Academia de Studii Economice din București
Facultatea de Cibernetică, Statistică și Informatică Economică
Specializarea Informatică Economică

Aplicație suport pentru studenții CSIE

Lucrare de licență

Coordonator
Prof. univ. dr. Cristian-Eugen CIUREA

Absolvent
Claudia Maria CIUREA

București 2023

Declarație privind originalitatea conținutului și asumarea răspunderii

Prin prezenta declar că rezultatele prezentate în această lucrare sunt în întregime rezultatul propriei mele creații cu excepția cazului în care se fac referiri la rezultatele altor autori. Confirm faptul că orice material folosit din alte surse (reviste, cărți și site-uri de internet) este în mod clar referit în lucrare și este indicat în lista de referințe bibliografice.

Conținut

Capitolul 1 Introducere

Obiectivul principal al lucrării este de a oferi studenților de la Facultatea de Cibernetică, Statistică și Informatică economică (CSIE) o aplicație mobilă interactivă prin care să își poată gestiona notele și obiectivele pe care aceștia doresc să le îndeplinească de-a lungul unui semestru universitar pentru a urmări progresul și a evidenția rezultatele. Există de asemenea și opțiunea de se personaliza orarul pentru a se corela informațiile introduse în aplicație cu materiile din orar.

În ultimele decenii tehnologia a evoluat, iar o dată cu ea calitatea și stilul de viață al unui individ. Metodele tradiționale de predare și învățare s-au schimbat și ele, nu se mai pliază pe nevoile societății actuale. Accesul la orice informație este acum doar la un click distanță, însă acest lucru poate fi atât un beneficiu, cât și un dezavantaj pentru un tânăr. Datorită faptului că există atât de multă informație în mediul online, mai multă decât cineva o poate asimila, este impetuos necesar să existe o îndrumare reprezentată de una sau mai multe entități (școală, universitate, profesori, părinți etc) care să ghideze tânărul în marea de informații care se așterne în fața sa. Rolul principal al profesorilor este acela de a ajuta studenții să-și îmbunătățească performanța academică, iar tehnologia este modalitatea prin care aceștia pot reuși să își dezvolte abilitățile tehnice necesare unui viitor loc de muncă. Tehnologia oferă acces în permanență la resurse educaționale, iar datorită pandemiei s-a realizat o modificare uluitoare la nivel mondial și anume implementarea orelor online [1] . O dată cu această schimbare a devenit evident că platformele educaționale și aplicațiile sunt necesare pentru o bună colaborare și interacțiune între profesor și student.

Astfel, s-a introdus noțiunea de e-learning, precum și aplicații și platforme educaționale, al căror obiectiv este îndrumarea studenților ghidați de profesor. Aceste platforme sunt utile prin faptul că înglobează materialele necesare pentru studiu, se vizualizează rapid temele și obiectivele ce trebuie realizate pentru o materie și este vizibil în permanență progresul studentului, prin vizualizarea notelor și a feedback-ului

din partea profesorilor. Informații precum progresul academic, notele, grafice create pe baza notelor, obiective de introdus, toate acestea pot fi ușor de urmărit pe o aplicație mobilă, întrucât toată lumea deține în prezent un telefon și are acces la internet.

În lucrarea de licență se va prezenta o aplicație android similară unei platforme educaționale, însă accentul va fi pus pe comunitățile de studenți, scopul principal al aplicației fiind acela de a sprijini studenții ca pe parcursul unui semestru să-și organizeze timpul și să-și țină evidența notelor și a progresului realizat.

Proiectul este împărțit în 6 capitole, primul fiind introducerea.

Al doilea capitol intitulat “Scopul, Necesitatea și Actualitatea dezvoltării aplicației mobile“ prezentă în detaliu tema care este abordată în lucrare, necesitatea construirii aplicației mobile și utilitățile pe care aceasta le are.

În al treilea capitol dezbate pe larg tehnologiile folosite în aplicație, menționându-se mediul de dezvoltare, limbajul în care este scrisă aplicația, implementarea unui API pentru crearea graficelor, serviciul de autentificare oferit de Firebase și utilitatea conectării la o bază de date la distanță, Firebase.

Al patrulea capitol are în componența sa descrierea arhitecturii aplicației mobile, a fluxului de activități, a modulelor în care este structurată aplicația și descrierea interfețelor.

Al cincilea capitol conține descrierea schemei cazurilor de utilizare pentru principalele funcții ale aplicației și prezentarea modului de implementare a claselor și a celor mai interesante funcții. De asemenea este prezentată și structura bazei de date, precum și modul în care este accesată.

Ultimul capitol îl reprezintă concluziile trase în urma finalizării lucrării, limitările aplicației și cauza apariției acestora. De asemenea, se propun funcționalități noi de dezvoltare a aplicației.

Capitolul 2 Scopul, Necesitatea și Actualitatea dezvoltării aplicației

2.1 Scopul construirii unei astfel de aplicații mobile

În această lucrare se va prezenta o aplicație Android similară unei aplicații de gestionare a notelor, materiilor și a progresului academic semestrial. Accentul se va pune pe personalizarea informațiilor introduse de către utilizator într-un mod estetic plăcut, având toate funcționalitățile necesare grupate într-un singur loc.

Aplicația suport pentru studenții CSIE face parte dintr-un sistem informațional încadrat în aria educației, respectiv de organizare a activității studenților, iar scopul acesteia este de a fi utilizată în cadrul facultății CSIE, oferind suport studenților prin funcționalitățile pe care le oferă.

2.2 Necesitatea utilizării aplicației

Necesitatea aplicației este dată de revoluția tehnologică pe care o traversăm, în care tehnicile clasice de învățare și menținere a comunicării între persoanele din cadrul unei comunități s-au schimbat. Odată cu apariția pandemiei, sistemul educațional a fost forțat să se mute complet în mediul online. Această modificare majoră în modul de predare și furnizare a informațiilor a venit cu avantaje și dezavantaje. Chiar dacă această schimbare a privat studenții și profesorii de interacțiunile sociale, a venit totuși cu un mare avantaj: sistematizarea și organizarea orelor în așa fel încât tot ceea ce este necesar unui student să se găsească în mediul online. Orele de curs, susținerea examenelor, predările de proiecte și prezentarea lucrării de licență, toate s-au mutat în mediul online. Mai mult decât atât, biblioteca a pus la dispoziția studenților exemplare de cărți în mediul virtual.

2.3 Actualitatea unei astfel de aplicații android

În prezent, chiar dacă totul a revenit la normal, se remarcă faptul că educația în mediul online rămâne în continuare actuală și utilă. Această modalitate de interacțiune a creat noi oportunități și a schimbat modul în care oamenii văd, aplică tehnicile de educație și se raportează la educație la modul general. De asemenea, în prezent se pune accent pe consolidarea unor abilitați noi, care în mod evident vor fi de necesare pentru locurile de munca din viitor, precum gândirea critică, creativitatea, adaptivitatea, managementul timpului, abilitați de lucru cu calculatorul și cu diferite aplicații software.

Se poate privi acest sistem al educației ca un *sistem adaptiv complex* caracterizat de *adaptivitate*, *coevoluție* și *conectivitate*. Aceste caracteristici s-au remarcat pe parcursul ultimilor doi ani. Abilitatea de adaptivitate a actorilor sistemului, respectiv a profesorilor și studenților, a fost impusă odată cu mutarea activităților în mediul online. Toată lumea a fost nevoită să se adapteze la un sistem nou, diferit și nefamiliar. Aici s-a testat creativitatea și abilitatea de a găsi soluții eficiente într-un timp scurt și pe termen lung. Coevoluția reprezintă adaptivitatea pe termen lung a sistemului care s-a transformat, mai exact a evoluat, fiind influențată atât de factori externi cât și interni și pune accent pe interacțiunea dintre student și profesor și student și mediul înconjurător. Mediul online a conectat toți actorii sistemului adaptiv complex, iar prin eforturi susținute din partea tuturor s-a reușit performanța de a nu se întrerupe ceea ce exista înainte în format fizic. Acesta a permis studenților să-și continue orele și să-și susțină examenele, însă într-un context nou, întrucât mediul de învățare a trecut de la băncile facultății la biroul de acasă al studenților. Astfel, și datorită sistemului online de învățământ care a devenit foarte popular, studenții sunt mereu conectați la tehnologie și internet, mereu cu telefonul în mână, iar această aplicație mobilă dezvoltată pentru lucrarea de licență reprezintă un mod inovativ, rapid și ușor de folosit de aceștia pentru a-și gestiona și urmări progresul, la un click distanță.

2.4 Similitudinea cu alte platforme sau aplicații

Aplicația ce urmează să fie prezentată a fost gândită după experiența a doi ani de studiu în mediul online și include mai multe funcționalități care să ușureze și să mențină într-un singur loc notele, materiile și progresul academic realizat de un student pe parcursul semestrului. Ceea ce face aceasta aplicație diferită de alte aplicații similare este faptul că se axează pe nevoile studenților, ușurând procesul de căutare a diferitelor informații, cu o mai bună gestionare a acestora. Pentru a explica mai bine funcționalitățile aplicației se va face o analiza comparativă cu alte soluții informatice relevante.

Slack este o platformă de comunicare folosită în mediul profesional menită să faciliteze comunicarea dintre membrii unei organizații. Are o interfață interactivă, ușor de folosit și permite crearea mai multor grupuri, numite workspace, în funcție de necesitățile utilizatorilor. O funcționalitate utilă este reprezentată de funcția 'Search' care permite căutarea după mai multe filtre (în funcție de grup, persoana care a primit sau trimis mesajul, după data calendaristică, după documentele salvate). Această opțiune este utilă în contextul unui număr mare de grupuri, mesaje și documente care se stochează în aplicație.

Edmodo este o platformă educațională care are ca public țintă elevii și profesorii din învățământul preuniversitar. Este o platformă de E-learning care permite trimiterea de chestionare, postarea sarcinilor de lucru, comunicarea prin mesaje și crearea unor grupuri între elevi, profesori și părinți. Asemenea aplicației Slack, Edmodo este o platformă gratuită, cu o interfață prietenoasă. Față de Slack, Edmodo vine în ajutorul instituțiilor de învățământ și oferă acestora în mod gratuit caracteristici specifice pentru a administra platforma în calitate de administrator. Platforma oferă caracteristici utile pentru sprijinul elevilor, profesorilor și părinților prin care se oferă posibilitatea de a lansa teste (quiz-uri), a oferi feedback la teste și teme, de a transmite fișiere de diverse

tipuri (materiale, clipuri video, link-uri, mesaje, fișiere), toate acestea într-un mediu controlat și vizibile profesorilor și părinților.

Clickup este o platformă care are ca scop gestionarea obiectivelor și organizarea acestora, astfel încât parcursul în vederea atingerii unui obiectiv să fie ușor de înțeles, bine structurat și transparent pentru a evita confuzii și depășirea tipului limită de predare a obiectivelor. Această platformă este un instrument care se bazează pe principiile și valorile metodologiei *agile*. Această aplicație are diferite funcționalități care sunt menite să îmbunătățească managementul proiectelor și procesul de dezvoltare și livrare a unei soluții finale care a fost dezvoltată și testată. O caracteristică importantă sunt iterațiile sprinturilor care se repetă pentru un număr predefinit de ori. Un sprint reprezintă un număr de pași în cadrul unei iterații. Pașii unui sprint sunt formați din planificare, implementare, testare și feedback-ul pe care un proces îl are în cadrul unei producții. Astfel, scopul acestor iterații este progresul continuu bazat pe feedback-ul anterior, experiența anterioară și adaptarea la cerințe noi pe parcursul dezvoltării unei soluții. Înainte de fiecare iterație, se analizează feedback-ul iterației anterioare și se stabilește o nouă planificare care va prezenta ceea ce se întâmplă în sprintul respectiv. Astfel, se pune accentul pe împărțirea unui proces mai mare în procese mai mici, numite sprinturi, prin care se iterează pentru un anumit număr de ori până se ajunge la o soluție finală bună.
Platforma are diferite funcționalități care au ca scop gestionarea unui proiect într-un mod eficient, urmărind pașii menționați anterior. Există diferite grafice și tabele care să ofere o perspectivă vizuală a procesului care are ca scop îndeplinirea unor obiective. Calendarele ajută la vizualizarea fluxului de proces pentru gestionarea mai bună a timpului de lucru.

Având în vedere cele menționate mai sus, s-a propus o aplicație mobilă care să vină în ajutorul studenților, însă să fie mai cuprinzătoare decât alte aplicații, având diverse funcționalități care să ușureze și să faciliteze gestionarea notelor, materiilor și a progresului pe parcursul unui semestru. Aplicația descrisă în această lucrare se

aseamănă cu funcționalitățile platformei Slack, întrucât are ca scop să ajute utilizatorii să se organizeze mai bine și mai eficient.

De asemenea, este similară și cu Clickup, pentru că are funcționalități de gestionare a obiectivelor, urmărind să existe o organizare cât mai clară și eficientă prin utilizarea calendarelor, a tabelelor care prezintă fluxul de lucru și de activitate având în permanență grijă de termenul de predare a proiectelor. Cu platforma Edmodo se aseamănă prin funcționalitatea de vizualizare a graficelor, urmărind progresul academic.

Aplicația mobilă prezentată în această lucrare este diferită de alte aplicații suport pentru studenți datorită unor particularități care pun accentul pe câteva funcționalități ale sistemului și pe fluxul de comunicare și care vor fi prezentate în cele ce urmează, astfel:

- Un utilizator își creează cont pentru a rămâne mereu în aplicație
- Acesta își personalizează orarul în secțiunea orar, introducând date despre materia respectiva precum sala, profesorul, data calendaristică de început și sfârșit
- Studentul introduce note la o anumită materie din orar, calculează media acestora, editează sau șterge. Se creează un grafic al notelor sub formă de PieChart colorat în care se vizualizează pe materii ponderea mediei notelor
- Pe baza materiilor din orar se pot introduce obiective care au o data limită de îndeplinit. Când un obiectiv este îndeplinit, utilizatorul îl bifează în aplicație și are posibilitatea de a vedea ce procent din sarcinile curente a fost îndeplinit. Aceste sarcini sau obiective pot fi ordonate după data de finalizare sau după nivelul de dificultate. Pe baza lor se creează un grafic de tip BarChart în care se vizualizează numărul de obiective și nivelul de dificultate pentru fiecare materie.

2.5 Utilitatea funcționalităților aplicației

O astfel de aplicație mobilă ajută studenții să se organizeze mai ușor, să țină evidența progresului lor academic și să le ușureze munca de planificare și organizare a obiectivelor ce trebuie duse la bun sfârșit. Într-un timp scurt pot găsi informațiile de care au nevoie accesând platforma destinată lor.

Funcționalitatea de a crea un grafic al notelor la o materie aleasă este utilă pentru a vizualiza statusul curent al progresului academic.

De asemenea, studentul are opțiunea să creeze o listă de obiective. Fiecare obiectiv este asociat unei materii din orar și se alege un nivel de dificultate și o dată până la care trebuie îndeplinit. Când data limită se apropie de data curentă, utilizatorul este anunțat printr-o notificare. În cazul în care un obiectiv este îndeplinit se bifează de pe listă.

Capitolul 3 Tehnologii utilizate

3.1 Aplicație mobilă pentru android

Lucrarea de licență are ca scop dezvoltarea unei aplicații mobile android și utilizează limbajul java cu o bază de date la distanță.

La nivel global s-a remarcat o creștere nemaiîntâlnită a numărului de utilizatori care folosesc un telefon în ultimul deceniu. Telefoanele inteligente au ajuns la milioane de oameni, iar acest lucru înseamnă faptul că odată cu dezvoltarea industriei telefoanelor mobile s-a dezvoltat și industria de dezvoltare a aplicațiilor mobile care să fie instalate de utilizatori. Datorită avansului tehnologic în ceea ce privește sistemul de operare și securitate și compatibilității cu mai multe dispozitive, aplicațiile de telefon Android susținute de Google au avut parte de o creștere susținută a numărului de utilizatori.

Din punct de vedere al unei strategii de afaceri, dezvoltarea unei aplicații mobile este un punct cheie al acesteia, pentru că ușurează experiența clientului în legătura cu achiziționarea sau informarea despre un anumit produs sau serviciu. Un alt beneficiu al dezvoltării unei aplicații mobile îl reprezintă faptul că este mai puțin costisitoare dezvoltarea și întreținerea unei aplicații mobile, față de angajarea unui număr mai mare de personal care să realizeze ceea ce o singură aplicație are posibilitatea de a face într-un mod eficient și sistematizat. Prin aplicația mobilă se centralizează eficient datele în legătură cu serviciul prestat de afacere. [2]

Un alt avantaj al dezvoltării de aplicații android este reprezentat de faptul că se adresează unui număr larg de utilizatori. Programatorii care dezvoltă aceste tipuri de aplicații au acces gratuit la codul sursă pentru că Android este o platformă sursă deschisă tuturor celor interesați. Acest lucru ajută la dezvoltarea de aplicații unice și inovative.

În plus, dezvoltarea unei aplicații android are ca avantaj posibilitatea de integrare a serviciilor de la Google precum *Google Maps, FireStore, Google Cloud Messaging*, oferind aplicației funcționalități care fac sa îi crească performanța.

3.2 Mediul de dezvoltare

Pentru dezvoltarea aplicației de licența s-a folosit Android Studio care este un IDE pentru dezvoltarea aplicațiilor android dezvoltat de IntelliJ IDEA. Acest mediu de dezvoltare are un editor de cod puternic și instrumente de dezvoltare create cu scopul de a minimiza redundanța și de a crește eficiența. De asemenea, are integrat un sistem Gradle build, un emulator și face conexiunea cu platforma GitHub, având suport pentru platforma Google Cloud. [3]

3.3 Limbajul Java

Aplicația de telefon care se prezintă este dezvoltată în java, pentru că acest limbaj se potrivește cel mai bine pentru funcționalitățile pe care aceasta le oferă. Limbajul Java a fost dezvoltat in anul 1995 pe vremea când internetul experimenta o creștere exponențiala. Java este un limbaj de programare orientat pe obiecte și un limbaj bazat pe multiparadigmă. [4]

3.4 Bază de date non relațională Firebase

Aplicația folosește o bază de date la distanță. Firebase a apărut în anul 2012 și de atunci și până în prezent s-a dezvoltat într-o platformă cloud care are funcționalități de sincronizare. În anul 2017 Google a anunțat apariția lui Firestore, ultima variantă de baza de date real-time, NoSql. Diferența dintre o bază de date SQL și NoSql este dată de faptul că nu exista tabele și coloane, ci colecții și documente. Documentele conțin seturi de perechi sub forma de cheie-valoare și sunt salvate în colecții. Documentele pot

conține și subcolecții și obiecte imbricate, iar fiecare poate include câmpuri primitive, cum ar fi obiecte de tipul string sau obiecte complexe. Colecțiile și documentele sunt create implicit în Firestore, prin atribuirea datelor care se doresc salvate pentru un document în cadrul unei colecții. Dacă colecția sau documentul nu există, Firestore le va crea automat. [5]

3.5 Autentificarea în FireBase

Prin intermediul lui Firebase Firestore s-a folosit o funcționalitate importantă: autentificarea, care e gestionată de clasa FirebaseAuthUI. Pentru procesul de autentificare este nevoie o instanță a clasei FirebaseAuth. Prin aceasta se creează conturi pentru utilizatori, se conectează și deconectează de la cont și se accesează informații despre un anumit utilizator.

Clasa FirebaseAuth este responsabilă de gestionarea procesului de autentificare. Ea are ca scop conectarea informațiilor utilizatorului la contul efectiv al acestuia.

La autentificare se utilizează metoda *getCurrentUser* a clasei *FireBaseAuth* pentru a obține un obiect de tipul *FirebaseUser*. Acest obiect conține informații despre utilizatorul autentificat, cum ar fi adresa de e-mail, ID-ul utilizatorului și alte detalii specifice furnizorului de autentificare utilizat (Figura 1), astfel că fiecare informație va fi diferită în funcție de utilizator (Figura 2).

```
//Register User using the credentials given
private void registerUser(String nume, String prenume, String dataNastere, String facultate,
                          String email, String parola, String parola2) {
    mAuth=FirebaseAuth.getInstance();//am initializat obiectul
    mAuth.createUserWithEmailAndPassword(email,parola).addOnCompleteListener( activity: SignUp.this,
            new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {

            if(task.isSuccessful()){
                //user was created
                Toast.makeText( context: SignUp.this,  text: "Cont creat", Toast.LENGTH_SHORT).show();

                //register user in firebase
                userID=mAuth.getCurrentUser().getUid();
                fStore=FirebaseFirestore.getInstance();
                DocumentReference documentReference=fStore.collection( collectionPath: "users").document(userID);

                //hashmap
                Map<String ,Object> user= new HashMap<>();
                user.put( k: "nume",nume);
                user.put( k: "prenume",prenume);
                user.put( k: "dNastere",dataNastere);
                user.put( k: "facultate",facultate);
                user.put( k: "email",email);
                documentReference.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void unused) {
                        Log.d(TAG, msg: "Succes:user profile is created for"+userID);
                    }
```

Figură 1 Înregistrarea unui utilizator nou folosind e-mail-ul și parola

```
FirebaseFirestore db = FirebaseFirestore.getInstance();
FirebaseAuth mAuth = FirebaseAuth.getInstance();
String userID = mAuth.getCurrentUser().getUid();
DocumentReference documentReference = db.collection( collectionPath: "users").document(userID);
   documentReference.addSnapshotListener( activity: this, new EventListener<DocumentSnapshot>() {
       @Override
       public void onEvent(@Nullable DocumentSnapshot value, @Nullable FirebaseFirestoreException error) {
           if(mAuth.getCurrentUser() !=null){
               textViewNume.setText(value.getString( field: "nume"));
               textViewPrenume.setText(value.getString( field: "prenume"));
           }

       }
   });
```

Figură 2 Accesare a unui utilizator și a informațiilor lui

Autentificarea se face prin e-mail și parolă. În ceea ce privește procesul de creare a contului, după înregistrarea utilizatorului se trimite un e-mail de verificare pentru confirmarea adresei de e-mail a utilizatorului (Figura 3) și pentru a asigura

securitatea și autenticitatea contului. Dacă e-mail-ul utilizatorului nu există acesta va fi trimis la pagina de LogIn (Figura 4). [6]

```
FirebaseAuth auth=FirebaseAuth.getInstance();
FirebaseUser user=auth.getCurrentUser();
if(user==null){
    Intent intent=new Intent(getApplicationContext(), LogIn.class);
    startActivity(intent);
    finish();
}
else{
```

Figură 3 Daca userul nu e autentificat, va fi trimis la Activitatea de LogIn

```
FirebaseUser firebaseUser= mAuth.getCurrentUser();
//Send verification email
firebaseUser.sendEmailVerification();
```

Figură 4 Verificarea e-mailului unui utilizator nou

3.6 Arhitectura Model View Controller

Model View Controller (MVC) este un design pattern care a fost introdus pentru prima dată în anii 1970 de către Trygve Reenskaug și ulterior dezvoltat de Glenn Krasner și Stephen Pope în 1988. Scopul principal al acestei arhitecturi este să lege modelul mental al utilizatorului de modelul digital al calculatorului. Acest model aduce multe beneficii prin structura modulară pe care o promovează și izolarea unităților funcționale, ceea ce facilitează înțelegerea și modificarea diferitelor parți ale unei aplicații fără a avea o cunoaștere deplină a tuturor unităților aplicației. [7] După modelul MVC s-au dezvoltat si arhitecturile MVP și Model-View-ViewModel (MVVM).

Arhitectura MVC este împărțită în 3 componente: Model, View și Controler. Componenta Model stochează date sau stări în clase complexe. Componenta View

este partea de interfață grafică pentru utilizatori. Componenta Controler preia datele introduse de utilizatori în view-uri și schimbă starea în model și trimite modificările înapoi la view-uri.

Un flux al arhitecturii MVC funcționează în felul următor: Controlerul primește datele introduse de utilizator. Acum sunt posibile 3 scenarii:

1. Controlerul anunța view-ul pentru modificarea stării
2. Controlerul anunța Model-ul, iar apoi acesta își anunță toate dependințele sale (perechile View-Controller)
3. Controlerul trimite un mesaj view-ului, care trimite cereri către Model pentru a putea să-și actualizeze cea mai recentă stare [8]

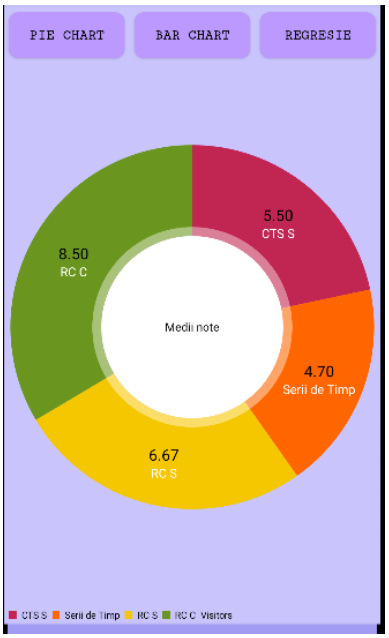
În ceea ce privește **implementarea MVC în Andoid,**

- componenta Model este, de regulă, o simplă clasă de obiecte Java și este responsabilă pentru gestionarea logicii aplicației la nivel foarte jos, mai exact cum se stochează, sortează, prelucrează datele. Nu are nicio cunoștință legată de interfața aplicației.
- Componenta View este o combinație între fișierul de resurse XML și activitatea/fragmentul. Responsabilitatea ei este să țină legătura cu componentele vizuale care apar pe ecran.
- Componenta Controller este activitatea sau fragmentul aplicației, iar rolul ei este de a crea o legătură între View și Model. Ea conține logica aplicației și este informată de comportamentul utilizatorului pentru a actualiza modelul când este necesar. De exemplu, evenimentul click este captat în metoda onClickListener din activitate. [8] [9]

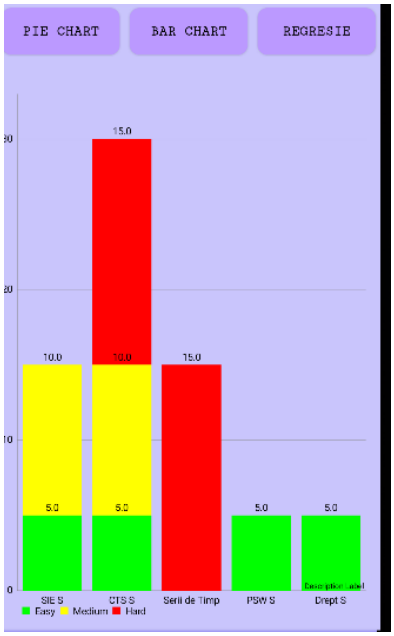
În aplicația mobilă prezentată în lucrare s-a folosit o arhitectură MVC. Avantajele acestei arhitecturi sunt reprezentate de ușurința cu care se testează codul și se implementează noi funcționalități.

3.7 Implementarea graficelor BarChart și PieChart

În cadrul aplicației există 2 tipuri de grafice: BarChart și PieChart. Scopul acestor grafice este de a vizualiza, dintr-o perspectiva de ansamblu, date precum media fiecărei materii unde au fost introduse note și câte obiective au fost atribuite unei materii, iar în funcție de dificultatea setată pentru fiecare obiectiv se gestionează efortul în mod diferit. De exemplu, unui obiectiv care are setat un nivel de dificultate mare îi vor fi alocate un număr mai mare de căsuțe, pe axa verticală, reprezentând nivelul de efort care trebuie depus de utilizatorul care a introdus obiectivul. În figura 5 și figura 6 se observă cele două grafice.



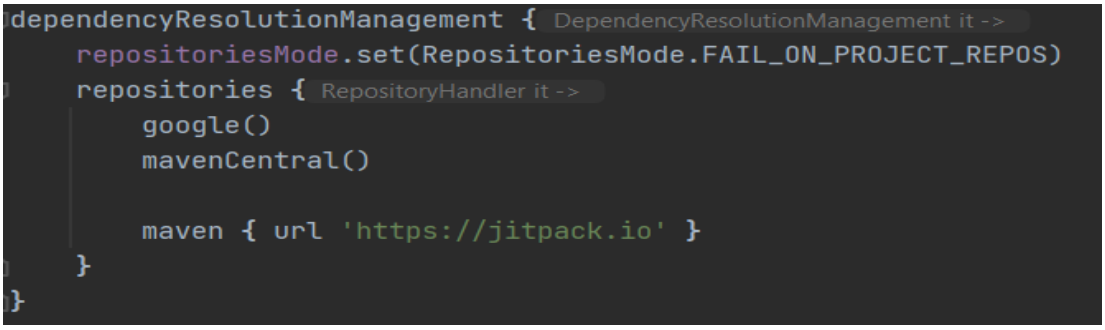
Figură 5 PieChart medii note pentru materii



Figură 6 BarChart Taskuri cu nivel dificultate diferit pentru fiecare materie la care s-a introdus cel puțin un obiectiv

Pentru a implementa graficele s-a folosit biblioteca MPAndroidChart. În figura 7 și figura 8 sunt arătate dependențele care sunt introduse în aplicație, astfel încât să funcționeze corect acest API. [10]

Figură 7 Dependența introdusă în build.grade



Figură 8 Dependența introdusă în settings.gradle

Pentru construirea efectivă a celor două grafice a fost necesară crearea unor algoritmi care să populeze o listă de obiecte care a fost dată ca parametru funcției de creare a graficelor. Un lucru important de menționat este faptul că graficele se schimbă în funcție de modificările aduse atât în zonele de introducere a obiectivelor cât și în cele pentru introducerea notelor. Acest lucru este posibil pentru că datele sunt preluate în mod dinamic din baza de date, iar orice modificare a listelor preluate din bază va produce o schimbare în aspectul graficelor.

În figura 9 este reprezentat algoritmul de construire a graficului PieChart. S-a inițializat o listă de obiecte de tip PieEntry, numită visitors. S-a parcurs lista de materii care a fost introdusă în secțiunea de note, iar de aici s-a preluat doar lista notelor pentru fiecare materie și s-a calculat o medie. S-a populat lista vistitors cu un obiect de tip PieEntry care primește ca parametru denumirea materiei și media calculată a notelor pentru materia respectivă.

```
ArrayList<PieEntry> visitors = new ArrayList<>();

String denumire;
List<Float> listaNote;
Float suma;
Float medie;
for (Materie m : materiiNote) {
    denumire = m.getDenumire();
    listaNote = m.getListanote();
    suma = Float.valueOf(0);
    for (Float f : listaNote) {
        suma += f;
    }
    if (listaNote.size() != 0) {
        medie = (suma / (listaNote.size()));
        visitors.add(new PieEntry(medie, denumire));
    }
}
```

Figură 9 Algoritm de construire a PieChart

În figura 10 este reprezentat modul de construire a graficului cu bare. În prima fază se inițializează un hashmap unde se vor salva numărul de obiective pentru fiecare nivel de dificultate. Din baza de date se preia lista de obiective, iar în lista taskCounts se contorizează numărul de obiective pentru fiecare grad de dificultate. Această lista va fi valoarea în hasmap, iar cheia asociată valorii este numele materiei. Pentru desenarea graficului s-au stabilit niște valori pentru fiecare nivel de dificultate.

```java
// Create lists for storing bar entries and labels
List<BarEntry> entries = new ArrayList<>();
List<String> labels = new ArrayList<>();

// Create a map to track the total task counts for each difficulty level
Map<String, float[]> taskCountsMap = new HashMap<>();

// Iterate through the toDoList and calculate task counts for each materie and difficulty level
for (ToDoModel model : toDoList) {
    String materie = model.getMaterie();
    int dificultate = model.getDificultate() - 1;

    float[] taskCounts = new float[0];
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        taskCounts = taskCountsMap.getOrDefault(materie, new float[3]);
    }
    assert taskCounts != null;
    taskCounts[dificultate]++;

    taskCountsMap.put(materie, taskCounts);
}
// Generate bar entries and labels based on the task counts
int barIndex = 0;
for (Map.Entry<String, float[]> entry : taskCountsMap.entrySet()) {
    String materie = entry.getKey();
    float[] taskCounts = entry.getValue();

    // Calculate the height for each difficulty level
    float easyHeight = taskCounts[0] * 5f;
    float mediumHeight = taskCounts[1] * 10f;
    float hardHeight = taskCounts[2] * 15f;

    // Create a stacked bar entry
    BarEntry barEntry = new BarEntry(barIndex, new float[]{easyHeight, mediumHeight, hardHeight});
    entries.add(barEntry);

    // Add the materie as a label
    labels.add(materie);
    barIndex++;
}
```

Figură 10 Algoritm de construire a barChart

Astfel, un obiectiv ușor va fi reprezentat de o bară și va fi mai mic în înălțime decât un obiectiv greu, fiecare având culori diferite.

Capitolul 4 Arhitectura soluției

În cadrul acestui capitol se vor descrie structura aplicației mobile și componentele acestei aplicații însoțite de capturi de ecran care să ilustreze ceea ce se descrie, iar la final se va exemplifica un flux complet pentru a se urmări fluiditatea aplicației dezvoltate.

4.1 Diagrama de componente

În general, o diagramă de componente arată modul în care componentele software se leagă între ele și cum comunică una cu cealaltă. De asemenea, ea modelează arhitectura de ansamblu și componentele locale din interiorul acesteia. [11]

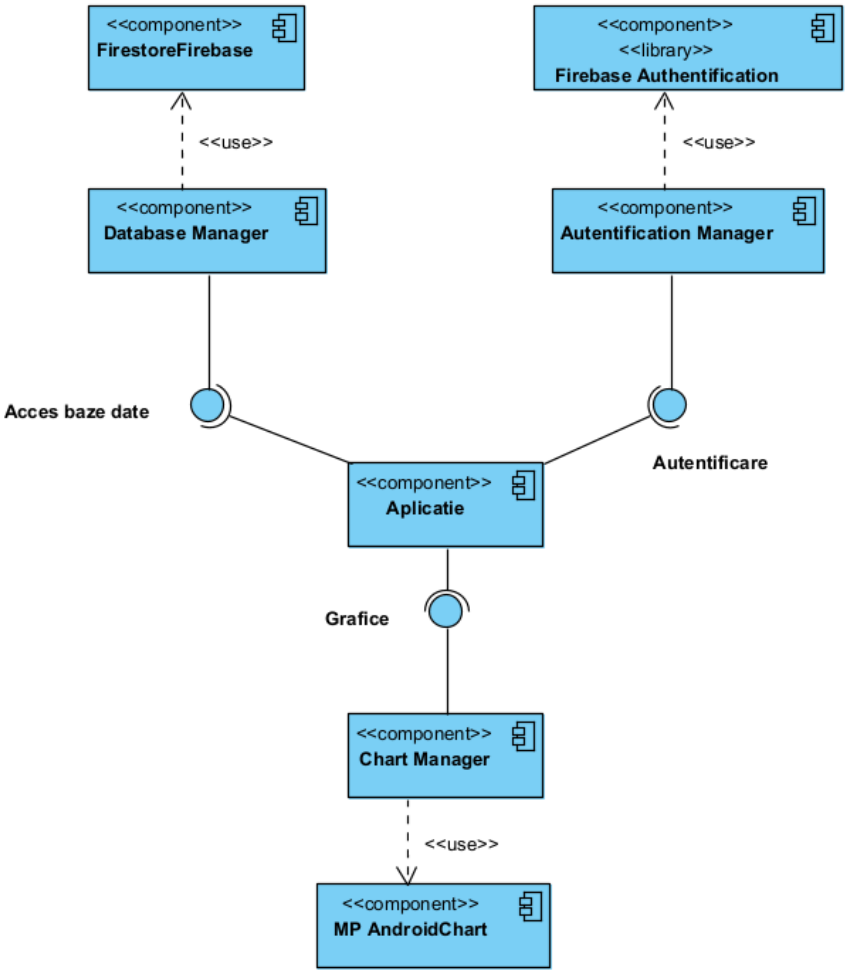
Aplicația prezentată în această lucrare are ca scop principal gestionarea notelor și a activităților care pot apărea pe parcursul unui semestru pentru a urmări progresul sau slabele performanțe academice.

Unicul actor care interacționează cu sistemul informațional este utilizatorul, respectiv studentul care este interesat să-și instaleze o astfel de aplicație mobila.

În figura 11 aplicația este componenta centrală principală. Accesul la baza de date la distanță este intermediată de o interfață care este solicitată de componenta aplicație.

Componenta Database Manager furnizează accesul la informație, oferind un nivel de abstractizare prin care se ușurează interacțiunea cu baza de date la distanță, Firestore.

Pentru autentificare, Autentification Manager utilizează funcționalitatea externă a aplicației oferita de Firebase.

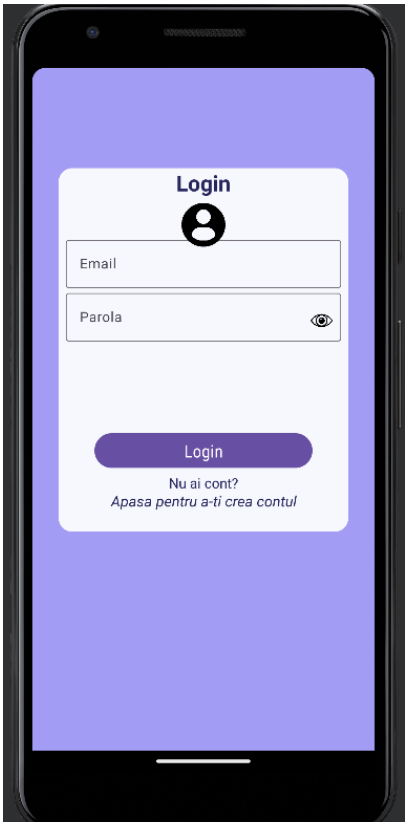


Figură 11 Diagramă de componente

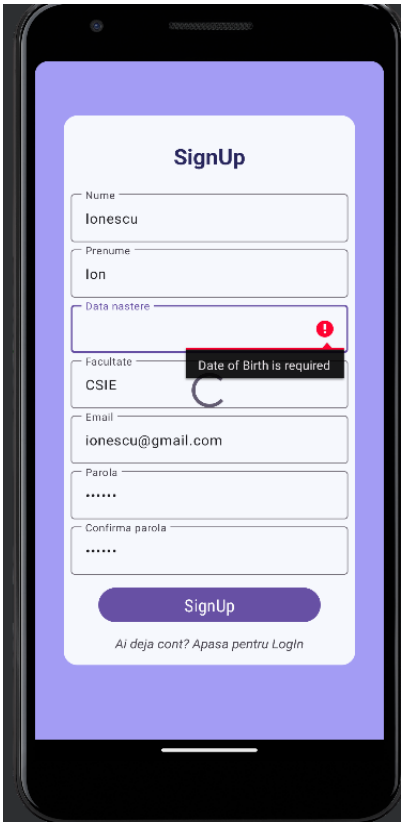
Graficele construite în aplicație folosesc un API extern numit MP AndroidChart. care furnizează metode de creare și personalizare a graficelor cu informații preluate din baza de date la distanță.

4.2 Introducere în aplicație

Un utilizator trebuie să își creeze un cont pentru a utiliza aplicația. Astfel, atunci când aplicația este deschisă pentru prima dată se deschide activitatea cu formularul LogIn.



Figură 12 LogIn



Figură 13 SignUp- excepție câmp necompletat

În cazul în care utilizatorul nu există, acesta trebuie să apese pe textView-ul *“Apasă pentru a-ți crea cont”* (Figura 12), unde se va deschide o activitate nouă cu formularul de SignUp. Ambele formulare conțin verificări pe câmpuri, astfel încât în cazul în care nu s-a completat un câmp se aruncă o excepție (Figura 13).

4.3 Structurarea interfețelor aplicației

În continuarea prezentării componentelor aplicației se va expune organizarea arhitecturală a interfețelor și se va explica funcționalitatea fiecărui buton împreună cu capturi de ecran reprezentative.

Arhitectura aplicației a fost gândită astfel încât sa fie cât mai intuitivă pentru un utilizator. Funcționalitățile aplicației sunt cuprinse în două tipuri de meniuri, iar la fiecare apăsare de buton din meniuri se va deschide o noua activitate sau fereastră.

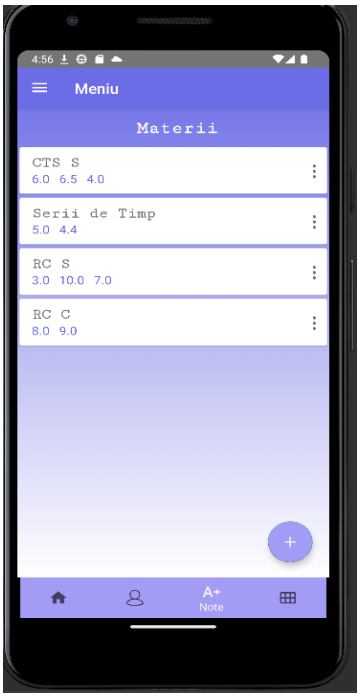
Meniul de josul ecranului*,"Bottom navigation drawer"* conține 4 elemente în următoarea ordine:

- pagina de acasă
- pagina personală a utilizatorului
- pagina unde se introduc note pentru o anumită materie
- orarul studentului pe semestrul respectiv.

Atunci când se apasă butonul “Home” din meniul de jos se deschide un fragment, reprezentând pagina de acasă. Aici este afișat un mesaj de “Bine ai revenit”, împreună cu data curentă. În funcție de această dată, pe ecran se va afișa ce materii se desfășoară în ziua respectivă. Materiile care sunt preluate din baza de date la distanță au fost introduse anterior în orar.

Butonul Profil deschide un fragment nou în care sunt afișate datele utilizatorului atunci când și-a creat contul.

Atunci când este apăsat al treilea buton din meniul de jos se deschide un fragment nou (Figura 14), unde utilizatorul introduce notele pe care le editează sau le șterge la o anumită materie din orar. În josul ecranului, rolul lui *Floating Action Button* este de a deschide un *AlertDialog* unde se vor introduce note (Figura 15).



Figură 14 Introducere note

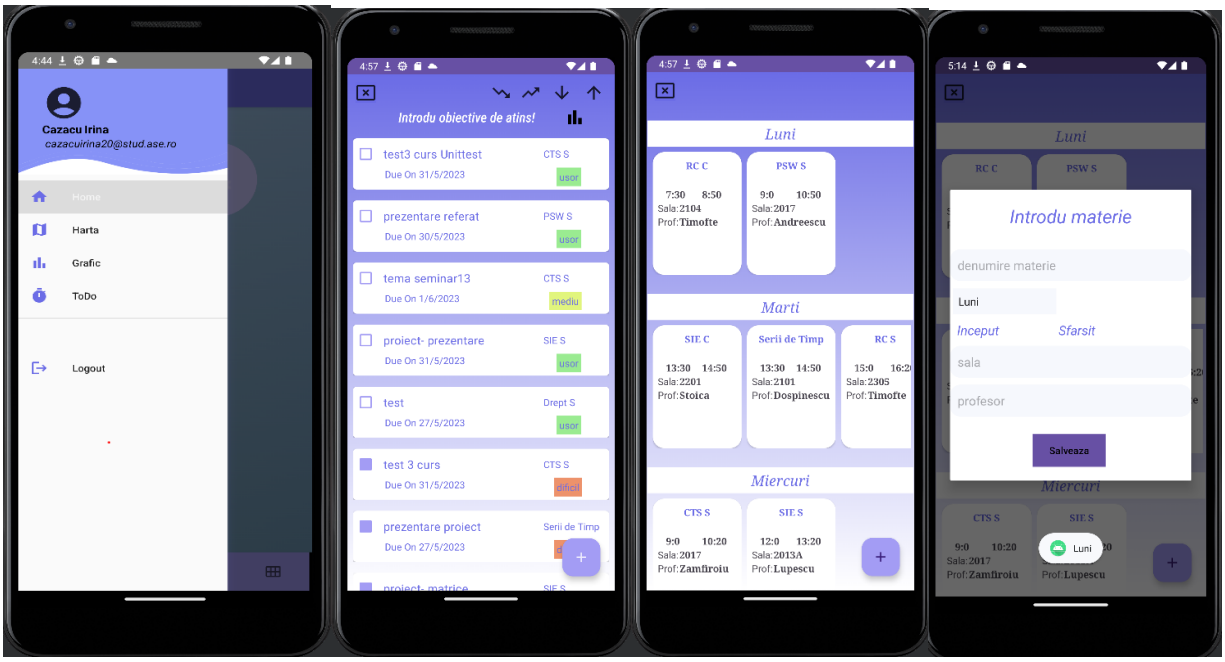
Figură 15 Alert Dialog

Figură 16 Home page

Ultimul buton din meniul de jos este reprezentat de orarul studentului. Se deschide o activitate nouă, iar aceasta este elementul central al aplicației, pentru că pe baza materiilor introduse aici se pot introduce note și obiective de îndeplinit (Figura 19). Asemănător la introducerea notelor există și aici un *Floating Action Button* care deschide un *AlertDialog* unde se vor introduce date despre materia care se dorește a fi introdusă în orar (Figura 20).

Meniul din stânga, *"Navigation Drawer"* (Figura 17) conține mai multe elemente care vor fi prezentate în cele ce urmează:

- Pagina de acasă
- Pagina cu grafice pentru a vizualiza progresul
- Pagina unde se salvează obiectivele de atins



Figură 17 Meniu *Figură 18 Obiective* *Figură 19 Orar* *Figură 20 AlertDialog*

Atunci când se apasă butonul “To do” din meniul din stânga (Figura 17) se deschide o activitate nouă unde vor apărea elementele din listă în ordinea introducerii lor, iar obiectivele pe care utilizatorul dorește să le atingă într-o anumita perioadă vor fi salvate în baza de date sub formă de stivă (Figura 18). Astfel, *Floating Action Button* deschide un *BottomSheetFragment* unde se pot introduce obiective noi. Fiecare obiectiv introdus se referă strict la o anumită materie din orar. Utilitatea acestei funcționalități este dată de faptul că la un obiectiv se introduc atât termenul limită, cât și nivelul de dificultate pe care utilizatorul îl consideră că acel obiectiv îl are. La afișarea obiectivelor se observă atât fiecare termen limită cât și nivelul de dificultate, colorate într-un mod cât mai sugestiv: ușor-verde, mediu-galben și dificil-roșu.

Graficele au fost construite cu un API și au fost explicate în capitolul anterior.

Ieșirea din cont a utilizatorului se face prin apăsarea butonului de log-out. Utilizatorul va fi redirecționat către activitatea LogIn.

4.4 Fluxul aplicației

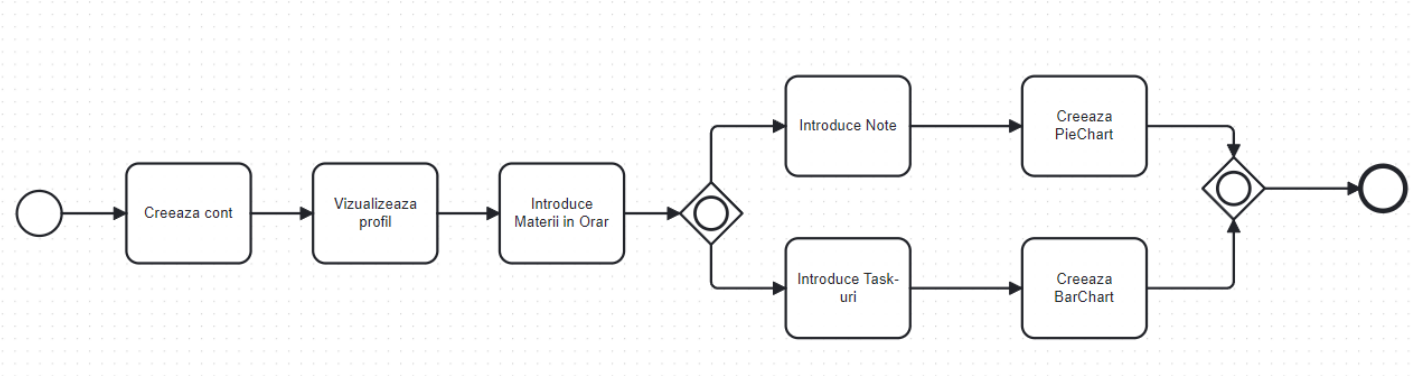
Până acum au fost prezentate funcționalitățile aplicației împreună cu arhitectura acesteia și modul în care ea a fost gândita din punct de vedere structural . În cele ce urmează se va prezenta un flux complet al aplicației și ordinea în care trebuie introduse datele.

Astfel, utilizatorul își creează contul, urmând ca acesta să meargă la secțiunea Orar pentru a-și introduce toate materiile din semestrul respectiv. Materiile vor fi aranjate în orar la o anumită zi, în ordinea crescătoare a orei la care începe fiecare materie. Ulterior, pe parcursul acumulării notelor, utilizatorul merge în secțiunea de Note unde își va introduce notele acumulate la o anumită materie pe parcursul semestrului. Notele pot fi editate sau șterse în funcție de necesitățile utilizatorului. La secțiunea Grafice, la PieChart se creează un grafic cu mediile notelor obținute la fiecare materie. Daca nu există nicio notă introdusă, atunci graficul nu va fi generat.

La secțiunea de *ToDo*, utilizatorul va introduce pe parcursul semestrului obiectivele pe care acesta intenționează să le îndeplinească într-un anumit timp stabilit de acesta la o anumită materie din orar. Se urmărește ușor progresul, întrucât utilizatorul va bifa un obiectiv atunci când el este îndeplinit. Exista posibilitatea ștergerii sau editării unui obiectiv prin glisarea elementului selectat la stânga sau la dreapta. De asemenea, în colțul din dreapta sus există un buton care generează un procent ce reprezintă câte obiective au fost bifate din toate cele existente. În funcție de nevoile utilizatorului, prin apăsarea săgeților din susul ecranului, obiectivele se ordonează crescător sau descrescător după data limita sau sunt sortate după nivelul de dificultate, de la cel mai ușor până la cel mai dificil sau invers.

La secțiunea grafice, la *BarChart* se generează un grafic pe baza obiectivelor introduse. Pe axa orizontală sunt materiile la care s-au introdus obiective de îndeplinit, iar pe axa verticală este reprezentat nivelul de dificultate pentru fiecare obiectiv. Astfel, graficul constituie o reprezentare vizuală ce ilustrează efortul ce trebuie alocat de către

student pentru fiecare obiectiv, în concordanță cu nivelul de dificultate care figurează în grafic.



Figură 21 Diagrama de flux si procese

În continuarea prezentării fluxului în cadrul aplicației, în figura 21 este reprezentată diagrama de flux de procese care are rolul de a exemplifica grafic, într-un mod simplist, fluxul prezentat mai sus. Poarta inclusivă semnifică faptul că fluxul de secvență merge doar pe o ramură sau pe amândouă.

Capitolul 5 Implementarea soluției

După cum a fost prezentat în Capitolul 4, aplicația prezinta două meniuri, iar din fiecare meniu, la apăsarea unui element din el se deschide fie o activitate, fie un fragment. Pentru a urmări cursul firesc al unui flux de activitate se vor prezenta implementarea algoritmilor în funcție de acest flux.

5.1 Implementarea orarului și structurarea pe clase

Această funcționalitate a fost implementată într-un mod complex, folosind mai multe recyclerview-uri. Recyclerview-ul părinte are în componența sa cinci recyclerView-uri asociate fiecărei zi lucrătoare a săptămânii. Datele sunt preluate din baza de date și afișate în recyclerViewurile corespunzătoare zilei săptămânii aleasă de utilizator sub formă de CardView. Pentru o vizualizare corectă a tuturor CardView-urilor care sunt afișate pe orizontală dedesubt la ziua săptămânii trebuie să se gliseze în dreapta.

Pentru realizarea acestui obiectiv a fost necesară implementarea a două adaptoare care au ca scop conectarea și vizualizarea datelor în recyclerViewuri. În figura 22 cadranul cu galben reprezintă recyclerView ul părinte, iar cadranele cu roșu reprezintă recyclerViewurile care sunt elemente pentru părinte.

În cele ce urmează se va prezenta într-o maniera succintă modul de construire a acestei funcționalități prin implementarea claselor și se vor explica relațiile dintre clase și modul de funcționare a acestora pentru reușita implementării.

Clasa MaterieOrar

Prin aceasta clasă se inițializează obiectul de tip materie care are ca atribute informațiile ce se doresc a fi afișate pe ecran folosind CardView-uri.

Clasa ParentModelClass

Clasa acesta are în componența sa doua atribute, un String și un Integer, ce reprezintă ziua săptămânii, ID-ul asociat acesteia și metode specifice de setteri și getteri.

Adapterul ListaMaterii

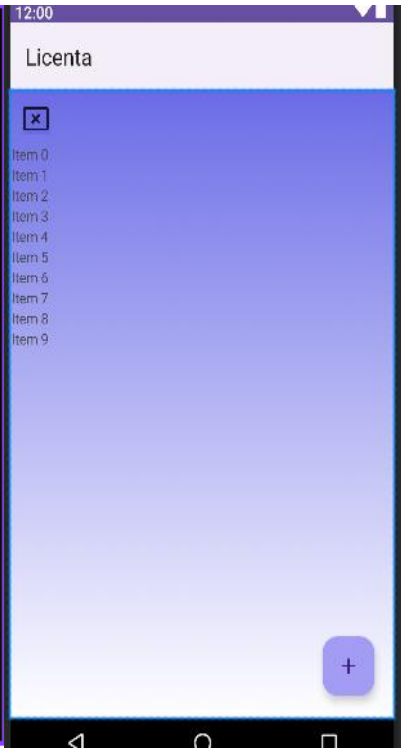
Această clasă este un adaptor pentru RecyclerViewurile copil. Va afișa lista de obiecte de tip MaterieOrar în CardViewuri. Funcția care populează un CardView este onBindViewHolder.

ParentAdapter

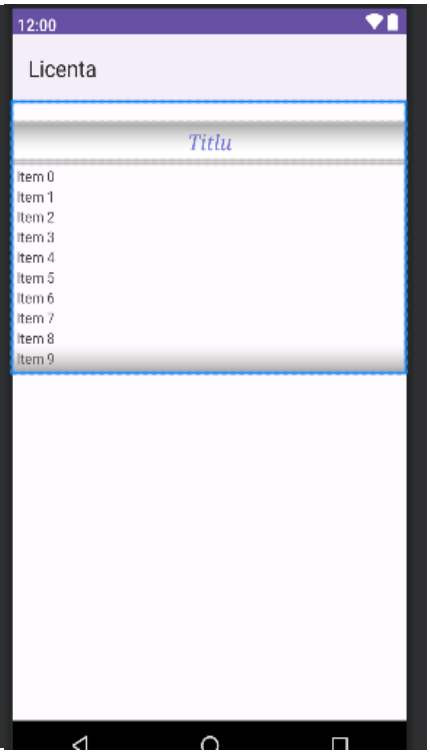
Acest adaptor populează recyclerView-ul părinte, fiecare element din acesta reprezentând un element populat mai sus în recyclerView copil ce conține cardViewuri. Se inițializează un hashmap care are ca chei obiecte de tipul ParentModelClass și reprezintă ziua săptămânii, iar valorile sunt lista de obiecte de tipul MaterieOrar. Astfel, fiecărei zi a săptămânii i se asociază o listă de materii.

În activitatea principala se inițializează un hashmap asemănător celui prezentat mai sus, cheia fiind o zi a săptămânii, iar valorile reprezintă o lista de materii. Preluarea din baza de date a materiilor deja existente se face cu metoda get apelată pe referința colecției unde sunt stocate datele. Datele sunt sortate după ora de început a fiecărei materii și salvate în hashmap în ordinea corectă.

Figură 22 RecyclerView



Figură 23 Fișier XML părinte



Figură 24 Fișier XML copil

Figură 25 CardView

Figurile 23, 24 și 25 prezintă modul în care arată fișierele XML pentru layoutul părinte, copil și Childview.

5.2 Implementarea introducerii notelor pentru o anumită materie

Această funcționalitate a fost implementată folosind un recyclerview iar pentru afișarea datelor pe ecran s-a implementat un adaptor.

NoteFragment este fragmentul care implementează această funcționalitate. Acesta conține o listă de note și o listă de obiecte de tip Materie. Pentru inserarea unei noi materii cu note asociate se deschide un AlertDialog atunci când un FloatingActionButton este apăsat. În cadrul ferestrei noi deschise există un spinner care conține denumirile materiilor preluate din baza de date în secțiunea orar și se alege materia la care se dorește introducerea notelor. Pentru fiecare nota introdusă se generează un nou editView. Fiecare element din recyclerview se editează atunci când este nevoie. Dacă se dorește acest lucru atunci reapare AlertDialog-ul având câmpurile completate cu datele originale. Pentru ștergere se selectează butonul de ștergere. Ambele opțiuni de editare și ștergere apar în dreptul fiecărui element din recyclerview sub forma de meniu. Salvarea în baza de date a notelor obținute se face prin obținerea unei referințe la colecția “NoteMaterii”. Pentru preluarea din baza de date a datelor din colecție se folosește metoda get aplicată referinței colecției. În continuare se parcurge fiecare document din colecție folosind o structură repetitivă. Fiecare document este preluat într-un obiect de tip Materie și se salvează într-o listă de obiecte de tip Materie. Atunci când se salvează în baza de date se folosește metoda add aplicată referinței colecției unde se va salva documentul. Pentru editarea elementelor se folosește aceeași logică, se preia referința colecției unde sunt salvate datele și se aplică metoda update.

Clasa Materie este clasa cu ajutorul căreia se creează obiecte care vor fi afișate pe ecran în recyclerview. Aceasta clasa conține un String pentru denumirea materiei și o listă de note.

Bineînțeles că pentru implementarea unui recyclerview este nevoie de un adapter. **Clasa RecyclerViewAdapterMaterii** face adaptarea dintre view-uri și datele

afișate pe ecran. Acesta conține o listă de materii iar în metoda onBindViewHolder se seteaza view-urile dintr-un element din recyclerview cu datele preluate din lista de materii. Tot aici se deschide și meniul pentru editare si ștergere a elementului în cazul în care se dorește acest lucru.

Un detaliu important de menționat este faptul ca fragmentul în android are un ciclu de viață diferit de cel al activităților, astfel că devine o necesitate atenția pentru sincronizare a datelor. Interfața “DataCallback” este utilizată pentru a notifica adaptorul atunci când lista de obiecte de tip Materie este modificată.

5.3 Implementarea creării de obiective

Activitatea principală care se ocupă de această funcționalitate cuprinde mai multe funcții de sortări ale elementelor din listă ce sunt afișate în recyclerview printr-un adaptor. Există sortări pentru ordonarea elementelor după nivelul de dificultate sau după data de finalizare a obiectivelor. Acest lucru s-a realizat folosind metoda sort aplicata clasei Collections. După fiecare modificare a listei de elemente, adaptorul care leagă datele de recyclerview este notificat pentru ca modificarea să fie vizibilă pe ecran.

Ceea ce este diferit față de funcționalitățile prezentate anterior este faptul că pentru a introduce un element nou în lista de elemente se va deschide un BottomSheetFragment la apăsarea unui floatingActionButton care se află în colțul din dreapta jos. În acest fragment au fost implementate diferite view-uri precum Radio Grup Button , Spinner și Calendar. Spinner-ul este construit cu elemente preluate din baza de date de la colecția orar. Acest lucru înseamnă că obiectivele pot fi puse doar pentru materiile care există în orar. La apăsarea butonului salvează se inițializează un hashmap unde se vor salva datele introduse în fiecare câmp sub formă de valoare, cheia fiind setată manual prin denumirea explicită a valorii. Se creează o referință la colecția unde vor fi salvate datele și se va aplica metoda add.

Un element diferit în aplicație îl reprezintă modul de editare a elementelor din listă. Astfel, în cazul în care utilizatorul dorește să editeze un obiectiv existent, acesta va selecta elementul și va glisa la dreapta. Se va deschide fragmentul de jos completat cu datele originale. În cazul în care se dorește ștergerea unui element din listă, elementul selectat trebuie glisat la stânga. Un AlertDialog va apărea pentru a întreba utilizatorul dacă este sigur că dorește ștergerea elementului. Clasa care se ocupă cu această funcționalitate se numește TouchHelper și extinde ItemTouchHelper.SimpleCallback. În metoda onSwiped se verifică direcția în care s-a glisat elementul și în funcție de asta este apelată metoda de ștergere sau editare folosind un obiect de tip ToDoAdaptor.

Clasa adaptor face legătura dintre lista de obiecte și recyclerView unde vor fi afișate obiectele din listă. Tot aici există și implementările funcțiilor de ștergere și editare. Când se șterge un element se preia referința colecției și a documentului respectiv și se apelează metoda delete, după care acesta este șters și din lista de obiecte, iar adaptorul este anunțat pentru a face modificările pe ecran. În cazul editării, se inițializează un obiect de tip Bundle care este populat cu datele originale, după care se inițializează un obiect nou de tipul fragmentului, iar obiectul de tip Bundle este transmis ca parametru. Pentru a face modificările vizibile pe ecran, adaptorul este anunțat de schimbările produse.

5.4 Structura bazei de date

Baza de date asociată acestei aplicații este una la distanță. Google cloud oferă un serviciu de bază de date non relațională care se numește *Firestore*. Acesta este o bază de date orientată pe documente, iar avantajele utilizării acesteia sunt scalabilitatea automată, performanța mare și sincronizarea în timp real pentru aplicația mobilă.
Datele sunt stocate în documente JSON care sunt organizate în colecții. Colecția nu există fără documente. Dacă se șterge ultimul document dintr-o colecție atunci colecția va dispărea. La crearea primului document dintr-o colecție se va crea automat colecția unde se va stoca respectivul document. Fiecare document conține câmpuri și are

asociat o cheie unică și este interogat folosind metode specifice: *get, set, delete, update.*

În cele ce urmează va fi prezentat modul de aranjare a datelor în baza de date cloud NoSql Firestore.

Atunci când un utilizator își creează contul pentru prima dată se creează un document cu o cheie unică în colecția numită “users”. Această colecție are un set de câmpuri ce reprezintă datele personale introduse de utilizator la crearea contului (data nașterii, e-mailul cu care se face autentificarea, nume, prenume, facultatea).

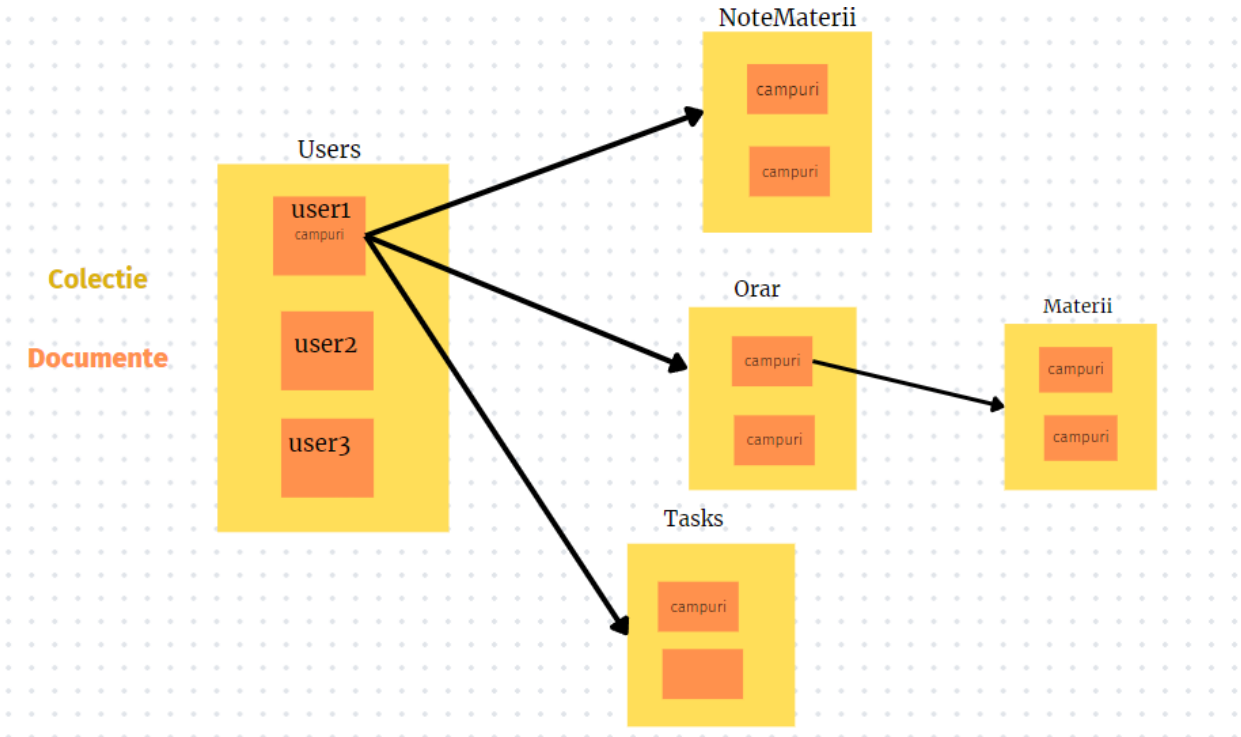
În funcție de datele introduse de utilizator pe parcursul aplicației, documentul asociat utilizatorului conține, pe lângă setul de câmpuri prezentat anterior, mai multe colecții.

Colecția numită **“NoteMaterii”** conține mai multe documente, fiecare document având o cheie unică. Un document are în componența sa două câmpuri salvate sub forma de hashmap. Primul reprezintă denumirea materiei, iar al doilea lista de note asociată respectivei materii. Atât lista de note, cât și denumirea materiei pot fi editate sau șterse, folosind operații specifice interogării bazei nerelaționate.

“Orar” este o colecție de documente, fiecare reprezentând o zi a săptămânii. Sunt 5 documente (de luni pana vineri), iar acestea au în componență un set de câmpuri ce reprezintă numele zilelor săptămânii și un ID asociat acesteia. ID-ul este de fapt un index, care pornește de la 0 pentru luni și se termină la 4 pentru ziua de vineri. Scopul său este de a asocia în mod corect materia introdusă într-o anumită zi, pentru că în baza de date aceste documente nu sunt salvate într-o ordine anume. Indexul ajută la identificarea corecta a zilei alese pentru o anumită materie. Atunci când utilizatorul introduce o materie în orar se creează colecția Materii, ale cărei documente reprezintă fiecare materie introdusa în acea zi. Fiecare document are numele materiei și are in componență un set de câmpuri (ora de început, final, sala, profesor).

“Tasks” este o colecție de documente ce reprezintă obiectivele introduse de utilizator. Structura unui document în acest caz este mai simplă decât documentele prezentate mai sus, întrucât conține doar un set de câmpuri (nivel de dificultate, termenul la care trebuie finalizat acel obiectiv, dacă a fost bifat sau nu, ziua când a fost introdus).

Astfel, la autentificare în aplicație, se preia documentul din colecția de utilizatori asociat respectivului utilizator iar de acolo, în funcție de necesitate, se accesează, se modifică sau se șterg documente doar de la acel utilizator. Pentru a accesa datele dintr-un document se preia referința colecțiilor și a documentelor necesare pentru a ajunge la datele dorite.



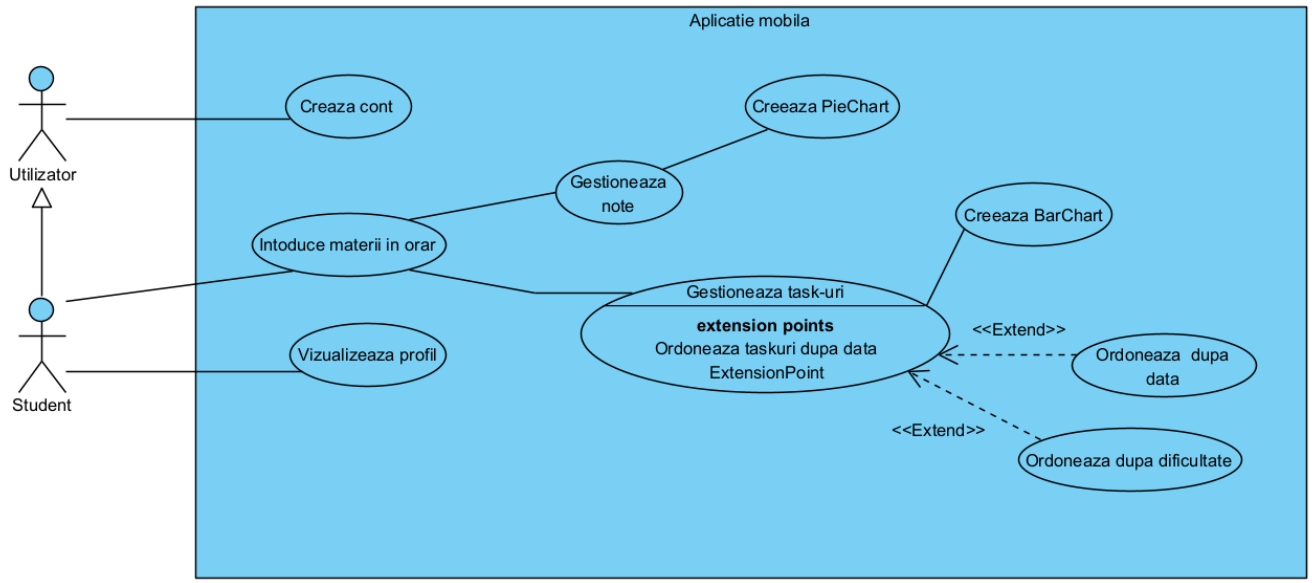
Figură 26 Structura bazei de date

Mai sus, în figura 26 este prezentată schema bazei de date NoSql.

5.5 Diagrama cazurilor de utilizare

Diagrama cazurilor de utilizare este o diagrama UML care descrie interacțiunea dintre actorii care folosesc sistemul și sistemul în sine. Cazurile de utilizare reprezintă funcționalități pe care sistemul informatic le oferă actorilor. Astfel, accentul nu este pus pe implementare, ci mai degrabă pe funcționalitățile și modul în care utilizatorul comunică cu sistemul. [12]

În figura 27 sunt analizate cazurile de utilizare a aplicației și modul în care acestea interacționează cu actorii sistemului. Actorul student moștenește funcționalitățile actorului utilizator. Studentul își introduce materii în orar și își vizualizează profilul cu datele introduse la crearea contului. Aplicația se bazează pe materiile introduse în orar. Pentru fiecare materie se introduc note, iar la solicitarea studentului se generează un grafic de tip PieChart cu notele aferente fiecărei materii.



Figură 27 Diagrama cazurilor de utilizare a sistemului informatic

În plus, pentru fiecare materie se introduc obiective care trebuie îndeplinite la o anumită dată, iar prin bifarea lor se urmărește progresul academic de pe parcursul anului școlar.

5.5 Descrierea cazului de utilizare „Gestionează obiective”

În tabelul 1 se va prezenta un caz de utilizare din diagrama cazurilor de utilizare care are o importanță semnificativă în aplicație.

Tabel 1 Descrierea cazului de utilizare „Gestionează obiective”

Element al cazului de utilizare	Descriere
Nume	Gestionează obiective
Scop	Salvarea si bifarea obiectivelor introduse de utilizator la o anumită materie
Actor principal	Studentul care este înregistrat în aplicație
Precondiții	Studentul a introdus deja materii în orar pentru a avea din ce materii să aleagă ca să introducă obiective
Post Condiții	Completarea corecta a tuturor câmpurilor atunci când se dorește înregistrarea unui nou obiectiv (atenție ca data de finalizare a obiectivului să depășească data curentă)
Declanșator	Selectarea din meniu a butonului care deschide activitatea de introducere a obiectivelor
Flux de bază	1. Utilizatorul își creează cont 2. Studentul introduce materii în orar 3. Studentul introduce obiective pentru anumite materii din orar 4. Studentul ordonează crescător sau descrescător obiectivele introduse 5. Studentul ordonează toate obiectivele după dificultate 6. Studentul verifică procentul de

	obiective bifate 7. Studentul generează o diagrama de tip bar chart
Relații cu alte cazuri de utilizare	• Acest caz de utilizare este extins de alte două cazuri de utilizare „Ordonează după dificultate ” și „Ordonează după dată”. • Acestea reprezintă două funcționalități care nu pot exista dacă cazul de utilizare pe care îl extind nu există, iar relația lor nu este implicită. • Cazul de utilizare extins există fără aceste două cazuri de utilizare, însă invers nu. • Există și cazul de utilizare „Creează BarChart” care este o funcționalitate ce se îndeplinește doar atunci când există obiective introduse.

Capitolul 6 Concluzii

Această aplicație a fost gândită și dezvoltată cu scopul de a oferi studenților de la CSIE un ajutor pe parcursul semestrului care constă în urmărirea progresului academic, a notelor, personalizarea orarului și vizualizarea graficelor de performanță. Totodată, aplicația a fost dezvoltată pentru Android, deoarece Android este cel mai popular sistem la nivel mondial de operare pentru dispozitivele mobile. Acest lucru înseamnă că are o posibilitate mai mare de penetrare a pieței care are ca public țintă studenții. De asemenea, Android rulează pe o gamă variată de dispozitive ceea ce conduce într-o mare măsură la extinderea publicului țintă. Un alt motiv pentru care s-a ales dezvoltarea unei aplicații Android este mediul de dezvoltare Android Studio care oferă unelte puternice pentru a scrie un cod, a verifica erorile și a testa codul scris.

Proiectul de licență bifează cerințele finale propuse. Acest lucru a fost posibil prin faptul că s-a reușit dezvoltarea unei aplicații mobile cu funcționalități diverse, care prezintă o estetică vizuală atractivă, bazată pe teme cromatice armonioase. Pe parcursul dezvoltării aplicației au intervenit diferite probleme cauzate fie de neînțelegerea cerințelor, fie de un mod de structurare mai puțin eficient a codului care pe alocuri au îngreunat rezolvarea respectivelor situații. Cu toate acestea, lucrul intens, răbdarea și perseverența în clarificarea și corectarea funcționalităților au fost calea către succesul realizării aplicației mobile, creată exclusiv prin forțe proprii. Astfel, timpul petrecut în realizarea acestui proiect a adăugat plus valoare cunoștințelor academice, fapt care a condus la dezvoltarea și a altor funcționalități pentru aplicație.

Cu toate acestea, au existat anumite limitări în ceea ce privește dezvoltarea aplicației neputându-se folosi funcționalități pe care Google le pune la dispoziție, de exemplu Google Maps. De aceea, o funcționalitate care nu a fost realizată este harta. Pe hartă se dorea să se salveze toate clădirile ASE, iar pe baza orarului să se creeze un traseu ținând seama de sălile pentru fiecare materie din ziua curentă. Într-o activitate specifică acestei funcționalități s-ar fi afișat pe hartă traseul care ar fi trebuit să fie urmat pentru schimbarea clădirilor în cazul în care ar fi fost necesar. Traseul ar fi fost desenat

pe hartă sub forma unei linii punctate și ar fi oferit cel mai scurt traseu pe care aplicația Google Maps nu o arată, întrucât sunt străduțe private, accesibile doar pietonilor. Datele despre materii ar fi fost preluate din baza de date, astfel încât acest traseu ar fi fost actualizat în permanență în funcție de ora și ziua curentă. Limitarea a constat în faptul că platforma Google Cloud care oferă un serviciu API pentru conectarea la hărți este contra cost și în funcție de interogările făcute, factura ar fi crescut substanțial.

Referitor la o posibilă extindere a aplicației ar putea exista diferite direcții de dezvoltare a acesteia. În primul rând, implementarea hărții în cadrul aplicației, în vederea personalizării acesteia pentru toți studenții de la ASE, nu doar pentru cei de la CSIE. Un argument ar fi faptul că studenții de la alte facultăți din ASE au cursuri în mai multe clădiri răspândite prin oraș, astfel încât implementarea hărții și-ar atinge mai rapid scopul. De asemenea, o funcționalitate unică ar fi dezvoltarea unui plan de gestiune a timpului în legătură cu deplasarea între cursuri, utilizând hărțile și sălile salvate pentru fiecare materie. Astfel, s-ar trimite notificarea utilizatorului înainte de ora de începere a cursului, iar în cazul în care acesta ar trebui să schimbe clădirea, aplicația ar trimite o notificare în momentul în care ar trebui să plece, precizându-se durata până la destinație. Pentru aceasta s-ar prelua locația curentă a utilizatorului pentru a se putea efectua calcularea distanțelor.

O altă funcționalitate interesant de implementat ar fi preluarea mai multor informații de la utilizator pentru calcularea unor parametri. De exemplu, introducerea regulata a timpului de studiu la o anumită materie, durata de realizare a anumitor proiecte, introducerea unor procente asociate fiecărei note. Toate acestea ar schimba focusul aplicației și către zona de lifestyle, dar și de gestiune mai eficientă a timpului dedicat studiului, astfel încât să poată fi mai bine administrat și timpul dedicat dezvoltării personale. Progresul sau declinul academic este în strânsă legătură cu stilul de viață al utilizatorului. De aceea, ar putea exista și un grafic cu o medie a orelor de somn. Pe baza tuturor acestor factori s-ar putea calcula o regresie din care să se previzioneze cu o probabilitate cât mai mare viitoarele note.

Bibliografie

[1] [Interactiv]. Available: https://soeonline.american.edu/blog/technology-in-education/. [Accesat 10 12 2022].

[2] [Interactiv]. Available: https://www.linkedin.com/pulse/importance-android-app-development-business-elizabeth-harris?trk=pulse-article. [Accesat 26 5 2023].

[3] [Interactiv]. Available: https://flathub.org/en-GB/apps/com.google.AndroidStudio. [Accesat 26 5 2023].

[4] Z. M. Sikora, Java: Practical Guide for Programmers, 2003.

[5] „Firestore Documentation- DataModel," Google Cloud, [Interactiv]. Available: https://cloud.google.com/firestore/docs/data-model. [Accesat 4 6 2023].

[6] A. S. Kumar, Mastering Firebase for Android Development, Packt Publishing, 2018.

[7] A. A. Dragos-Paul Pop, „Designing an MVC Model for Rapid Web Application Development," *Procedia Engineering,* vol. 69, pp. 1172-1179, 2014.

[8] L. Tian, „A comparison of Android native app architecture MVC, MVP and MVVM," *Eindhoven University of Technology,* p. 45, 2016.

[9] R. Mishra, „MVC (Model View Controller) Architecture Pattern in Android with Example," [Interactiv]. Available: https://www.geeksforgeeks.org/mvc-model-view-controller-architecture-pattern-in-android-with-example/. [Accesat 3 6 2023].

[10] „How to use MPAndroidChart in Android Studio!," [Interactiv]. Available: https://medium.com/@codingInformer/how-to-use-mpandroidchart-in-android-studio-c01a8150720f. [Accesat 19 06 2023].

[11] „What is Component Diagram?," [Interactiv]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/. [Accesat 19 06 2023].

[12] „Visual Paradigm," [Interactiv]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/. [Accesat 18 06 2023].

[13] B. Hambling și A. Samaroo, Software Testing: An ISEB Intermediate Certificate, British Computer

Society, 2009, p. 178.

[14] B. Hughes, Exploring IT for Business Benefit, Swindon: BCS Learning & Development Limited, 2008.

[15] M. Kobayashi-Hillary, Global Services: Moving to a Level Playing Field, Swindon: BCS Learning & Development Limited, 2007.

[16] [Interactiv]. Available: https://www.learnworlds.com/online-learning-platforms/. [Accesat 2 12 2022].

[17] [Interactiv]. Available: https://gmolsolutions.com/en/blog/what-are-educational-platforms-for/. [Accesat 3 12 2022].

[18] [Interactiv]. Available: https://www.educations.com/articles-and-advice/5-reasons-online-learning-is-future-of-education-17146. [Accesat 4 12 2022].

Tabel de figuri

Tabel de tabele

Tabel de acronime

ASE	Academia de studii economice
CSIE	Cibernetică Statistică și informatică economică
API	Application Programming Interface
IDE	Integrated development environment
SQL	Stuctured Query Language
NoSql	Not only SQL
MVC	Model-View-Controller
UML	Unified Modeling Language