Android aplikacija za evidenciju prisustva na nastavi pomoću QR koda

Kovačević, Marko

Undergraduate thesis / Završni rad

2023

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:200:540558

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-02-28**

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

IZRADA ANDROID APLIKACIJE ZA EVIDENCIJU PRISUTNOSTI NA NASTAVI POMOĆU QR KODA

Završni rad

Marko Kovačević

Osijek, 2023.

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

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

Osijek, 31.08.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Marko Kovačević
Studij, smjer:	Programsko inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4518, 27.07.2020.
OIB Pristupnika:	10676448530
Mentor:	izv. prof. dr. sc. Ivica Lukić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Android aplikacija za evidenciju prisustva na nastavi pomoću QR koda
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak završnog rad:	Aplikacija treba imat dvije različite uloge: profesor i student. Profesor ima mogućnost dodavanja svojih kolegija i termina predavanja. Student može odabrati kolegij na kojem želi evidentirati svoju prisutnost. Nakon odabira kolegija u izborniku može odabrati predavanje i nudi mu se QR kod za evideniciju prisutnosti. Tema je rezervirana za: Marko Kovačević
Prijedlog ocjene 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 od strane mentora:	31.08.2023.
Datum potvrde ocjene od strane Odbora:	08.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 09.09.2023.

Ime i prezime studenta:	Marko Kovačević
Studij:	Programsko inženjerstvo
Mat. br. studenta, godina upisa:	R4518, 27.07.2020.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija za evidenciju prisustva na nastavi pomoću QR koda**

izrađen pod vodstvom mentora izv. prof. dr. sc. Ivica Lukić

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 2

2. PREGLED PODRUČJA TEME RADA 3

2.1. Postojeća slična rješenja 3

3. OPIS KORIŠTENIH TEHNOLOGIJA 6

3.1. Operacijski sustav Android 6

3.2. Android Studio 7

3.3. Kotlin programski jezik 8

3.4. XML 9

3.5. Firebase Firestore 10

4. RAZVOJ APLIKACIJE 12

4.1. Biblioteke 12

4.2. Uvod u rad aplikacije 13

4.3. Implementacija baze podataka 14

4.4. Dodatne značajke aplikacije 14

5. STRUKTURA APLIKACIJE 16

5.1. Početni fragment (engl. *Welcome Fragment*) 16

5.2. Registracijska aktivnost (engl. *Register Activity*) i aktivnost za prijavu (engl. *Login Activity*) 18

5.3. Glavna aktivnost (engl. *Main Activity*) 20

5.4. Fragment s kolegijima (engl. *My Subjects Fragment*) 21

5.5. Fragment s detaljima o kolegiju za profesore (engl. *Subject Details Professor Fragment*) 24

5.6. Fragment s detaljima o kolegiju za studente (engl. *Subject Details Student Fragment*) 25

5.7. Fragment za generiranje QR koda (engl. *QR Code Fragment*) 28
5.8. Aktivnost za očitavanje QR koda 29
5.9. Fragment s povijesti aktivnosti (engl. *Lecture History Fragment*) 32
5.10. Fragment s detaljima aktivnosti (engl. *Lecture Details Fragment*) 33
6. ZAKLJUČAK 35
LITERATURA 36
SAŽETAK 37
ABSTRACT 38
ŽIVOTOPIS 39

1. UVOD

Većina fakulteta u Republici Hrvatskoj u današnje vrijeme vodi evidenciju prisutnosti studenata na nastavi. Na taj način fakulteti se osiguravaju da studenti prisustvuju na aktivnostima te samim time olakšaju studentima polaganje kolegija. Uvriježeno je pravilo od 70 posto koje govori da bi student trebao prisustvovati 70 posto aktivnosti kolegija kako bi imao pravo izlaska na ispit i polaganja kolegija. Neke od mogućih aktivnosti, ovisno o fakultetu i kolegiju su: predavanja, auditorne vježbe, laboratorijske vježbe, konstrukcijske vježbe itd. Najčešći način vođenja evidencije na fakultetima je slanjem papira po učionici koji prisutni studenti zatim potpisuju. Neki profesori zatim prepisuju evidenciju prisutnosti s papira u *Excel* tablicu koju objave studentima kako bi mogli pratiti svoju prisutnost. Drugi profesori nakon svih održanih aktivnosti objave popis studenata te tko je zadovoljio minimalni kriterij a tko nije. Takav način evidencije prakticira i većina profesora Osječkog FERIT-a.

U ovom završnom radu pruža se alternativna mogućnost evidencije prisutnosti studenata. Tema ovog završnog rada je izrada Android aplikacije za evidenciju prisutnosti studenata na nastavi pomoću QR koda. Aplikacija je namijenjena i profesorima i studentima. Profesori imaju mogućnost generiranja QR koda te imaju uvid u razne podatke o dolaznosti studenata kao i sam popis studenata za svaku aktivnost. Kako bi student evidentirao svoju prisutnost mora otvoriti čitač unutar aplikacije te očitati generirani QR kod. Nakon uspješne evidencije student dobija potvrdu evidencije. Također student može lakše pratiti svoju prisutnost na različitim kolegijima te može lakše saznati je li postigao minimalan prag dolaznosti.

Prilikom izrade aplikacije korišteno je razvojno okruženje Android Studio, a programski kod pisan je u programskom jeziku Kotlin. Za dizajn je korišten alat Figma u kojemu se dizajnira izgled aplikacije. Zatim se pomoću dizajna iz Figme implementira dizajn u Android Studio razvojnom okruženju u jeziku za označavanje podataka XML (engl. *Extensible Markup Language*). Za pohranu podataka u bazu koristi se Firebase Cloud usluga koja se naziva Firestore Database. Firestore Database je *NoSQL* baza podataka.

Glavni dio rada podijeljen je u tri poglavlja: opis korištenih tehnologija, razvoj aplikacije i struktura aplikacije. U prvom poglavlju navode se korištene tehnologije, u drugom se opisuje proces razvijanja aplikacije a u trećem se navode fragmenti i aktivnosti koje čine aplikaciju.

1.1. Zadatak završnog rada

Pružiti alternativu klasičnom načinu evidencije priusutnosti studenata na nastavi. Izraditi Android aplikaciju koja će omogućiti profesorima lakše praćenje dolaznosti studenata te olakšati sam proces evidencije. Aplikacija će studentima pružiti brži i lakši pristup podatcima o vlastitoj prisutnosti na aktivnostima fakulteta.

2. PREGLED PODRUČJA TEME RADA

U ovom poglavlju spomenuti su i opisani razni aktualni primjeri sličnih primjena i rješenja kao i rješenje ove aplikacije. Neka od navedenih rješenja koriste QR kod isto kao i ova aplikacija, a neka koriste različite tehnologije ali služe sličnoj svrsi. Navedena rješenja koriste se u različite svrhe, neka su namijenjena evidentiranju prisutnosti radnika, neka evidentiranju prisutnosti studenata te za potrebne vođenja evidencije drugih organizacija.

2.1. Postojeća slična rješenja

QR TIGER:

QR Tiger pruža rješenje koje se oslanja na Google formu koja sadrži QR kod. Ovaj alat transformira poveznicu za Google formu u QR kod koji se zatim može očitati. Pomoću ovog sustava moguće je pratiti vrijeme dolaska, vrijeme odlaska, izostanke i ukupan broj sati proveden na poslu, fakultetu ili za potrebe drugih organizacija. Kako bi se QR kod osposobio za očitavanje prvo je potrebno napraviti par koraka. Prvo treba označiti formu, dodati naslov, sliku itd. Moguće je dodati ime svakog zaposlenika kako bi zaposlenici mogli označiti svoje ime prije očitanja. Zatim je potrebno unijeti poveznicu koji će se koristiti za pristup QR kodu. Zadnji korak je napraviti testno očitanje nakon čega se QR kod može koristiti.

Slika 2.1. QR Tiger - Qr kod generator (dostupno na: https://www.qrcode-tiger.com/)

Tvrtka HID Global:

Ova tvrtka nudi sustav koji pruža evidenciju prisutnosti radnika na radnom mjestu. Koristeći ovaj sustav poslodavac može pratiti koliko sati su zaposlenici proveli na poslu, kada su došli na radno mjesto, kada su otišli te razne statistike povezane s dolaznosti zaposlenika. Kako bi zaposlenici mogli koristiti ovaj sustav moraju preuzeti mobilnu aplikaciju HID Mobile Access. Kada se zaposlenik prijava u aplikaciji dobije svoju digitalnu akreditaciju koja se pohranjuje na mobilnom uređaju. Implementacija infrastrukture može biti različita od tvrtke do tvrtke no jedan primjer bi bio kada bi

tvrtka ugradila čitač na ulazu u zgradu. Zaposlenik bi u tom slučaju na ulazu i izlazu iz zgrade imao mogućnost povezivanja s čitačem pomoću *Bluetooth* tehnologije. Taj proces se odvija automatski kada se zaposlenik dovoljno približi čitaču. Aplikacija na mobitelu i čitač razmjenjuju enkrptirane autentifikacijske podatke kako bi se potvrdio identitet zaposlenika. Nakon identificiranja korisnika u sustav se unose podatci o vremenu dolaska te identitetu osobe. Ovakav sustav nudi prikladan i siguran način praćenja prisutnosti zaposlenika neke tvrtke.

Slika 2.2. Logo tvrtke HID Global (dostupno na: https://www.hidglobal.com/)

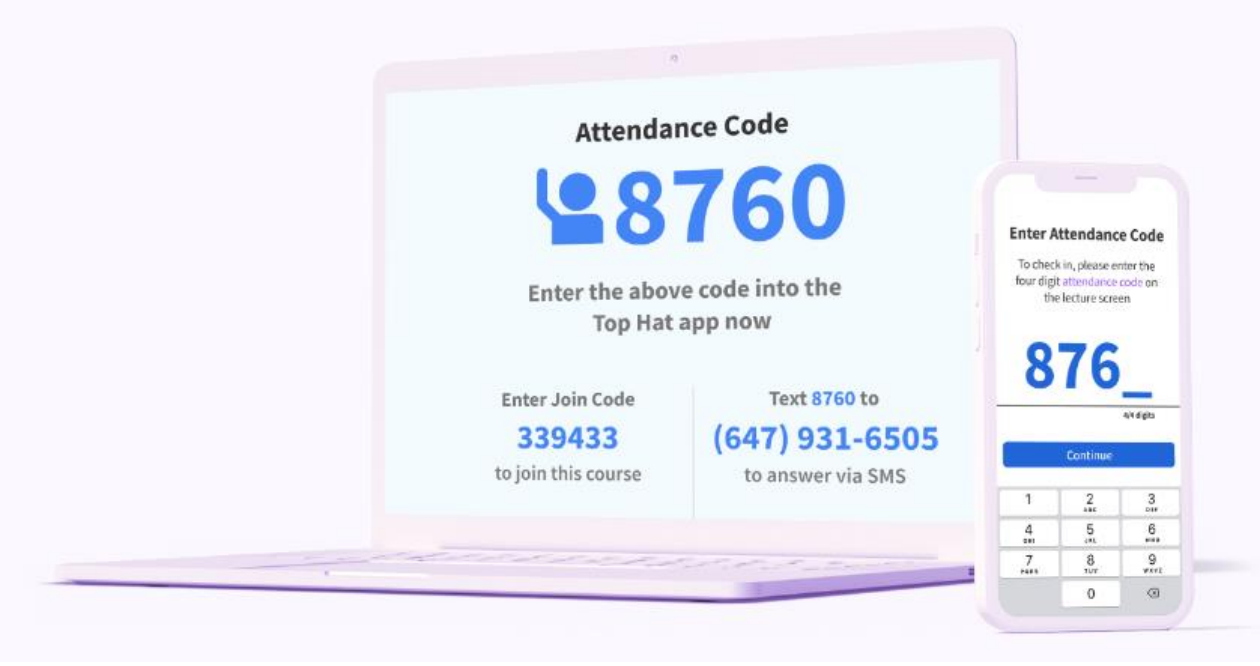
Tvrtka Connecteam:

Connecteam pruža uslugu praćenja vremena koje su zaposlenici proveli u tvrtci koristeći mobilnu aplikaciju. Ovo rješenje omogućava poslodavcima GPS praćenje kako bi znali gdje se zaposlenici nalaze tijekom radnog vremena. Svi podatci se spremaju u tablicu u kojoj su pohranjeni podatci o broju radnih sati, pauzama, prekovremenim satima i neradnim satima.

Slika 2.3. Logo tvrtke Connecteam (dostupno na: https://connecteam.com/)

TOP HAT:

TOP HAT je alat namijenjen za evidenciju studenata. Glavna svrha mu je olakšati i ubrzati proces evidentiranja studenata. Ovaj alat omogućuje automatiziranje evidentiranja prisutnosti za online ili predavanja uživo. Studenti imaju mogućnost poslati poruku na određeni broj koji će im profesor prikazati ili mogu putem aplikacije unijeti kod za prisutnost na određenoj aktivnosti. Moguće je koristiti geolokaciju kako bi se osigurali od toga da se studenti ne prijavljuju na aktivnosti od kuće. Putem aplikacije studenti mogu pristupiti statističkim podatcima o svojoj dolaznosti. Ovaj alat također nudi i evidentiranje performansi studenata, dijeljenje zadaće te razne pomoćne alate kao što su ankete, rasprave itd.



Slika 2.4. TOP HAT Kod za evidenciju studenata (isječak s https://tophat.com)

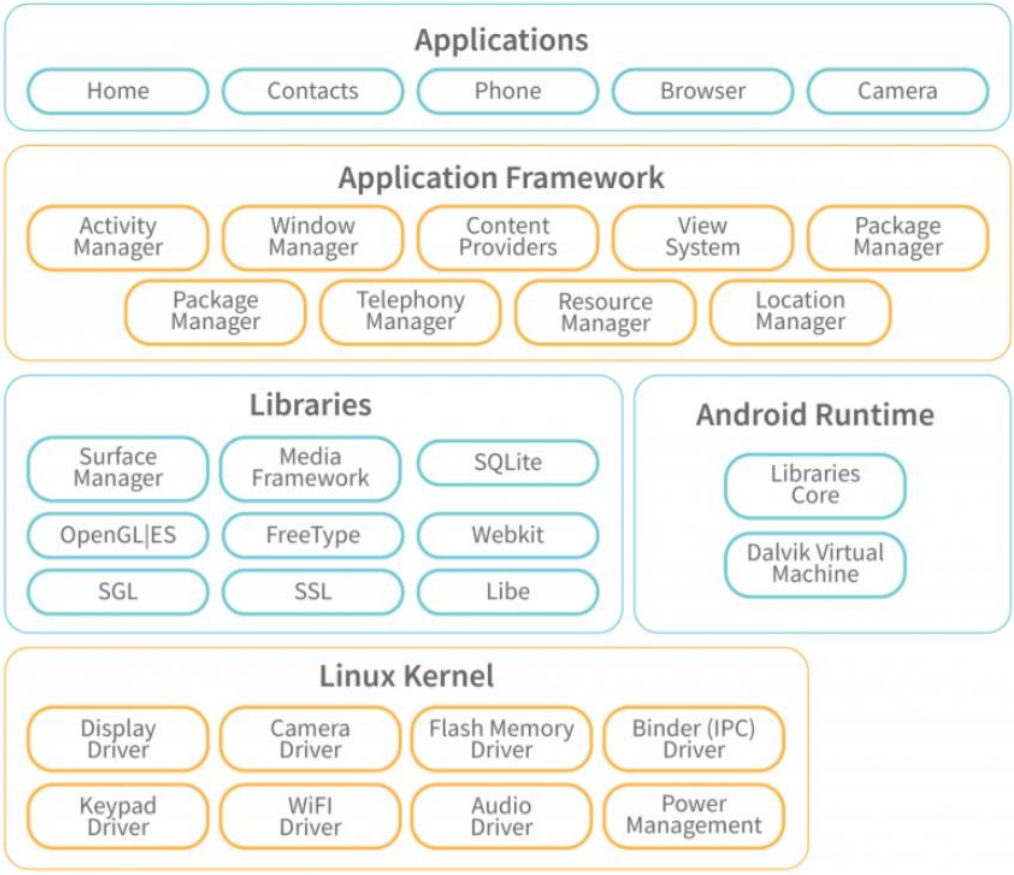
3. OPIS KORIŠTENIH TEHNOLOGIJA

U ovom poglavlju su opisane tehnologije koje su korištene prilikom izrade završnog rada. Aplikacija je razvijena za operacijski sustav Android. Za pisanje koda i *debagiranje* korišten je Android Studio. Kod je pisan u programskom jeziku Kotlin te je korišten XML za uređivanje korisničkog sučelja aplikacije. Baza podataka je realizirana u Google-ovom Firebase Firestore-u.

3.1. Operacijski sustav Android

Android je Google-ov operacijski sustav primarno korišten za mobilne uređaje s ekranom na dodir. Također postoje implementacije Android-a za razne uređaje kao što su pametni satovi, televizori te medijski sustav u automobilima. Ovaj operacijski sustav se razvio u jedan od najkorištenijih operacijskih sustava na svijetu te pokreće milijune uređaja različitih proizvođača. Temelji se na modificiranoj verziji Linux jezgre i drugih *open-source software-a* te je samim tim cijeli operacijski sustav *open-source*. To znači da je izvorni kod operacijskog sustava dostupan javnosti i moguće ga je modificirati. Upravo ta značajka Andorid sustava je najviše je doprinijela njegovom širenju te su različiti proizvođači imali mogućnost prilagođavanja sustava svojim *hardverskim* i dizajnerskim preferencama. Android arhitektura se sastoji od 4 sloja: aplikacije, *Application Framework*, sloj s bibliotekama i virtualnim strojem te Linux jezgra. Aplikacijski sloj je najviši sloj Android arhitekture, na ovom sloju su instalirane sve aplikacije. Na tom sloju nalaze se unaprijed instalirane aplikacije (kontakti, kamera, galerija, itd.) kao i aplikacije preuzete s *Google Store-a*. *Applications Framework* sloj pruža apstrakciju za *hardverski* pristup. Sadrži različite upravitelje i pruža razne usluge u obliku Java klasa. Na primjer, na ovom sloju se nalazi Upravitelj obavijesti koji omogućuje aplikaciji prikaz obavijesti korisnicima. Na trećem sloju nalaze se razne C, C++ i Java biblioteke te virtualni stroj dizajniran kako bi podržao pokretanje i rad Android sustava. Linux jezgra je najniži sloj Android arhitekture i on je zaslužan za upravljanje svim driverima kao što su kamera, ekran, zvuk, pohrana, itd. Android Inc. tvrtka je nastala u 2003. godini a 2005. tvrtku je kupio Google. Google je bio u procesu razvijanja Android sustava koji je tada bio namijenjen za mobitel s fizičkom tipkovnicom no pojavom prvog Iphone-a 2007. godine Google se morao okrenuti ekranima osjetljivim na dodir. HTC Dream je bio prvi komercijalni mobitel kojega je pokretao Android operacijski sustav, pušten je u prodaju 2008. godine. Tada je nastalo rivalstvo između Apple-ovog IOS-a i Google-ovog Androida koje i danas traje. Nedostatak Android sustava je što postoji velik broj njegovih distribucija koji su namijenjeni za različite uređaje što otežava razvoj aplikacija. Android *developerima* je otežan razvoj

aplikacija jer mnogo uređaja nema instaliranu najnoviju verziju te postoji jako puno različitih uređaja što nije slučaj kod Apple-a.

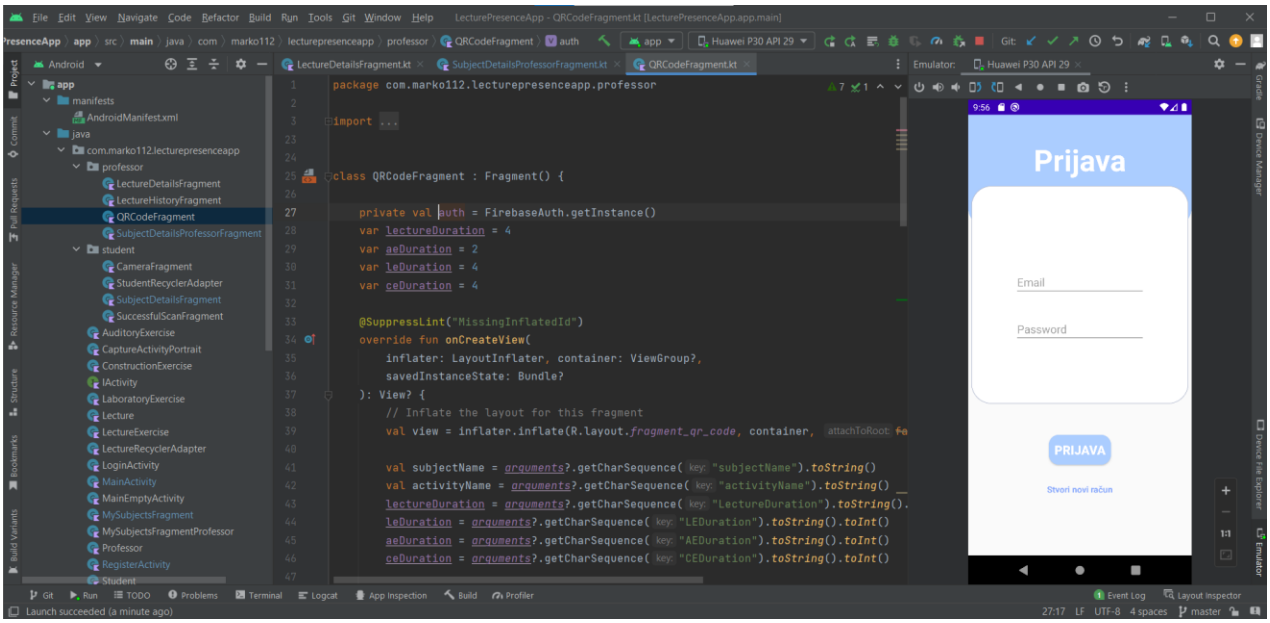


Slika 3.1. Arhitektura Android sustava (isječak s https://www.interviewbit.com/blog/android-architecture/)

3.2. Android Studio

Android Studio je službeno razvojno okruženje za Android operacijski sustav razvijen od strane Google-a. Svrha Android Studio okruženja je izrada i razvoj Android aplikacija. Ovo okruženje pruža različite alate koji značajno olakšavaju pisanje koda, testiranje te kreiranje završne verzije za objavu na *Google Store-u*. Postoji mogućnost korištenja emulatora pomoću kojega možemo testirati aplikaciju na virtualnom uređaju, a također je moguće spojiti fizički uređaj te testirati aplikaciju na njemu. Moguće je razvijati aplikacije za sve Android uređaje te je moguće specificirati s kojim verzijama Android sustava će aplikacija u izradi biti kompaktibilna. Struktura projekta u Android Studio-u se sastoji od 3 modula: Android *app* moduli, moduli s bibliotekama i *Google App Engine* moduli. Svaki *app* modul sadrži *manifest* datoteku koja predstavlja konfiguracijsku datoteku. Ona

sadrži bitne informacije (metapodatke) o aplikaciji kao što su komponente aplikacije, dopuštenja te razne druge detalje. *App* moduli sadrže i Java datoteke koje se sastoje od Java i Kotlin izvornih kodova. Također *app* moduli se sastoje od *res* datoteka koje su pisane u XML-u i sadrže razne vizualne dijelove aplikacije kao što su *layout*, *drawable* direktorij u koji se pohranjuju razne ikonice, *values* direktorij u koji se pohranjuju boje i sav tekst iz aplikacije, itd. Cijela aplikacija se zasniva na *Gradle build* sustavu koji automatizira proces izgradnje (engl. *build*) projekta. Proces izgradnje projekta uključuje *kompajliranje* izvornog koda, upravljanje ovisnostima, izvođenje testova i pakiranje aplikacije za distribuciju.



Programski kod 3.1. Isječak iz Android Studio razvojnog okruženja

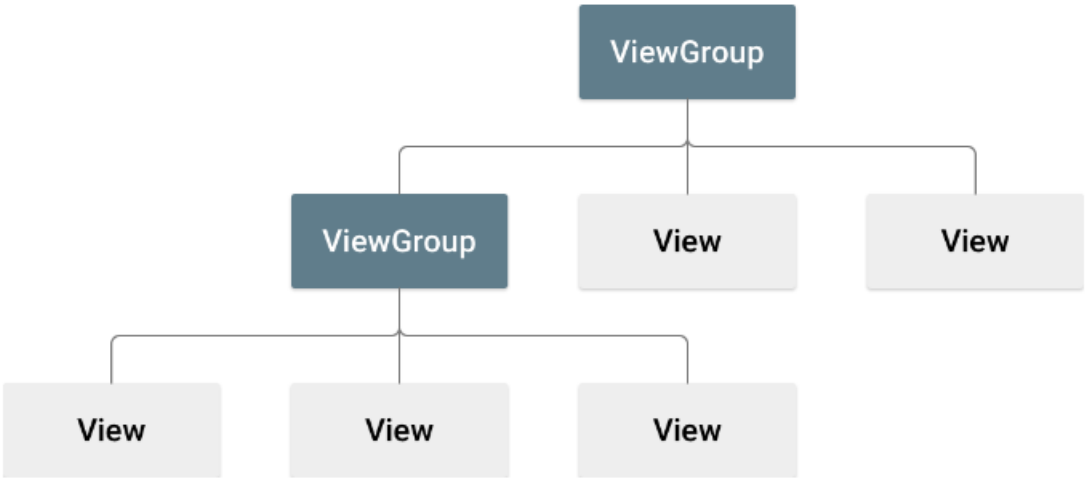
3.3. Kotlin programski jezik

Kotlin je moderan programski jezik sa statičkim tipovima koji je dizajniran za razvoj Android aplikacija. Radi sa statičkim tipovima što znači da tip svake varijable mora biti poznat prije *kompajliranja* izvornog koda. JetBrains tvrtka je započela s razvojem Kotlina u 2010. godini te su ga pustili u javnost kao *open-source* jezik u 2012. godini. Prva verzija Kotlina je izašla 2016. godine a od 2019. Google ga je proglasio službenim jezikom za razvoj Android aplikacija. Prije pojave Kotlina najznačajniji programski jezik u razvoju Android aplikacija bio je Java. Iako Kotlin ima ključne riječi *val* i *var* s kojima se deklarira nepromijenjivu i promijenjivu varijablu i dalje je *strongly typed* kao i Java što znači da za svaku varijablu mora biti poznat tip. Podržani su razni elementi funkcijskog

programiranja kao što su anonimne funkcije, lambde, generičke funkcije, *inline* funkcije itd. Iako Kotlin pruža mogućnost funkcijskog programiranja, također je zadržao i elemente objektno-orijentiranog programiranja. Velika prednost Kotlina nad Javom je upravaljanje *null-pointer* referencama. Standardni tipovi ne mogu biti *null*, no dodavanjem upitnika standardni tipovi postaju *nullable* te je moguće na elegantan način pozvati svojstva i metode na *nullable* tipovima bez iznimki i pogrešaka. Također je podržano i asinkrono programiranje koristeći ključne riječi *async* i *await*. Postoje i korutine koje možemo koristiti kako bi postigli asinkrono izvođenje koda te samim time i responzivnije iskustvo korištenja aplikacije. Moguće je u već postojeći Java projekt dodavati Kotlin klase i mogućnosti. Prednost Kotlina nad Javom je što sam kod izgleda elegantnije i značajno je manje linija koda nego u Javi.

3.4. XML

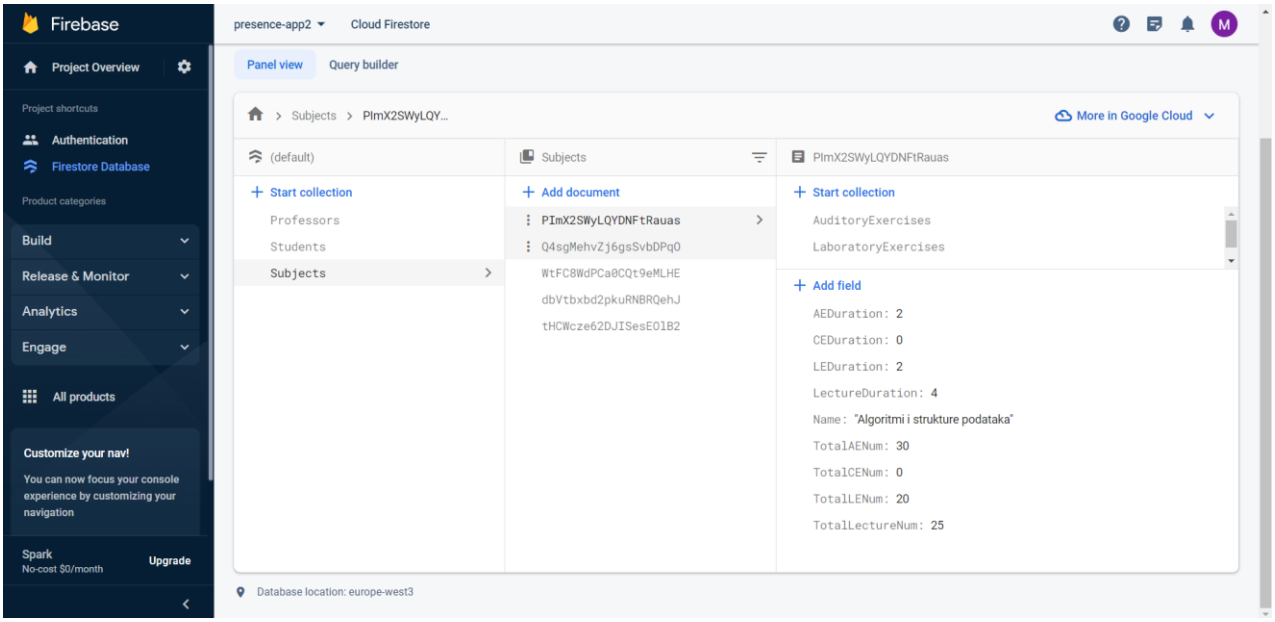
XML (engl. *Extensible Markup Language*) je „ozačni jezik“ pomoću kojega se opisuju podatci, sličan je HTML-u. Prilikom razvoja Android aplikacija XML se najviše koristi za kreiranje elemenata korisničkog sučelja. XML datoteke nalaze se unutar *res* direktorija unutar kojega su najčešći poddirektoriji *drawable, layout, values, mipmap,* itd. U *drawable* poddirektoriju nalaze se razne vrste slika uključujući JPEG, PNG, itd. Poddirektorij *layout* služi za pohranu XML datoteka koje predstavljaju izgled aplikacije (svaka *layout* datoteka se sastoji od elemenata korisničkog sučelja). Korisničko sučelje Android aplikacije temelji se na hijerarhijskoj strukturi koja se sastoji od *layout-a* (*ViewGroup*) i *widgeta* (*View*). Pomoću *layout-a* možemo pozicionirati *widgete* kao što su gumb, tekst, itd. *Values* direktorij sadrži sve boje i sav tekst koji se koristi u aplikaciji. Imati sav tekst aplikacije u jednoj datoteci (*strings.xml*) je korisno ukoliko je potreno prevesti aplikaciju na više jezika. U *mipmap* direktorij pohranjuju se slike koje se koriste za ikonu aplikacije namijenjene za različite rezolucije ekrana.



Slika 3.2. Primjer hijerarhije komponenti *layout-a* (isječak s https://developer.android.com/)

3.5. Firebase Firestore

Firebase je Google-ova platforma koja pruža raznovrsne usluge za razvoj mobilnih aplikacija te skaliranje istih. Jedna od ključnih usluga Firebase-a je Firebase Firestore – fleksibilna *NoSQL cloud* baza podataka. Firestore je *NoSQL* baza podataka što znači da nema unaprijed definirane strukture podataka te se po potrebi mogu dodavati polja u bazu. Takva baza se ne oslanja na relacije među tablicama kao tradicionalne *SQL* baze podataka. Podatci u Firestori bazi su pohranjeni u kolekcije dokumenata. Dokument se sastoji od polja koja mogu sadržavati jednostavne podatke kao tekst, broj ili složenije strukture kao polje ili parove ključa i vrijednosti. Također, dokument može u sebi imati pohranjenu kolekciju te se tako podatci mogu povezivati te se mogu slagati strukture podataka prema želji. Još jedna bitna usluga koju pruža Firebase je autentifikacija. Firebase autentifikacija pruža više načina autentifikacije korisnika kao što je korištenje email-a i lozinke, broj mobitela te prilagođeni način autentifikacije. Firebase pruža mogućnost slanja email-a za potvrdu kreiranja računa. Također postoji mogućnost provjere identiteta korisnika bilo gdje u aplikaciji.



Slika 3.3. Firebase Firestore baza podataka (isječak s https://console.firebase.google.com/)

4. RAZVOJ APLIKACIJE

U ovom poglavlju navedene su vanjske biblioteke koje su korištene kako bi se realizirale neke mogućnosti aplikacije. Korištene su razne biblioteke pomoću kojih su implementirane neke od ključnih komponenti aplikacije kao što su QR kod čitač i QR kod generator. U drugom potpoglavlju ovog poglavlja nalazi se uvod u rad aplikacije u kojemu su objašnjeni ciljevi aplikacije te način na koji su isti realizirani. U trećem potpoglavlju objašnjen je model baze podataka te sama struktura baze podataka. U zadnjem potpoglavlju četvrtog poglavlja navedene su neke dodatne značajke aplikacije.

4.1. Biblioteke

Vanjske biblioteke su u suštini već napisan kod koji programeri mogu uključiti u svoj kod kako bi uštedili vrijeme i kako ne bi morali pisati kod koji je već napisan. U ovoj cjelini navedene su biblioteke koje su korištene te je objašnjena njihova svrha i specifična implementacija za ovaj završni rad. U programskom kodu 4.1. prikazane su sve biblioteke koje su korištene za izradu aplikacije ovog završnog rada. *RecyclerView* biblioteka namijena je prikazu više elemenata na ekranu kroz koje se može navigirati pomicanjem prsta po ekranu. Prije *RecyclerView-a* koristio se *ListView* no potonji je zamijenjen jer je bio memorijski zahtjevan a *RecyclerView* je doskočio tome problemu tako što u danom trenutku učitava samo onoliko elemenata koliko stane na ekran. Za uspješnu realizaciju *RecyclerView-a* potrebna je *adapterska* klasa koja prilagođava podatke za prikaz pojedinog elementa. Za svaki pojedini element koristi se klasa *ViewHolder* koja predstavlja jedan element u kojoj se polja korisničkog sučelja postavljaju na željene vrijednosti. *RecyclerView* je u ovoj aplikaciji korišten za prikaz kolegija, aktivnosti (predavanje, auditorna vježba, laboratorijska vježba ili konstrukcijska vježba) te za prikaz studenata. Firebase ovisnost omogućuje komunikaciju s bazom, dohvaćanje podataka, spremanje novih podataka te autentifikaciju. *ConstraintLayout* služi za uređivanje korisničkog sučelja tako što omogućuje raspoređivanje i fiksiranje vizualnih komponenti. Zxing-android-embedded je vanjska biblioteka koja pruža klase koje znatno olakšavaju generiranje i čitanje QR koda.

```
dependencies {
    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.6.0'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'com.google.firebase:firebase-firestore-ktx:24.4.1'
    implementation 'androidx.annotation:annotation:1.5.0'
    implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.5.1'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.5.1'
    implementation 'com.google.firebase:firebase-auth-ktx:21.1.0'
    implementation 'androidx.recyclerview:recyclerview-selection:1.1.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'com.journeyapps:zxing-android-embedded:4.2.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

Programski kod 4.1. Biblioteke koje su korištene za izradu aplikacije

4.2. Uvod u rad aplikacije

Cilj ove aplikacije je pružiti jedinstveno sučelje koje će olakšati evidenciju i praćenje aktivnosti na fakultetu profesorima i studentima. Profesori na vrlo jednostavan način mogu kreirati jednu od četiri dostupne vrste aktivnosti (predavanje, auditorne, laboratorijske i konstrukcijske vježbe). Kada se aktivnost kreira generirani QR kod se prikaže na ekranu. QR kod generator kao ulaz dobije nekakav tekst te na temelju tog teksta generira QR kod – jedinstveni uzorak piksela koji se može očitati te na taj način pristupiti tekstu kojim je generiran. U slučaju ove aplikacije za tekst QR koda koriste se ID aktivnosti, vrsta aktivnosti te trajanje aktivnosti što se može vidjeti iz programskog koda 4.2. Podatci koji se dobiju očitanjem QR koda se zatim mogu koristiti kako bi student upisao svoju pristnost na određenoj aktivnosti. Na taj način dobiveno je vrlo jednosavno korisničko sučelje preko kojega student očitava QR kod a u pozadini se studentu prisutnost upisuje u bazu podataka. Profesor ima mogućnost dijeljenja QR koda na svoju e-mail adresu te može prikazati QR kod svim studentima koristeći projektor. Nadalje, profesori imaju uvid u sve održane aktivnosti iz svakog kolegija koji oni drže. Nakon što profesor odabere vrstu aktivnosti čiju povijest želi vidjeti i pritisne gumb „Povijest“ dobit će prikaz svih prijašnjih aktivnosti. Za svaku aktivnost čuvaju se ključne informacije kao što su vrijeme održavanja, profesor koji je održao aktivnost, popis studenata, broj studenata te redni broj predavanja. Sa studentske strane, aplikacija je izuzetno korisna jer studenti imaju jedinstven uvid u svoju prisutnost na svim kolegijima koje pohađaju. Njihova prisutnost prikaza je brojčano i pomoću trake za napredak (engl. *progress bar*). Studenti mogu lako vidjeti koliko sati određene aktivnosti se

do sada održalo te koliko su sati bili prisutni te u skladu s tim informacijama lakše mogu planirati i uskladiti sve svoje obveze.

```
//this string will be used for qr code creation
val qrCodeString = "$lessonId/${lessonInfo?.activityType}${lessonInfo?.activityDuration}"
```

Programski kod 4.2. Tekst na temelju kojeg je generiran QR kod

4.3. Implementacija baze podataka

Baza podataka je implementirana u Firebase Firestore-u. Firebase Firestore pohranjuje podatke kao dokumente koji se sastoje od raznih tipova polja. Ta polja mogu biti jednostavna kao običan broj ili kompleksna kao niz brojeva ili struktura ključeva i vrijednosti te je moguće ići i u još veću dubinu i imati pohranjene strukture unutar struktura. Dokumenti se pohranjuju u kolekciji ali i sami dokumenti mogu imati pohranjene kolekcije dokumenata u sebi. Struktura aplikacije ovog završnog rada je osmišljena na način da postoje četiri vrste kolekcija od čega su tri kolekcije na vrhu hijerarhije. Kolekcije su sljedeće: Profesori, Studenti, Kolegiji i Aktivnosti. Kolekcija profesora je najjednostavnija, sastoji se od tri polja a to su: ime profesora, e-mail profesora te popis kolegija koje drži. Kolekcija studenata sastoji se od četiri polja: ime, e-mail, popis kolegija te prisutnost studenta. Prisutnost se sastoji od dva sloja parova vrijednosti. Kod unutarnjih parova ključ je tip aktivnosti a vrijednost je broj koji predstvalja prisutnost studenta. Kod vanjskih parova ključ je ID kolegija a vrijednost je unutarnji par ključa i vrijednosti naveden u rečenici prije. Kolekcija kolegija sadrži razne podatke o kolegijima kao i dodatne četiri kolekcije pohranjene u svakom dokumentu. Te četiri kolekcije predstavljaju aktivnosti pa samim time imamo sljedeće kolekcije: Predavanja, Auditorne vježbe, Laboratorijske vježbe te Konstrukcijske vježbe. U svakoj od tih kolekcija nalazi se niz dokumenata, a svaki dokument sadrži ključne podatke o pojedinoj aktivnosti koji se prikazuju profesorima i pružaju im uvid u povijest aktivnosti.

4.4. Dodatne značajke aplikacije

Na vrhu korisničkog sučelja aplikacije nalazi se aplikacijska traka (engl. *App bar*) na kojoj se nalazi naziv aplikacije. Na naziv aplikacije postavljen je mehanizam koji čeka na pritisak (engl. *Listener*). Kada je pritisnut naziv aplikacije na aplikacijskoj traci pokreće se transakcija fragmenta (engl. *Fragment transaction*) i otvara se fragment dobrodošlice ukoliko korisnik već nije na tom fragmentu. Na taj način naslov djeluje kao tipka koja vodi na početni ekran (engl. *Home Button*). Prilikom kreiranja svake transakcije fragmenta poziva se metoda *addToBackStack* koja omogućuje vraćanje

natrag pritiskom na gumb natrag na uređaju. Ukoliko se korisnik nalazi na početnom fragmentu pritisak na gumb natrag rezultirati će izlaskom iz aplikacije.

5. STRUKTURA APLIKACIJE

U ovoj cjelini prikazana je i objašnjena struktura aplikacije i sam tijek izvođenja aplikacije. Struktura aplikacije uključuje sve Android aktivnosti i fragmente koji su kreirani i korišteni za rad aplikacije. Aktivnost najčešće predstavlja jedan ekran aplikacije koji pruža korisničko sučelje i funkcionalnost. Aktivnosti omogućuju navigiranje kroz razne ekrane aplikacije tako što jedna aktivnost pokrene drugu koristeći *Intent* – apstrakcija koja predstavlja operaciju koja se treba izvesti. Fragmenti se koriste kako bi kompleksno sučelje razbilo u više manjih komponenti. Svaki fragment mora biti dio neke aktivnosti te je samim time ovisan o aktivnosti unutar koje se nalazi. Životni vijek fragmenta traje onoliko koliko i životni vijek aktivnosti unutar koje se nalazi. Za razliku od fragmenta, aktivnosti su nezavisne. Lakše je i brže upravljati fragmentima nego aktivnostima što je doprinijelo njihovoj popularnosti. Struktura aplikacije ovog završnog rada sastoji se od tri aktivnosti: Aktivnost za registraciju, aktivnost za prijavu te glavna aktivnost. Glavna aktivnost sadrži *FragmentContainerView* pomoću kojeg je moguće mijenjati fragmente ovisno o unosu korisnika. U ovoj aplikaciji koristi se niz fragmenata koji su opisani u sljedećim potpoglavljima.

5.1. Početni fragment (engl. *Welcome Fragment*)

Prvi fragment koji se otvara i studentima i profesorima nakon prijave je fragment dobrodošlice. Ovo je vrlo jednostavan fragment koji se sastoji od dva gumba te naslova koji prikazuje ime i prezime trenutno prijavljenog korisnika.

Slika 5.1. Početni fragment (engl. *Welcome Fragment*)

Gumb „Moji kolegiji“ vodi korisnika na novi fragment na kojemu su prikazani svi kolegiji na koje je korisnik upisan. Drugi gumb je gumb sa slikom (engl. *Image Button*) koji se koristi za odjavu. Kada se pritisne gumb za odjavu šalje se zahtjev za odjavu na Firebase te se nakon uspješne odjave otvara aktivnost za prijavu (engl. *Login Activity*). Firebase autentifikacija omogućuje dohvaćanje ID-a prijavljenog korisnika u svakom trenutku korištenja aplikacije. Kako bi se pristupilo ID-u studenta potrebno je imati instancu klase *FirebaseAuthentication* te pozvati svojstvo te klase koje sadrži instancu prijavljenog korisnika (engl. *current user*). Klasa korisnika sadrži svojstvo uid (engl. *user ID*) koje sadrži ID prijavljenog korisnika. U ovoj aplikaciji prvo se provjerava postoji li takav ID u kolekciji profesora, ako ne postoji onda se provjerava postoji li takav id u kolekciji studenata. Korištenjem Firebase autentifikacije na način opisan u prethodnim rečenicama moguće je doći do

imena prijavljenog korisnika i prikazati ga na fragmentu a to se može vidjeti u programskom kodu 5.2.

```
private suspend fun getName(): String{
    var name = ""
    auth.currentUser?.let {  it: FirebaseUser
        val professorDocument = db.collection( collectionPath: "Professors") CollectionReference
            .document(it.uid) DocumentReference
            .get() Task<DocumentSnapshot!>
            .await()
        val professorName = professorDocument.getString( field: "Name")
        if(professorName != null && professorName != ""){
            name = professorName.toString()
        }
        else{
            val studentDocument = db.collection( collectionPath: "Students") CollectionReference
                .document(it.uid) DocumentReference
                .get() Task<DocumentSnapshot!>
                .await()
            val studentName = studentDocument.get("Name")
            if(studentName != null && studentName != ""){
                name = studentName.toString()
            }
        }
    }
```

Programski kod 5.1. Dohvaćanje imena korisnika

5.2. Registracijska aktivnost (engl. *Register Activity*) i aktivnost za prijavu (engl. *Login Activity*)

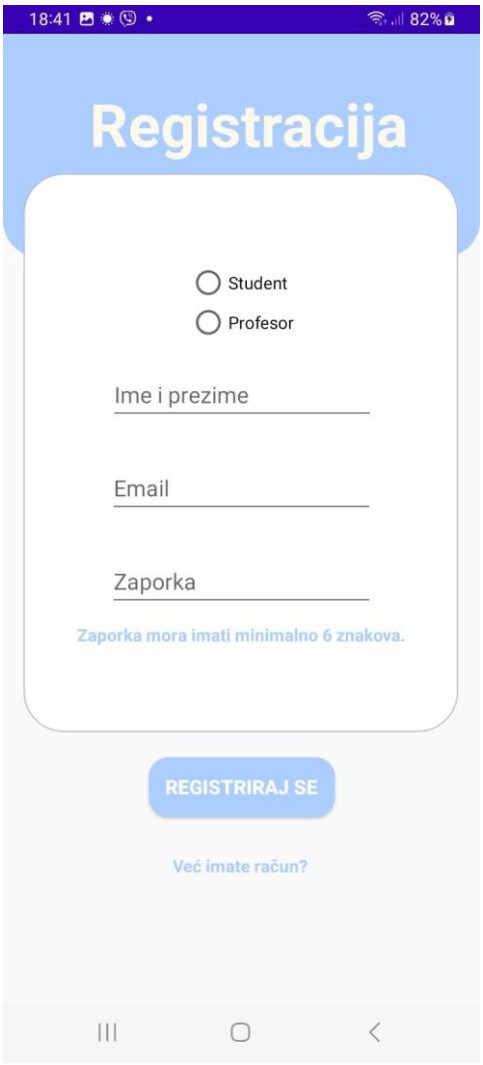
Prilikom otvaranja aplikacije uvijek se prvo prikazuje ekran za prijavu (aktivnost za prijavljivanje). Aktivnost za prijavu sadrži dva polja u koja korisnik unosi svoje podatke, gumb za prijavu te gumb koji vodi na aktivnost za registraciju. Ukoliko korisnik nema već kreiran račun klikom na tekst „Stvori novi račun“ pokreće se nova aktivnost za registraciju koristeći *Intent*. Pritiskom na gumb „Prijava“ vrši se autentifikacija i autorizacija te se pokreće glavna aktivnost koristeći *Intent*. Autentifikacija u ovoj aplikaciji se vrši koristeći Firebase autentifikaciju a također je implementirana i autorizacija kako bi se korisniku prikazalo točno određeno korisničko sučelje. Provjerava se je li korisnik profesor ili je student te se sukladno toj informaciji prikazuje odgovarajuće korisničko sučelje. Studentsko korisničko sučelje pruža uvid studentima u sve kolegije na koje su upisani te u svoju evidenciju za

svaki pojedinačni kolegij, kao i mogućnost očitanja QR koda. Sučelje za profesore također pruža pristup kolegijima na kojima profesor sudjeluje no osim toga profesori imaju pristup povijesti svih aktivnosti koje su održane iz pojedinog kolegija, detaljima o tim aktivnostima te imaju mogućnost kreiranja nove aktivnosti. Na slici 5.2. može se vidjeti izgled aktivnosti za prijavu.

Slika 5.2. Aktivnost za prijavu (engl. *Login Activity*)

Aktivnost za registraciju sastoji se od tri polja za korisnikov unos podataka, mogućnost izbora uloge (profesor ili student) te dva gumba. Za uspješnu registraciju potrebno je unijeti ime i prezime, e-mail, lozinku te ulogu. Nakon ispunjavanja svih polja za unos potrebno je pritisnuti gumb za registraciju. Nakon pritiska gumba za registraciju šalje se zahtjev na Firebase za registraciju korisnika, Firebase usluga šalje korisniku e-mail za potvrdu kreiranja računa te se nakon uspješne registracije otvara

aktivnost za prijavu. Na registracijskoj aktivnosti još se nalazi i tekst „Već imate račun?“ koji kada se pritisne vodi na aktivnost za prijavu. Slika 5.3. prikazuje izgled aktivnosti za registraciju korisnika.



Slika 5.3. Aktivnost za registraciju (engl. *Register Activity*)

5.3. Glavna aktivnost (engl. *Main Activity*)

Glavna aktivnost sastoji se od samo dvije komponente: aplikacijska traka i kontejnera za fragmente (engl. *Fragment Container View*). Na aplikacijskoj traci nalazi se naziv aplikacije koji je vidljiv u svim dijelovima aplikacije. Kontejner za fragmente služi za izmjenu fragmenata ovisno o unosu korisnika. Postupak prijelaza s jednog fragmenta na drugi sastoji se od četiri obavezna koraka te jednog opcionalnog. Prvo je potrebno instancirati fragment koji želimo otvoriti. Zatim je potrebno instancirati transakciju fragmenta (engl. *Fragment Transaction*) a to se postiže na način da se pozove metoda koja započinje transakciju (engl. *beginTransaction*) na upravitelju fragmenata (engl.

Fragment Manager) koji se nalazi u roditeljskoj aktivnosti fragmenta. Sljedeći korak je opcionalan te ga radimo samo ako želimo dijeliti podatke između trenutnog fragmenta i fragmenta koji treba otvoriti. Za dijeljenje podatka između fragmenata koristi se klasa *Bundle* u koju se pohranjuju podatci koje želimo dijeliti. Svaka klasa framenta u sebi ima svojstvo *arguments* koje je potrebno postaviti na *Bundle* koji smo kreirali te na taj način možemo pristupiti dijeljenim podatcima u novo otvorenom fragmentu pozivajući metodu dohvaćanja (engl. *get method*) na *Bundle* objektu. Sljedeći korak uključuje pozivanja metode zamjene fragmenata (engl. *replace*) na objektu transakcije fragmenta koji je kreiran u drugom koraku. Metoda zamjene prima dva argumenta, prvi argument predstavlja ID fragment kontejnera a drugi argument je objekt novog fragmenta koji je kreiran u prvom koraku. Također je na objektu transakcije pozvana metoda *addToBackStack* koja pamti sve fragmente koji su posjećeni te omogućuje vraćanje na fragmente koji su bili otvoreni. Zadnji korak je pozivanje metode izvršavanja (engl. *commit*) koja primjenjuje podešene promjene na korisničko sučelje. U programskom kodu 5.2. prikazana je metoda za prijelaz s jednog fragmenta na drugi.

```
private fun goToHistoryFragment(){
    val lectureHistoryFragment = LectureHistoryFragment()
    val fragmentTransaction : FragmentTransaction = requireActivity().supportFragmentManager.beginTransaction()

    bundle.putString("subjectId", arguments?.getCharSequence( key: "Id").toString())
    bundle.putString("activityType", activityToCreate)
    bundle.putString("subjectName", subjectName)
    bundle.putString("activityName", activityName)
    lectureHistoryFragment.arguments = bundle

    fragmentTransaction.replace(R.id.fragmentContainerView, lectureHistoryFragment).addToBackStack( name: null).commit()
}
```

Programski kod 5.2. *goToHistoryFragment* metoda

5.4. Fragment s kolegijima (engl. *My Subjects Fragment*)

Fragment s kolegijima je identičan za profesore i za studente. Glavna komponenta ovog fragmenta je *RecyclerView* koja omogućuje prikaz svih dostupnih kolegija za prijavljenog korisnika. Svaki element sastoji se od samo jednog polja koji sadrži naziv kolegija.

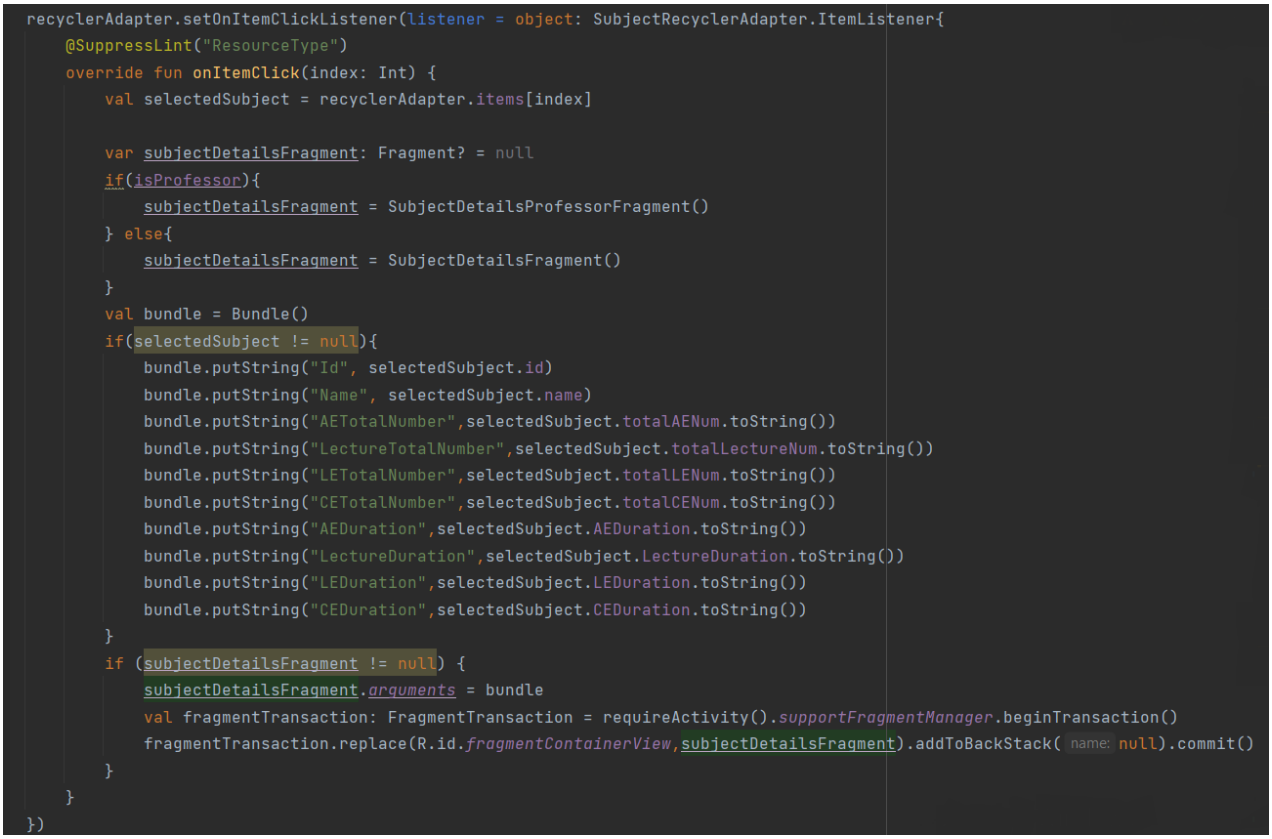
Slika 5.4. Fragment s kolegijima (engl. *MySubjectsFragment*)

U programskom kodu 5.3. prikazan je isječak koda koji prikazuje metodu *getSubjects* pomoću koje se dohvaćaju svi kolegiji na koje je korisnik prijavljen. Ovo je realizirano na način da se dohvate svi kolegiji iz baze te se izfiltriraju kako bi ostali samo kolegiji na koje je korisnik prijavljen. Svaki kolegij koji je dohvaćen se sprema u objekt klase *Subject*.

```
private fun getSubjects(subjectIds : ArrayList<String>, callback : (List<Subject>) -> Unit){
    db.collection( collectionPath: "Subjects") CollectionReference
        .get() Task<QuerySnapshot!>
        .addOnSuccessListener {  it: QuerySnapshot!
        val subjectList = mutableListOf<Subject>()
            for(document in it.documents){
                if(document.id in subjectIds){
                    val subject = Subject(
                        document.id,
                        document.data?.get("Name") as String,
                        document.data?.get("TotalAENum") as Long,
                        document.data?.get("TotalLectureNum") as Long,
                        document.data?.get("TotalLENum") as Long,
                        document.data?.get("TotalCENum") as Long,
                        document.data?.get("AEDuration") as Long,
                        document.data?.get("LectureDuration") as Long,
                        document.data?.get("LEDuration") as Long,
                        document.data?.get("CEDuration") as Long
                    )
                    subjectList.add(subject)
                }
            }
            callback(subjectList)
        }
}
```

Programski kod 5.3. Metoda za dohvaćanje kolegija (engl. *getSubjects*)

Na svaki element *RecyclerView-a* postavljen je tzv. *ItemListener*. *Listener* općenito predstavlja mehanizam koji omogućuje čekanje na nekakav događaj i implementiranje željenog ponašanja nakon što se dogodi neki događaj. *ItemListener* implementiran u kodu čeka na pritisak korisnika na jedan od elemenata *RecyclerView-a* te će se ovisno o tome koji je element kliknut otvoriti fragment s odgovarajućim podatcima. U programskom kodu 5.4. prikazan je kod koji na pritisak nekog kolegija prelazi s jednog fragmenta na drugi. Vrši se provjera je li prijavljeni korisnik profesor ili student te se sukladno tome otvara sljedeći fragment. Ova provjera je bitna jer je sljedeći fragment fragment s detaljima o kolegiju, a taj fragment se razlikuje profesorima i studentima. Klasa kolegij (engl. *Subject*) koja je populirana vrijednostima dohvaćenim iz baze podataka korsti se za prijenos podataka s jednog fragmenta na drugi. Prijelaz s jednog fragmenta na drugi je objašnjen u potpoglavlju 5.3.



```
recyclerAdapter.setOnItemClickListener(listener = object: SubjectRecyclerAdapter.ItemListener{
    @SuppressLint("ResourceType")
    override fun onItemClick(index: Int) {
        val selectedSubject = recyclerAdapter.items[index]

        var subjectDetailsFragment: Fragment? = null
        if(isProfessor){
            subjectDetailsFragment = SubjectDetailsProfessorFragment()
        } else{
            subjectDetailsFragment = SubjectDetailsFragment()
        }
        val bundle = Bundle()
        if(selectedSubject != null){
            bundle.putString("Id", selectedSubject.id)
            bundle.putString("Name", selectedSubject.name)
            bundle.putString("AETotalNumber",selectedSubject.totalAENum.toString())
            bundle.putString("LectureTotalNumber",selectedSubject.totalLectureNum.toString())
            bundle.putString("LETotalNumber",selectedSubject.totalLENum.toString())
            bundle.putString("CETotalNumber",selectedSubject.totalCENum.toString())
            bundle.putString("AEDuration",selectedSubject.AEDuration.toString())
            bundle.putString("LectureDuration",selectedSubject.LectureDuration.toString())
            bundle.putString("LEDuration",selectedSubject.LEDuration.toString())
            bundle.putString("CEDuration",selectedSubject.CEDuration.toString())
        }
        if (subjectDetailsFragment != null) {
            subjectDetailsFragment.arguments = bundle
            val fragmentTransaction: FragmentTransaction = requireActivity().supportFragmentManager.beginTransaction()
            fragmentTransaction.replace(R.id.fragmentContainerView,subjectDetailsFragment).addToBackStack( name: null).commit()
        }
    }
})
```

Programski kod 5.4. Isječak koda koji omogućuje „slušanje“ elemenata *RecyclerView-a*

5.5. Fragment s detaljima o kolegiju za profesore (engl. *Subject Details Professor Fragment*)

Profesori i studenti nemaju istu ulogu u stvarnom svijetu pa tako ni u ovoj aplikaciji, iz tog razloga od ovog fragmenta pa na dalje profesori i studenti imaju različito korisničko sučelje. Ovaj fragment sastoji se od četiri trake koje prate napredak (engl. *progress bar*) održavanja aktivnosti uz odgovarajući numerički prikaz održanih i ukupnog broja aktivnosti. Također na ovom fragmentu se nalazi i radio grupa koja pruža četiri moguća izbora: Predavanje, AV, LV i KV. Nadalje, na dnu se nalaze dva gumba. Gumb „Povijest“ vodi na novi fragment koji prikazuje povijest izabrane vrste aktivnosti. Gumb „+“ vodi na novi fragment na kojemu se generira QR kod nove odabrane vrste aktivnosti te se u bazi kreira nova aktivnost. U bazi se pohranjuju podatci o vremenu kreiranja aktivnosti, ID kreatora aktivnosti (profesor), redni broj aktivnosti te popis prisutnih studenata koji je u trenutku kreiranja aktivnosti samo prazna lista. Ukoliko korisnik pritisne gumb „Povijest“ ili „+“ prije nego što je odabrao vrstu predavanja pojavit će se *Toast* poruka (poruka koju pruža povratnu

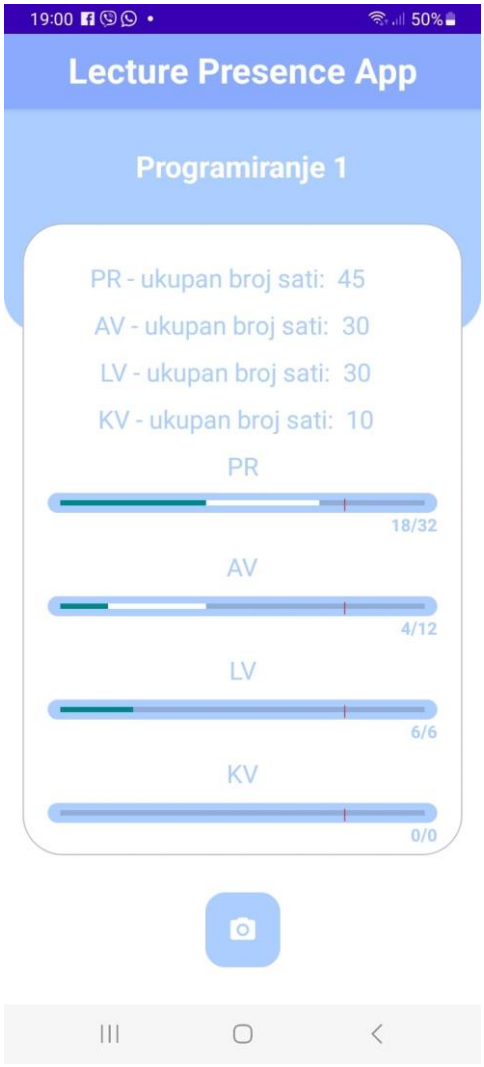
informaciju korisniku) koja obaviještava korisnika da treba odabrati vrstu aktivnosti prije pritiska na neki od gumba.

Slika 5.5. Fragment s detajima o kolegiju za profesore (engl. *Subject Details Professor Fragment*)

5.6. Fragment s detaljima o kolegiju za studente (engl. *Subject Details Student Fragment*)

Ovaj fragment najbitniji je fragment za studente te im pruža informaciju o njihovoj prisutnosti na određenom kolegiju za svaku vrstu aktivnosti. Studentima je njihova prisutnost prikazana pomoću traka za napredak (engl. *progress bar*) i numerički. Iako se na slici 5.6. čini kao da ih ima četiri, na ovom fragmentu nalazi se osam traka za napredak od kojih četiri prate prisutnost studenta na pojedinoj aktivnosti a druge četiri prate broj održanih aktivnosti. Trake koje prate studentovu prisutnost su

pozicionirane iznad trakaza koje prate održana predavanja koristeći svojstvo elevacije (engl. *elevation*) u XML-u. Također, boja traka koje su iznad postavljena je na transparentnu kako bi se mogle vidjeti trake ispod. Na taj način je postignuto željeno korisničko kroz koje korisnik vidi jednu traku za napredak koja prati dvije vrijednosti što nije moguće napraviti sa zadanim trakama za napredak Android Studio-a. Još jedan detalj na trakama za napredak je mala vertikalana crvena crtica koja je pozicionirana na sedamdeset posto cijele trake. Na taj način studenti lako mogu vidjeti kada su ostvarili sedamdeset posto prisutnosti iz određene aktivnosti. Osim traka za napredak studenti mogu vidjeti ukupan broj sati određenih za svaku aktivnost. I na dnu fragmenta nalazi se gumb sa slikom (engl. *Image Button*) koji kada je pritisnut vodi na aktivnost za očitanje QR koda.



Slika 5.6. Fragment s detajima o kolegiju za studente (engl. *Subject Details Student Fragment*)

U programskom kodu 5.5 prikazan je kod koji omogućuje pokretanje aktivnosti za očitanje te način na koji se rukuje rezultatom očitanja QR koda. Metoda *registerForActivitryResult* postavlja mehanizam koji čeka na rezultat aktivnosti za očitavanje te kada dobije rezultat izvršava se lambda funkcija koja rukuje rezultatom. Ukoliko je rezultat valjan pozivaju se dvije metode, jedna koja upisuje prisutnost studentu te druga koja studenta upisuje na popis prisutnih studenata na aktivnosti. Ukoliko rezultat nije valjan ispisat će se poruka „Neuspješno skeniranje“. Drugi dio koda predstavlja mehanizam koji čeka na pritisak na gumb za očitanje. Kada je gumb za očitanje pritisnut kreira se instanca *IntentIntegrator-a* koja omogučuje jednostavno podešavanje postavki QR kod čitača. Na kraju se kreira *Intent* koji se koristi za pokretanje aktivnosti za skeniranje s postavkama očitanja podešenim na integratoru.

```
val scanLauncher = registerForActivityResult(ActivityResultContracts.StartActivityForResult()){ result ->
    val resultCode = result.resultCode
    val data = result.data

    if(resultCode == RESULT_OK && data != null){
        val scanResult = IntentIntegrator.parseActivityResult(resultCode, data)

        if(scanResult != null && scanResult.contents != null){
            val qrCodeString = scanResult.contents

            CoroutineScope(Dispatchers.Main).launch {  this: CoroutineScope
                addStudentsAttendance(qrCodeString)
                addStudentToActivity(qrCodeString)
                refreshData()
            }
        }else{
            Toast.makeText(this.context,  text: "Neuspješno skeniranje.",Toast.LENGTH_SHORT).show()
        }
    }
}
goToScanScreenButton.setOnClickListener {  it: View!
    val integrator = IntentIntegrator(this.activity)
    integrator.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE)
    integrator.setPrompt("Skeniraj QR kod")
    integrator.setCameraId(0) // Use the rear camera
    integrator.captureActivity = CaptureActivityPortrait::class.java
    integrator.setOrientationLocked(true)

    val scanIntent = integrator.createScanIntent()
    scanLauncher.launch(scanIntent)
}
```

Programski kod 5.5. Isječak koda koji pokreće čitač QR koda te rukuje rezultatom istog

5.7. Fragment za generiranje QR koda (engl. *QR Code Fragment*)

Fragment za generiranje QR koda može se pronaći u korisničkom sučelju za profesore. Profesoru se nakon pritiska na gumb za dodavanje nove aktivnosti (predavanje, AV, itd.) otvara ovaj fragment. Glavna komponenta ovog fragmenta je slikovna komponenta (engl. *Image View*) pomoću koje se prikazuje generirani QR kod. Na vrhu fragmenta može se vidjeti naziv kolegija iz kojega je kreirana aktivnost te vrsta aktivnosti koja je kreirana. Na ovom fragmentu još se nalaze i dva gumba, gumb „Detalji“ koji vodi na novi fragment na kojemu su prikazani detalji aktivnosti (vrijeme održavanja, ime profesora koji je održao aktivnost, popis studenata, itd.) te gumb koji omogućuje slanje e-maila s generiranim QR kodom profesoru. Na taj način profesor može lako otvoriti QR kod na računalu te ga projicirati na platnu kako bi ga studenti mogli očitati te se upisati na listu prisutnih studenata.

Slika 5.7. QR kod fragment (engl. *QR Code Fragment*)

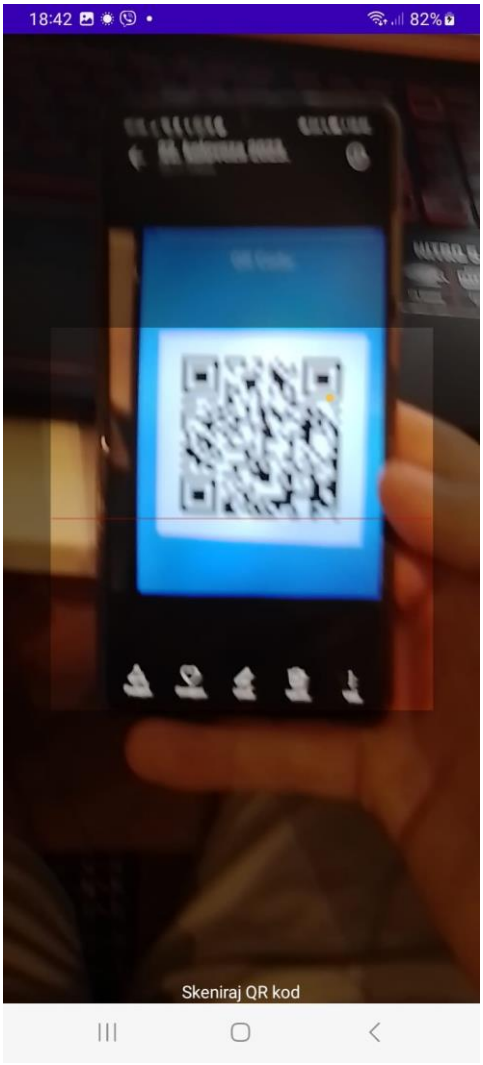
Za generiranje QR koda koristi se Zxing-android-embedded biblioteka koja pruža potrebne klase za generiranje QR koda. *MultiFormatWriter* klasa koristi se za generiranje raznih vrsta barkoda. U matricu se pohranjuje kodirani QR kod na temelju teksta koji je predan metodi za kodiranje koja pripada *MultiFormatWriter* objektu. Na kraju se koristi instanca klase *BarcodeEncoder* koja konvertira bit matricu (engl. *Bit Matrix*) u *bitmap* sliku koja se može koristiti za prikaz koristeći *ImageView* u XML-u.

```
private fun generateQRCode(qrCodeString: String): Bitmap? {
    val writer = MultiFormatWriter()
    try {
        val matrix: BitMatrix = writer.encode(qrCodeString, BarcodeFormat.QR_CODE,  width: 600,  height: 600)
        val encoder = BarcodeEncoder()

        return encoder.createBitmap(matrix)

    } catch (e: WriterException) {
        Log.d(TAG,  msg: "$e, Pojavila se greška prilikom generiranja QR koda")
        return null
    }
```

Programski kod 5.6. Metoda za generiranje QR koda

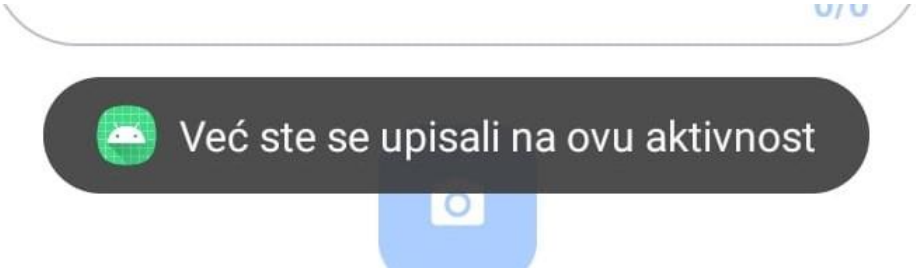
5.8. Aktivnost za očitavanje QR koda

Nakon pritiska na gumb za skeniranje na fragmentu s detaljima o kolegiju, studentu se otvara kamera s prilagođenim sučeljem za očitanje QR koda.



Slika 5.8. Aktivnost za očitavanje QR koda

Kada korisnik usmjeri kameru prema QR kodu te se kod uspješno očita prikaže se poruka da je QR kod uspješno očitan te da je evidencija uspješno odrađena i korisnik se vraća na fragment s detaljima o kolegiju automatski. Ukoliko se dogodila greška prilikom očitanja prikazat će se prikladna poruka. Onemogućeno je očitanje QR koda od drugog kolegija kako ne bi došlo do grešaka u bazi podataka, ukoliko student pokuša očitati krivi QR kod dobit će sljedeći poruku: „Skeniranje neuspješno. Provjerite jeste li na ispravnom kolegiju.“. Također je onemućeno očitanje QR koda iste aktivnosti više puta. Ukoliko student očita QR kod jedne aktivnosti drugi put prikazat će se prikladna poruka što se može vidjeti na slici 5.9.



Slika 5.9. Obavijest da je student već upisan na aktivnost

Kada aktivnost za očitavanje QR koda vrati rezultat pozvat će se tri metode. Prva metoda, *addStudentsAttendance* u bazu upisuje prisutnost studentu koju student može vidjeti.

```
private suspend fun addStudentsAttendance(qrCodeString: String){
    val activityType = qrCodeString.substringAfter( delimiter: "/").dropLast( n: 1)
    val activityDuration = qrCodeString.last().toString().toLong()
    val activityTypeShort = getActivityTypeShort(activityType)

    try {
        if(checkIfAlreadyAttended(qrCodeString)){
            Toast.makeText(this.context,  text: "Već ste se upisali na ovu aktivnost", Toast.LENGTH_LONG).show()
        }else{
            auth.currentUser?.let { student ->
                db.collection( collectionPath: "Students") CollectionReference
                    .document(student.uid) DocumentReference
                    .update( field: "Attendance.$subjectId.$activityTypeShort", FieldValue.increment(activityDuration)) Task<Void!>
                    .await()
            }
        }

    }catch(e: Exception){
        Toast.makeText(this.context, text: "Skeniranje neuspješno",Toast.LENGTH_LONG).show()
    }
}
```

Programski kod 5.7. Metoda *addStudentsAttendance*

Druga metoda, *addStudentToActivity* u bazu upisuje studenta na listu prisutnih studenata koju profesor može vidjeti.

```
private suspend fun addStudentToActivity(qrCodeString: String){
    val activityId = qrCodeString.substringBefore( delimiter: "/")
    val activityType = qrCodeString.substringAfter( delimiter: "/").dropLast( n: 1)

    val studentId = auth.currentUser?.uid
    try {
        db.collection( collectionPath: "Subjects") CollectionReference
            .document(subjectId) DocumentReference
            .collection(activityType) CollectionReference
            .document(activityId) DocumentReference
            .update( field: "StudentIds",FieldValue.arrayUnion(studentId)) Task<Void!>
            .await()
    }catch (e: Exception){
        Toast.makeText(this.context, text: "Skeniranje neuspješno. Provjerite jeste li na ispravnom kolegiju.",Toast.LENGTH_LONG).show()
    }
}
```

Programski kod 5.8. Metoda *addStudentToActivity*

I treća metoda, *refreshData* se poziva kako bi aplikacija bila responzivna – ona omogućava da se podatci ažuriraju odmah nakon očitanja QR koda, a ne tek kada se opet otvori fragment s detaljima kolegija.

5.9. Fragment s povijesti aktivnosti (engl. *Lecture History Fragment*)

Ovaj fragment je dio korisničkog sučelja za profesore. Omogućuje profesorima uvid u sve održane aktivnosti iz nekog kolegija. Glavna komponenta ovog fragmenta je *RecyclerView* koji sadrži listu svih aktivnosti odabrane vrste za neki kolegij. Profesor na fragmentu s detaljima kolegija ima izbor između četiri vrste aktivnosti. Nakon što izabere jednu od četiri mogućnosti i pritisne gumb „Povijest“ otvara se Fragment s povijesti aktivnosti odabrane vrste. Za svaku pojedinu aktivnost prikazane su sljedeće informacije: Vrsta aktivnosti, redni broj aktivnosti i vrijeme održavanja aktivnosti. Na svaku aktivnost postavljen je mehanizam koji čeka na pritisak (engl. *Listener*). Nakon što korisnik pritisne jednu od aktivnosti otvara se novi fragment na kojemu je prikazano više detalja o odabranoj aktivnosti.

Slika 5.10. Fragment s povijesti aktivnosti (engl. *Lecture History Fragment*)

5.10. Fragment s detaljima aktivnosti (engl. *Lecture Details Fragment*)

Ovo je posljednji fragment korisničkog sučelja za profesore. Prošireni prikaz detalja aktivnosti koja je odabrana prikazan je na ovom fragmentu. Prikazane su sljedeće informacije o aktivnosti: vrsta aktivnosti, redni broj aktivnosti, ime predavača, vrijeme održavanja, broj prisutnih studenata te popis prisutnih studenata. Popis prisutnih studenata realiziran je koristeći još jedan *RecyclerView* kroz koji je moguće listati (engl. *scroll*) kako bi se dobio prikaz svih studenata.

Slika 5.11. Fragment s detaljima aktivnosti (engl. *Lecture Details Fragment*)

6. ZAKLJUČAK

Cilj ovog završnog rada je kreirati Android aplikaciju koja bi mogla olakšati proces evidencije prisutnosti studenata profesorima te praćenje svoje prisutnosti studentima. Cilj je bio realizirati ovo ponašanje koristeći tehnologiju generiranja QR koda te očitanja istog. Baza podataka je implementirana koristeći Firebase Firestore uslugu. Struktura baze je osmišljena na način da pohrani i omogući pristup najbitnijim informacijama kako bi se postigao cilj završnog rada. U bazi su pohranjene informacije o studentima, profesorima, kolegijima te o aktivnostima i sve te kolekcije su strukturirane na način koji omogućuje povezivanja podataka kada je to potrebno. Aplikacija se sastoji od tri glavne aktivnosti od kojih je jedna zaslužna za registraciju, druga za prijavu a treća sadrži spremnik za sve ostale fragmente. Aplikacija pruža profesorima jednostavno sučelje za kreiranje aktivnosti, dijeljenje QR koda te praćenje povijesti aktivnosti. Studenti mogu pratiti svoju prisutnost na svakom kolegiju, napredak održavanja aktivnosti te granicu kada su zadovoljili minimalan prag prisutnosti. Ova aplikacija je namijenjena za evidenciju prisutnosti studenata na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek te je prilagođena načinu održavanja nastave istog. Kada bi htjeli aplikaciju koristiti za drugi fakultet ili u drugu svrhu bilo bi potrebno raditi izmjene. Na primjer, na FERIT-u kolegiji često imaju tri do četiri vrste aktivnosti (predavanja, auditorne vježbe, laboratorijske vježbe, konstrukcijske vježbe) što možda nije slučaj za neki drugi fakultet. Ova aplikacija trenutno uspješno radi ono za što je i namijenjena na bazi podataka kreiranoj za testiranje funkcionalnosti no nije spojena na sustav koji se koristi u stvarnom svijetu na FERIT-u za evidenciju studenata. Kako bi se upotpunila svrha ove aplikacija bilo bi potrebno spojiti se na bazu podataka FERIT-a te dohvaćati stvarne podatke o studentima, profesorima i kolegijima.

LITERATURA

[1] Android Developers, Kotlin, dostupno na: https://developer.android.com/kotlin/ , srpanj – kolovoz 2023.

[2] Android Developers, Android, dostupno na: https://developer.android.com/docs , srpanj – kolovoz 2023.

[3] Firebase Firestore Documentation, dostupno na: https://firebase.google.com/docs/firestore , srpanj – kolovoz 2023.

[4] Zxing-android-embedded dokumentacija, dostupno na: https://github.com/journeyapps/zxing-android-embedded , kolovoz 2023.

[5] Android arhitektura, dostupno na: https://www.interviewbit.com/blog/android-architecture/ , kolovoz 2023.

[6] Marko Gargenta, „Learning Android“, O'Reilly, dostupno na: https://www.protechtraining.com/static/archive/Learning_Android.pdf , kolovoz 2023.

SAŽETAK

Ovaj završni rad pruža rješenje za efikasnije, urednije i praktičnije evidentiranje prisutnosti studenata od već postojećeg sustava evidencije na FERIT-u. Trentuačno se na većini kolegija prisutnost evidentira slanjem papira za potpis te unošenja u Excel tablicu. Studenti često nemaju pristup svojoj prisutnosti te ukoliko žele pratiti svoju prisutnost moraju samostalno voditi evidenciju. Ova aplikacija pruža rješenje za oba problema. Profesori mogu za kolegij koji vode kreirati aktivnost po želji te podijeliti QR kod kako bi se studenti mogli evidentirati. Studenti imaju mogućnost očitanja dobivenog QR koda i samim time evidentiranja na aktivnosti. Također, studenti mogu vrlo lako pristupiti i pregledati svoju prisutnost za svaki kolegij na koji su upisani. U prvom poglavlju opisan je zadatak završnog rada. U drugom poglavlju spomenuta su i opisana postojeća slična rješenja. U trećem poglavlju spomenute su i opisane tehnologije koje su korištene u ovom završnom radu. U četvrtom poglavlju opisan je razvoj aplikacije te struktura baze podataka. U petom poglavlju detaljno je razrađena i opisana struktura aplikacije. Naposljetku, u zaključku su opisani cilj i ograničenja aplikacije te način moguće primjene.

Ključne riječi: Android, aplikacija, evidencija prisutnosti, Firebase, Kotlin

ABSTRACT

Title: Creation of Android application for record of class attendance using QR code

This final thesis provides a solution for a more efficient, orderly and practical way of recording of students attendance than the already existing system of records at FERIT. Currently, in most subjects attendance is recorded by sending a paper for signature and entering it into an Excel table. Students offten don't have access to their attendance so if they want to monitor their attendance, they must keep records independently. This application provides a solution for both of these problems. Professors can create an activity of their choice for the subject they teach and share a QR code so the students could be registered. Students have the ability to scan the received QR code and thereby register for activity. Also, students can access and review their attendance for each subject they are enrolled in very easily. The first chapter describes the task and existing similar solutions. In the second chapter, the technologies used in this final work are mentioned and described. The third chapter describes the development of the application and the structure of the database. In the fourth chapter, the structure of the application is elaborated and desribed in detail. Finally, the goal and the limitations of the application and the method of possible application are described in the conclusion.

Key words: Android, application, attendance record, Firebase, Kotlin

ŽIVOTOPIS

Marko Kovačević rođen je 3.2.2002. godine u Đakovu. Osnovnu školu od prvog do četvrtog razreda pohađao je u Osnovnoj školi Josipa Kozarca u Vrbici. Od petog do osmog razreda pohađao je Osnovnu školu Stjepana Cvrkovića u Starim Mikanovcima. U srednju školu kreće 2016. godine kada je upisao Graditeljsko-geodetsku školu Osijek, smjer geodetski tehničar. 2020. godine upisuje se na preddiplomski studij Programsko inženjerstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. 2023. godine obavio je stručnu praksu u tvrtci Mono Software na kojoj je proširio svoja znanja o web tehnologijama, razvoju aplikacija korištenjem ASP.NET razvojnog okvira, radu s relacijskim bazama podataka te React biblioteke.

_____

Potpis autora