Android mobilna aplikacija za evidenciju prisutnosti na nastavi

Ević, Ena

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:335381

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-08-10**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

ANDROID MOBILNA APLIKACIJA ZA EVIDENCIJU PRISUTNOSTI NA NASTAVI

Završni rad

Ena Ević

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 23.09.2019.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Ena Ević
Studij, smjer:	Preddiplomski sveučilišni studij Elektrotehnika i informacijska tehnologija
Mat. br. studenta, godina upisa:	4199, 27.09.2019.
OIB studenta:	78608905741
Mentor:	Doc.dr.sc. Josip Balen
Sumentor:	
Sumentor iz tvrtke:	Goran Luketić
Naslov završnog rada:	Android mobilna aplikacija za evidenciju prisutnosti na nastavi
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	23.09.2019.
Datum potvrde ocjene Odbora:	25.09.2019.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum: 2. rujna 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 02.09.2020.

Ime i prezime studenta:	Ena Ević
Studij:	Preddiplomski sveučilišni studij Elektrotehnika i informacijska tehnologija
Mat. br. studenta, godina upisa:	4199, 27.09.2019.
Turnitin podudaranje [%]:	12

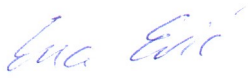
Ovom izjavom izjavljujem da je rad pod nazivom**: Android mobilna aplikacija za evidenciju prisutnosti na nastavi**

izrađen pod vodstvom mentora Doc.dr.sc. Josip Balen

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:



SADRŽAJ

1. UVOD....1
1.1. Zadatak završnog rada....1
2. KORIŠTENE TEHNOLOGIJE....2
2.1. Android operacijski sustav....2
2.2. Android Studio razvojno okruženje....4
2.3. Adobe XD vektorski grafički program....5
2.4. Firebase platforma....5
3. RAZVOJ APLIKACIJE....8
3.1. Primjena Material Designa i stvaranje prototipa....8
3.2. Razvoj aplikacije....10
3.2.1. Povezivanje s bazom podataka....10
3.2.2. Prijava u aplikaciju i implementiranje FirebaseUI biblioteke....11
3.2.3. Prikaz upisanih kolegija i detaljan pregled svakog....12
3.2.4. Odabir novih kolegija....14
4. ZAKLJUČAK....18
SAŽETAK....20
ABSTRACT....20
ŽIVOTOPIS....21

1. UVOD

Na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek (FERIT) studenti su obavezni prisustvovati na najmanje 70% nastave od ukupne predviđene satnice kolegija, a ukoliko student ne ostvari navedeni postotak na određenom kolegiju mora isti ponovno upisati iduće godine. Taj se postotak često realizira uz pomoć kolega koji slušaju ista predavanja i koji u trenutku evidentiranja nazočnosti potpisivanjem na listu dodaju i potpis studenta koji nije u tom trenutku prisutan. Iako ta metoda osigurava željeni potrebni postotak, dovodi studenta do gubitka kontrole nad prisutnošću, a u slučaju nepouzdanih kolega ne osigurava ni potreban broj nazočnih sati. U slučaju da student ne traži pomoć kolega, no zbog nekog razloga nije u mogućnosti doći na nastavu, korisna mu je informacija koliko još može izostati. Jedna od metoda praćenja nazočnosti je prebrojavanje skupljenih potpisa na potpisnoj listi, no to je moguće samo ukoliko su na listi navedeni svi datumi održanih predavanja.

Ovaj se završni rad bavi izradom Android aplikacije za praćenje nazočnosti studenata u koju student može samostalno unijeti za svaki željeni kolegij koliko je bio prisutan, odsutan, ali i koliko ga je puta netko upisao. Svaki se korisnik prijavljuje u aplikaciju, te se prikazuju podaci koji odgovaraju svakom korisniku. Prijava i spremanje podataka se obavlja preko Firebase platforme. Za izradu aplikacije korišteno je službeno integrirano razvojno okruženje - Android Studio, aplikacija je napisana programskim jezikom Java, a izgled aplikacije definiran je XML (engl. *Extensible Markup Language*) jezikom za označavanje podataka. U drugom poglavlju se teorijski opisuju korištene tehnologije i njihove glavne karakteristike koje su značajne za izradu aplikacije, dok se u trećem poglavlju govori o korištenoj arhitekturi aplikacije, implementaciji tehnologija iz drugog poglavlja, prototipu dizajna i izradi same aplikacije.

1.1. Zadatak završnog rada

Zadatak završnog rada je proučiti i opisati tehnologije za izradu mobilnih aplikacija za Android platformu i izraditi Android mobilnu aplikaciju koja će služiti studentima za evidenciju prisutnosti na nastavi. Aplikacija treba omogućiti prijavu i odjavu iz aplikacije, spremanje podataka u bazu i prikaz statistike.

2. KORIŠTENE TEHNOLOGIJE

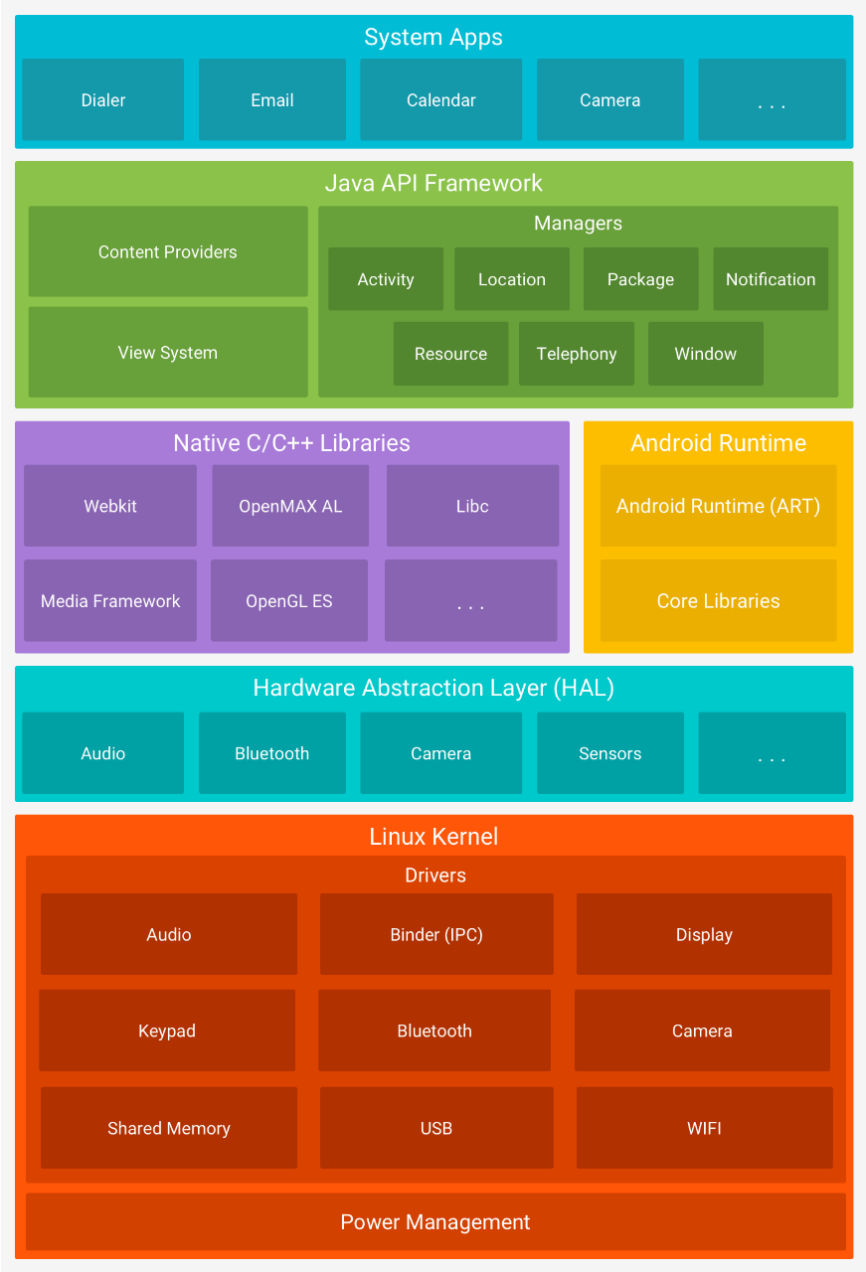
2.1. Android operacijski sustav

Android je naziv mobilnog operacijskog sustava (OS) otvorenog koda (engl. *Open-source*), baziranog na Linux kernelu, inicijalno razvijen u tvrtki Android Inc. [1], te je 2005. godine kupljen od tvrtke Google (Google). Google ga od tad intenzivno razvija - ne samo za pametne telefone, već i za tablete, pametne satove, stereo uređaje za aute, televizore i mnoge IoT uređaje. Upravo zahvaljujući toj raznolikosti brzo postaje najpopularniji operacijski sustav na cijelome svijetu. Iako je otvorenog koda i programeri ne moraju imati licencu za njega, proizvođači uređaja ne smiju koristiti zaštićen naziv „Android“ ako Google ne certificira uređaj, što znači da uređaj u tom slučaju iako radi na Android OS, ne može se nazvati Android uređajem niti može implementirati *PlayStore* i ostale Googleove aplikacije koje je potrebno naknadno instalirati jer ne dolaze ugrađene u sam uređaj [2]. Već od samih početaka, verzije Android operacijskog sustava dobivaju imena po slatkišima. Redom kako su verzije izlazile [3], to su:

- Android 1.5: Android Cupcake
- Android 1.6: Android Donut
- Android 2.0: Android Eclair
- Android 2.2: Android Froyo
- Android 2.3: Android Gingerbread
- Android 3.0: Android Honeycomb
- Android 4.0: Android Ice Cream Sandwich
- Android 4.1 to 4.3.1: Android Jelly Bean
- Android 4.4 to 4.4.4: Android KitKat
- Android 5.0 to 5.1.1: Android Lollipop
- Android 6.0 to 6.0.1: Android Marshmallow
- Android 7.0 to 7.1: Android Nougat
- Android 8.0 to Android 8.1: Android Oreo
- Android 9.0: Android Pie

Postoji mnogo načina za izradu mobilnih aplikacija, uključujući i mnoge alate koji su prilagođeni za više različitih operacijskih sustava, no takvi alati ne pružaju pogodnosti mnogih Android mogućnosti. Suprotno tome se razvijaju nativne (engl. *Native*) Android aplikacije koje imaju bolji pristup svim sadržajima uređaja kako bi se razvila što bolja aplikacija. Na taj način

aplikacija ima izravan pristup sučeljima za programiranje Android aplikacija. Čest je slučaj prelaska aplikacija programiranih za više sustava u nativne radi poboljšanja rada aplikacije za svaki sustav [4].



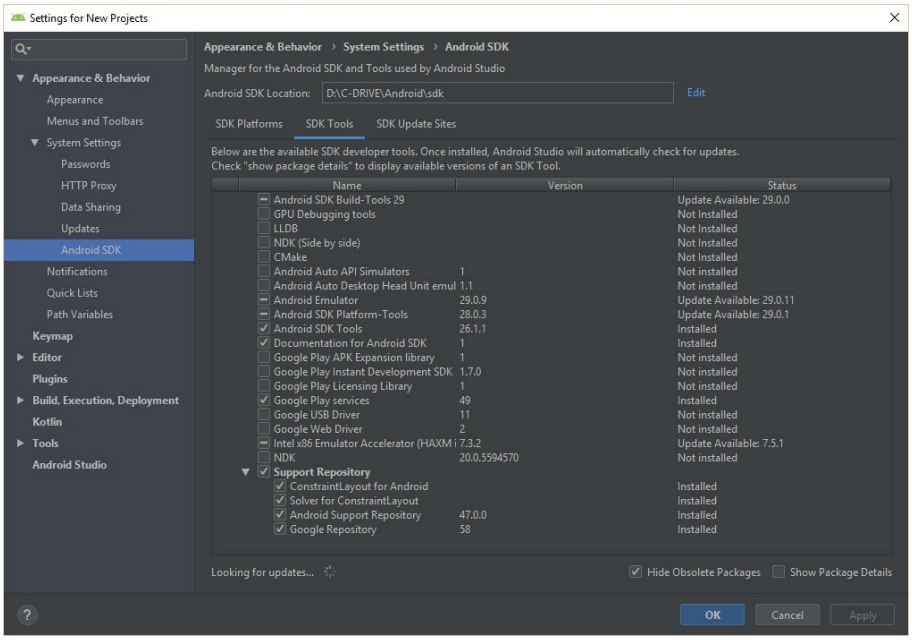
Slika 2.1 Arhitektura Android platforme [5]

Arhitektura Android platforme je prikazana na slici 2.1. Temelj Android platforme je Linux kernel zahvaljujući kojem proizvođači mogu lakše razvijati upravljačke programe, koji je oslonac Android Runtimeu za temeljne funkcionalnosti poput upravljanja nitima i upravljanje memorijom na nižoj razini, a njegovo korištenje također ima mnoge sigurnosne značajke. Hardware Abstraction Layer (HAL) je standardno sučelje koje sadrži i pruža mogućnosti sklopovlja višim

slojevima Androida, sadrži više modula biblioteka koje implementiraju sučelje za specifičnu komponentu sklopovlja (poput kamere, bluetootha...). Android Runtime služi za pokretanje višestrukih virtualnih strojeva na uređajima s malom količinom memorije izvršavajući DEX datoteke. Nativne C/C++ biblioteke pozivaju viši slojevi, a mogu ih koristiti zahvaljujući Android NDK (engl. *Native Development Kit*) koji je set alata koji omogućava korištenje C i C++ koda u Androidu i omogućava izravan pristup nativnim bibliotekama. Sljedeći je Java API Framework koji omogućava sva sučelja za programiranje aplikacija (engl. *Application Programming Interface,* API) napisana u Java programskom jeziku, a ta su sučelja gradivni blokovi za razvoj Android aplikacija. Zadnji sloj ove arhitekture su aplikacije sustava (engl. *System Apps*) koje dolaze pred-instalirane (kalendar, kamera, e-mail, i druge), a programeri mogu pozivati ove aplikacije umjesto da ih moraju sami implementirati [5]. Od 2017 Google je proglasio Kotlin službenim jezikom za razvoj Android aplikacija. Kotlin programski jezik je razvijen od JetBrains tvrtke koja je također razvila i Android Studio programsko okruženje, koje je u početku podržavalo samo Javu.

2.2. Android Studio razvojno okruženje

Android Studio je službeno integrirano razvojno okruženje (eng. *Integrated Development Enviroment* - IDE) za razvoj Android aplikacija koje se bazira na JetBrains IntelliJ IDEA softveru. Android studio je najavljen na Google I/O konferenciji u svibnju 2013. godine, a prva stabilna verzija je objavljena u prosincu 2014.

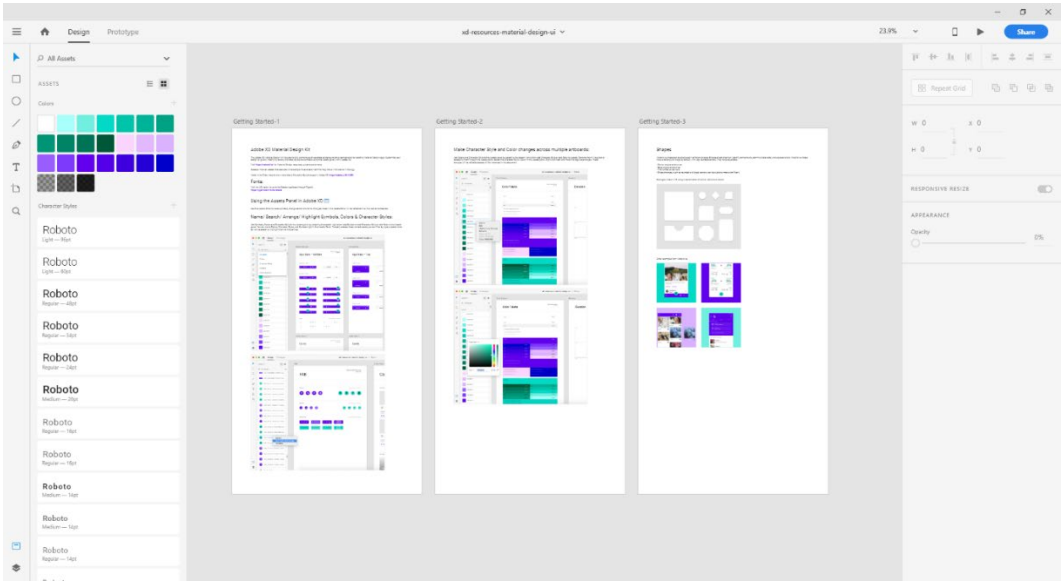


Slika 2.2. Prikaz SDK postavki unutar Android Studija

Danas ga je moguće instalirati na Windows, macOS i Linux operacijskim sustavima [6]. Kako bi se Android Studio mogao koristiti, potrebno je uz njega dodatno instalirati skup razvojnih alata Android SDK (eng. *Software Development Kit*) koji uključuju potrebne biblioteke, *debugger,* emulator, dokumentaciju i predloške koda [7].

2.3. Adobe XD vektorski grafički program

Adobe XD je vektorski grafički program koji je razvijen i u vlasništvu Adobe Inc. Koristi se za razvoj web i mobilnih aplikacija. Prednost mu je jednostavno i intuitivno korisničko sučelje, kao i mogućnost stvaranja animiranih promjena do kojih bi došlo stvarnim korištenjem aplikacije. Ima predefinirane dimenzije mobilnih ekrana, ali i mogućnost samostalnog definiranja dimenzija. Google Material Design, koji definira pravila po kojima bi Android aplikacije trebale biti dizajnirane, je integriran u Adobe XD te ga je jednostavno primijeniti [8].

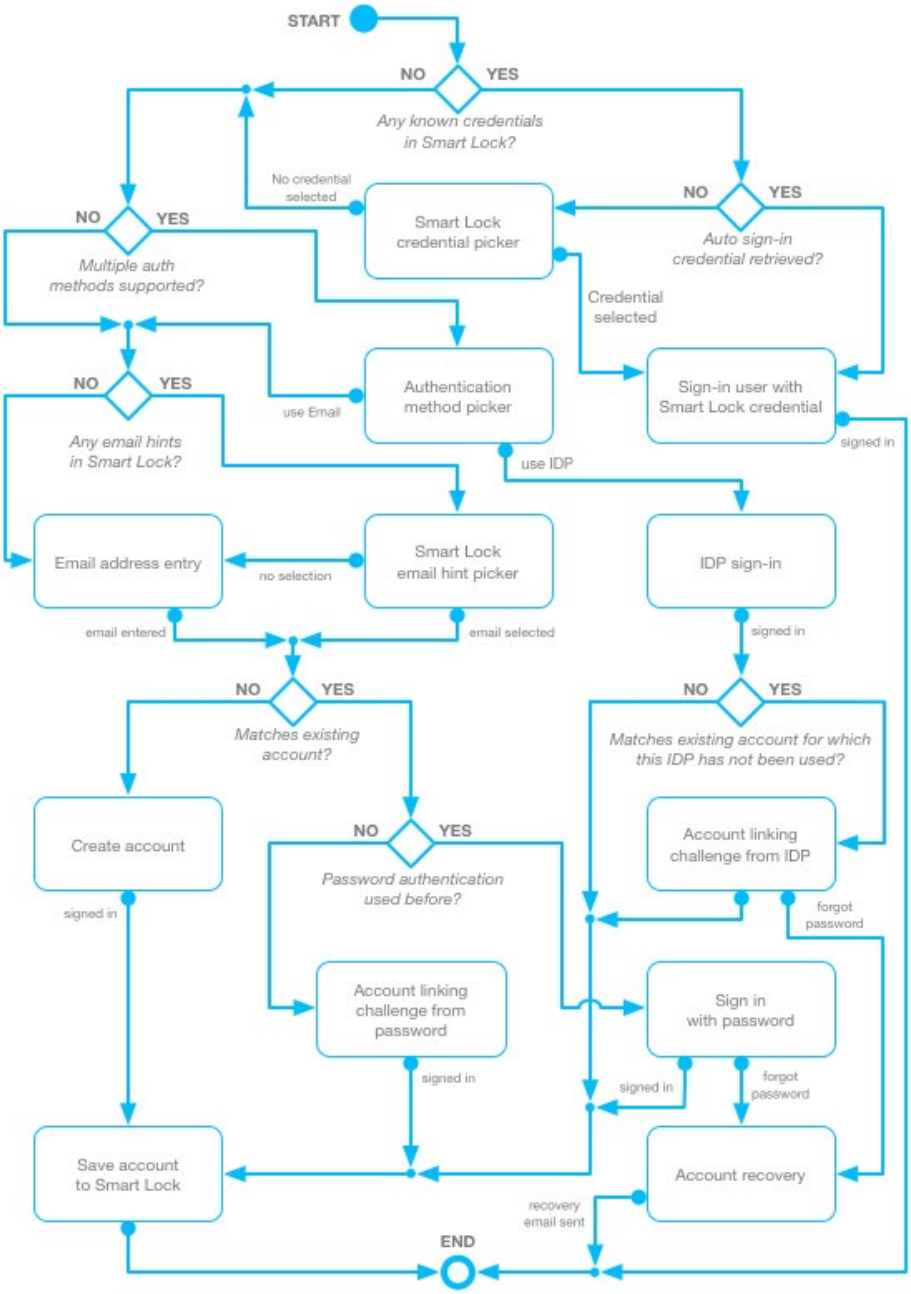


Slika 2.3. Izgled Adobe XD

2.4. Firebase platforma

Firebase je BaaS (engl. *Backend-as-a-Service)* platforma, namijenjena mobilnim i web aplikacijama. Razvija ju 2011. godine Firebase Inc, a 2014. ju kupuje tvrtka Google. Omogućava programerima da se usredotoče na to što će i kako će njihova aplikacija raditi, a ne gdje i kako će podaci iz aplikacije biti spremljeni. Za izradu ovoga rada najbitnije su Firebase baza podataka (Firebase Realtime Database) i Firebase sustav za autentifikaciju (Firebase Authentication). Još neki od korisnih proizvoda Firebase platforme su Firebase Analytics, Firebase Storage, Performance i još mnoge druge, a sveukupno ih je osamnaest [9]. Firebase Realtime Database je

no-SQL baza podataka koja se ažurira u stvarnom vremenu. Njezina je najveća prednost dohvaćanje i spremanje podataka putem jednostavnog aplikacijskog programskog sučelja. Podaci se čitaju i stavljaju u bazu na temelju Firebase pravila koja se nazivaju *Firebase Realtime Database Security Rules.* Zahvaljujući tim pravilima možemo reći tko može i pod kojim uvjetima pristupiti bazi, što dodatno dobiva na značaju kada se spoji s autentifikacijom korisnika. Prednost no-SQL baze je što koristi JSON (engl. *JavaScript Object Notation*) format za organizaciju baze, što znači da se podaci uz dobru organizaciju baze mogu dohvaćati trenutno koristeći nazive čvorova [10].



Slika 2.4 Tok prijave korisnika [11]

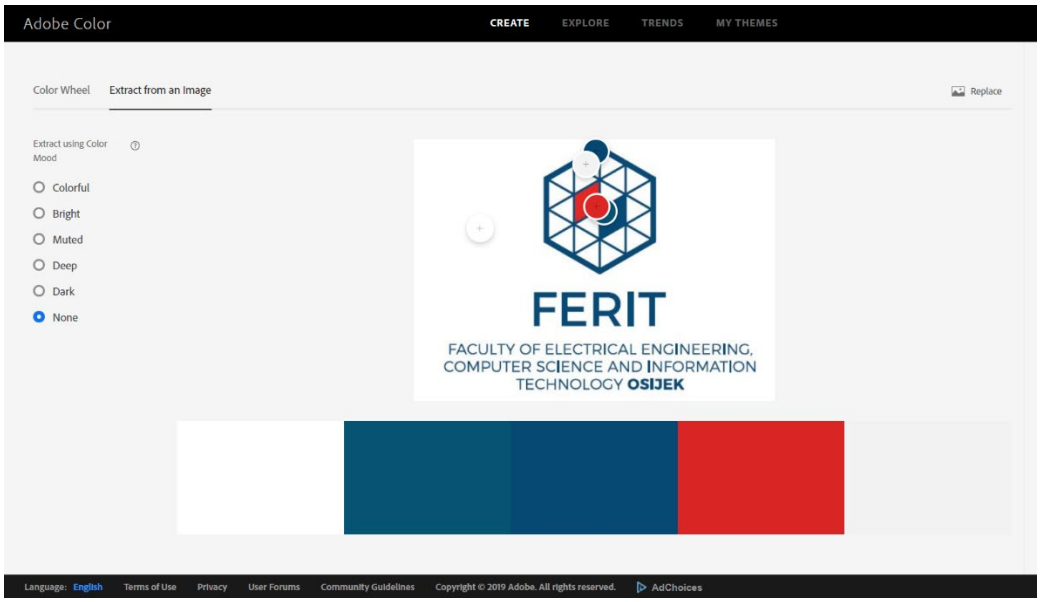
Većina aplikacija zahtjeva neki oblik identifikacije kako bi se podaci mogli spremati u bazu i kako bi dobili personalizirani sadržaj u aplikaciji. Firebase Authentication omogućava prijavu u aplikaciju bez da se moramo brinuti hoće li korisnički podaci biti sigurni i kako ćemo ih dohvatiti. Korisnik može anonimno pristupiti aplikaciji bez prijave, no najveća prednost su predefinirani načini prijave putem maila, telefonskog broja ili putem već kreiranog korisničkog računa na nekoj od društvenih mreža ili drugih web stranica (Google, Facebook, Twitter, Github). Te predefinirane načine prijave se može implementirati u sustav za prijavu koji smo sami napravili ili se može koristiti FirebaseUI - programska biblioteka koja između ostalog sadrži tijek prijave u aplikaciju koji je opsežan i ima puno raznih situacija i njihovih ishoda (slika 2.4) poput obnove podataka računa u slučaju zaboravljene lozinke, a FirebaseUI to sve kontrolira. Također je moguće promijeniti dizajn predefiniranog ekrana prijave i prilagoditi ga dizajnu aplikacije. Prilikom prijave stvara se jedinstveni identifikacijski ključ za novog korisnika koji se može iskoristiti za spremanje i dohvaćanje korisnikovih podataka u bazi [12]. Kako bi sve navedeno funkcioniralo, potrebno je u Firebase platformi odabrati na koje načine omogućavamo korisniku da se prijavi, u suprotnom prijava neće biti moguća. Može se omogućiti *Smart Lock for Passwords* koji sprema korisničke podatke (elektroničku poštu i lozinku) radi brže prijave.

3. RAZVOJ APLIKACIJE

Kako bi se aplikacija što uspješnije programirala, korisno je unaprijed znati izgled aplikacije – koliko ekrana će biti korišteno, koji će biti elementi na ekranima i kako će se ti elementi ponašati kada korisnik napravi interakciju na ekranu. Kako bi aplikacija bila intuitivna i kako bi kretnje elemenata i ekrana aplikacije bili korisniku jasni bez ikakvih uputa, potrebno je primijeniti principe Material Designa. Nakon što se pripremi vizualni izgled aplikacije, odabire se arhitektura programiranja aplikacije – u ovoj aplikaciji koristi se Model-Pogled-ModelPogleda (engl. Model-View-ViewModel, MVVM) arhitekturni obrazac koju Google preporučuje za stvaranje Android aplikacija. Prednost takve arhitekture je što odvaja logiku odabira i stvaranja sadržaja od prikaza sadržaja na ekranu.

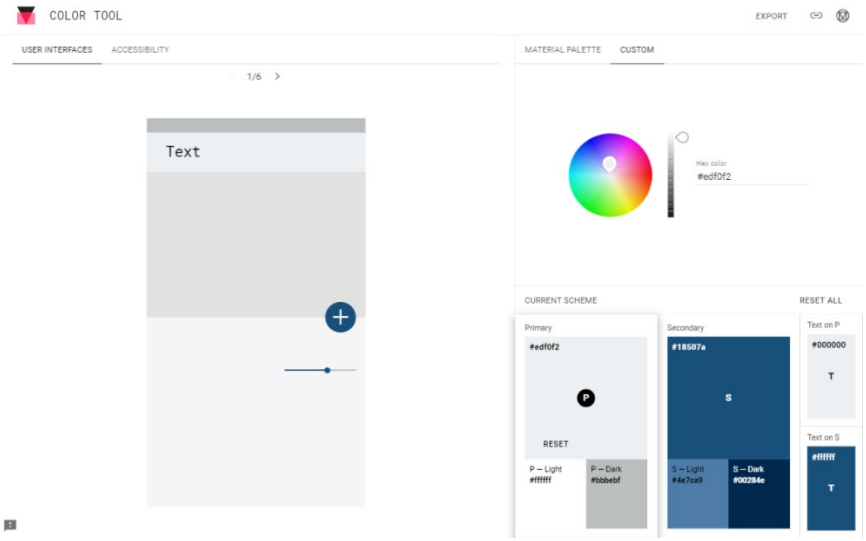
3.1. Primjena Material Designa i stvaranje prototipa

Material Design je naziv za jezični dizajn kojeg je razvio Google. Objedinjuje klasične principe dobrog dizajna koristeći tehnološke inovacije i znanost. Cilj Material Designa je stvoriti unificirani dizajn koji je prepoznatljiv i razumljiv korisnicima na svim uređajima i platformama. Smjernice i pravila su napisani tako da se mogu prilagoditi svakom brendu, pritom zadržavajući korisnicima poznate oblike, pokrete i interakcije. Iz logotipa FERIT-a pomoću besplatnog alata Adobe Color (slika 3.1) izdvojeno je pet boja, od kojih je odabrana jedna nijansa plave koja će se koristiti u dizajnu aplikacije.



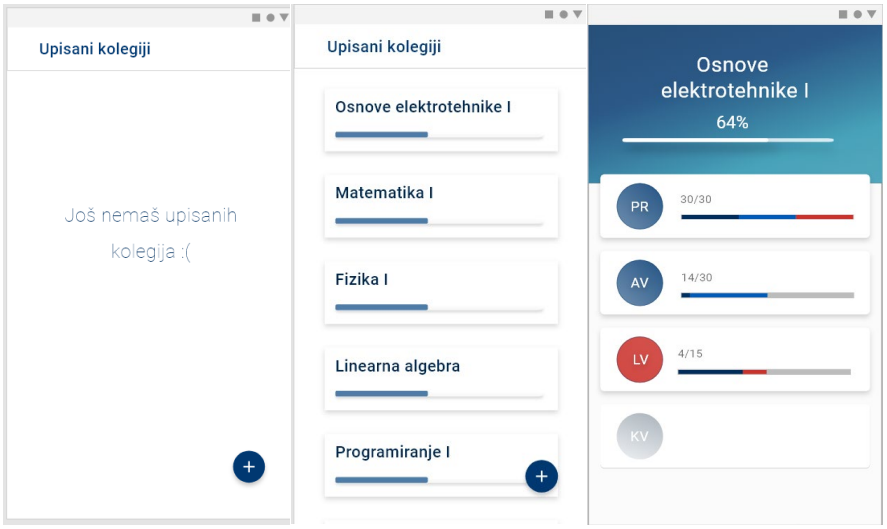
Slika 3.1 Adobe Color sučelje

Sve boje će se izražavati kao heksadekadske notacije komponenti boja (tzv. hex vrijednosti), koja definira kombinaciju crvene, zelene i plave (engl. Red, green, blue - *RGB*). Zapis izgleda kao kombinacija te tri vrijednosti - #RRGGBB, gdje je RR oznaka crvene komponente, GG zelene, i BB plave [13]. Hex vrijednost odabrane plave boje je #18507a. Prateći smjernice *Material Designa* u Adobe XD izrađujemo prototip kako će aplikacija izgledati. Potom, koristeći se alatom Color Tool (slika 3.2), odabiremo predefiniranu boju „Blue Grey“ u najsvjetlijoj varijanti za primarnu, te našu ranije spomenutu plavu za sekundarnu boju.



Slika 3.2 Sučelje Color Tool alata za primjenu Material Design*a*

Nakon što su odabrane boje i nakon što je proučen Material Design može se krenuti sa izradom prototipa u programu Adobe XD. Zahvaljujući FirebaseUI biblioteci ne moramo stvarati prototip ekrana za prijavu i registraciju jer su oni već definirani. Korisniku se nakon prijave, ili registracije, pokreće glavni zaslon na kojemu ima pregled svih kolegija koje sluša i za koje prati svoju nazočnost. U slučaju da se korisnik tek registrirao ili je obrisao sve kolegije, prikazuje mu se isti ekran no bez liste kolegija. Pritiskom na FAB (engl. *Floating Action Button*) otvara se novi prozor koji sadrži sve kolegije te omogućavaju odabir istih. U slučaju da korisnik ima upisanih kolegija, oni se na listi svih kolegija ne prikazuju. Jednom kada su kolegiji odabrani, korisnik jednim pritiskom otvara novi prozor gdje može dobiti vizualni pregled koliko je ostvareno nazočnosti, koliko ima izostanaka i koliko sati ostalo do kraja izvođenja nastave. Odabirom na tip predavanja može dodavati koliko je bio prisutan, odsutan ili koliko je upisan. Ti se ekrani mogu vidjeti na slici 3.3. Tijekom programiranja aplikacije neke značajke dizajna su se promijenile kako bi se korisniku dodatno olakšalo korištenje aplikacije.



Slika 3.3 Prototip dizajna aplikacije

3.2. Razvoj aplikacije

Kao što je ranije napisano, razvojno okruženje za izradu ove aplikacije je Android Studio, programski jezik je *Java*, a za definiranje izgleda koristi se *XML*. Potrebno je povezati aplikaciju s Firebase platformom kako bismo mogli omogućiti prijavu, spremanje i dohvaćanje podataka iz baze. Zatim stvaramo *.xml* datoteke u kojima kreiramo izgled aplikacije, i naposljetku stvaramo *.java* kod sukladno ranije spomenutom arhitekturnom obrascu MVVM. Aplikacija sadrži tri ključna zaslona: pregled upisanih kolegija, odabir novih kolegija i pregled sadržaja odabranog kolegija. Sva tri zaslona prikazuju se kao fragmenti koji se dodaju na *MainActivity*.

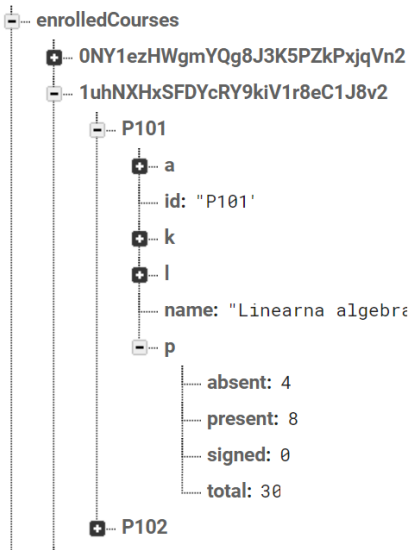
3.2.1. Povezivanje s bazom podataka

Prije nego se krene u povezivanje Android projekta s Firebaseom potrebno je provjeriti je li Android Studio verzija ažurna. Zatim je potrebno da projekt cilja na API razinu 16 (*Jelly Bean*) ili višu, i da je *Gradle* verzija 4.1 ili kasnija. Zadnji uvjet je posjedovanje Google računa putem kojega se prijavljuje na Firebase. Nakon što su ti preduvjeti zadovoljeni, pomoću Firebase konzole stvaramo projekt koji ćemo kasnije povezati s aplikacijom. Povezujemo aplikaciju tako da unutar novostvorenog Firebase projekta odaberemo dodavanje Android aplikacije i unesemo ID aplikacije (npr. *hr.domena.projekt*).

```
implementation 'com.google.firebase:firebase-database:19.1.0'
implementation 'com.google.firebase:firebase-auth:19.0.0'
implementation 'com.google.firebase:firebase-storage:19.0.1'
```

Programski kôd 3.1 Firebase dependencies

Zatim dodajemo konfiguracijsku datoteku (*google-services.json*i nadopunjujemo *Gradle* datoteku kako bi sve radilo (programski kôd 3.2.1). Zadnji je korak dodavanje Firebase SDK. Dio konfiguracijskih postupaka moguće je proći i iz samog Android Studija i tako ubrzati postupak. Svi podaci u aplikaciji su povučeni iz Firebase baze podataka. Za razliku od SQL baza podataka, u Firebase bazi podataka podaci se spremaju u JSON datoteku čija je prednost što se podaci mogu hijerarhijski slagati, gdje se svaki novi element prikazuje kao novi „čvor“. Potrebno je definirati čvor sa svim kolegijima, čvor s upisanim kolegijima i čvor s korisničkim podacima. U čvoru s kolegijima pod šifrom kolegija unosimo podatke o kolegiju: naziv, šifru, količinu sati za svaki tip izvođenja kolegija. Kod upisanih kolegija, potrebno je za svakog korisnika izdvojiti novi čvor, a za svaki upisani kolegij uz ukupan broj sati za tip predavanja dodati i broj dolazaka, izostanaka i upisa. Konačna struktura upisanih kolegija se može vidjeti na slici 3.4.



Slika 3.4 Struktura baze podataka

3.2.2. Prijava u aplikaciju i implementiranje FirebaseUI biblioteke

Jednom kada smo povezani s Firebase Platformom, sljedeći je korak omogućiti korisniku prijavu. Korisnik se prijavljuje putem Firebase Authentification, a tok prijave prati i kontrolira FirebaseUI, koji osigurava izgled ekrana za prijavu i registraciju, i također omogućava izbor između implementiranja prijave putem elektroničke pošte, broja telefona, Google računa, Facebook računa, Twitter računa ili pristup aplikaciji bez prijave (anonimno). U ovoj aplikaciji prijava se odvija putem elektroničke pošte ili Google računa. U *MainActivity.java* koji se pokreće prilikom pokretanja aplikacije, koji je ujedno glavni i jedini zaslon u aplikaciji, pozivamo funkciju *initializeFireBaseAuth* (programski kôd 3.2). Unutar te funkcije provjeravamo je li netko u aplikaciji prijavljen i ako je pokrećemo funkciju za provjeru je li korisniku to prva prijava. Ako

korisnik nije prijavljen pokreće se *startActivityForResult* koji otvara predefinirani prozor za prijavu i u *setAvailableProviders* unesemo koji tip prijave želimo omogućiti, sa *setTheme* postavljamo boje ekrana za prijavu. Funkcija *setUserRepository* dohvaća čvor s korisnicima u bazi podataka i provjerava je li korisnik nov. U slučaju da je to korisniku prva prijava, njegovi se podaci spremaju u bazu, a ukoliko nije prva prijava znači da su podaci već spremljeni i da aplikacija može krenuti s radom. Nakon što je prijava potvrđena, na *MainActivity* se dodaje fragment koji sadrži listu kolegija koje korisnik sluša, ako takvih ima. Ako nema kolegija koje sluša, tada se prikazuje poruka kako nema upisanih kolegija. U slučaju odjave je najbitnije prestati dohvaćati podatke tog korisnika, maknuti sve *listenere*, očistiti listu kolegija koji su se prikazivali i ponovo pokrenuti ekran za prijavu.

```
private void initializeFirebaseAuth() {
    mFirebaseAuth = FirebaseAuth.getInstance();
    mAuthStateListener = new FirebaseAuth.AuthStateListener() {
        @Override
        public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
            mFirebaseUser = mFirebaseAuth.getCurrentUser();
            if (mFirebaseUser != null) {
                mUserID = mFirebaseUser.getUid();
                setUserRepository();
            } else {
                onSignedOutCleanup();
                startActivityForResult(
                        AuthUI.getInstance()
                                .createSignInIntentBuilder()
                                .setIsSmartLockEnabled(false)
                                .setAvailableProviders(Arrays.asList(
                                        new uthUI.IdpConfig.GoogleBuilder().build(),
                                        new uthUI.IdpConfig.EmailBuilder().build()))
                                .setTheme(R.style.MyTheme)
                                .build(),
                        RC_SIGN_IN);
            }
        }
    };
}
```

Programski kôd 3.2 Prijava u aplikaciju

3.2.3. Prikaz upisanih kolegija i detaljan pregled svakog

Jednom kada je korisnik prijavljen, prvo što vidi na ekranu je lista upisanih kolegija. Ta se lista nalazi na fragmentu *EnrolledCoursesFrafment*, čiji *.xml* sadrži *ListView* kojemu predajemo listu upisanih kolegija i tekst koji se prikazuje ako nema kolegija. Izgled svakog elementa liste definiramo u posebnoj *.xml* datoteci koja sadrži naziv kolegija, numerički i grafički prikazan postotak nazočnosti. Sadržaj dohvaćamo putem funkcije *setLiveData* (programski kôd 3.3) od *EnrolledCoursesViewModel* koji obavlja svu logiku vezanu uz dohvaćanje sadržaja iz baze, i taj

sadržaj preko *LiveData* šalje fragmentu koji ima *observer* za taj konkretni *LiveData* koji u našem slučaju sadrži listu upisanih kolegija.

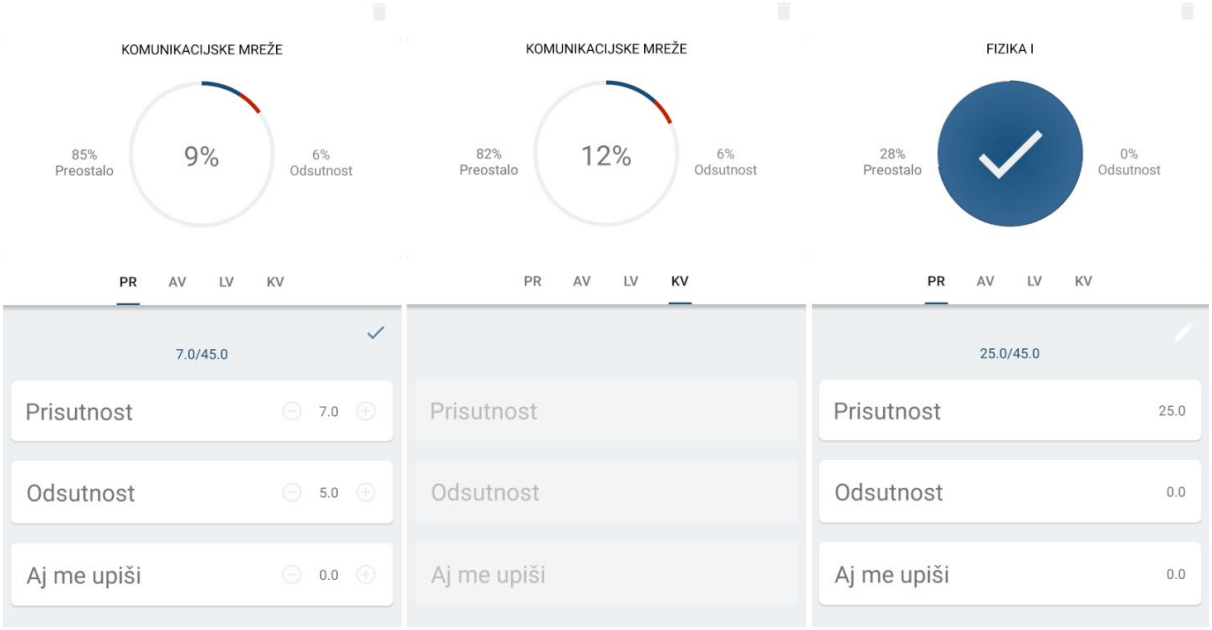
```
private void setLiveData() {
    mCourses = new ArrayList<>();
    mCourseAdapter = new CourseAdapter(getContext(), R.layout.item_course,
mCourses);
    mCourseAdapter.setNotifyOnChange(true);
    mCourseListView.setAdapter(mCourseAdapter);
    mViewModel = ViewModelProviders.of(this).get(EnrolledCoursesViewModel.class);
    mViewModel.init(mUserID);
    if (mObserver == null) {
        mObserver = new Observer<List<EnrolledCourse>>() {
            @Override
            public void onChanged(@Nullable List<EnrolledCourse> enrolledCourses) {
                mCourses = mViewModel.getEnrolledCourses().getValue();
                mCourseAdapter.clear();
                mCourseAdapter.addAll(mCourses);
                if (mCourses.size() == 0) {
                    mNoEnrolledCourses.setVisibility(View.VISIBLE);
                } else {
                    mNoEnrolledCourses.setVisibility(View.GONE);
                }
            }
        };
    }
    mViewModel.getEnrolledCourses().observe(this, mObserver);

}
```

Programski kôd 3.3 Dohvaćanje podataka iz EnrolledCoursesViewModel

Slika 3.5 Vizualne obavijesti ovisno o ostvarenom postotku

Kako bismo prikazali te podatke koristimo prilagođeni adapter. Svaki element liste prikazuje ikonu upozorenja ako je broj izostanaka prešao 20% ili prikazuje ikonu završenosti ako je broj nazočnosti prešao 70% (slika 3.5). Kada pritisnemo na jedan od elemenata liste otvara se novi fragment koji se naziva *CourseOverviewFragment* koji sadrži informacije o kolegiju kao što su naziv, postotak nazočnosti, postotak izostanaka i postotak koliko je još nastave preostalo i *PagerView* koji sadrži svaki od tipova izvođenja nastave (predavanja, auditorne, laboratorijske i konstrukcijske vježbe). Svaki element *PageViewera* sadrži i tri gumba na čiji se dodir broj sati

povećava za jedan, a pritiskom na ikonu za uređivanje prikazuju se tipke koje omogućavaju izmjenu trenutnog stanja u slučaju da je došlo do krivog unosa. Ako se kolegij ne izvodi na neki od četiri tipa izvođenja nastave tada gumbe u toj sekciji nije moguće pritisnuti, a kada se ostvari 70% nazočnosti nestaje ostvareni postotak i prikazuje oznaka završenosti (treći zaslon na slici 3.6.). Ako korisnik želi obrisati kolegij u gornjem desnom kutu se nalazi ikona za brisanje.



Slika 3.6 Mogući izgledi zaslona pregleda kolegija

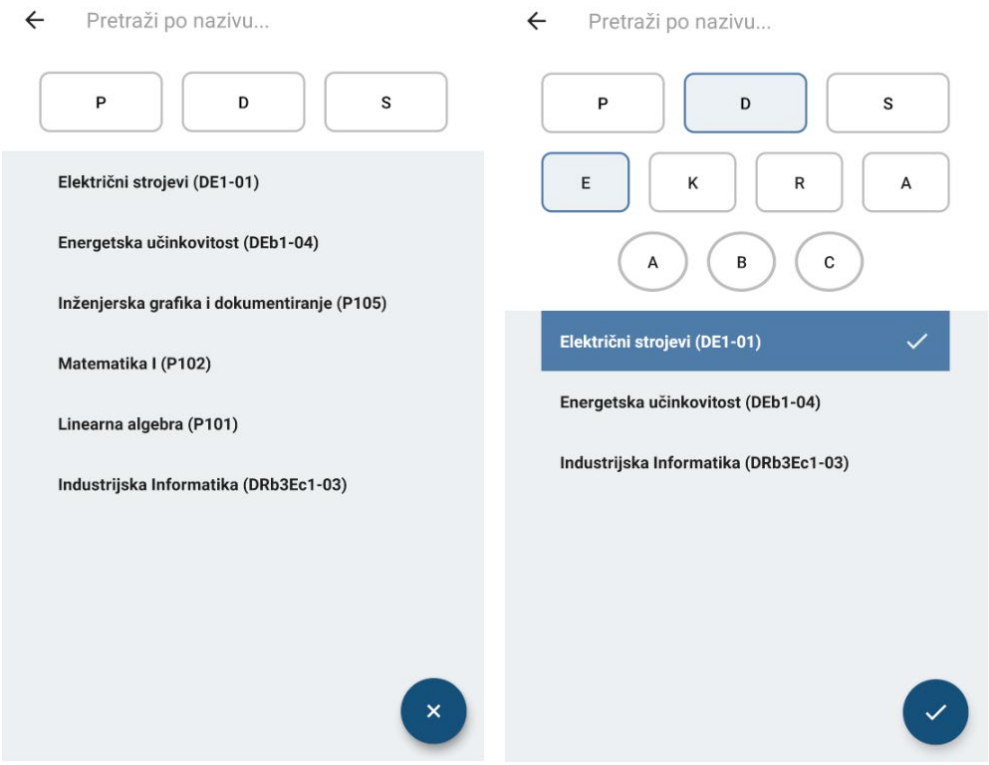
3.2.4. Odabir novih kolegija

Fragment odabira novih kolegija dohvaća podatke na način kao i fragment koji prikazuje upisane kolegije – poziva ViewModel koji komunicira s repozitorijem koji dohvaća podatke iz baze, samo što je njegov zadatak prikazati sve kolegije koji se mogu upisati. Ako korisnik već ima neke odabrane kolegije tada se isti neće prikazivati na ekranu za odabir. S obzirom na velik broj kolegija, korisnik može na dva načina pretražiti tu listu. Prvi je način da pritiskom na tipku za pretragu upiše naziv kolegija, a drugi je da - isto nakon pritiska gumba za pretragu – odabere po smjeru koji ga kolegiji zanimaju. Ako korisnik ne odabere niti jedan kolegij, FAB prikazuje ikonu za prekid radnje, i pritiskom na FAB se vraća na prethodni fragment. Ako je neki novi kolegij odabran FAB prikazuje ikonu za dodavanje kolegija, pritiskom na njega vraća se na prethodni fragment koji sada prikazuje i te nove kolegije i oni se odmah dodaju u bazu. Da je kolegij odabran se naznačuje promjenom boje pozadine i dodavanjem oznake u obliku kvačice. Elementi liste su prikazani preko *RecyclerViewa* za koji smo prilagodili adapter koji filtrira kolegije po nazivu uz filter koji smo nazvali *courseNameFilter* (programski kôd 3.4), a unos teksta se prati preko *setOnQueryTextListener* listenera koji je postavljen na *SearchView*.

```
private Filter courseNameFilter = new Filter() {
    @Override
    protected FilterResults performFiltering(CharSequence constraint) {
        List<Course> filteredList = new ArrayList<>();
        if (constraint == null || constraint.length() == 0) {
            filteredList.addAll(adapterCourseListFull);
        } else {
            String filterPattern = constraint.toString().toLowerCase().trim();
            for (Course : adapterCourseListFull) {
                if (course.getName().toLowerCase().contains(filterPattern)) {
                    filteredList.add(course);
                }
            }
        }
        FilterResults results = new FilterResults();
        results.values = filteredList;
        return results;
    }

    @Override
    protected void publishResults(CharSequence constraint, FilterResults results) {
        adapterCourseList.clear();
        adapterCourseList.addAll((List) results.values);
        notifyDataSetChanged();
    }
};
```

Programski kôd 3.4 Filter za pretragu kolegija po nazivu

Sve oznake su prikazane kao gumbi na čiji se pritisak pokreće funkcija *searchCourseById* (programski kôd 3.5) koja iz globalnih varijabli zna što mora filtrirati. Posebno pazi na slučaj kada kolegij nema drugu oznaku osim tipa studija, što znači da taj kolegij svi slušaju, te da se mora prikazati neovisno o odabranom smjeru. Ista logika se primjenjuje i za odabir pod-smjerova na diplomskom studiju. Na slici 3.7 se može vidjeti kako ovaj ekran izgleda, a u trenutku izrade završnog rada u bazi nisu bili svi kolegiji, stoga ih se ni na slici ne nalazi puno.



Slika 3.7 Izgled zaslona za odabir kolegija

```
private void searchCourseById () {
    List<Course> list = new ArrayList<>();
    List<Course> degreeList = new ArrayList<>();
    List<Course> electiveList = new ArrayList<>();
    List<Course> electiveGradList = new ArrayList<>();
    for (Course : allButEnrolledCourses) {
        if (course.getId().startsWith(DEGREE_LEVEL)) {
            degreeList.add(course);
        }
    }
    if (MODULE == null && GRAD_MODULE == null) {
        list = degreeList;
    } else if (MODULE != null) {
        for (Course course : degreeList) {
            Character ch = course.getId().charAt(1);
            if (!ch.equals('E') && !ch.equals('K') && !ch.equals('R') &&
!ch.equals('A') && !ch.equals('I')) {
                electiveList.add(course);
            } else if (course.getId().contains(MODULE)) {
                electiveList.add(course);
            }
        }
        list = electiveList;

        if (GRAD_MODULE != null){
            for (Course course : electiveList) {
                String mGm =MODULE+GRAD_MODULE;
                if (!course.getId().contains("a") && !course.getId().contains("b")
&& !course.getId().contains("c") && !course.getId().contains("d")) {
                    electiveGradList.add(course);
                }else if (course.getId().contains(mGm)) {
                    electiveGradList.add(course);
                }
                list = electiveGradList;
            }
        }
    }
    mAdapter.putData(list);
}
```

Programski kôd 3.5 Pretraživanje kolegija po šifri kolegija

4. ZAKLJUČAK

U ovome završnom radu proučene su i opisane tehnologije za izradu mobilnih aplikacija za Android platformu i izrađena je Android mobilna aplikacija koja služi studentima za evidenciju prisutnosti na nastavi. Opisane tehnologije koje su se primijenile za izradu aplikacije: Android operacijski sustav, Android studio razvojno okruženje, Adobe XD vektorski program za izradu prototipa web i mobilnih aplikacija, te Firebase platforma od koje smo koristili bazu podataka i sustav za autentifikaciju. Zatim je opisan izgled aplikacije, korištena arhitektura aplikacije i objašnjen je je korišteni programski kod u određenim dijelovima programa. Konačan izgled aplikacije se razlikuje od prototipa jer se prilikom izrade i testiranja predviđeni dizajn nije pokazao dobrim. Ova aplikacija je predviđena samo za studente FERIT-a, no u budućnosti je moguće nadograditi i za druge fakultete sveučilišta, dodati gamifikacijske elemente, ili nadograditi aplikaciju s dodatnim mogućnostima.

LITERATURA

[1] M. Karch, „What Is Google Android?“ https://www.lifewire.com/what-is-google-android-1616887, [lipanj 2019.].

[2] „Android (operacijski sustav)“ https://hr.wikipedia.org/wiki/Android_(operacijski_sustav), [rujan 2019.].

[3] J. Callaham, „The history of Android OS: its name, origin and more“ https://www.androidauthority.com/history-android-os-name-789433, [lipanj 2019.].

[4] J. Talbot i J. Mclean, Learning Android Application Programming, Addison-Wesley Professional, 2014. [rujan 2019.]

[5] „Platform Architecture“ https://developer.android.com/guide/platform, [lipanj 2019.]

[6] “Android Studio“ https://en.wikipedia.org/wiki/Android_Studio, [lipanj 2019.].

[7] “Android SDK” https://www.techopedia.com/definition/4220/android-sdk, [lipanj 2019.].

[8] “Adobe XD” https://en.wikipedia.org/wiki/Adobe_XD, [lipanj 2019.].

[9] “Firebase” https://en.wikipedia.org/wiki/Firebase, [lipanj 2019.].

[10] A. K. S, Mastering Firebase for Android Development, Packt Publishing , 2018. [rujan 2019.]

[11] FirebaseUI for Auth” https://firebaseopensource.com/projects/firebase/firebaseui-android/auth/readme.md, [rujan 2019.].

[12] N. Smyth, Firebase Essentials - Android Edition, CreateSpace Independent Publishing Platform, 2017. [rujan 2019.]

[13] “HTML kodovi boja” https://www.mojwebdizajn.net/skriptni-jezici/vodic/html/html-kodovi-boja.aspx, [lipanj 2019.].

[14] H. Yahiaoui, Firebase Cookbook, Packt Publishing, 2017. [rujan 2019.]

SAŽETAK

Cilj ovog rada je bio proučiti i opisati tehnologije za izradu mobilnih aplikacija za Android platformu i izraditi Android mobilnu aplikaciju koja će služiti studentima za evidenciju prisutnosti na nastavi. Završni rad proučava i opisuje Android operacijski sustav, službeno razvojno okruženje Android Studio i Firebase platformu. Izrađena Android aplikacija omogućava prijavu i odjavu iz aplikacije, spremanje unosa nazočnosti ili odsutnosti na kolegijima u bazu podataka, prikaz statistike nazočnosti i pretraživanje kolegija po nazivu ili šifri prilikom odabira.

Ključne riječi: pametni mobilni uređaji, mobilna aplikacija, Android, Java, Firebase

ABSTRACT

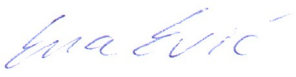
Android mobile application for class attendance records

The goal of this final paper was to study and describe technologies needed for making mobile application for Android platform and to make Android mobile application that will help students to keep track of their class attendance records. This final paper studies and describes Android operating system, the official Integrated Development Environment Android Studio, and Firebase platform. Android application that was made enables logging in and out of the app, it saves new record entries, it shows class attendance statistic and it enables searching courses by their name or by their id.

Keywords: Smart phones, mobile application, Android, Java, Firebase

ŽIVOTOPIS

Ena Ević rođena je 18. rujna 1997. u Osijeku. Upisuje Isusovačku klasičnu gimnaziju s pravom javnosti u Osijeku 2012. godine. Tijekom srednje škole volontira u Gradskom društvu Crvenog križa Osijek. Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek (FERIT) upisuje 2016. godine, a 2017. godine se pridružuje studentskom zboru FERIT-a gdje aktivno sudjeluje u organizaciji raznih događanja. Dobiva nagradu fakulteta za postignut akademski uspjeh 2019. godine. Član je IAESTE i IEEE udruga. Tečno govori engleski jezik i poznaje osnove njemačkog jezika.



Ena Ević