**Abduraman Ciller Melis**

**CSIE, SERIA A INFO, GRUPA 1066**

**DISPOZITIVE ȘI APLICAȚII MOBILE**

**DOCUMENTAȚIE PROIECT**

**GESTIUNEA DATORIILOR FAȚĂ DE ALTE PERSOANE**

1. **INTRODUCERE**

* **Definire problemă (Ce problemă rezolvă proiectul?)**

Această aplicație a fost gândită inițial pentru a gestiona datoriile unei persoane față de diverse întreprinderi, prieteni, etc. Ulterior, am încercat să dezvolt aplicația și să gestionez si datoriile unor persoane față de utilizator, aceste sume înregistrându-se pe post de venituri. Mai mult, utilizatorul va beneficia și de o listă de dorințe care cumva îl va motiva să economisească bani pentru a-și permite obiectele pe care le dorește.

Aplicația încearcă să simplifice viața utilizatorilor ei prin stocarea tuturor informațiilor din această arie a gestiunii datoriilor într-un singur loc, cu o interfață intuitivă, user-friendly. Mi-am pus amprenta în special printr-un design colorat, dar simplu în același timp pentru a fi o aplicație atrăgătoare și ușor de gestionat.

* **Definire obiective (Ce face proiectul pentru a rezolva problema?)**

Obiectivul acestei aplicații este de-a veni în ajutorul persoanelor ocupate, foarte active în viața de zi cu zi, care au o agendă plină de activități solicitante. Aplicația poate fi privită ca o multitudine de sticky notes cu remindere privind data scadentă a facturilor sau chiar și a promisiunilor făcute persoanelor cărora le solicităm ajutorul în ceea ce privește o anumită sumă de bani.

Fiecare utilizator va cunoaste cele mai semnificative detalii privind o anumită datorie, astfel încât își va organiza și gestiona bugetul în funcție de data scadentă și suma de bani corespunzătoare.

* **Definire cadru general (În ce mediu va fi folosită aplicația?)**

Această aplicație se adresează unui public destul de larg în ceea ce privește vârsta, deoarece poate fi utilizată atât de copii, adolescenți, dar și de către persoane mature cu un stil de viață agitat.

Poate fi folosită ori de câte ori în viața utilizatorilor apare un împrumut, o obligație la stat sau față de alte întreprinderi. Prin utilizarea aplicației, utilizatorii ei vor fi liniștiți că nu vor uita o dată scadentă, nu vor face o impresie mai puțin bună față de alte persoane cărora le datorează bani, nu vor întârzia cu plățile și vor avea o viziune de ansamblu asupra bugetului lor.

1. **DESCRIERE SOLUȚIE**

* Nu se fac referiri la tehnologiile folosite
* Descriere scenariu utilizare
* Descriere funcționalități

La prima utilizare a aplicației, se solicită înregistrarea utilizatorului și crearea unui cont pentru ca ulterior să se logheze prin intermediul unui username și a unei parole. I se solicită adresa de mail, numele, prenumele, vârsta și alegerea unui username și a unei parole, iar după ce este de acord cu politica de securitate și condițiile stabilite de fiecare aplicație de acest gen, i se va crea un cont pe care poate să-l acceseze oricând, ba chiar să rămână logat.

Primul meniu ce apare dupa logare, îi oferă posibilitatea utilizatorului să aleagă una dintre opțiuni:

* Adăugare datorie
* Vizualizare wishlist
* Adăugare creanță (datoriile altor persoane față de utilizator)
* Vizualizarea tuturor datoriilor înregistrate
* Oferirea unui feedback dezvoltatorilor aplicației pentru a știi ce să îmbunătățească

ADD DEBT : Primul formular îi permite utilizatorului introducerea unei datorii față de prieteni, bancă, stat și îi solicită cantitatea de bani, unitatea valutară, data împrumutului, data scadentă, modalitatea de plată (cash/card) și opțiunea de a primi un reminder message pe adresa de mail pentru a-i reaminti utilizatorului data scadentă.

MY DEBTS – LIST : După înregistrarea fiecărei datorii, ea va fi stocată în această listă într-un mod organizat pentru a-i oferi utilizatorului o viziune asupra cantității de bani pe care trebuie să o plătească. Din acest formular, utilizatorul poate vedea locația fiecărei întreprinderi, bănci, adresa prietenilor pentru a se orienta mult mai ușor în spațiu și pentru a-și gestiona timpul într-un mod eficient.

MY WISH LIST: Acest formular a fost gândit ca o motivație pentru utilizatori. Astfel, având posibilitatea de a-și nota cărțile dorite, acesta va încerca să economisească din buget pentru a-și permite achiziționarea cărților din acest wish list cât mai repede.

REVIEW – Pentru a îmbunătăți aplicația în funcție de părerile utilizatorilor, le-am oferit posibilitatea acestora de a ne lăsa un feedback. De asemenea, pentru realizarea unor statistici privind calitatea aplicației, utilizatorii sunt rugați să răspundă la câteva întrebări.

ADD RECEIVABLE – Acest formular îi permite utilizatorului introducerea unei creanțe, a unei sume de bani pe care o anumită persoană/întreprindere i-o datorează. Principiul de funcționare este asemănător cu cel descris la ADD DEBT.

1. **IMPLEMENTARE SOLUȚIE**

Am început implementarea functionalităților pe care mi le-am propus prin realizarea inițială a 6 activități, ajungând în final la un număr de 11 activități, o bază de date locală SQLite și o bază de date în cloud – Firebase.

În momentul rulării, aplicația pornește cu pagina de Log In pentru accesarea conturilor de către utilizatori prin introducerea unui username și a unei parole, dar oferă și posibilitatea creării unui cont nou, în cazul persoanelor neînregistrare. Introducerea datelor se realizează prin apăsarea butonului Sign Up care îl va conduce către o pagină nouă. Trecerea dintr-o activitate în alta se realizează prin următoarea secvență de cod :

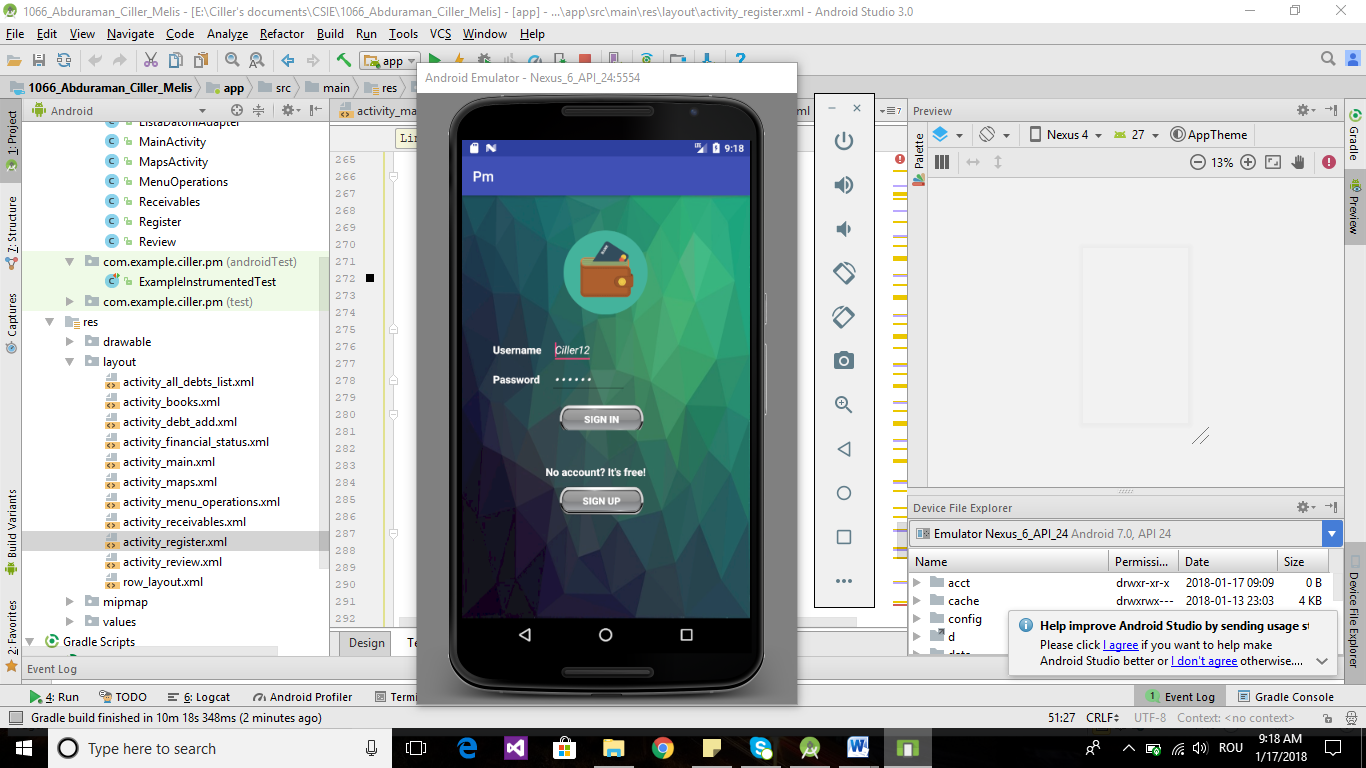
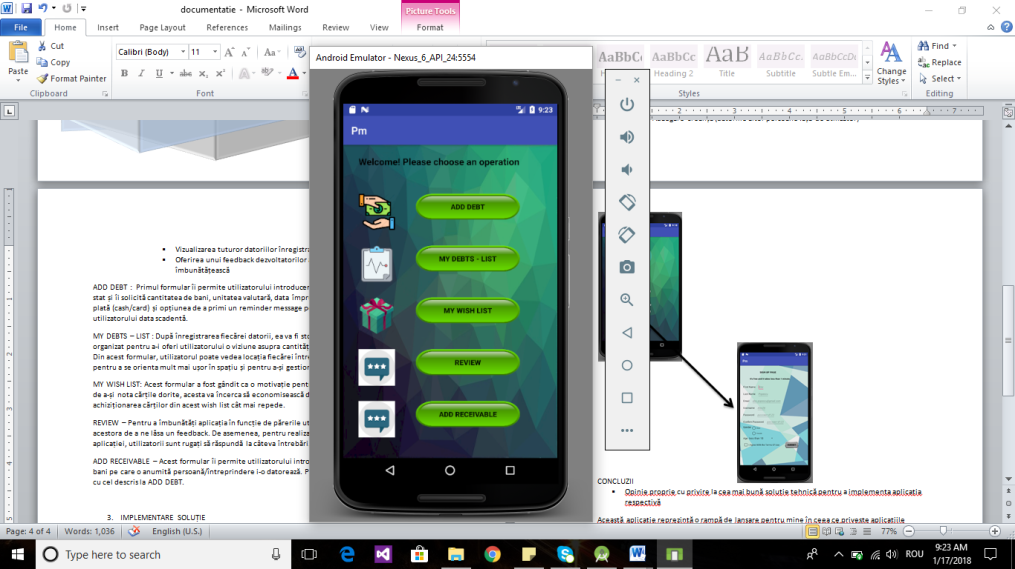
***protected void*** *metodaSignUp(View view)  
{  
 Intent it =* ***new*** *Intent(getApplicationContext(),Register.****class****);  
 startActivity(it);  
  
}*

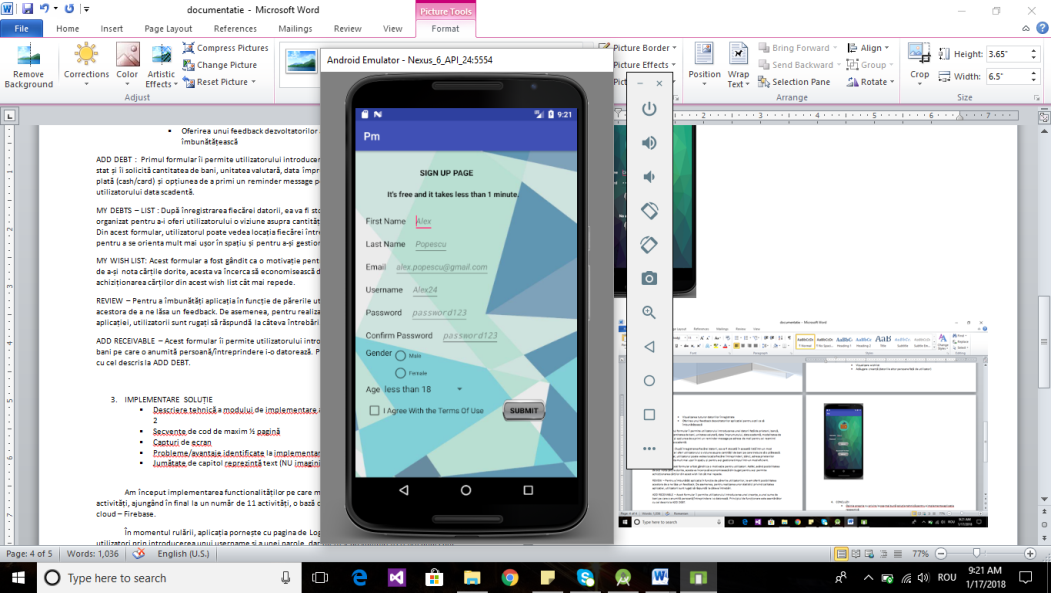
În pagina principală se rețin datele utilizatorului, pentru a nu fi nevoie de introducerea lor la fiecare utilizare a aplicației.

*SharedPreferences shared = getSharedPreferences(****"login"****, Context.****MODE\_PRIVATE****);  
((EditText)findViewById(R.id.****editTextMain1****)).setText(shared.getString(****"username"****,* ***""****));  
((EditText)findViewById(R.id.****editText2****)).setText(shared.getString(****"password"****,* ***""****));*

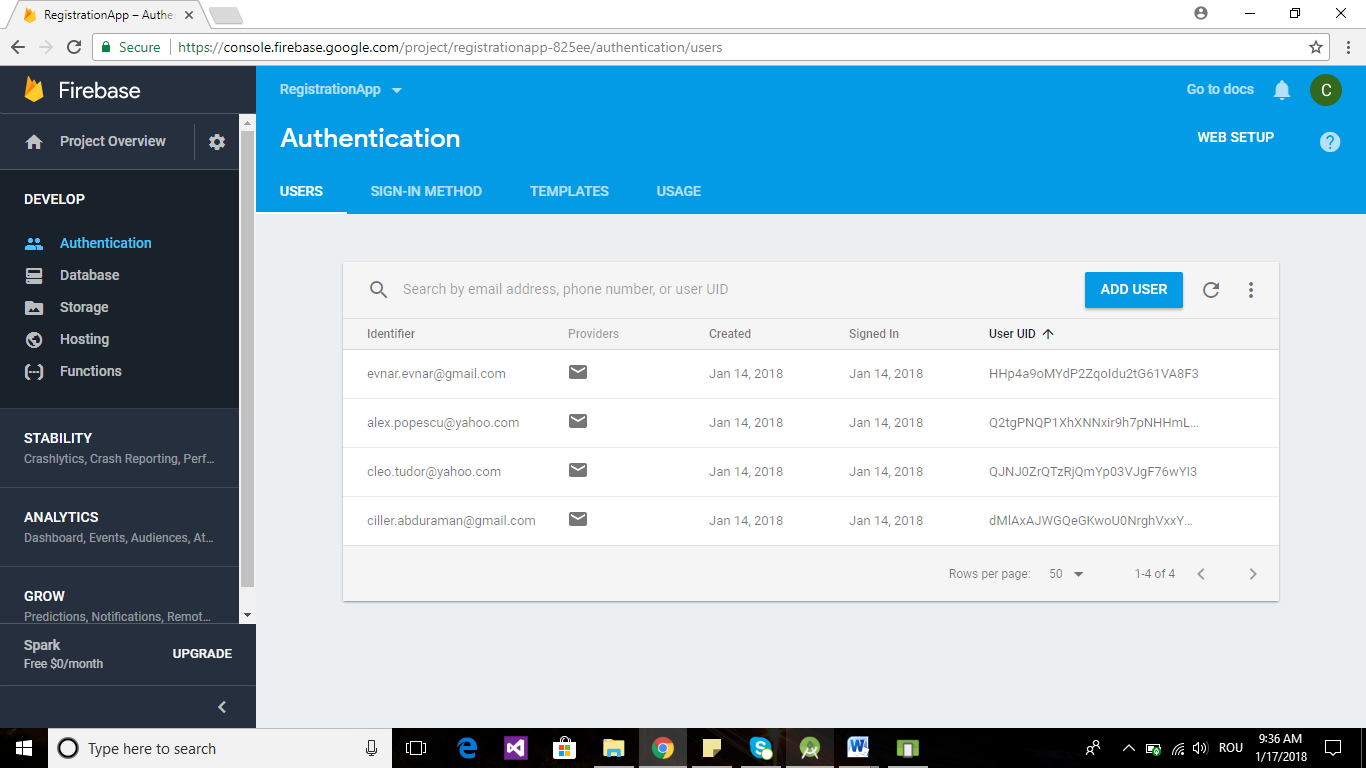
Am realizat și câteva validări pentru a mă asigura ca utilizatorul completează toate câmpurile relevante.

***private boolean*** *isValidPassword(String pass)  
{* ***if****(pass!=* ***null*** *&& pass.length() > 4){* ***return true****;  
 }* ***return false****;  
}*

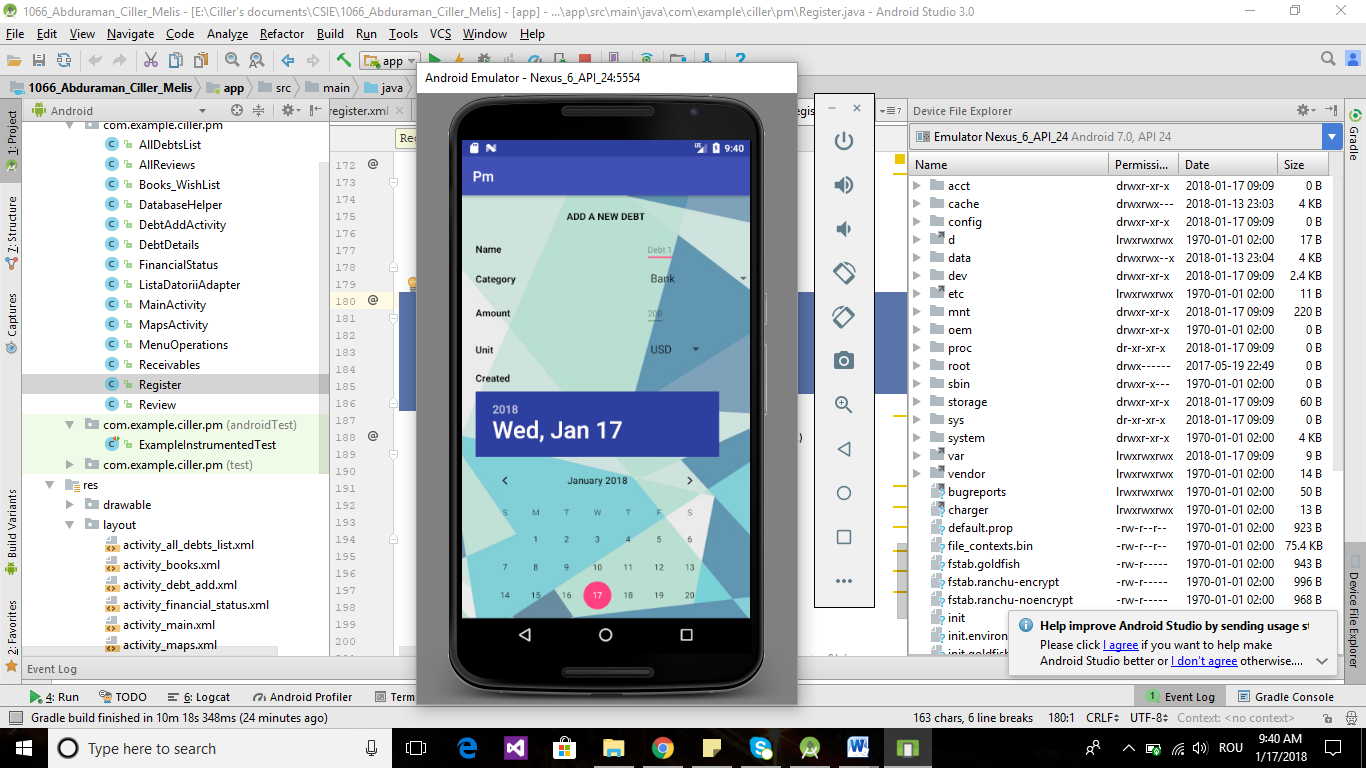
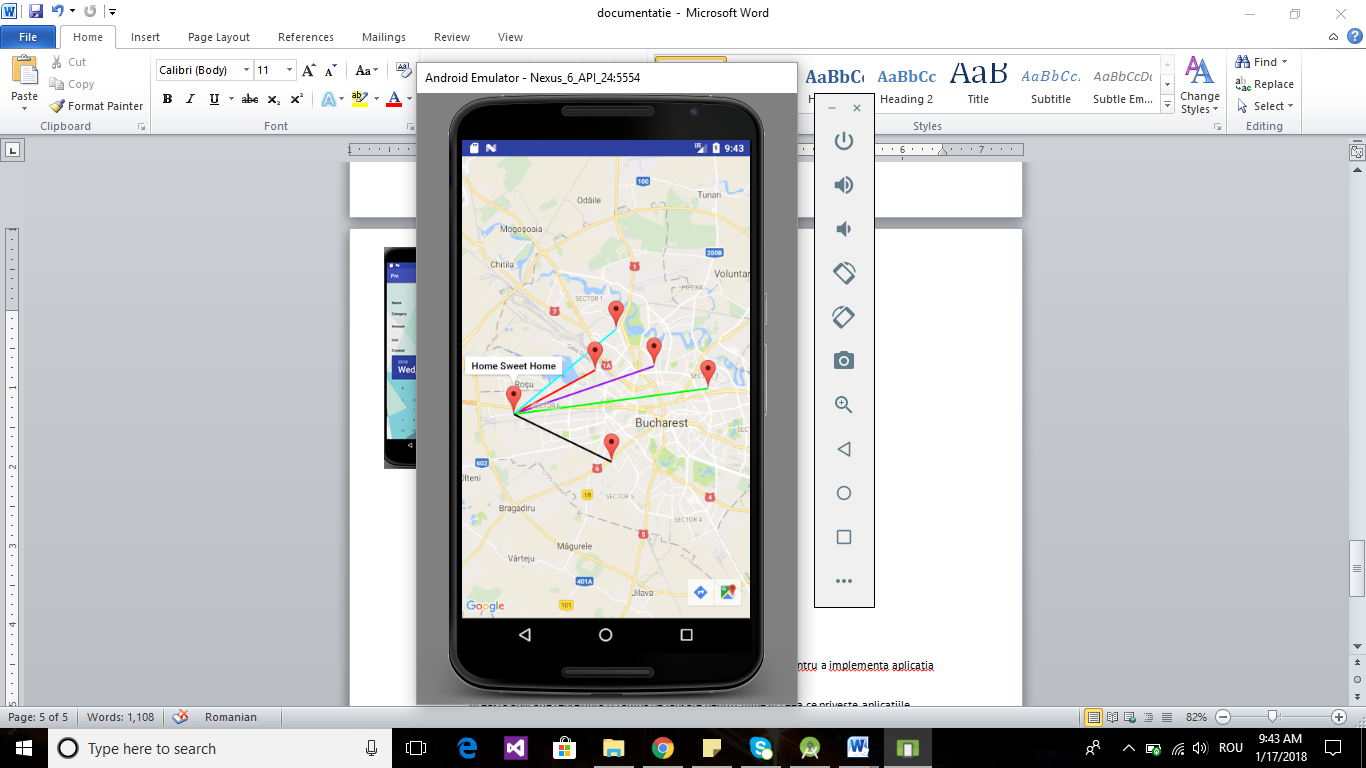
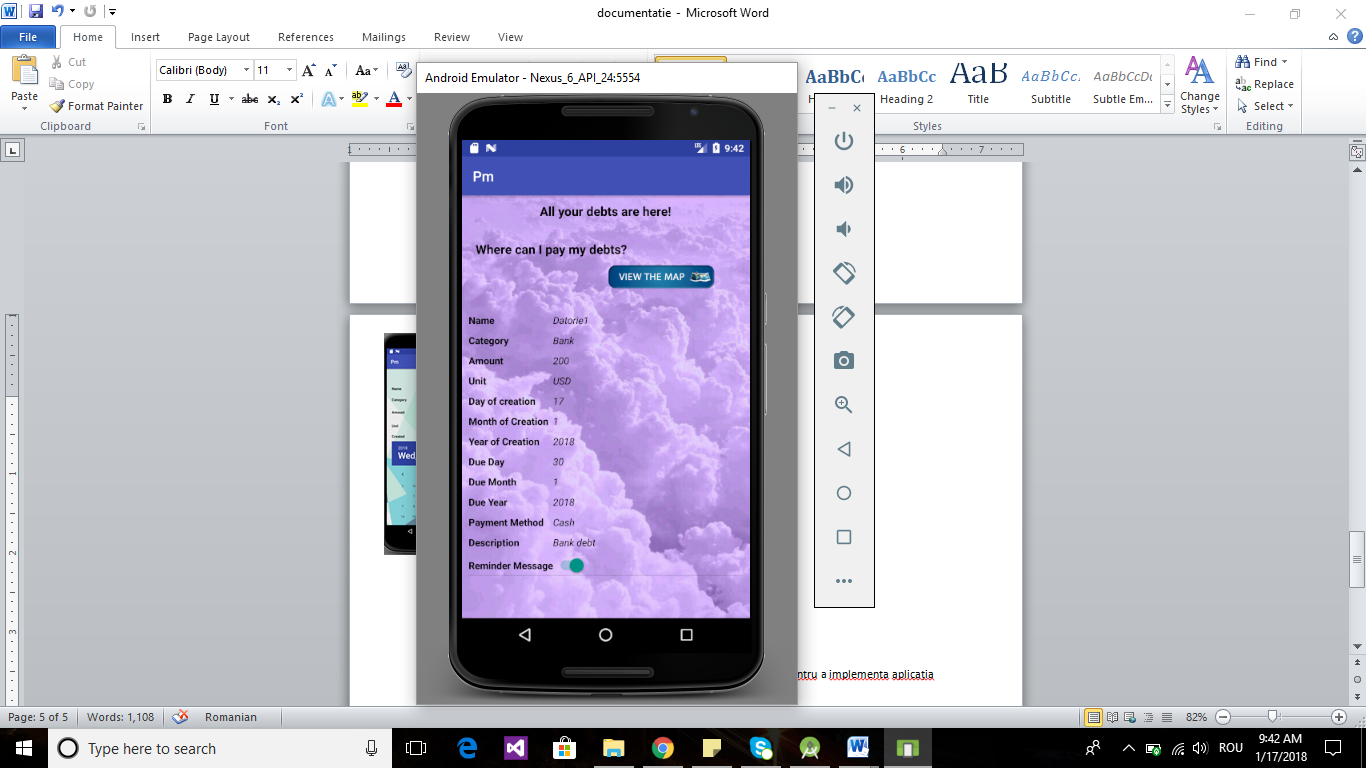




Datele introduse vor fi stocate în Firebase, în secțiunea Authentification, dar vor fi vizibile și utilizatorului într-un TextBox.

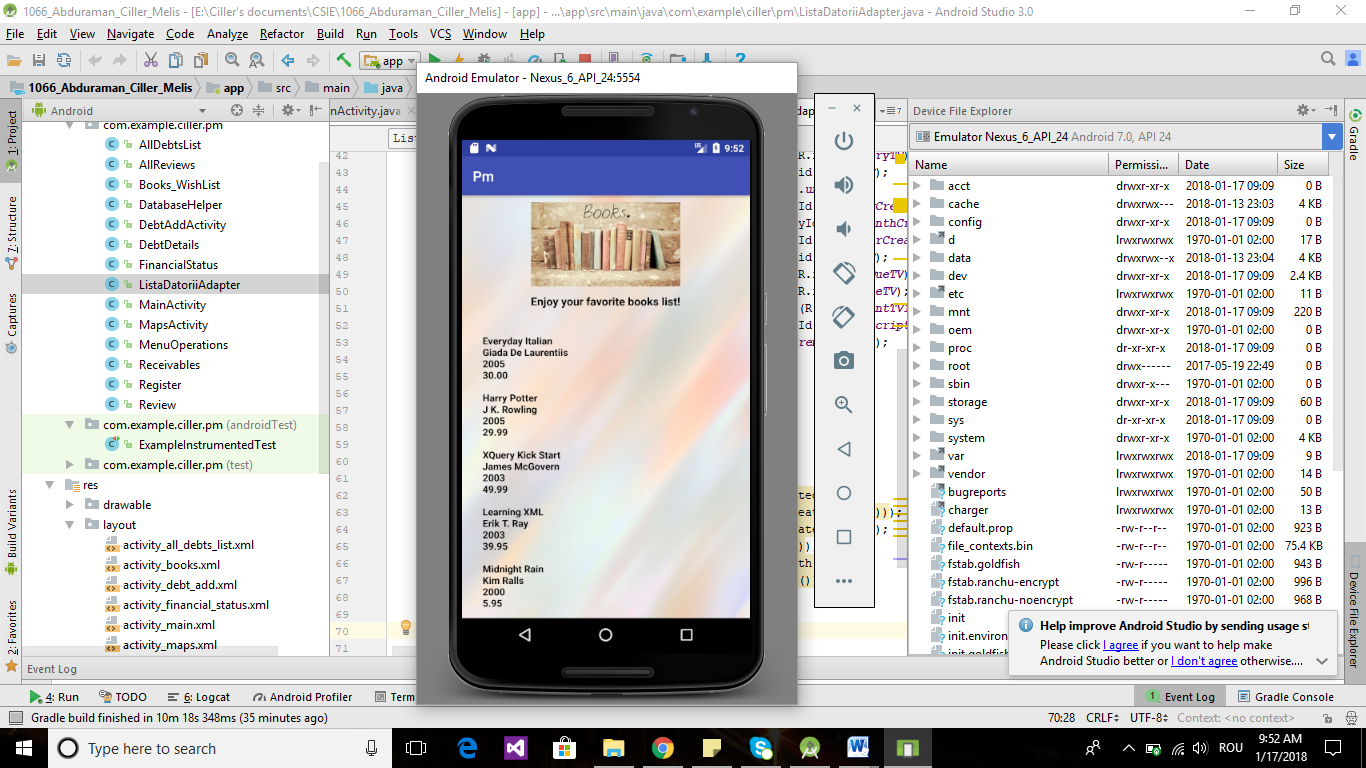


Administratorul va avea posibilitatea să reseteze o parolă la cererea utilizatorului și să ștearga conturi.



Datele introduse în pagina de Add Debt vor fi preluate din EditText-uri prin metode precum getText().toString(), getCheckedItem().toString(), getSelectedItem().toString() și vor fi stocate într-o listă ce va fi afișată într-o nouă activitate prin ajutorul unui adapter personalizat ListaDatoriiAdapter care va identifica textbox-urile prin metoda findViewById și le va completa cu datele utilizatorului prin metoda setText.

Am folosit și elemente precum Spinner(variantele posibile au fost stocate într-un array în subfolderul values - strings), CheckBox, RadioButton(au fost puse într-un RadioGroup pentru a nu permite utilizatorului alegerea mai multor opțiuni)

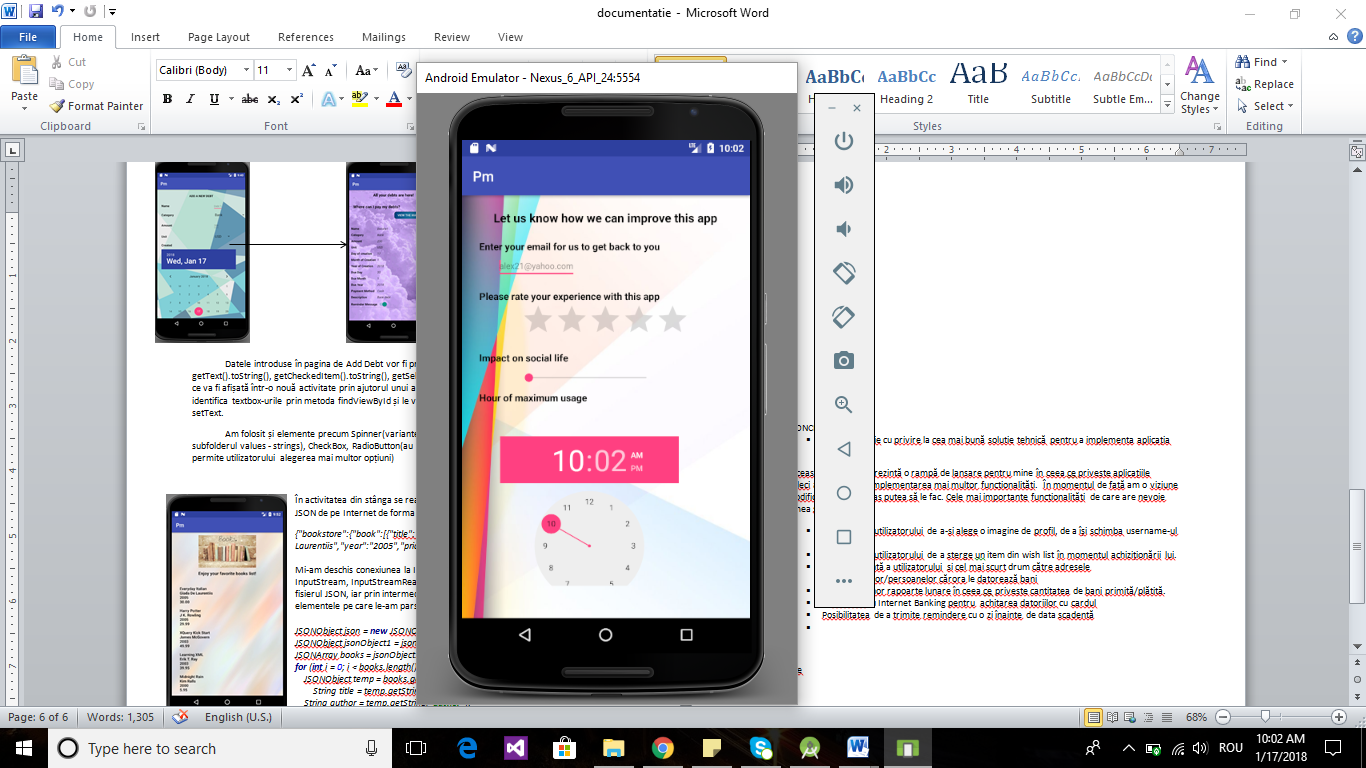
În activitatea din stânga se realizează preluarea cărților dorite dintr-un fișier JSON de pe Internet de forma :

*{"bookstore":{"book":[{"title":"Everyday Italian","author":"Giada De Laurentiis","year":"2005","price":"30.00”}*

Mi-am deschis conexiunea la Internet prin HttpURLConnection, am folosit InputStream, InputStreamReader și un BufferedReader pentru a citi datele din fisierul JSON, iar prin intermediul StringBuilder construirea unui String cu elementele pe care le-am parsat conform modelului

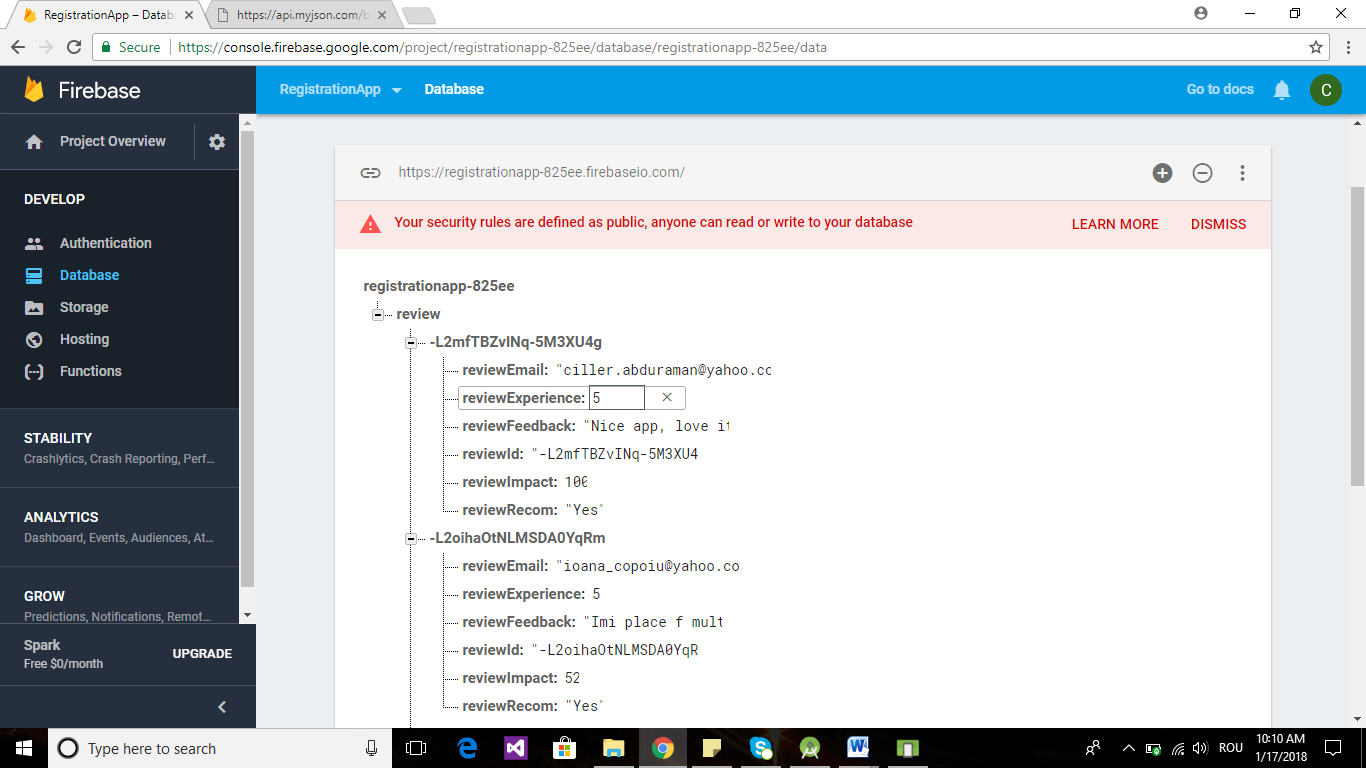
*JSONObject json =* ***new*** *JSONObject(fisier.toString());  
JSONObject jsonObject1 = json.getJSONObject(****"bookstore"****);  
JSONArray books = jsonObject1.getJSONArray(****"book"****);****for*** *(****int*** *i = 0; i < books.length(); i++) {  
 JSONObject temp = books.getJSONObject(i);  
 String title = temp.getString(****"title"****);  
 String author = temp.getString(****"author"****);  
 String year = temp.getString(****"year"****);  
 String price = temp.getString(****"price"****);*

Permisiunea la Internet am realizat-o în AndroidManifest.xml <**uses-permission android:name="android.permission.INTERNET"** />

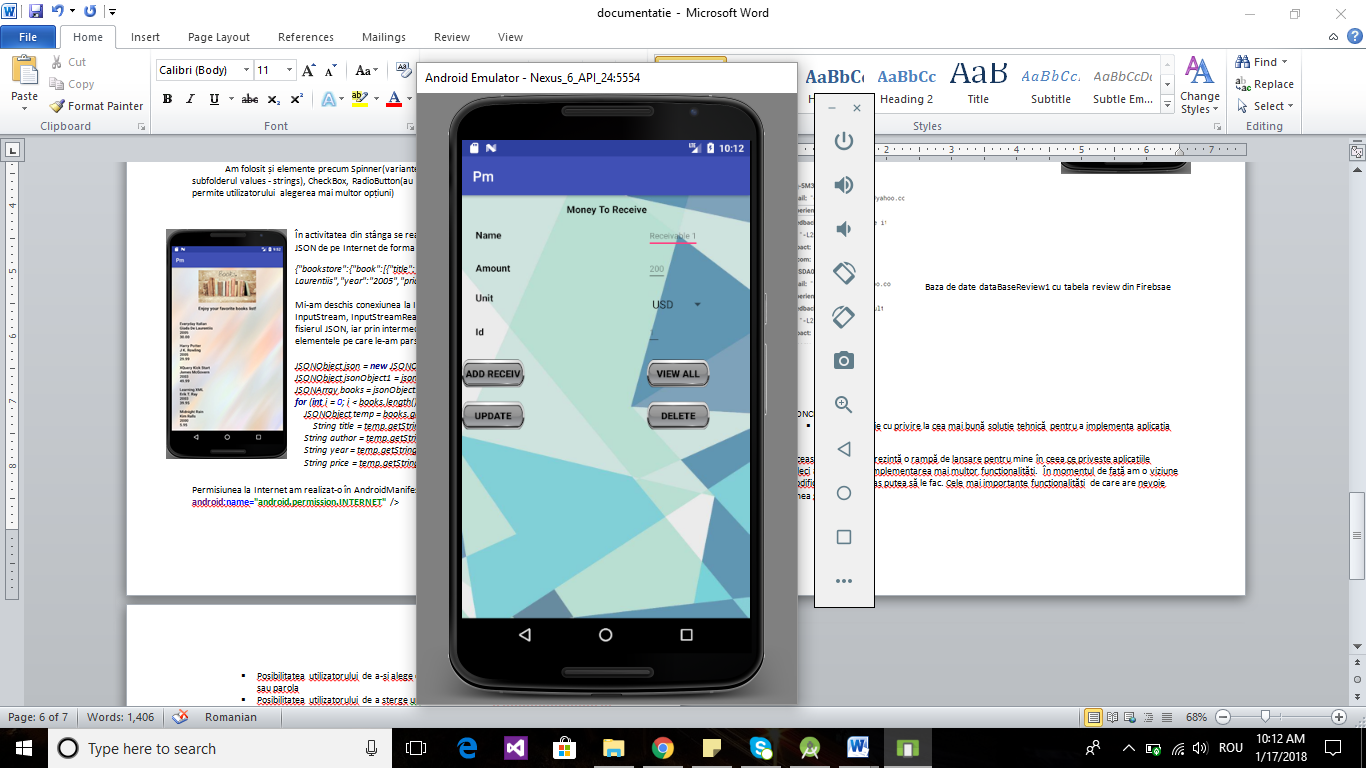
Utilizatorul poate oferi un review aplicației prin intermediul formularului din stânga ce conține următoarele elemente: RatingBar(maxim 5 stele), SeekBar, TimePicker, ToggleButton, EditText-uri ale căror informații sunt preluate și stocate în FireBase. *(DatabaseReference* ***databaseReviews1***).

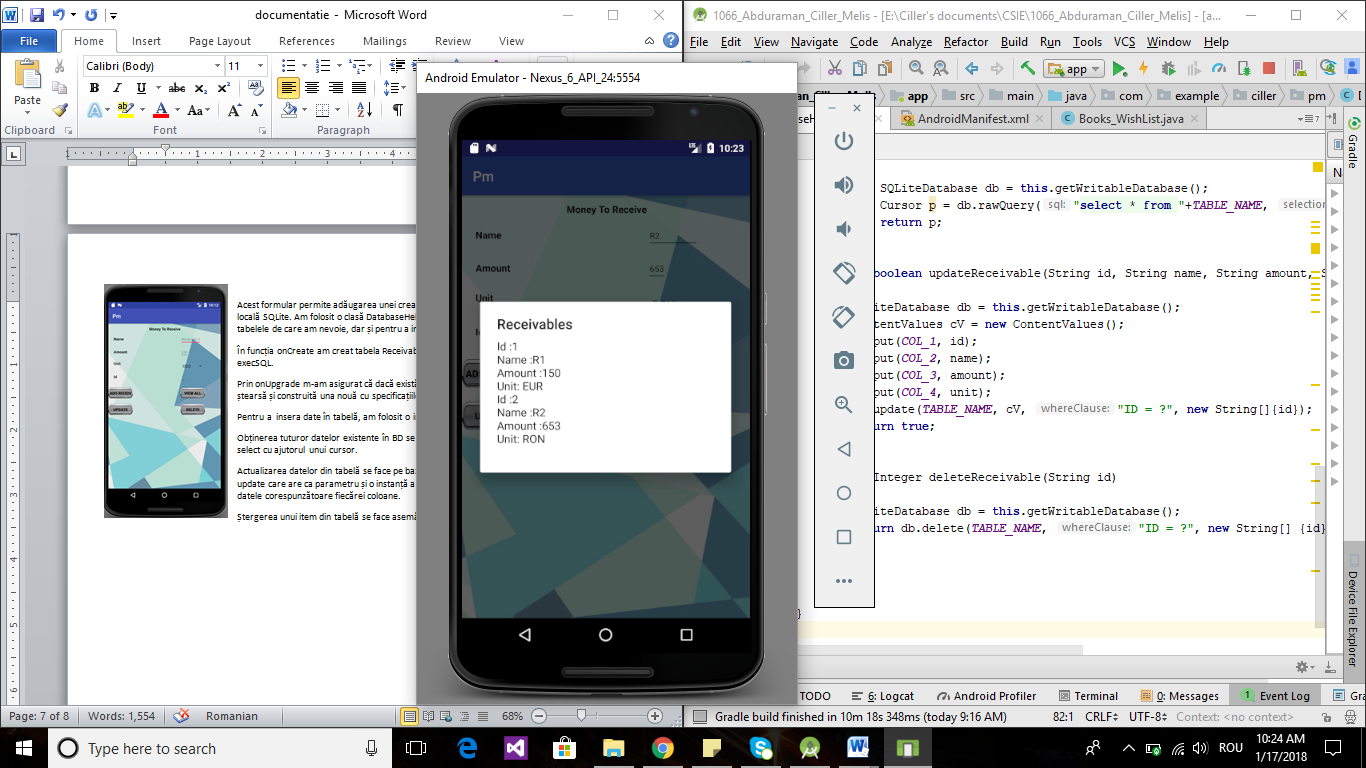
După apăsarea butonul de Send Feedback, acesta va primi un mesaj de confirmare sau de reintroducere a datelor prin metoda makeText corespunzător clasei Toast.

Pentru stocarea datelor, am folosit clasa AllReviews cu următoarele atribute: reviewId, reviewEmail, reviewExperience, reviewImpact, reviewRecom, reviewFeedback. În această clasă am implementat constructorul și getterii și setterii pe care îi voi utiliza pentru a extrage datele.



Baza de date dataBaseReview1 cu tabela review din Firebsae

Acest formular permite adăugarea unei creanțe ce va fi stocată în baza de date locală SQLite. Am folosit o clasă DatabaseHelper, pentru a crea baza de date și tabelele de care am nevoie, dar și pentru a implementa operațiile DML și DDl

În funcția onCreate am creat tabela Receivables\_Table cu 4 coloane prin execSQL.

Prin onUpgrade m-am asigurat că dacă există o tabelă cu același nume, va fi ștearsă și construită una nouă cu specificațiile făcute mai sus.

Pentru a insera date în tabelă, am folosit o instanță a ContentValues.

Obținerea tuturor datelor existente în BD se face printr-o interogare de tip select cu ajutorul unui cursor și afișarea acestora cu ajutorul StringBuffer și a unei funcții personalizate showMessage.

Actualizarea datelor din tabelă se face pe baza cheii unice ID și prin metoda update care are ca parametru și o instanță a ContentValues unde sunt stocate datele corespunzătoare fiecărei coloane.

Ștergerea unui item din tabelă se face asemănător actualizării acesteia.

Fiecărui buton îi este asociată metoda setOnClickListener.

*public void deleteReceivable()  
{  
 deleteReceivButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Integer randuriSterse = myDatabase.deleteReceivable(etId.getText().toString());  
 if (randuriSterse > 0)  
 Toast.makeText(Receivables.this, "Receivable removed", Toast.LENGTH\_LONG).show();  
 else  
 Toast.makeText(Receivables.this, "Error", Toast.LENGTH\_LONG).show();  
 }  
 });  
}*

* **Probleme/avantaje identificate la implementare pentru platforma Android**

Avantaje :

* Mediul de programare Android Studio are o interfață intuitivă, cu un aspect plăcut.
* Există numeroase tutoriale și materiale care mi-au venit în ajutor în momentul în care am avut erori sau nu am știut sa folosesc opțiunile pe care Android Studio mi le pune la dispoziție.
* Limbajul de programare Java îmi este mai cunoscut și pot implementa mai ușor diverse funcționalități în mai puține linii de cod, datorită bibliotecilor și librăriilor existente.

Probleme:

* Emulatorul : Mi-am instalat pe parcurs mai multe emulatoare cu diverse versiuni de Android pentru a avea acces la Google Maps
* Fiind o fană iOS, nu am avut posibilitatea de a îmi testa aplicația pe un device și am pierdut foarte mult timp până ce îmi pornea emulatorul de fiecare dată când făceam o modificare sau implementam o nouă funcționalitate.

1. **CONCLUZII**

* Opinie proprie cu privire la cea mai bună soluție tehnică pentru a implementa aplicația respectivă

Această aplicație reprezintă o rampă de lansare pentru mine în ceea ce privește aplicațiile Android, deci are nevoie de implementarea mai multor funcționalități. În momentul de față am o viziune asupra modificărilor pe care aș putea să le fac. Cele mai importante funcționalități de care are nevoie aplicația mea sunt:

* Posibilitatea utilizatorului de a-și alege o imagine de profil, de a își schimba username-ul sau parola
* Posibilitatea utilizatorului de a sterge un item din wish list în momentul achiziționării lui.
* Locația curentă a utilizatorului și cel mai scurt drum către adresele întreprinderilor/persoanelor cărora le datorează bani
* Realizarea unor rapoarte lunare în ceea ce privește cantitatea de bani primită/plătită.
* O asociere cu Internet Banking pentru achitarea datoriilor cu cardul
* Posibilitatea de a trimite remindere cu o zi înainte de data scadentă
* Realizarea unor statistici și grafice privind datoriile și creanțele pe o anumită perioadă sau pe alte criterii

Bibliografie

* <https://stackoverflow.com/questions/40268805/how-to-update-google-play-service-on-emulator-android-studio/40375645#40375645>
* [www.pdm.ase.ro](http://www.pdm.ase.ro)
* <https://www.androidauthority.com/android-studio-tutorial-beginners-637572/>

Anexe