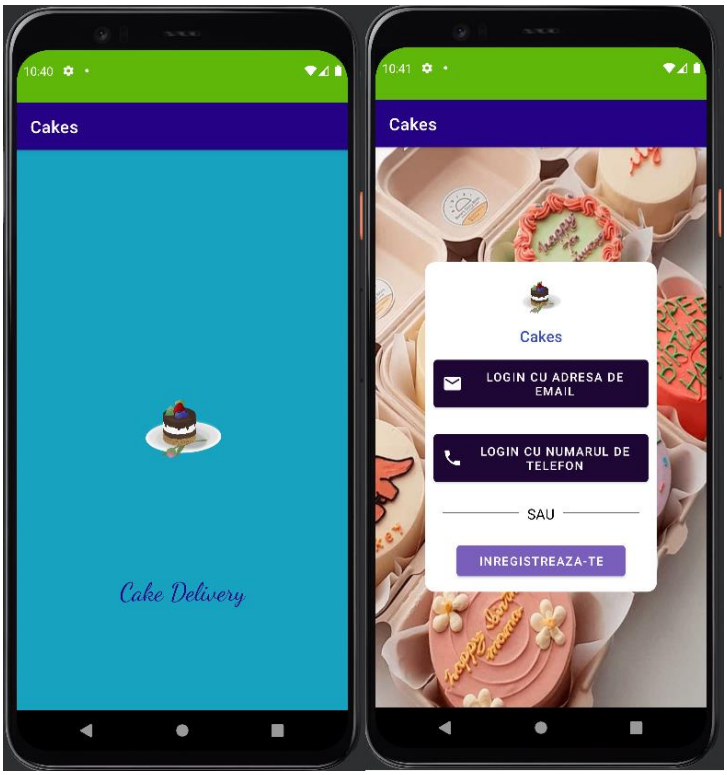
PROIECT MDS – *Cake Delivery*



Boghiu Alexandra-Adriana
Cristea Ștefania-Alexandra
Lăcătuș Cătălin-Petru
Neagu Anastasia-Elena
Sora Andreea-Ioana
Grupa 234

Cuprins

1. Descriere proiect.....3
2. Prezentarea aplicației.....3
3. Gestionarea datelor.....9
4. User stories.....9
5. Design și arhitectură.....10
6. Source control.....11
7. Bug reporting.....13
8. Build tool.....16
9. Refactoring și code standards.....18
10. Design patterns.....19
11. Concluzii.....20

1. Descriere proiect

Proiectul constă într-o aplicație de cake delivery. Clientul are opțiunea de a comanda torturi pregătite din rețete proprii, realizate de cofetari de excepție, sau de a-și alege în totalitate ingredientele dorite. De asemenea, aplicația oferă posibilitatea de a adăuga comenzii tale platouri de prajituri, șampanie și lumânări. Comenzile pot fi livrate la domiciliu sau ridicate personal la locația noastră.

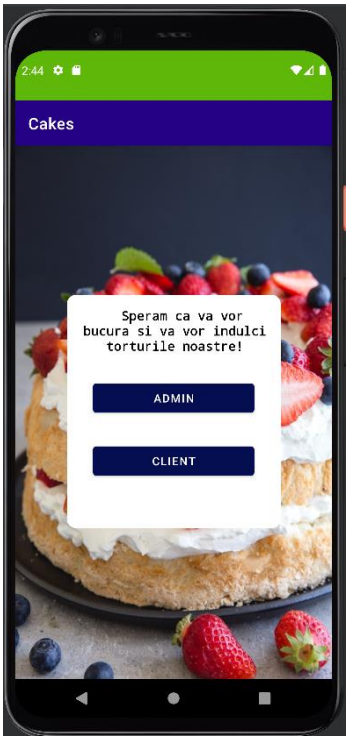
2. Prezentarea aplicației

Aplicația noastră a fost dezvoltată folosind Android Studio (limbajul Java). Android Studio este bazat pe software-ul IntelliJ IDEA de la JetBrains, este instrumentul oficial de dezvoltare a aplicațiilor Android. Acest mediu de dezvoltare este disponibil pentru Windows, OS X și Linux.

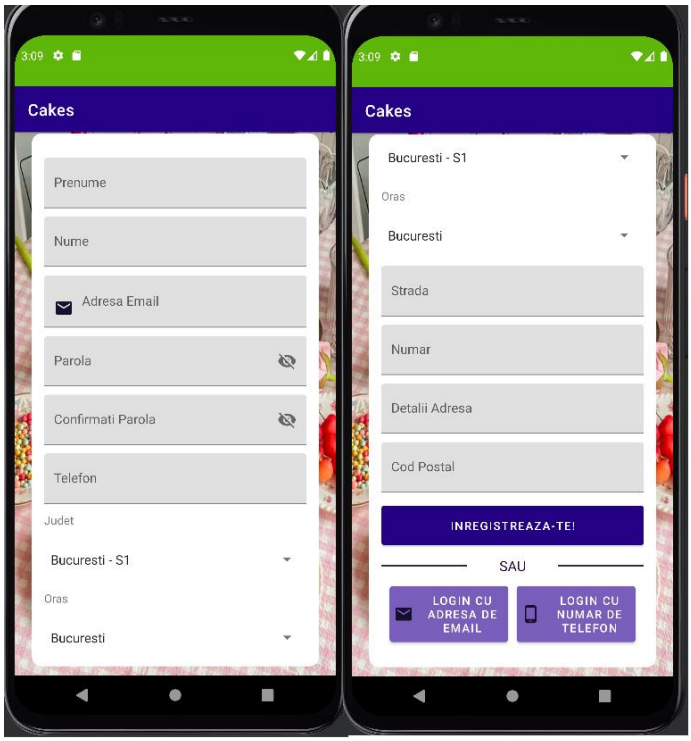
În continuare vă prezentăm funcționalitățile aplicației noastre:

- Inițial, vom avea un meniu în care putem alege dacă ne logăm cu adresa de email, numărul de telefon sau ne înregistrăm:

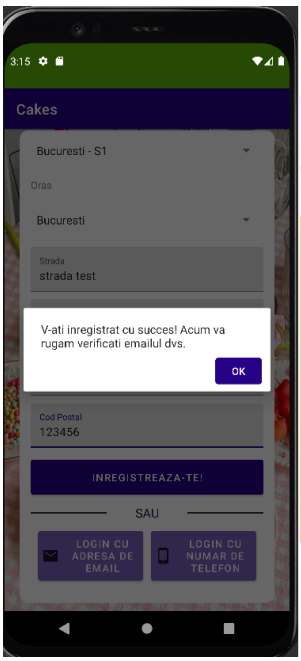
- Indiferent de opțiunea aleasă, vom fi puși să alegem dacă acțiunea are loc pentru admin sau pentru client:

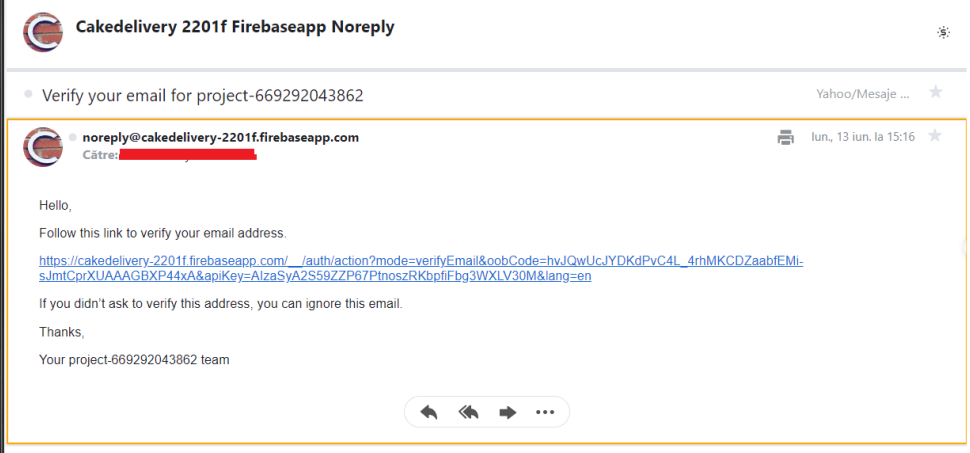


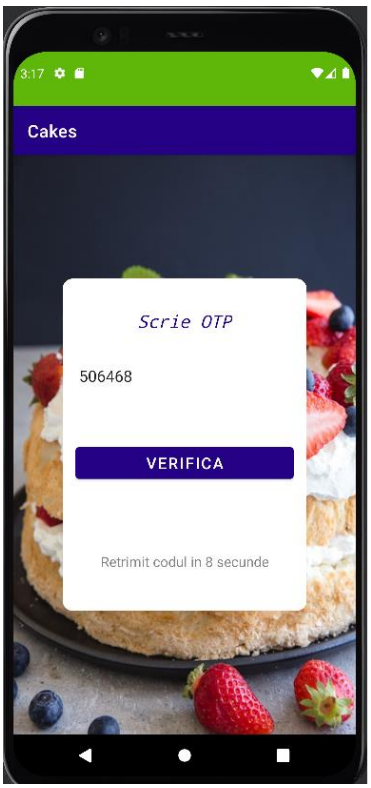
- Pentru înregistrare, vom completa o serie de câmpuri care trebuie să respecte anumite reguli precum: email valid, parolă de minim 8 caractere cu minim o literă și o cifră, numele și prenumele nu poate conține cifre etc:

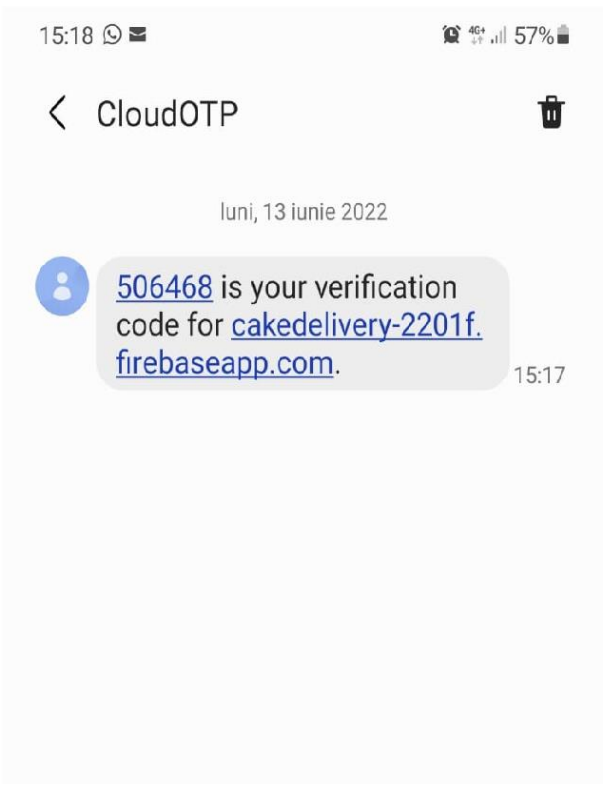


- După înregistrare, user-ul va primi mail pentru confirmarea adresei de email și mesaj pe telefon pentru confirmarea numărului de telefon:

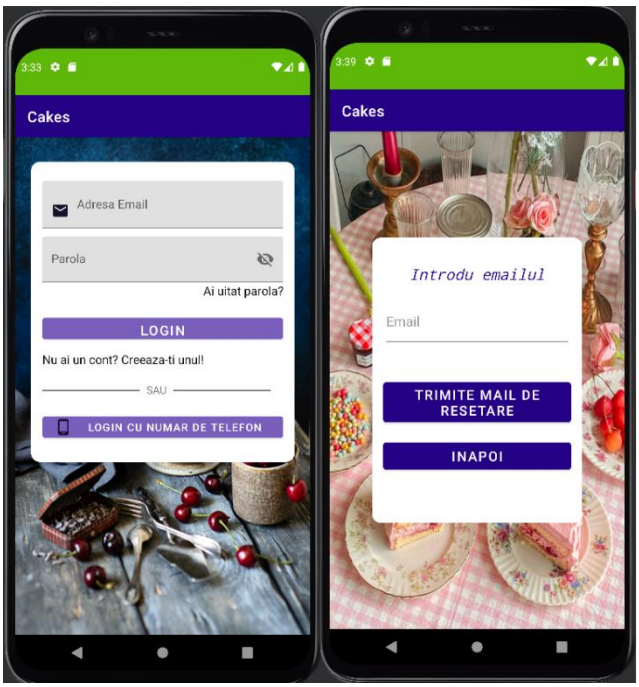


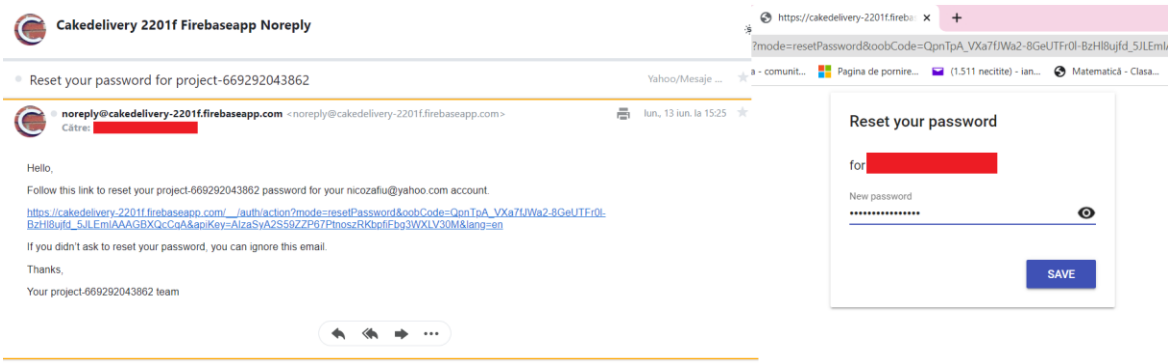




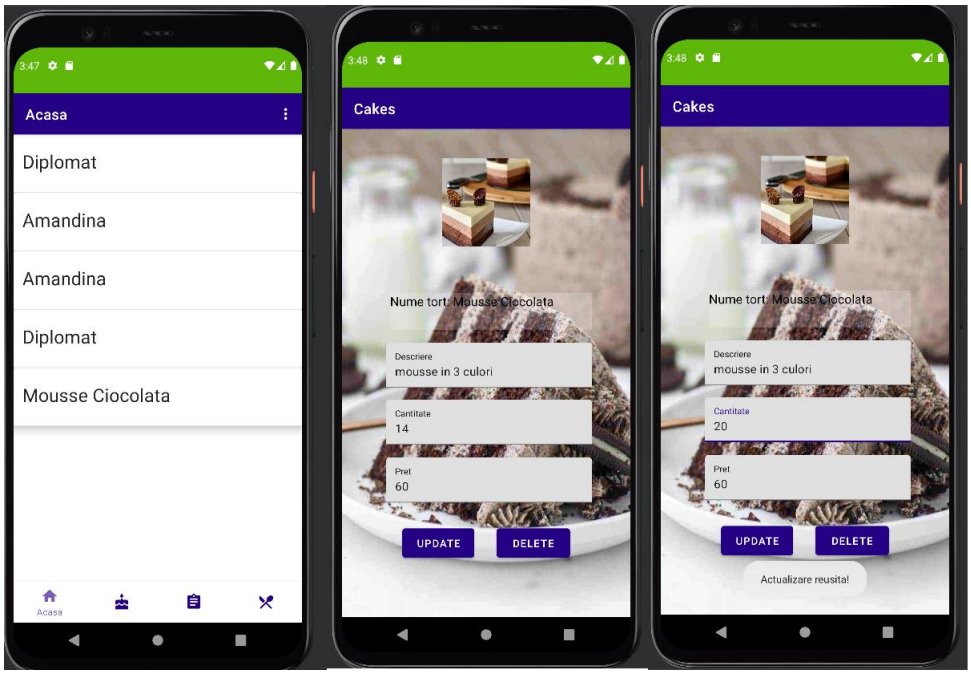


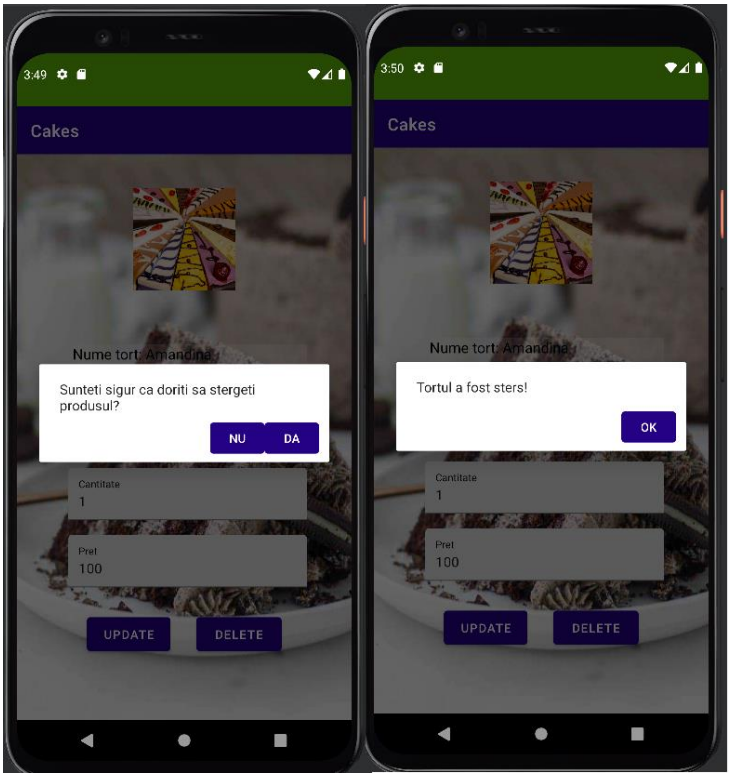
- În continuare ne putem loga. Vom demonstra și opțiunea de „Ai uitat parola?”



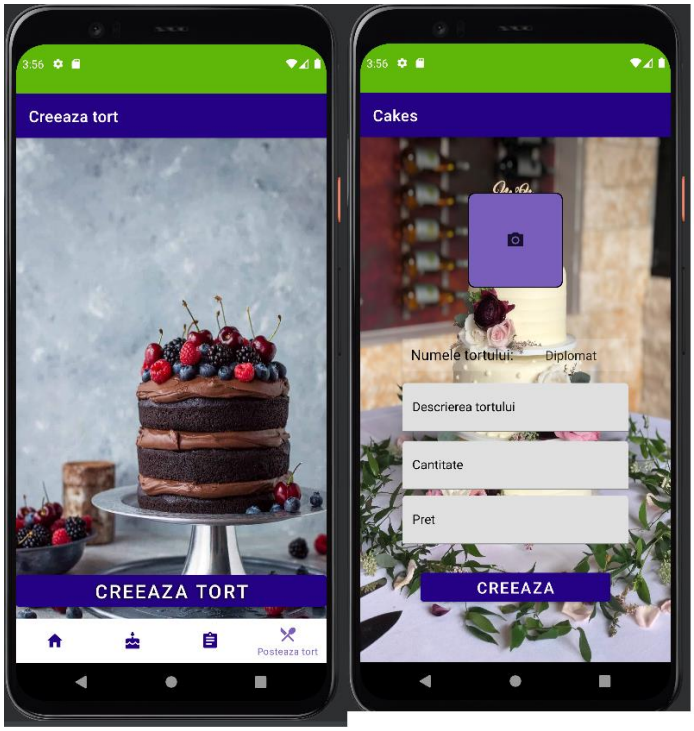


- În contul adminului, pe prima pagină vor apărea torturile create de acesta, fiind disponibile opțiunile de update și delete pe un tort selectat:

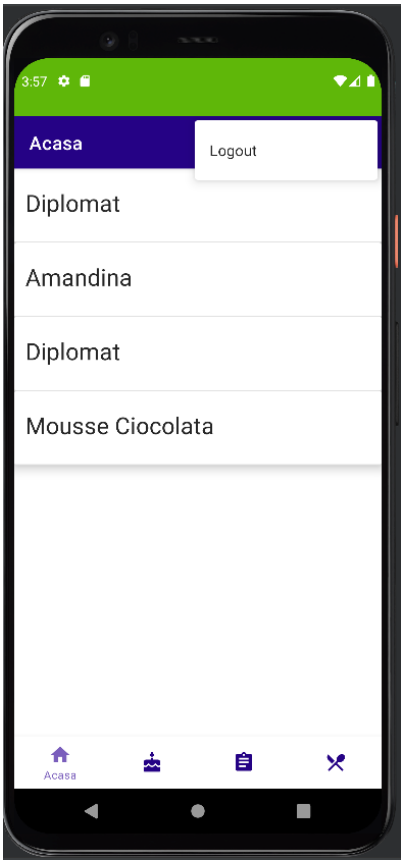




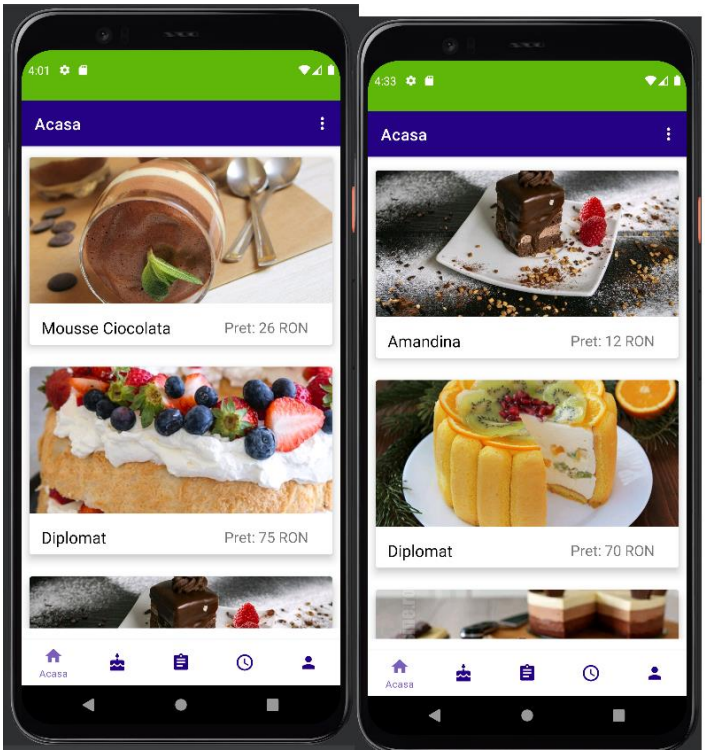
- Adminul poate să creeze un tort nou:



- Opțiunea de logout se află în colțul din dreapta sus:

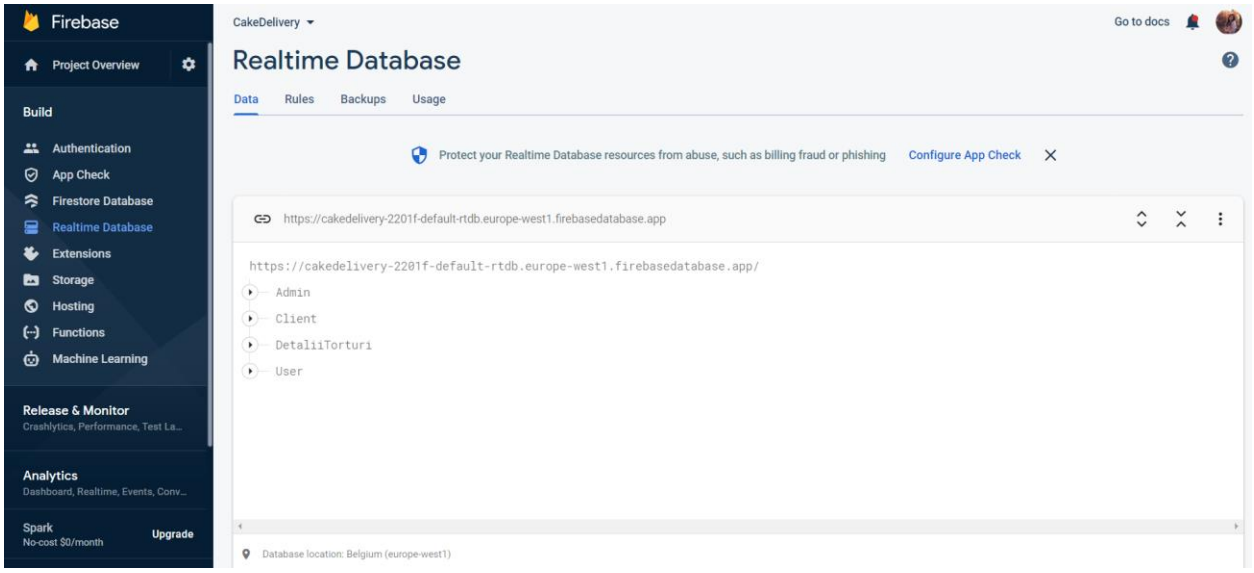


- Pe contul clientului, în pagina Acasa vor apărea torturile postate de către toți administratorii:



3. Gestionarea datelor

Pentru stocarea datelor am utilizat Firebase. Firebase Realtime Database este o bază de date în cloud NoSQL care este utilizată pentru a stoca și sincroniza datele. Datele din baza de date pot fi sincronizate simultan pe toate platformele, cum ar fi Android, web și IOS. Datele din baza de date sunt stocate în format JSON și se actualizează în timp real cu fiecare user conectat.



Exemplu date stocate pentru un admin:

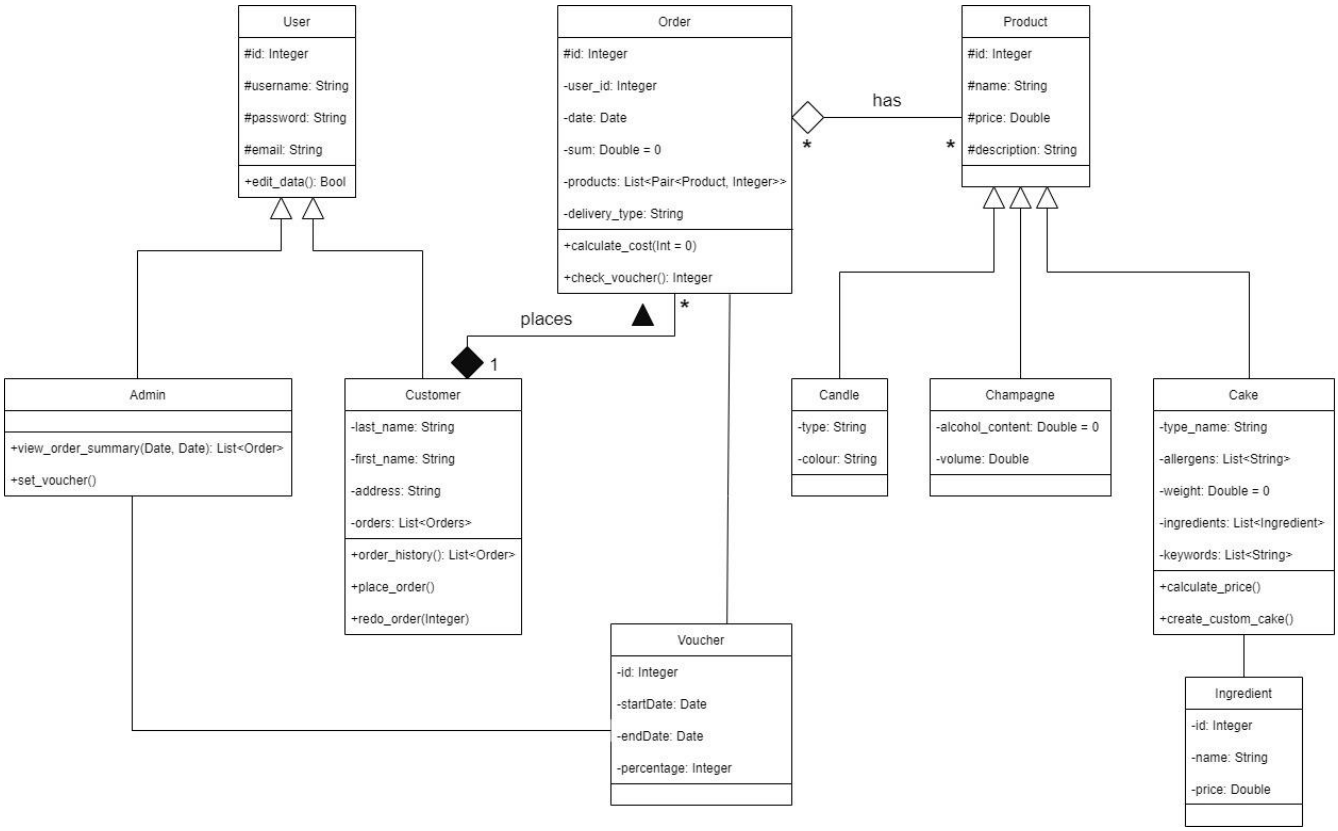
4. User stories

1. Clienții pot să își selecteze ingredientele dorite în cazul în care au anumite alergii, preferințe alimentare etc.
2. Clienții pot să își personalizeze tortul după buget și preferințe pentru a crea tortul ideal pentru ei.
3. Clienții pot adăuga, după alegerea tortului, un set de lumânări sau/și o sticlă de șampanie pentru a evita deplasările/comenzile multiple.

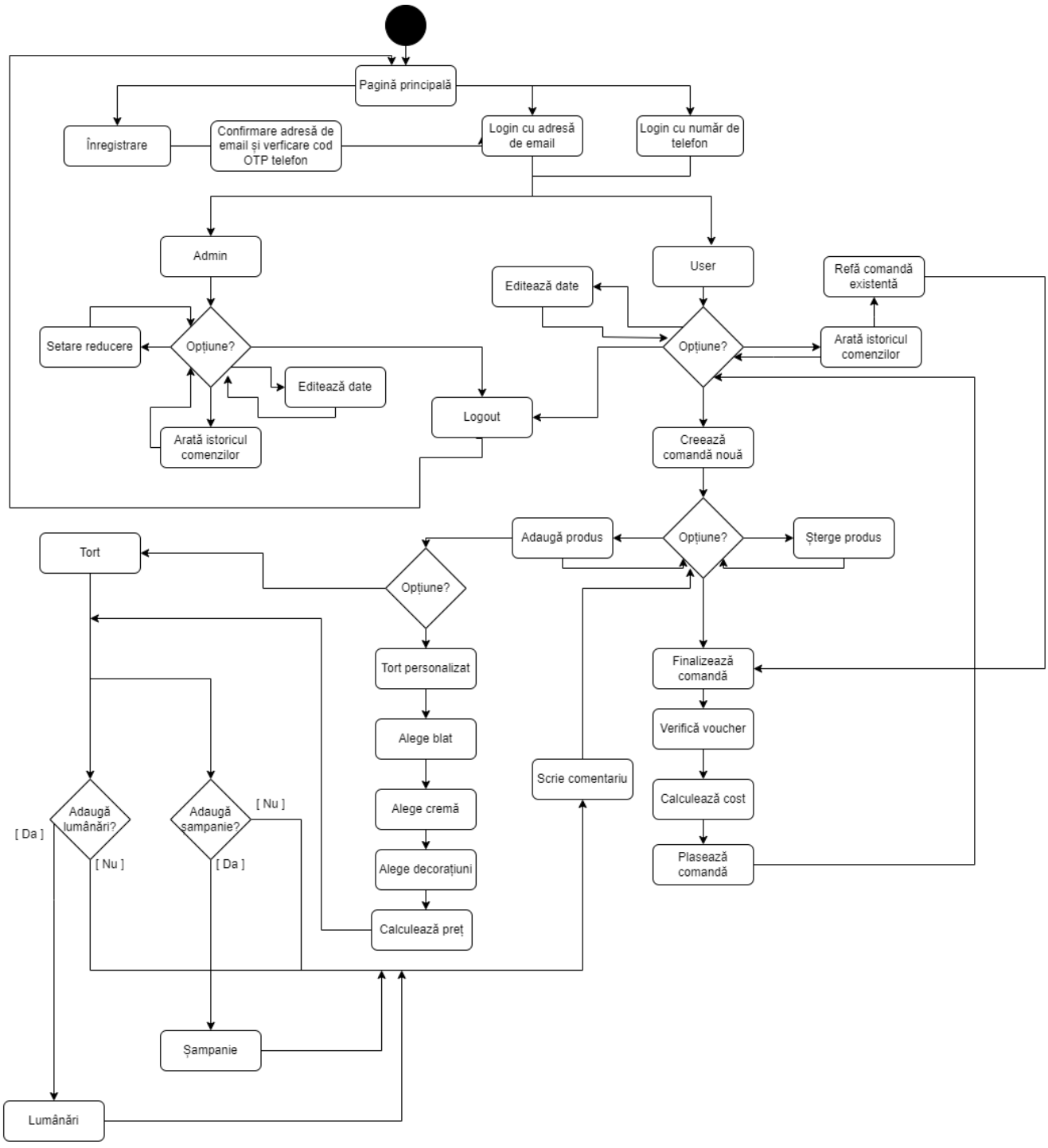
4. Un administrator poate adăuga coduri promoționale pentru a atrage mai mulți clienți.
5. Clienții își pot vizualiza un istoric și pot reveni cu o comandă identică în cazul în care dorește un tort comandat anterior.
6. Clienții primesc reduceri după un anumit număr de comenzi, fiind răsplătiți pentru fidelitatea lor.
7. Clienții pot alege ridicarea comenzii din locație sau livrarea la o adresă dată contra cost. Ridicarea din locație oferă flexibilitate clientului, fiind disponibilă de-a lungul întregii zile programate, iar livrarea este benefică pentru cei ce nu dispun de timpul necesar alocat ridicării comenzii.
8. Ca administrator, pot vizualiza comenzile plasate în aplicație pentru a avea o evidență clară asupra contabilității firmei.
9. Administratorul poate cere un raport al comenzilor filtrate după o perioadă de timp pentru a vedea dacă strategiile de marketing aplicate au fost eficiente.
10. Administratorul poate seta cuvinte cheie (ciocolată, caramel, căpșuni, vegan etc) pentru a eficientiza căutarea clientului.
11. Administratorul poate aplica opțiunile de CRUD pe baza de date.

5. Design și arhiectură

- Diagrama pe clase:

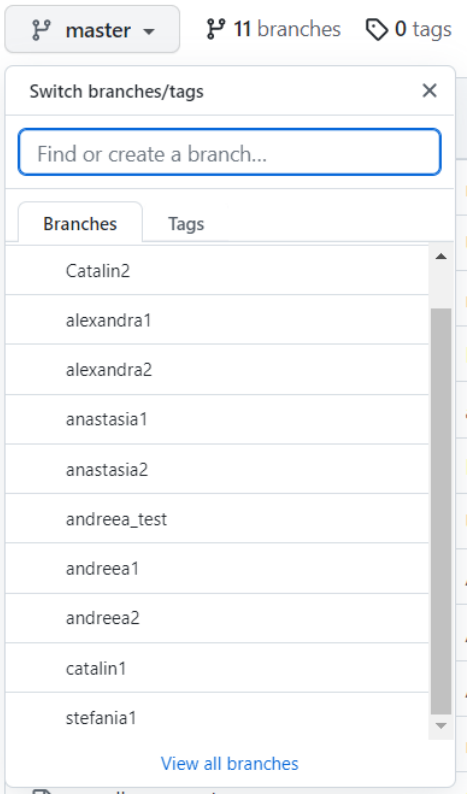


➢ Diagrama interacțiunii utilizatorului cu aplicația:

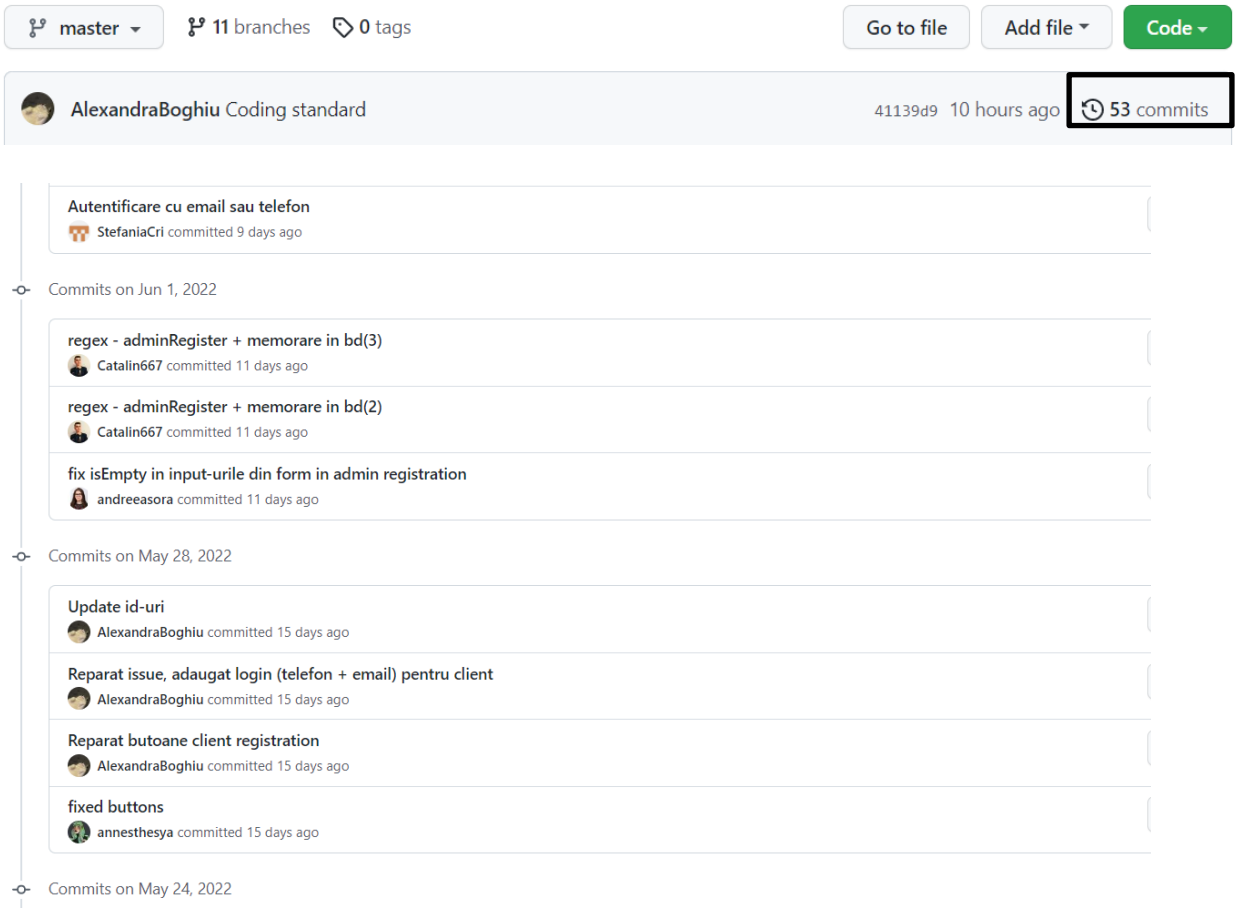


6. Source control

Pentru a ușura munca colaborativă pe cod, am folosit încă de la început Github. La fiecare pas al proiectului s-a creat un branch pentru anumite task-uri (task-uri atribuite unei anumite persoane), iar apoi, după o verificare amănunțită de către toți membrii echipei, s-a făcut merge în master.



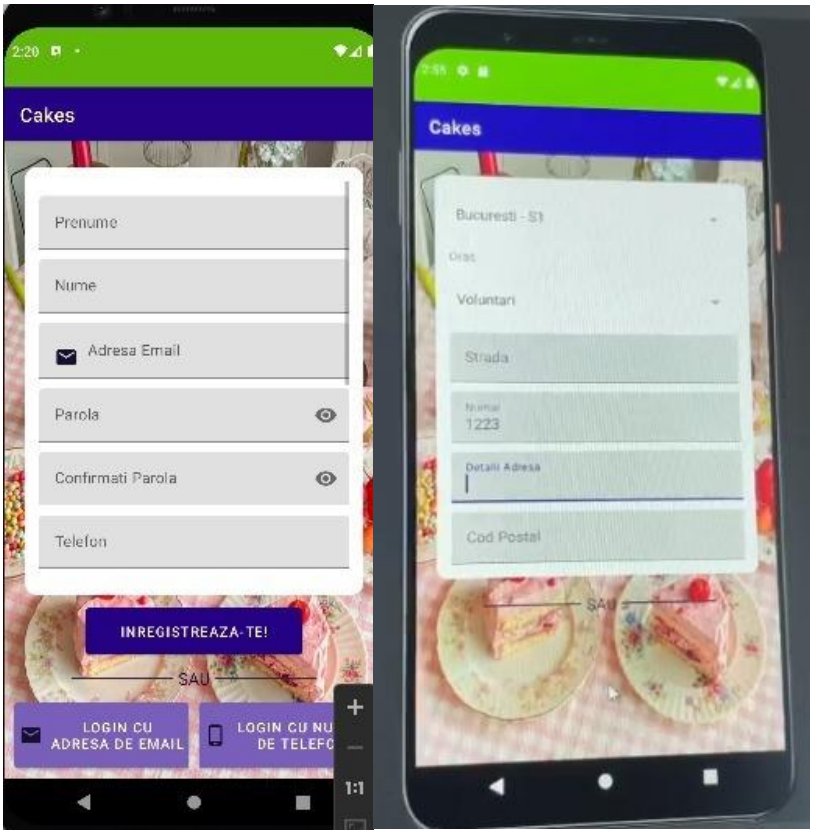
De asemenea, toată munca noastră a fost formată dintr-un număr de commit-uri destul de mare.



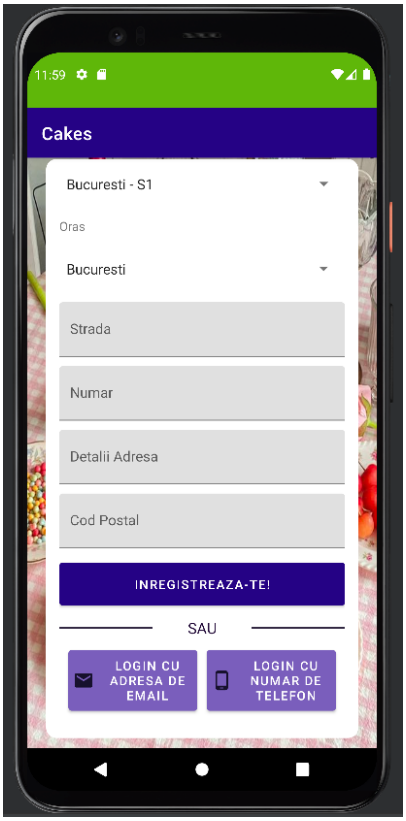
7. Bug reporting

În timpul dezvoltării aplicației, ne-am confruntat cu diverse erori. Vom enumera mai jos problemele cele mai semnificative:

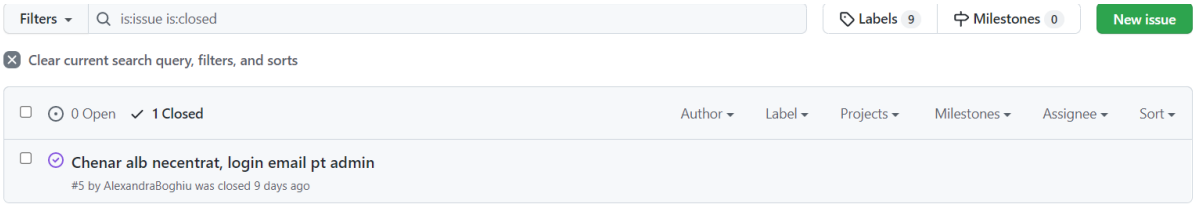
- Librăria ImageCropper, folosită de noi în procesul de creare tort, unde adminul decupa o imagine pentru a fi salvată în baza de date, nu mai era actualizată pentru versiunile noi de Android Studio, motiv pentru care a trebuit să descărcăm separat modulul cropimg (sursă: https://github.com/ArthurHub/Android-Image-Cropper).
- În meniul de înregistrare, utilizatorul nu mai avea disponibile butoanele de "Inregistreaza-te", "Login cu adresa email", "Login cu numar de telefon" după ce completa câmpurile necesare, deoarece la deschiderea tastaturii, ecranul își modifica dimensiunea, iar butoanele ajungeau în afara ecranului.

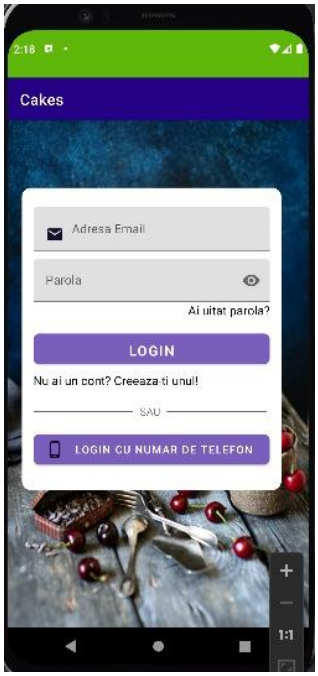


Rezolvarea a fost mutarea butoanelor în interiorul scroll-ului:



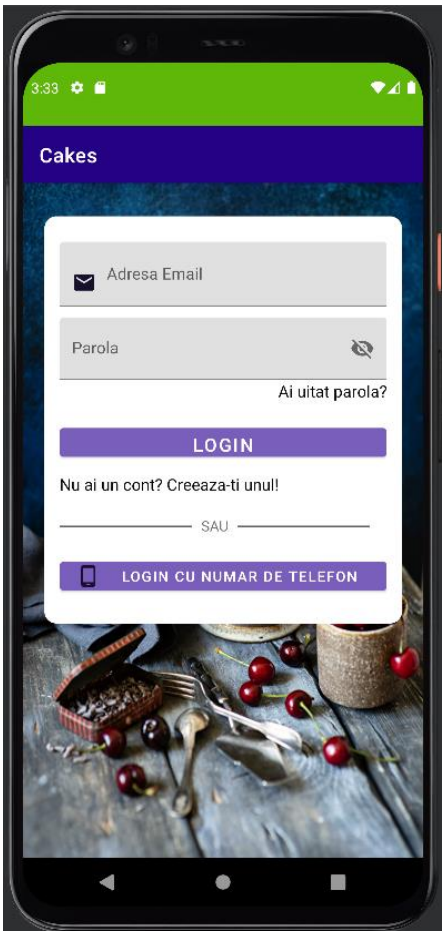
- Chenarul de la login cu adresă de email pentru admin era centrat necorespunzător:





Rezolvarea a fost ștergerea constrângerii de la linia 22 (app:layout_constraintRight_toRightOf) care făcea ca elementul să se așeze mereu în dreapta ecranului și adăugarea liniilor 20-23 care au făcut ca elementul să rămână sus, în centrul ecranului indiferent de dimensiunea ecranului:

```
22  -   app:layout_constraintRight_toRightOf="parent">
20  +   app:layout_constraintStart_toStartOf="parent"
21  +   app:layout_constraintTop_toTopOf="parent"
22  +   app:layout_constraintVertical_bias="0.125"
23  +   app:layout_goneMarginTop="0dp">
```

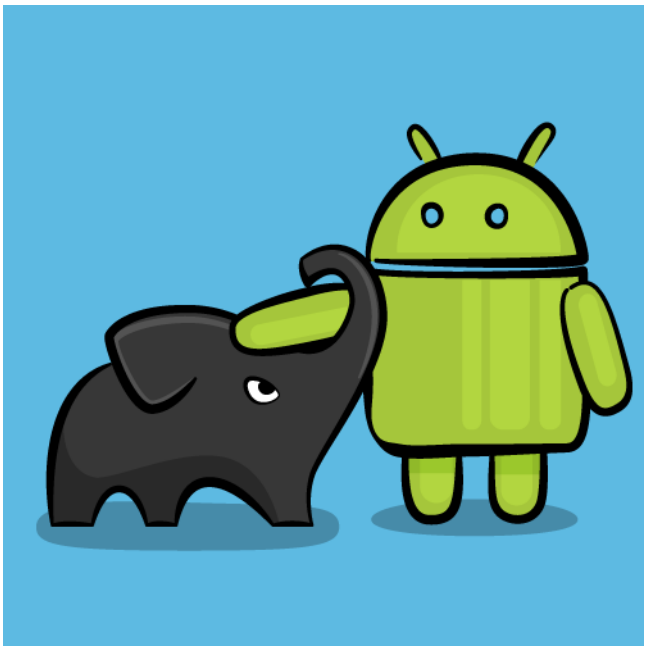


- Am întâmpinat multe probleme legate de id-urile elementelor, din neatenția noastră, fiind rezolvate rapid cu o simplă revizuire a codului.
- De asemenea, am avut parte de multe erori legate de Firebase și dependențele din build gradle, însă cu ajutorul unor căutări riguroase pe Internet, am gasit rezolvarea lor.

8. Build tool

Gradle este un build tool cunoscut pentru flexibilitatea de a construi software. Bazele acestuia sunt: Apache Ant, Apache Maven. De asemenea, acesta introduce un limbaj de script bazat pe Groovy, pentru a se detașa de scrierea bazată pe XML, folosită de Maven.

Sistemul utilizat de Android pentru a construi aplicația face următoarele lucruri: compilează resursele aplicației și codul sursă, le împachetează într-un fișier de tip APK, pe care poți să îl testezi, lansezi pe magazinul oficial (Google Play) etc. Sistemul de compilare Android Studio se bazează pe Gradle, iar pluginul Android Gradle adaugă câteva funcții specifice pentru construirea de aplicații Android. Deși pluginul Android este de obicei actualizat în pas cu Android Studio, pluginul (și restul sistemului Gradle) poate rula independent de Android Studio și poate fi actualizat separat.



Fișierul build.gradle din cadrul proiectului nostru:

```
plugins {
    id 'com.android.application'
    id 'com.google.gms.google-services'
}
```

```
android {
    compileSdk 32

    compileSdkVersion = 32
    buildToolsVersion = '32.1.0-rc1'


    defaultConfig {
        multiDexEnabled true
        applicationId "com.example.app"
        minSdk 20
        targetSdk 32
        versionCode 1
        versionName "1.0"
        vectorDrawables.useSupportLibrary = true
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

}

dependencies {
    "com.android.support:appcompat-v7:22.1.0"
//    implementation 'com.github.jesseruder:Android-Image-Cropper:2.1.3'
    'cropper'
    'com.theartofdev.edmodo:android-image-cropper:2.4.+'
    implementation 'com.google.firebase:firebase-auth:21.0.5'
    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    implementation 'com.google.firebase:firebase-analytics:21.0.0'
    implementation 'com.google.firebase:firebase-auth:21.0.4'
    implementation 'com.google.firebase:firebase-database:20.0.5'
    implementation 'com.google.firebase:firebase-storage:20.0.1'
    implementation 'com.google.firebase:firebase-messaging:23.0.4'
    implementation 'com.google.firebase:firebase-database-ktx:20.0.5'
    implementation 'com.google.firebase:firebase-firestore:24.1.2'
    implementation 'com.google.firebase:firebase-firestore-ktx:24.1.2'
    implementation 'com.google.firebase:firebase-storage-ktx:20.0.1'
    implementation 'com.google.firebase:firebase-auth-ktx:21.0.5'
    implementation 'com.android.support:multidex:1.0.3'
    implementation 'androidx.browser:browser:1.0.0'

//    implementation 'com.google.firebase:firebase-auth'
//    implementation platform('com.google.firebase:firebase-bom:30.1.0')

    implementation "androidx.navigation:navigation-fragment:2.3.0"
    implementation "androidx.navigation:navigation-ui:2.3.0"
    implementation 'androidx.swiperefreshlayout:swiperefreshlayout:1.1.0'
```

```
    implementation files('glide-3.6.0.jar')
    implementation project(path: ':cropimg')

    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
//    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

}
```

9. Refactoring și code standards

În procesul de dezvoltare al aplicației am întâlnit deseori situații în care a trebuit să refacem anumite părți din cod, marea lor majoritatea fiind probleme minore.

Un prim exemplu de refactoring a fost redenumirea tuturor fișierelor în același format, gruparea lor în pachete corespunzătoare și utilizarea coding standard-ului de la Google (https://google.github.io/styleguide/javaguide.html). Fiind un proiect de grup, fiecare a avut un stil de codare/denumire (camelCase, snake_case etc) diferit, motiv pentru care am ales să reorganizăm structura aplicației.

Un alt exemplu de refactoring a fost ridicarea gradului de complexitate al parolei pe care o alege user-ul. Inițial, se putea alege o parolă care să aibă minim 8 caractere.


```
            }
        }
//              check password
        if (TextUtils.isEmpty(password)) {
            Password.setErrorEnabled(true);
            Password.setError("Creati o parola. Aceasta nu poate lipsi.");
        } else {
            if (password.length() < 8) {
                Password.setErrorEnabled(true);
                Password.setError("Parola dvs. este slaba. Creati o parola care are cel putin 8 caractere.");
            }
            isValidePassword = true;
        }
//              check confirm password
```

Ulterior, am impus utilizatorului să folosească o parolă de minim 8 caractere, cu cel puțin o cifra și cel putin o literă, folosind un regex.

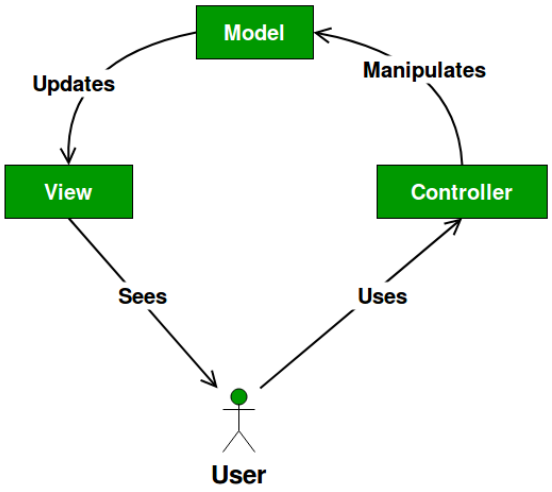
```
String passwordPattern = "^(?=.*[A-Za-z])(?=.*\\d)[A-Za-z\\d]{8,}$";
```

```
//              check password
        if (TextUtils.isEmpty(password)) {
            Password.setErrorEnabled(true);
            Password.setError("Creati o parola. Aceasta nu poate lipsi.");
        }
        else {
            if (password.matches(passwordPattern)) {
                isValidePassword = true;
            }
            else {
                Password.setErrorEnabled(true);
                Password.setError("Parola dvs. este slaba. Creati o parola
care are cel putin 8 caractere (cel putin o litera si o cifra).");
```

```
            }
        }
```

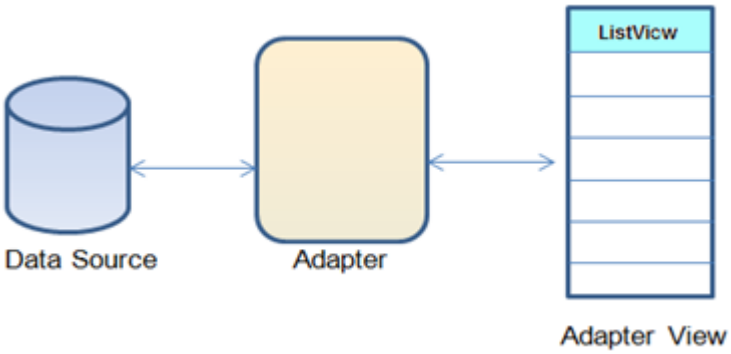
10. Design patterns

Model-view-controller (MVC) este un model arhitectural utilizat în ingineria software. În MVC, modelul conține informațiile (datele) și regulile business; view conține elemente din interfața utilizator (texte, input-uri ale formularelor etc); controller-ul gestionează comunicarea dintre model si view.



În proiectul nostru am folosit conceptul MVC în felul următor:

- View: fișierele de tip xml din app/res/layout. (exemplu: *admin_update_delete.xml*)
- Model: exemplu: *UpdateTortModel*
- Controller: Rolul controller-ului este jucat de Adaptor. Acesta face legătura între model și view. (exemplu: *AdminAcasaAdaptor*)



11. Concluzii

Dezvoltarea acestui proiect a venit cu o serie de avantaje pentru noi. În primul rând, am învățat niște tehnologii noi (Android Studio, Firebase), am învățat cum se dezvoltă o aplicație în viața reală, ne-am dezvoltat cunoștințele despre Java. În al doilea rând, faptul că a fost un proiect în echipă ne-a ajutat foarte mult deoarece am învățat să colaborăm pe cod, să folosim mult mai bine Github și să ne apropiem de o experiență din industrie.