația să aibă succes prin faptul că persoanele ar alege mereu aplicația cu interfața mai intuitivă și care arată mai bine dacă ar fi de ales între două aplicații cu aceeași funcționalitate. Din acest motiv am ales să adaptez interfața pentru a se asemana cu una din aplicațiile care sunt cel utilizate la noi în țară, și anume Facebook. Acest obiectiv a putut fi posibil pentru că funcționalitățile se aseamănă însă scopul final este diferit.

## Scurtă descriere a soluției

Aplicația este una pentru platforma mobilă Android ce se conectează la un server și are următoarele componente:

1. **Aplicația client:** prin intermediul acesteia utilizatorii interacționează cu aplicația server printr-un protocol TCP. Ei fac solicitări și primesc răspunsuri pe care aplicația client le interpretează și afișează un răspuns corespunzător.
2. **Aplicația server:** acesta este punctul de unire dintre aplicația client și baza de date. Primește informațiile de la aplicația client, procesează informațiile respective și apoi actualizează informațiile din baza de date, trimițând mesaj înapoi la client cu datele de care are nevoie în urma actualizării bazei de date, dacă este cazul.
3. **Baza de date:** reprezintă mediul de stocare a datelor al aplicației, constituit dintr-o bază de date relațională.

## Abordare tehnică

### Instrumente software

**Android Studio:** este IDE-ul oficial, oferit de Google, bazat pe IDE-ul IntelliJ oferit de JetBrains, iar din această cauză este și foarte asemănător cu acesta. Este esențial să folosești acest IDE pentru dezvoltarea unei aplicații pentru platforma Android din mai multe motive. Unul dintre ele l-am enumerat mai sus și anume că este oferit de Google, dezvoltatorii platformei Android, și poți fi sigur că există sprijin suficient din partea lor cât și din partea

comunității. Un alt motiv este suportul nativ pentru limbajul Kotlin, o alternativă a limbajului Java, însă motivul care m-a făcut să aleg Android Studio este similaritatea pe care o are cu IDE-ul IntelliJ, un IDE pe care îl folosesc de mult timp.

**Visual Studio:** acest IDE este dezvoltat de Microsoft și oferă suport pentru crearea a diferitor tipuri de aplicații, de la aplicații mobile până la aplicații desktop și aplicații web. Am folosit acest IDE pentru crearea aplicației server de care are nevoie aplicația client pentru răspunsuri și actualizări asupra bazei de date. Pe de altă parte sugestiile de formatare a codului și sugestiile pe care le oferă sunt de mare ajutor pentru a scrie o aplicație rapid și ușor de înțeles.

**Microsoft SQL Server Management Studio:** este o aplicație care este foarte de folos în gestionarea unei baze de date relaționale, în cazul meu fiind SQL Server Developer, deoarece ajută la crearea rapidă a tabelelor, cât și modificarea acestora incluzând inserarea și modificarea datelor dintr-o tabelă.

## **Limbaje și medii de programare folosite**

**SqlClient:** reprezintă un pachet NuGet realizat de Microsoft pentru .NET Core care asigură o conexiune stabilă cu baza de date și creat special pentru baza de date Sql Server, astfel lucrul cu acesta este foarte ușor.

**Android Emulator:** după cum descrie și numele, aceasta este o aplicație integrată în Android Studio ce simulează un telefon Android pentru a ușura procesul de dezvoltare a unei aplicații de acest tip. Neavând nevoia de a conecta un telefon la mediul de lucru, sau nedeținând un telefon care are ca sistem de operare Android, această cale îți oferă calea de a testa schimbările aplicației tale. Acest emulator poate fi văzut în *Figura 1* de mai jos.

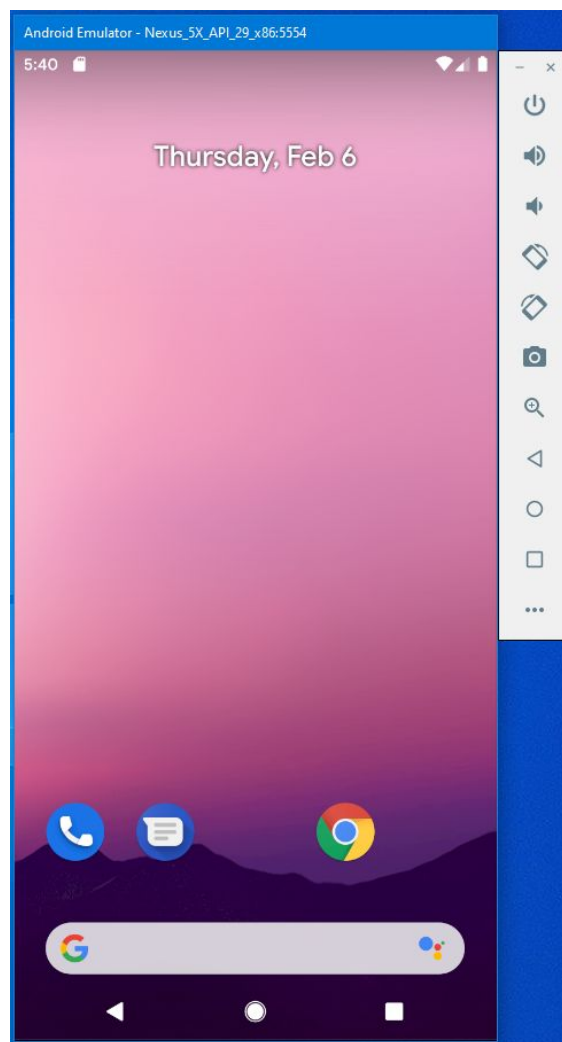


Figura 1: Emulatorul Android

Limbajele de programare folosite sunt următoarele: Java, C#, XML, SQL

## Contribuții

Deoarece am un părinte care are ca job secundar profesia de instalator, mi-am oferit ajutorul de fiecare dată când a fost nevoie și am ajuns să înțeleg în mare parte cum stau lucrurile în acest mediu. Zeci de telefoane în fiecare zi, uneori chiar deranjante și fiindcă toată lumea prefera una din cele mai cunoscute persoane din domeniu. De ce? Pentru că nimeni nu cunoaște pe altcineva, iar atunci când li se oferă pe altcineva pentru că persoana respectivă nu are timpul necesar pentru rezolvarea problemei acestea preferă să aștepte și chiar două săptămâni, dacă problema nu e urgentă.



Astfel, gândindu-mă, am realizat că dacă ești o persoană nouă într-un anumit domeniu, șansele ca oamenii să afle cât mai repede de aptitudinile tale sunt destul de mici, sau dacă ești o persoană localizată într-un oraș nou și ți se întâmplă o problemă sau ai nevoie de un serviciu, nu este prea ușor să găsești o persoană care să te ajute.

Pentru rezolvarea acestei probleme am decis să realizez o aplicație pentru platforma Android, pentru că la noi în țară această platforma Android ocupă un procentaj de peste 82%, după cum se poate observa în *Figura 2* de mai jos.

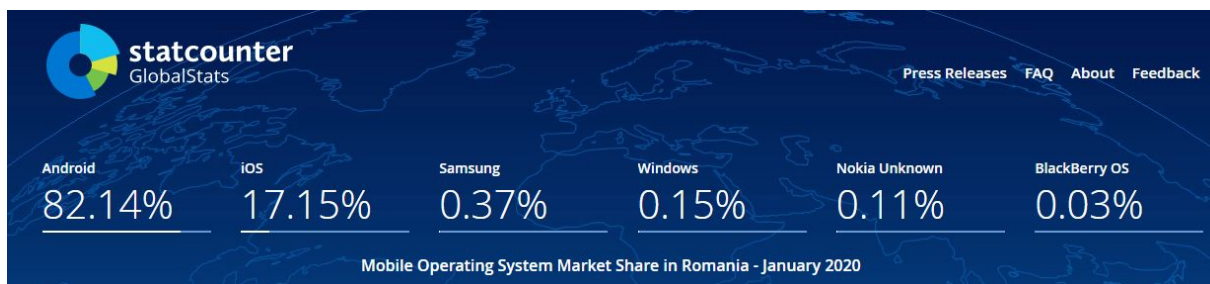


Figura 2: Mobile Operating System Market Share in Romania - January 2020

Această aplicație are ca scop ajutarea persoanelor care se află la începuturile practicării unui domeniu cât și ajutarea persoanelor care au nevoie de un serviciu însă se află în dificultate în găsirea unei persoane.

Dezvoltarea aplicației a început prin imaginarea și crearea interfeței, apoi urmând crearea serverului și a bazei de date. Conexiunea dintre aplicația client și server se face prin intermediul protocolului de comunicare TCP în favoarea protocolului UDP, deoarece fiecare pachet de date este important și este nevoie de confirmare pentru primirea fiecărui pachet (Protocolul UDP nu verifică recepționarea mesajului). După crearea comunicării între aplicația client și server a fost creată baza de date și apoi stabilită conexiunea dintre server și baza de date. Odată ce baza a fost creată, funcționalitățile mai rămân de implementat. Astfel procesul de implementare a unei funcționalități reprezintă definirea acesteia în aplicația client, apoi crearea cazurilor de răspuns în aplicația server și actualizarea bazei de date din server.

Prin urmarea acestui mod de lucru am reușit să creez o aplicație ce are ca funcționalitate solicitarea de servicii de către utilizatori și răspunderea la solicitările aferente de către alți utilizatori în vederea soluționării problemei. De asemenea acest lucru a putut fi posibil și datorită respectării convențiilor programării și a bunelor practici ale acesteia realizând o aplicație care este ușor de înțeles dar și ușor de implementat noi funcționalități.

În timpul dezvoltării acestei aplicații am folosit cunoștințe dobândite de la materii studiate în timpul facultății, acestea fiind:

- **Programare avansată:** introducerea în limbajul Java și a bunelor practici ale acestuia.
- **Tehnici de programare pe platforma Android:** introducerea în crearea de aplicații pentru platforma Android.
- **Introducere în .NET:** introducerea în limbajul C# și a principiilor acestuia.
- **Baze de date:** crearea și gestionarea unei baze de date.
- **Programare orientată obiect:** practicile în realizarea unei aplicații orientată obiect.
- **Rețele de calculatoare:** protocolul TCP și UDP, și care dintre acesta este mai potrivit pentru această aplicație.
- **Ingineria programării:** gestionarea și eficientizarea modului de lucru cât și realizarea diferitelor tipuri de diagrame.

## 1 Descrierea problemei

Atunci când ești nou într-un oraș, șansele ca să ai nevoie de un serviciu sunt destul de ridicate și găsirea persoanei potrivite nu este un lucru ușor. Cel mai comun lucru este apelarea unui prieten sau a unei cunoștințe explicându-le situația cu speranța că poate au pe cineva în minte care te-ar putea ajuta cu problema pe care o ai.

Într-o altă ordine de idei, atunci când începi să activezi într-un domeniu ce reprezintă oferirea de servicii, șansele ca să ai succes încă de la început sunt destul de mici și chiar sunt șanse din lipsa de activitate să fii nevoit să renunți și să urmezi altă cale.

Din pricina acestor motive am ales să încerc să schimb situația prin crearea acestei aplicație care are ca scop ajutarea ambelor tipuri de persoane. Prin această aplicație rapiditatea găsirii persoanei potrivite pentru soluționarea problemei crește față de metodele curente cât și siguranța capabilităților acesteia deoarece cunoaște problema în avans și cunoaște calea spre soluționarea acesteia.

## 2 Abordări anterioare

În acest moment nu există aplicații, fie web, fie mobile, care să aibă acest mod de funcționare, însă sunt anumite aplicații care se apropie de această funcționalitate.

Cea mai apropiată este aplicația OLX ce are în componența sa funcționalitatea ca persoanele să își listeze serviciile pe care acestea le oferă, după cum se poate observa în *Figura 3* de mai jos.

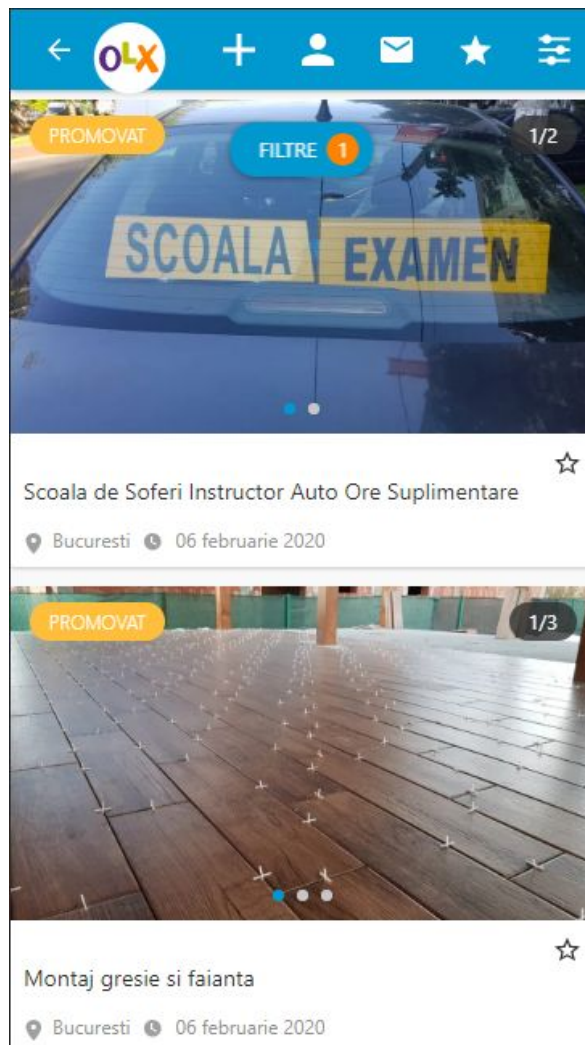


Figura 3: Pagina OLX pentru listarea serviciilor

Modul în care aceasta funcționează este că o persoană fizică sau juridică adaugă un anunț în care își listează serviciile pe care le oferă, alături de informații pentru contact. Este o modalitate foarte bună pentru a-ți anunța serviciile pe care le oferi însă deseori persoanele alese sunt printre primele opțiuni iar acele opțiuni sunt anunțuri promovate, după cum se poate vedea în *Figura 3* de mai sus, limitând posibilitatea altor persoane care oferă aceleași servicii să fie aleși.

O altă modalitate apropiată pentru a face rost de un serviciu este prin intermediul platformei de socializare Facebook, prin grupurile dedicate acestor situații, după cum se poate observa în *Figura 4* de mai jos.

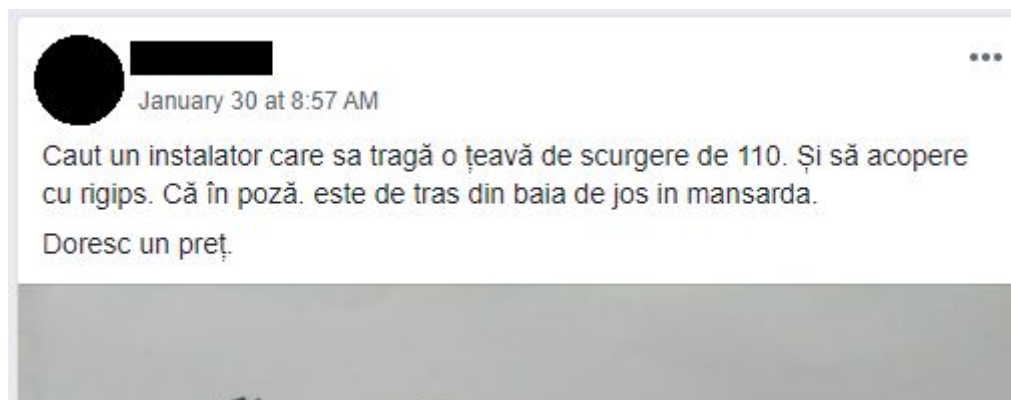


Figura 4: Postare pe un grup de pe rețeaua de socializare Facebook

Pentru a putea face parte dintr-un asemenea grup este nevoie, în marea majoritate a cazurilor, de un proces de înscriere. Cazurile diferă de la apăsarea unui buton, până la răspunderea unor anumite întrebări și așteptarea de a aprobarea pentru a face parte din grupul respectiv sau chiar respingerea, în unele cazuri.

Răspunsurile la asemenea postări sunt în mare parte recomandări de persoane din partea utilizatorilor ce nu oferă servicii și chiar răspunsuri în batjocură.

Aplicația pe care am dezvoltat-o, combină într-o oarecare măsură aceste două funcționalități în care utilizatorii postează detalii despre serviciile de care au nevoie, iar persoanele ce pot oferi aceste servicii răspund prin mesaje, discutând detalii anterioare.

## 3 Descrierea soluției

### 3.1 Principalele funcționalități

Atunci când utilizatorul deschide pentru prima dată aplicația, ceea ce apare este pagina pentru introducerea datelor de autentificare, prezentată în *Figura 5* de mai jos, în care se cere pentru a fi introduse adresa de email a contului, cât și parola aferentă a acestuia. După ce datele au fost introduse se apasă pe butonul de Autentificare. În funcție de răspunsul primit se pot întâmpla două lucruri: se trece la pagina următoare, în care se pot vedea postările utilizatorilor, sau se afișează un mesaj în care este specificat că datele introduse nu sunt corecte. Dacă utilizatorul are nevoie de un cont pentru a-i fi permisă utilizarea aplicației, acest lucru poate fi realizat prin apăsarea textului ce are mesajul aferent, vizibil în *Figura 5* de mai

jos. După apăsarea textului respectiv utilizatorului i se afișează un formular în care sunt solicitate câteva date (nume, prenume, adresa de email și o parolă) pentru identificarea contului, prezentat în *Figura 6* de mai jos. După completarea câmpurilor, toate fiind obligatorii, se apasă pe butonul “Înregistrare” și, la fel ca și la pagina de autentificare, se așteaptă un răspuns. Din nou, dacă datele introduse nu sunt în regulă, se afișează un mesaj corespunzător, specificându-se care din câmpuri nu sunt valide. Contrar, utilizatorul este autentificat automat și redirecționat spre pagina în care se pot vedea toate postările.

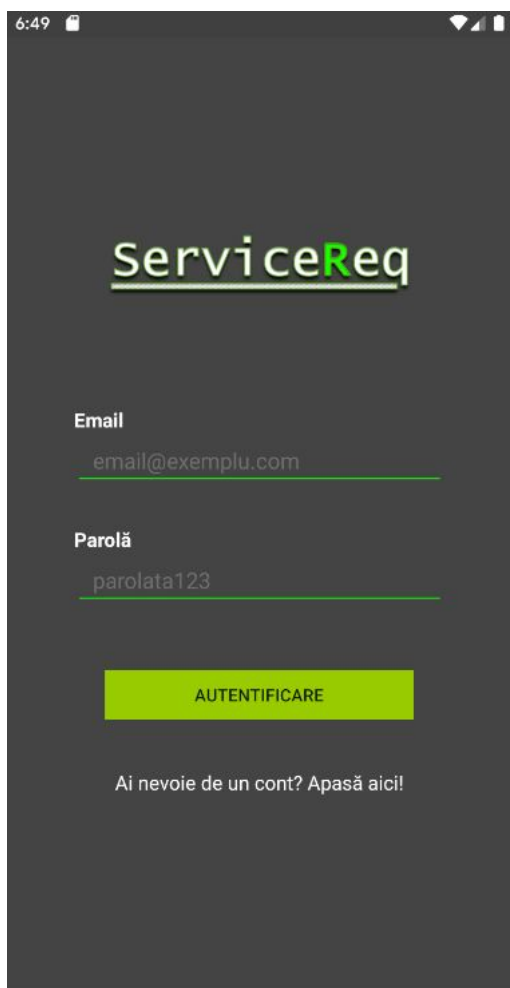


Figura 5: Autentificare

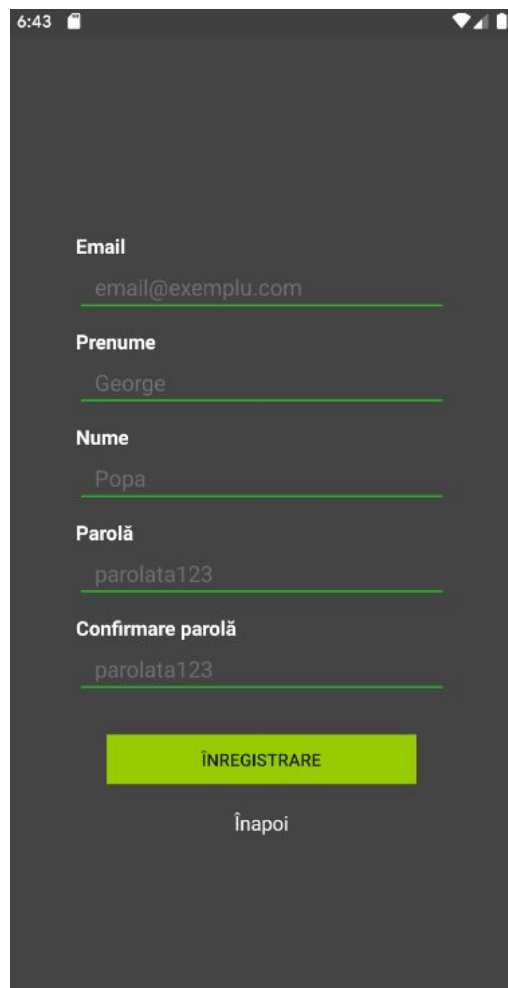


Figura 6: Înregistrare

În continuare, după autentificare sau autentificare, utilizatorul îi va fi afișată pagina în care se află toate postările adăugate de utilizatori. Fiecare postare conține aceleași detalii însă singura diferență din punct de vedere compozițional poate fi existența sau nu a unei imagini atașate unei postări, acest lucru putând fi văzut în *Figura 6* de mai jos. Odată autentificat, poți naviga prin toate componentele aplicației folosind bara de navigare poziționată jos (vizibil tot în *Figura 7* de mai jos). Aceste componente sunt:

1. **Profil:** pagina în care vezi detaliile profilului utilizatorului autentificat, alături de posibilitatea editării acestuia cât și deconectarea de pe acest cont.
2. **Postări:** pagina în care se pot vedea postările utilizatorilor și posibilitatea de a adăuga o nouă postare.
3. **Mesaje:** pagina în care vezi toate mesajele trimise și primite pe acel cont



Figura 7: Pagina cu postări

În această pagină, pentru a adăuga o postare nouă, vei apăsa pe butonul “Adaugă o postare” și se va afișa pagina în care introduci detaliile postării, incluzând o descriere, o imagine dacă este nevoie și categoria din care consideri că face parte serviciul de care ai nevoie, pagină ce poate fi văzută în *Figura 8* de mai jos.

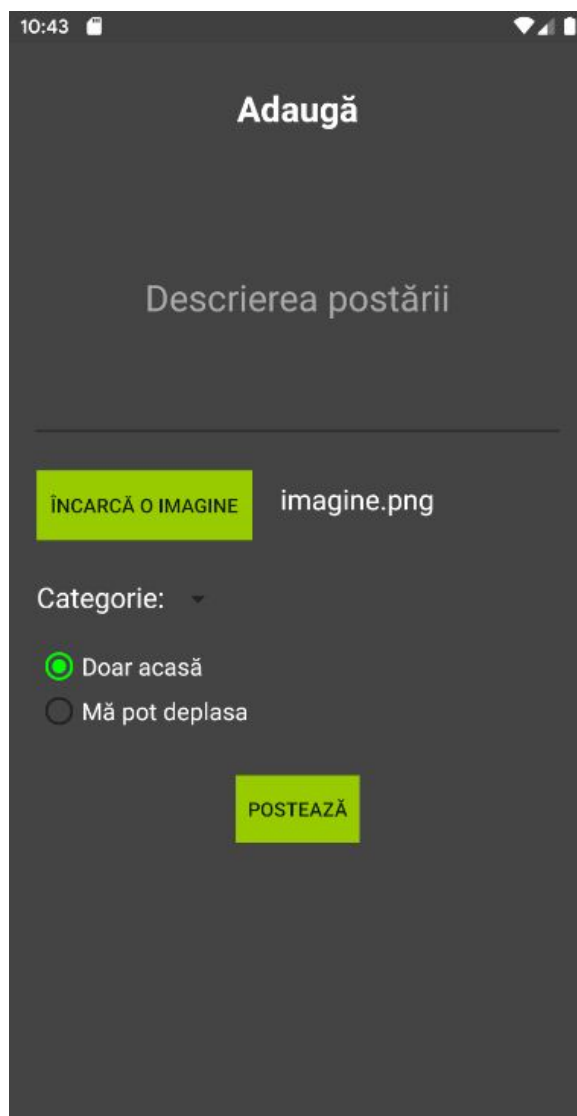


Figura 8: Pagina pentru adăugarea unei postări

Din pagina cu postări, dacă dorești să trimiți un mesaj persoanei pe care dorești să o ajuți, tot ce trebuie să faci este să apeși pe postarea respectivă și automat vei fi redirecționat către pagina în care vei putea trimite un mesaj utilizatorului respectiv discutând eventual detaliile serviciului pe care îl poți oferi.

În pagina de vizualizare a mesajelor se află toate convorbirile pe care le-ai avut cu alți utilizatori, detaliile fiind numele acestora și ultimul mesaj primit sau trimis către acesta. Aici se mai află și posibilitatea de a oferi o evaluare a utilizatorului respectiv, marcată prin butonul ce are drept simbol două stelute galbene, după cum se poate observa în *Figura 9* de mai jos, alături de detaliile menționate mai sus.

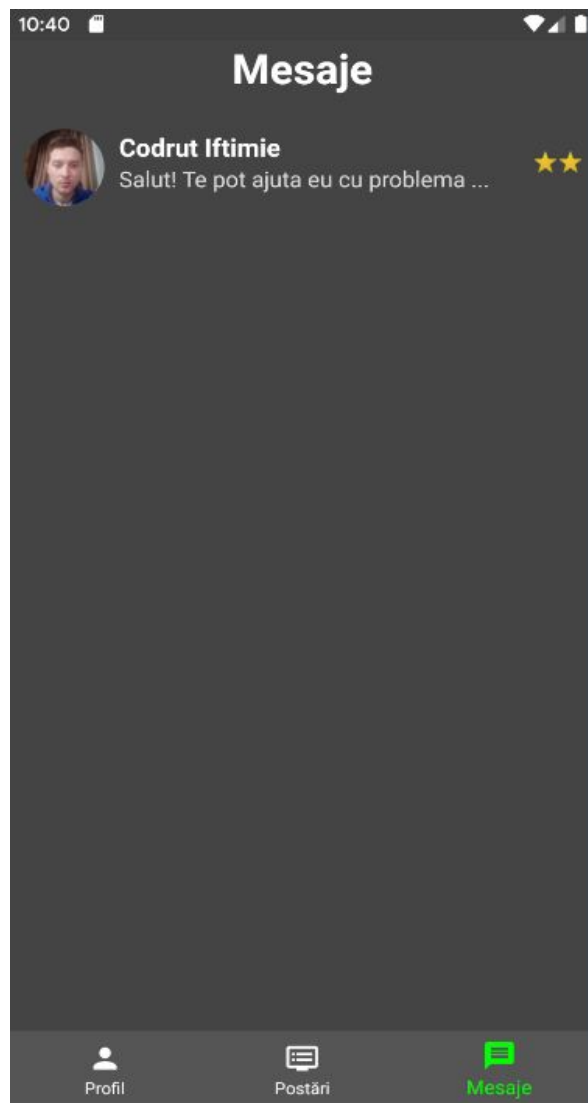


Figura 9: Pagina ce conține mesajele trimise sau primite

Odată cu apăsarea pe butonul pentru evaluare, va apărea o fereastră în care se va preciza persoana pentru care urmezi să oferi o evaluare alături de selectarea numărului de stele pe care dorești să le oferi persoanei respective. Dacă dorești să amâni această evaluare, sau a fost o apăsare din greșeală, există opțiunea de anulare, vizibilă în *Figura 10* de mai jos, alături de lucrurile menționate anterior.



Figura 10: Evaluarea unei persoane



Pentru a răspunde sau a continua conversația cu o persoană, după ce apeși pe mesajul descris anterior, va apărea pagina în care vor apărea toate mesajele interschimbate cu persoana respectivă și în partea de jos locul unde poți trimite un nou mesaj, alături de butonul pentru expediere, obiecte vizibile în *Figura 11* de mai jos.



Figura 11: Pagina pentru interschimbare de mesaje

Diferențierea dintre mesajele trimise și mesajele primite se face prin alinierea în partea stângă sau dreaptă a mesajelor. Pentru mesajele trimise alinierea se face în partea dreaptă unde, în dreapta mesajului apărând poza setată profilului tău cât și culoarea fundalului mesajului fiind o culoare deschisă. Același lucru se întâmplă și pentru mesajele primite, însă ordinea este inversă. În stânga mesajului apare poza de profil a persoanei care v-a trimis mesajul, iar culoarea fundalului mesajului este una închisă, diferențele fiind vizibile în *Figura 11* de mai sus.

În pagina de vizualizare a profilului apar, de sus în jos, butonul pentru editarea profilului, butonul pentru deconectarea de pe contul curent, urmând apoi poza aferentă

profilului urmată de numele care a fost setat profilului. În josul numelui apare ratingul pe care îl ai, compus dintr-un număr maxim de 5 stele. După aceste detalii urmează lista în care apar toate postările făcute de pe acest profil, lucruri vizibile în *Figura 12* de mai jos.

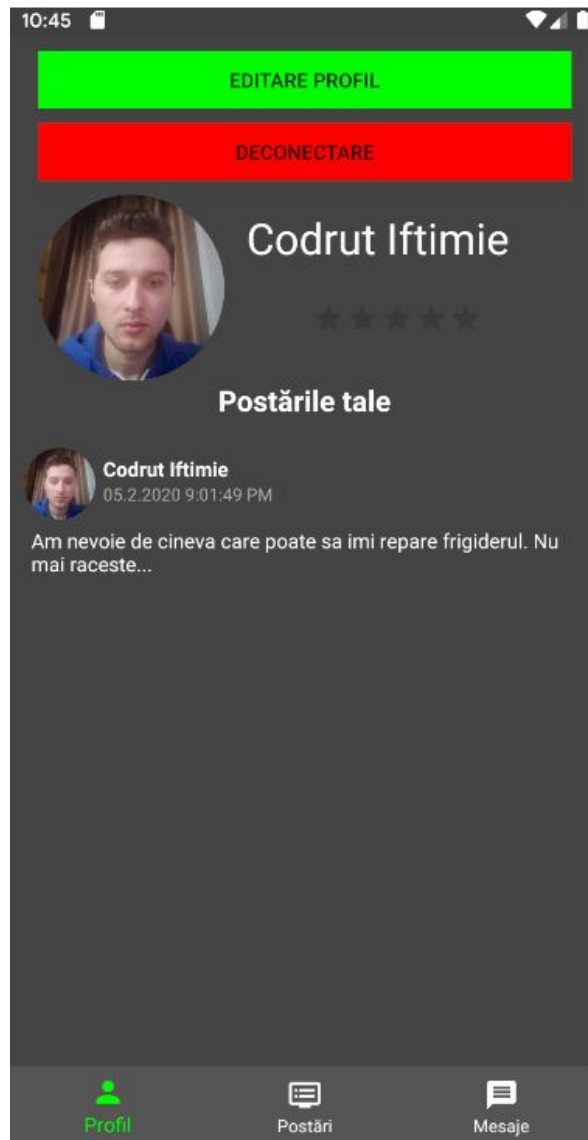


Figura 12: Vizualizarea profilului

Pentru schimbarea pozei profilului, nu este nevoie de a intra pe pagina de editare a profilului, ci să apeși pe poza curentă. După apăsare se va afișa o scurtă listă în care poți alege modul în care încarci noua imagine de profil. Prima opțiune este “Camera”, care va deschide aplicația telefonului pentru fotografiat, după care ți se va permite să faci o poză care va fi în final poza curentă profilului. Următoarea opțiune este “Alege din Galeria foto” în care alegi o poză salvată în telefonul mobil să fie poza de profil. Ultima opțiune este “Anulare”, care după cum sugerează și numele, anulează acțiunea curentă. Aceste opțiuni pot fi văzute în *Figura 13* de mai jos.

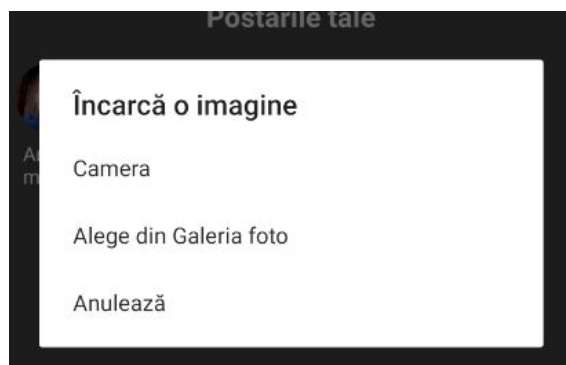


Figura 13: Încărcarea unei imagini

Pentru editarea profilului, după apăsarea butonului aferent, va apărea pagina de editare a profilului care conține un formular în care poți schimba numele, prenumele și parola curentă dar și activarea opțiunii “Helper” care îți va permite să primești notificări atunci când este adăugată o postare ce face parte dintr-o categorie pe care ai selectat-o. Această pagină poate fi văzută mai jos, în *Figura 14*.

Figura 14: Pagina pentru editarea profilului

După bifarea opțiunii “Helper” se va afișa o listă cu opțiuni multiple, din care pot fi selectate mai mult decât una din ele. Aceste opțiuni pot fi văzute în *Figura 15* de mai jos.

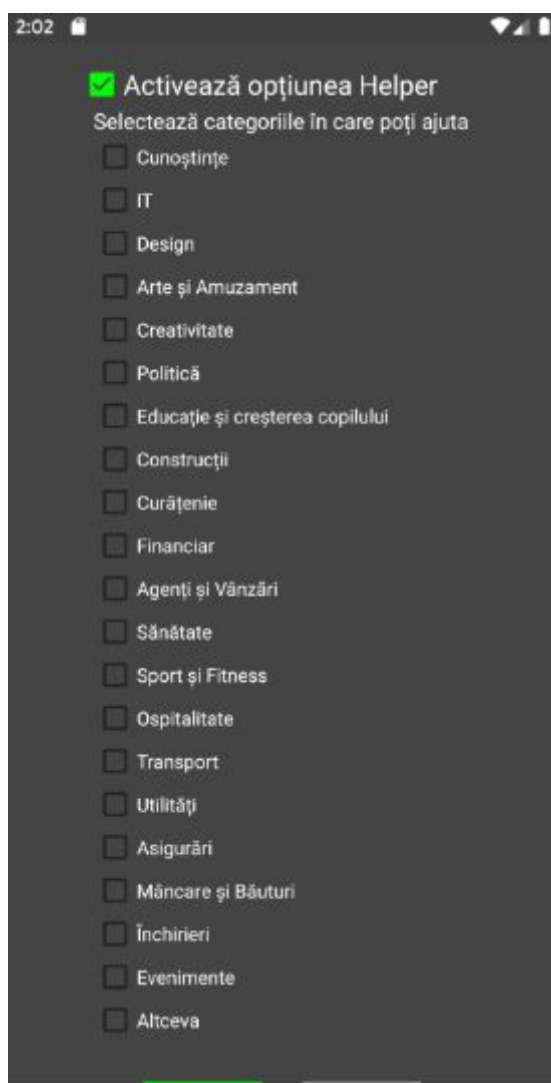


Figura 15: Opțiunile modului Helper

Funcționalitățile enumerate mai sus alcătuiesc aplicația care încearcă să soluționeze problemele enumerate mai sus, însă aceasta poate fi îmbunătățită deoarece s-ar putea spune că reprezintă doar baza unei astfel de aplicații.

## 4 Detalii de implementare

### 4.1 Aplicația server

Începând cu aplicația server, care este una standalone, lucrurile decurg într-un mod foarte simplu. Se deschide portul specificat și se așteaptă conectarea clienților. Un client odată conectat, este preluat de clasa `ClientHandler` care caută un thread optim pe care să fie adăugat

clientul. În acest moment limita de clienți pe un thread este de 100, lucru care poate fi ajustat în funcție de puterea sistemului pe care rulează serverul.

Serverul, pentru partea de tratare a clienților, este o combinație dintre un server concurent și unul iterativ, deoarece acesta asignează câte un thread la fiecare 100 clienți, aceștia fiind tratați în mod iterativ. După depășirea pragului de 100 clienți se creează un nou thread unde noii clienți vor fi asigurați acolo, de aici fiind partea de concurență.

```
AppThread bestThread = threads[0]; //assume that the best thread is the first
foreach (var thread in threads) //search all the threads
{
    if (thread.Clients.Count < 100 && thread.Clients.Count < bestThread.Clients.Count)
        bestThread = thread;
}
if (bestThread.Equals(threads[0])) //if the best thread was found to be the first
{
    if (bestThread.Clients.Count >= 100) //check if it already has 100 clients
    {
        threads.Add(new AppThread()); //then create a new thread and add it to
```

Figura 16: Asignarea clientului la cel mai bun thread

În *Figura 16* de mai sus se poate observa modul de asignare a unui client către un thread. Întâi se caută thread-ul cu cei mai puțini clienți, presupunând că primul thread are cei mai puțini clienți. După găsirea threadului respectiv, se verifică dacă acel thread are deja numărul maxim de clienți. În caz pozitiv, se creează un nou thread iar apoi clientul este asignat noului thread.

```
public static bool InsertQuery(string table, string[] fields, string[] values)
{
    using (SqlCommand cmd = new SqlCommand())
    {
        cmd.Connection = Database;
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = $"INSERT INTO {table}(";
        foreach (var field in fields)
            cmd.CommandText += $"{field},";
        cmd.CommandText = cmd.CommandText.Remove(cmd.CommandText.Length - 1);
        cmd.CommandText += ") VALUES (";
        foreach (var value in values)
            cmd.CommandText += $"{value},";
        cmd.CommandText = cmd.CommandText.Remove(cmd.CommandText.Length - 1);
        cmd.CommandText += ")";

        if (cmd.ExecuteNonQuery() > 0)
        {
```

Figura 17: Funcția pentru inserarea datelor într-o tabelă

În *Figura 17* de mai sus v-am prezentat funcția care se ocupă cu inserarea datelor într-o tabelă. Parametrii acesteia sunt alcătuiți dintr-un string care reprezintă numele tabelului în care dorești să introduci date, un array de stringuri care să conțină câmpurile pentru care dorești să introduci valori, și încă un array de stringuri care trebuie să conțină valorile pentru câmpurile respective.

## 4.2 Aplicația client

Un aspect important, pe partea aplicației client, este pasarea informațiilor necesare a unei postări pentru crearea view-ului pe partea de vizualizare a mesajelor trimise sau primite de către utilizator și crearea acelui view atunci când se primește mesaj de la server că a fost primit un nou mesaj. Pentru această funcționalitate s-a folosit metoda `setTag()` pentru un view din informațiile pasate de Intent, modul de utilizare fiind vizibil în *Figura 18* de mai jos.

```
toBeAdded.setTag(viewTagId);
```

Figura 18: Metoda `setTag()`

În cazul de mai sus, parametrul “viewTagId” reprezintă ID-ul utilizatorului de la care s-a primit mesajul pentru ca mai apoi, atunci când se primește sau se trimite un mesaj către același utilizator acest view să fie actualizat și nu creat altul nou pentru a nu crea o confuzie între mesaje.

După acest pas, pentru afișarea conversațiilor anterioare a fost folosită implementarea interfeței `SharedPreferences`, în care pentru fiecare conversație cu persoane diferite, drept cheie pentru găsirea acesteia este combinația “CONVO” + id-ul persoanei cu care este conversația respectivă, iar ca valoare a fost setat obiectul serializat care conține toate mesajele interschimbate cu aceasta.

Serializarea obiectului respectiv s-a realizat prin utilizarea clasei `Gson`, oferită de Google, pentru conversiunea obiectului respectiv în format JSON. Structura acestui obiect o puteți observa în *Figura 19* de mai jos.

```
public class Conversation implements Serializable {
    String firstName;
    String lastName;
    String receiverId;
    List<ExchangedMessage> conversation;
}

class ExchangedMessage {
    String senderId;
    String message;
    String date;
}
```

Figura 19: Structura clasei `Conversation`

Atunci când este deschisă pagina pentru vizualizarea conversațiilor se verifică dacă există salvată o conversație. În caz pozitiv, se încarcă stringul JSON în memorie, se deserializează folosind clasa Gson și se instanțiază obiectul de tipul Conversation. Apoi se parcurg elementele din lista “conversation” și se creează un câte un view pentru fiecare element din listă, aceste view-uri diferind în funcție de persoana care a trimis mesajul respectiv, verificându-se stringul “senderId” cu ID-ul utilizatorului curent. Aceste view-uri pot fi vizibile în *Figura 11* de mai sus. Dacă nu există o conversație cu utilizatorul respectiv (caz posibil atunci când se dorește începerea unei conversații din pagina cu postări sau prin primirea unui mesaj de la server indicând un mesaj nou), se creează una imediat după ce a fost trimis un mesaj către persoana respectivă, sau atunci când se primește un mesaj de la persoana respectivă.

Alt aspect este structura mesajelor interschimbate între client-server. Fiecare astfel de mesaj are o structură specifică și un număr de “parametri” exacți. Fiecare mesaj începe cu o literă majusculă care reprezintă acțiunea pe care urmează să o facă serverul sau aplicația client. Acțiunile curente sunt:

- ➔ **L**: verificarea datelor pentru autentificare
- ➔ **R**: crearea unui nou cont de utilizator
- ➔ **N**: adăugarea unei noi postări
- ➔ **U**: actualizarea datelor utilizatorului autentificat
- ➔ **O**: deconectarea de la utilizatorul autentificat
- ➔ **M**: trimiterea sau primirea unui nou mesaj
- ➔ **I**: actualizarea imaginii de profil a utilizatorului autentificat

Toate aceste astfel de mesaje conțin și ID-ul utilizatorului care dorește să facă una din acțiunile menționate mai sus.

Pentru cazul în care un utilizator adaugă o postare nouă, pe lângă faptul că adăugăm acea postare în baza de date trebuie atenționați și ceilalți utilizatori că a apărut o astfel de postare, astfel, pe partea de server, după ce a fost introdusă noua postare în baza de date, se trimite un mesaj către toți utilizatorii autentificați, atenționându-i că a fost adăugată o postare nouă.

Pe partea de client lucrurile sunt puțin mai complicate și este nevoie de lucru cu thread-uri, principalul motiv fiind pentru a nu bloca interfața aplicației. Un prim thread este

pentru a citi mesajele care vin de la server. Acest thread decodează mesajul și în funcție de acțiunea necesară trimite mesajul către adapterul specific, în acest caz adapterul Postărilor numit “PostsAdapter”. Acest adapter are în componența sa o listă cu toate postările pe care trebuie să le creeze un view, această listă fiind constant verificată într-un thread și dacă conține elemente, apelează funcția care creează view-ul pentru acea nouă postare pasând datele necesare.

Am menționat anterior că serverul atunci când trimite mesaj tuturor utilizatorilor, verifică mai întâi dacă utilizatorul respectiv este autentificat. Acest lucru este necesar pentru că dacă nu s-ar face această verificare ar trimite mesaj acelui utilizator cu postarea respectivă, iar atunci când se autentifică, acesta primește toate postările care au fost adăugate până în acest moment, inclusiv această nouă postare, rezultând în duplicarea postării respective pe partea de client.

## 4.3 Diagrame

### 4.3.1 Diagrame use-case

În *Figura 20* de mai jos se pot observa cazurile de folosință a aplicației pentru un utilizator neautentificat.

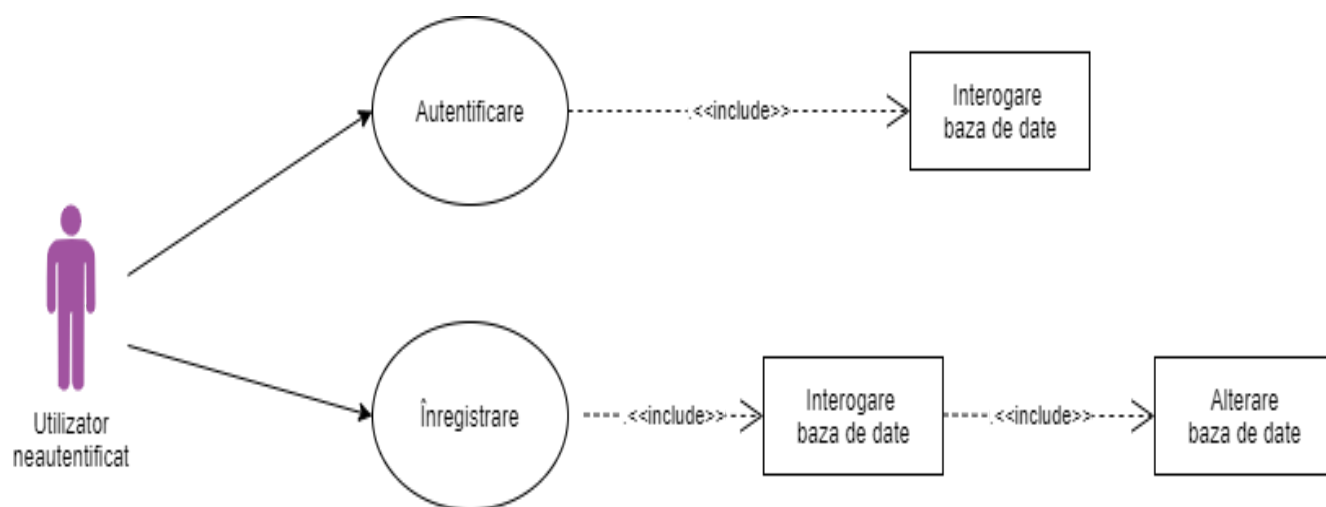


Figura 20: Diagrama use-case a unui utilizator neautentificat

În *Figura 21* de mai jos se observă cazurile de folosință a aplicației după ce utilizatorul s-a autentificat.



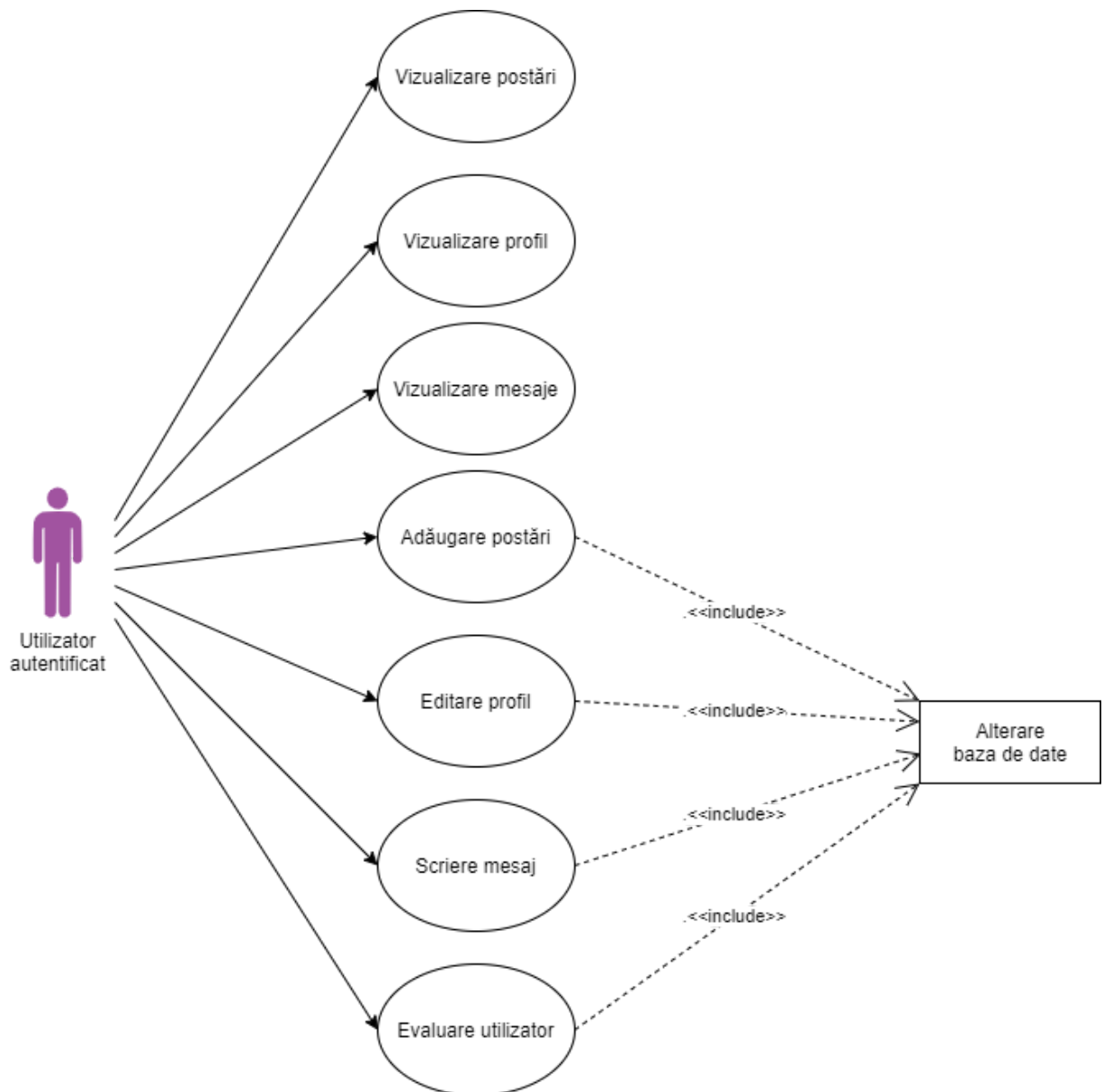


Figura 21: Diagrama use-case a unui utilizator autentificat

#### 4.3.2 Diagrame arhitecturale

Arhitectura aplicației client poate fi văzută în *Figura 22* de mai jos. Acțiunile sunt destul de simple. Utilizatorul începe interacțiunea cu activitatea care se ocupă cu autentificarea (“LoginActivity”). După autentificare pornește “MainActivity” care deschide inițial “FeedFragment”, dar poți interschimba fragmentele între ele. Fiecare fragment pornește cel puțin o activitate, fiecare dintre aceste activități folosind clasa “Server” pentru a trimite mesaje către aplicația server. Clasa “Server” interacționează cu cele două adaptere, care la rândul lor modificând view-urile fragmentelor.

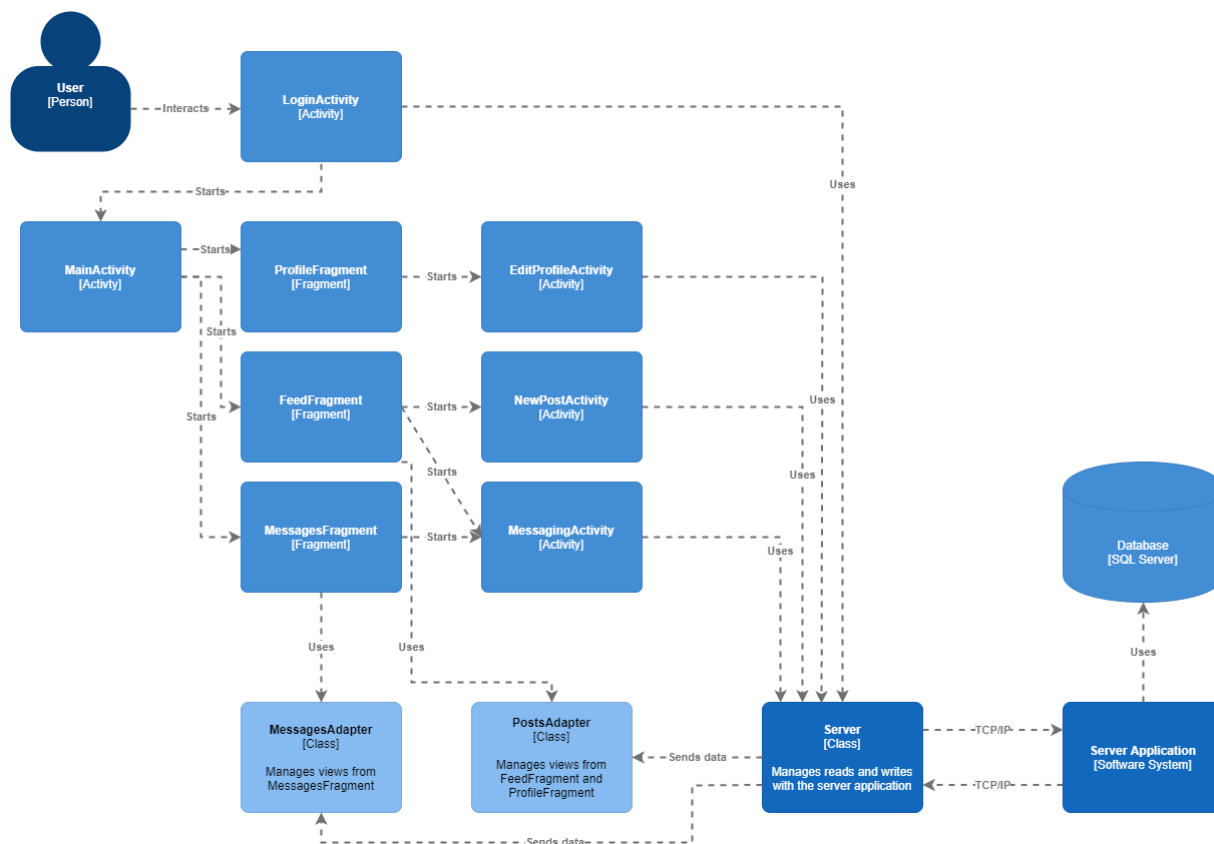


Figura 22: Arhitectura aplicație client

În *Figura 23* de mai jos este descrisă arhitectura aplicației server. Clasa “Server” este clasa care se ocupă de interschimbarea datelor cu aplicația client și de asemenea face citiri și modificări asupra bazei de date. Când un client se conectează acesta este redirecționat către clasa “ClientHandler” care decide ce thread este mai bun pentru client. Clientul este adăugat în lista unei instanțe a clasei “AppThread” iar structura clientului va fi într-un final de tipul “AppClient”.

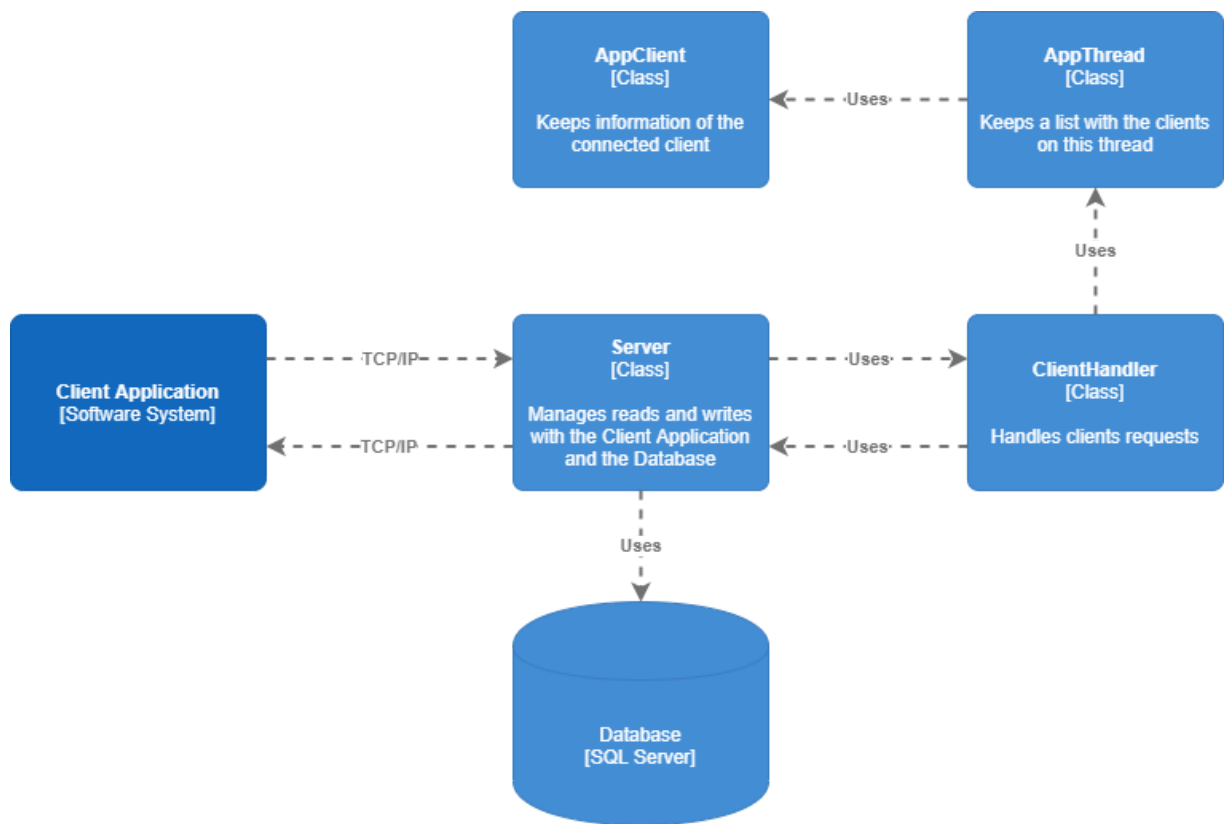


Figura 23: Arhitectura serverului

În *Figura 24* de mai jos este descrisă structura bazei de date precum și relațiile dintre tabelele acesteia. Tabele care compun baza de date sunt “Users”, “Posts” și “Messages”.

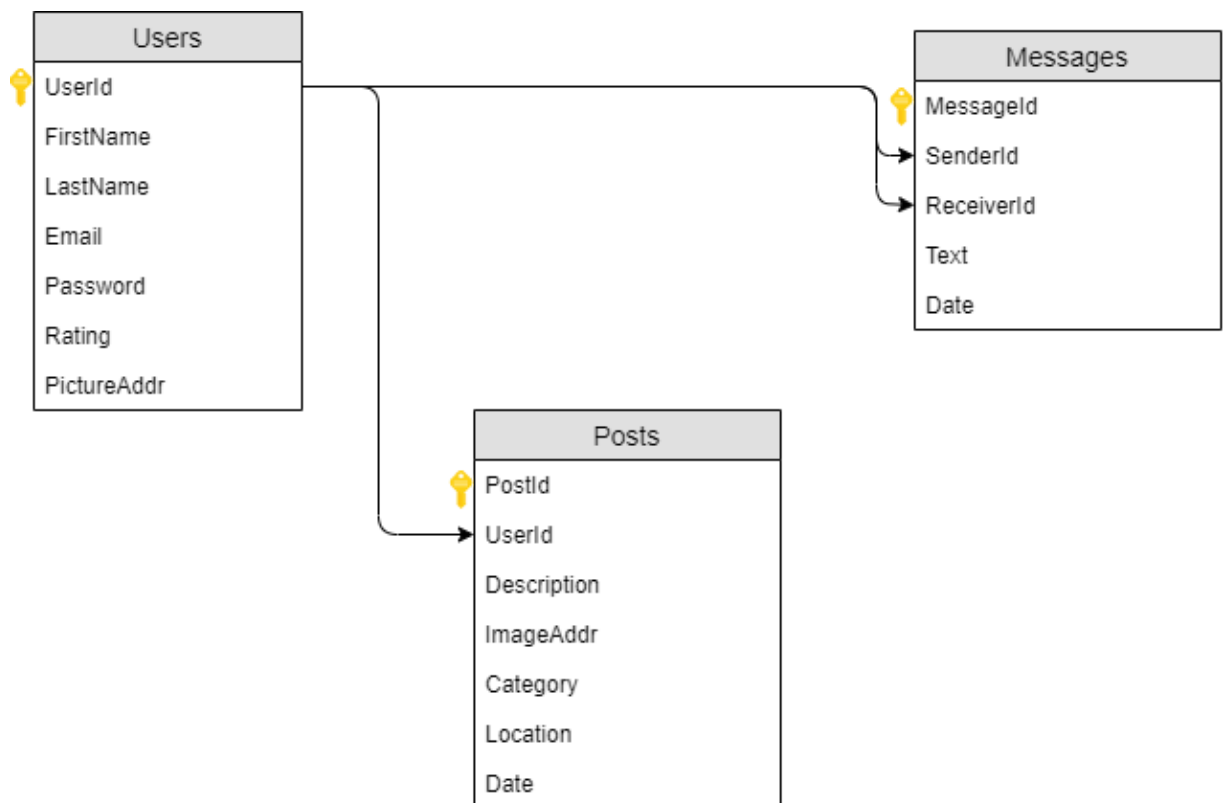


Figura 24: Structura bazei de date

## Concluziile lucrării

Prin această lucrare de licență doresc să afirm faptul că poți crea o aplicație de care pot beneficia o multitudine de persoane și care să ajute aceste persoane în dezvoltarea profesională, personală și a cunoștințelor acestora, poate chiar să promoveze spiritul de ajutor.

Prin utilizarea platformelor cel mai des întâlnite, asigurăm posibilitatea câtor mai multe persoane să utilizeze orice aplicație, iar fiecare dintre aceste aplicații sunt create cu scopul de a ne ajuta și de a ne ușura munca pe care trebuie să o depunem, indiferent de situație.

După cele menționate mai sus, consider că aplicația **“ServiceReq”** vine în ajutorul tuturor persoanelor, indiferent de vârstă care necesită obținerea unui serviciu ce în anumite situații se poate descoperi a fi un lucru dificil.

Alt ajutor pe care îl oferă această aplicație este persoanelor care încep să activeze într-un domeniu ce reprezintă oferirea de servicii prin șansa de a fi printre primii care pot oferi serviciul de care alte persoane au nevoie, ajungând astfel cunoscuți în acest scop.

Această aplicație poate fi îmbunătățită prin implementarea următoarelor lucruri:

- Postările să ofere o locație exactă, cu o rază specifică, pentru care utilizatorii “Helper” să primească notificare dacă se află în raza specificată.
- Îmbunătățirea sistemului de evaluare prin adăugarea de comentarii atunci când o persoană este evaluată.
- Posibilitatea de blocare a mesajelor din partea anumitor persoane.
- Adăugarea la pagina profilului numărul de postări soluționate iar în pagina de postări, în dreptul postării să afișeze de cine a fost soluționat.
- Adăugarea de estimare a primirii unui răspuns pentru categoria unde a fost adăugată o postare calculată pe baza postărilor anterioare.

## Bibliografie

1. Android Developers Documentation - <https://developer.android.com/reference/packages.html>
2. C# Documentation - <https://docs.microsoft.com/en-us/dotnet/csharp/>
3. SqlClient Documentation - <https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient>
4. Java Sockets - <https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html>
5. TcpClient - <https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.tcpclient>
6. SqlServer Documentation - <https://docs.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql?view=sql-server-ver15>
7. Gson Documentation - <https://github.com/google/gson/blob/master/UserGuide.md>
8. Mobile OS Market Share - <https://gs.statcounter.com/os-market-share/mobile/romania>