ANDROID APLIKACIJA ZA EVIDENCIJU DOLAZAKA NA NASTAVU

Gelešić, Tomislav

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

ANDROID APLIKACIJA ZA EVIDENCIJU DOLAZAKA NA NASTAVU

Završni rad

Tomislav Gelešić

Osijek, 2019.

SADRŽAJ

1. Uvod

Danas svi govore o informatici, digitalizaciji, programiranju i svijetloj budućnosti tehničke grane znanosti. Programirati nije najteža stvar za raditi, ali da je programirati lako, svatko bi to činio. [1] U ovom radu objašnjene su osnove programiranja u Android Studiju korištenjem objektno orijentiranog programskog jezika Java. [2] Upravo ovaj par je odabran zbog velike raširenosti Android uređaja i paradigme koju promovira - *„open-source coding"*. To je način programiranja koji se zasniva na dostupnosti samog programskog koda koji se može dijeliti, unaprjeđivati, čitati i komentirati na forumima ili nekim drugim zbirnim mjestima zainteresiranih pratitelja a ne samo unutar poslovnih krugova ili unutar tvrtki koje čuvaju kôd kao poslovnu tajnu. Samo okruženje Android Studio, koje se mora instalirati kako bi krenuli programirati uopće, sasvim je besplatno. Iako danas Android Studio podržava programiranje u raznim programskim jezicima poput Java, C, C++, C#, Python ali i najnoviji Kotlin. Od početka se krenulo s Java programskim jezikom kao osnovnim jer je bio opće prihvaćen u programerskoj zajednici. Standardizacijom internetskih protokola i stvaranje JVM-a (*„Java Virtual Machine"*) dijela sklopovlja omogućilo se uniformno prikazivanje sadržaja na internetu bez obzira u kojem jeziku je izvorno napisan, tj. nastao je *„Java bytecode"* kao programski jezik u koji se sve pretvaralo kako bi se prikazalo na internetu. Dakle, prije Android platforme i Android Studija, već je postojala dobro razrađena razvojna baza sa popratnom dokumentacijom. Sama činjenica zašto je tvrtka Sun Microsystems (Java) ušla u dogovor sa tvrtkom Google (Android) i definirala blisku povijest na određeni način nije tematski bitna za ovaj rad. U radu je dan sažet pregled Android Studija i jezika Jave sa linkovima na službene stranice gdje se nalaze dodatne informacije. Na kraju se nalazi implementacija rješenja zadatka kao primjera potpune izrade android aplikacije. Uključujući izradu dizajna i pisanje logike problemskog zadatka, instalaciju aplikacije na stvaran uređaj i slike iz prakse korištenja.

1.1. Zadatak završnog rada

Kratko opisati specifičnosti izrade aplikacije za Android platformu. Izraditi Android aplikaciju pomoću koje korisnik može voditi evidenciju pohađanja nastave za vrijeme studija. Omogućiti korisniku kreiranje profila studija zadavanjem svih oblika i satnice kolegija koje pohađa u određenoj akademskoj godini. Opisati postupak izrade aplikacije kao i njene

funkcionalnosti. Aplikaciju testirati zadavanjem testnih podataka nekog stvarnog ili izmišljenog studija.

2. Android Studio

Integrirano razvojno okruženje (IDE - „*eng. Integrated Development Environment*") za Android operacijski sustav naziva se Android Studio. Svaka aplikacija je novi projekt koji otvaramo u okruženju. Omogućuje Gradle sustavno postavljanje što znači da je vrlo lako unijeti dodatne funkcionalnosti u aplikaciju preko vanjskih biblioteka ili uključivanja dodatnih mogućnosti okruženja ako se preuzmu dodatni paketi. Gradle datoteka se temelji se na objektno orijentiranoj paradigmi i „*key - value*" parovima uz neke ključne riječi koje su obavezne. Okruženje pruža dvije vrste dizajniranja projekta: „*drag and drop*" i unos programskih naredbi u obliku teksta. Drugo omogućuje preciznije i konkretnije programiranje jer korištenjem „*drag and drop*" metode okruženje automatski generira programski kôd u obliku teksta. Ponekad okruženje zna generirati više kôda no što je potrebno i kasnije zna prouzročiti probleme i greške koje se teško otkrivaju ili popravljaju. Tako da treba obratiti pažnju pri odabiru metode jer je stvar ukusa koju ćemo odabrati a kvaliteta dolazi iz razumijevanja onoga s čim radimo.

Android Studio nudi:

- mnoge alate za razna mjerenja efikasnosti aplikacije (vrijeme odziva, memorija, itd.)
- otklanjanje pogrešaka („*debugging*")
- uređivač izgleda zaslona (grafičko sučelje za „*drag and drop*")
- gotove predloške izgleda ili predloške sa rješenjima za neke česte funkcionalnosti
- integrirana sučelja za komunikaciju s nekim online bazama podataka (npr. Firebase)
- Google Cloud kompatibilnost
- predpregled izgleda zaslona
- Emulator, tj. virtualnu simulaciju za sve vrste uređaja sa Android operacijskim sustavom

Najnovija verzija Android Studija je: „Android Studio 3.4" (lipanj, 2019.) i okruženje podržava novi programski jezik Kotlin.

3. Java - programski jezik

Java je objektno orijentirani (OO) programski jezik (iako ne potpuno čist OO jezik jer sadrži primitivne tipove podataka, npr. *int, char, float*, dok su u nekom čistom OO jeziku svi tipovi podataka objekti). Razvili su ga James Gosling, Patrick Naughton i drugi inženjeri tvrtke Sun Microsystems 1995. godine. Velika prednost u odnosu na većinu dotadašnjih programskih jezika je uniformnost programa na cjelokupnom tržištu uređaja. Aplikacije pisane u Javi mogu se izvoditi bez prevođenja u strojni jezik uređaja a to se postiže korištenjem programa JRE („*Java Runtime Environment*") na JVM („*Java Virtual Machine"*) sklopovlju uređaja, koji omogućuje prikaz aplikacija bez pretvaranja kôda u strojni jezik konkretnog uređaja i platforme koju koristi. Taj koncept je poznat pod kraticom WORA („*Write Once Read Anywhere*") i drastično ubrzava rad uređaja i razmjenu podataka uređaja s internetom što omogućuje veliku elegantnost i ugodnost korištenja samog uređaja.

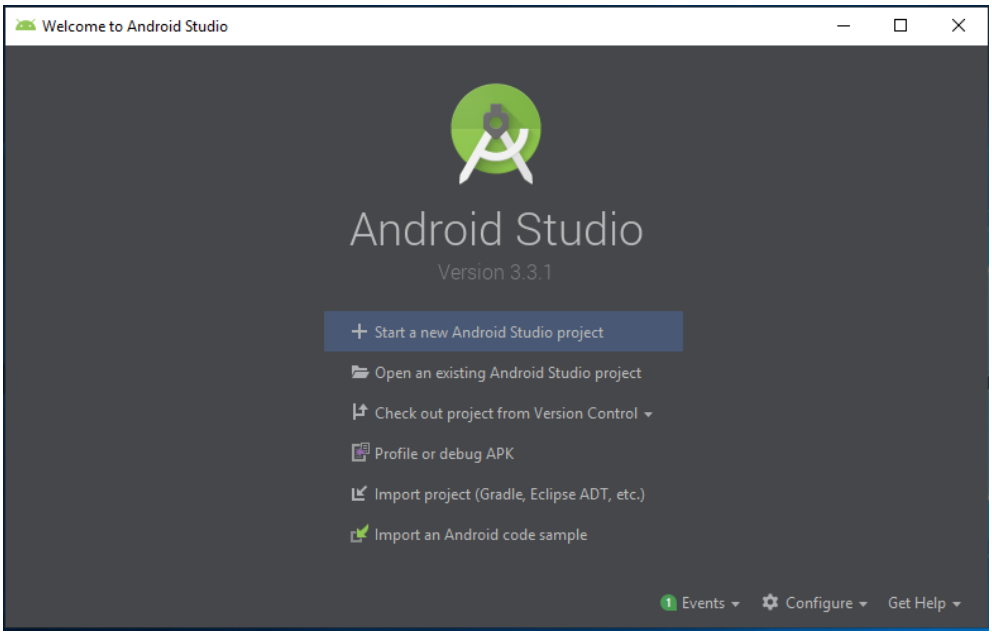
Najnovija verzija je: Java 12 (travanj 2019.).

4. Implementacija aplikacije

U nastavku slijedi korak po korak stvaranja aplikacije koji su popraćeni slikama i komentarima što se na njima nalazi. Zbog velikog broja mogućnosti kako se nešto moglo učiniti treba ove korake shvatiti okvirno te svatko ima slobodu implementirati i prilagoditi dijelove kako njemu najbolje odgovara jer ne postoji univerzalno pravilo u programiranju, to je poput stila pisanja, ovisi o pojedincu.

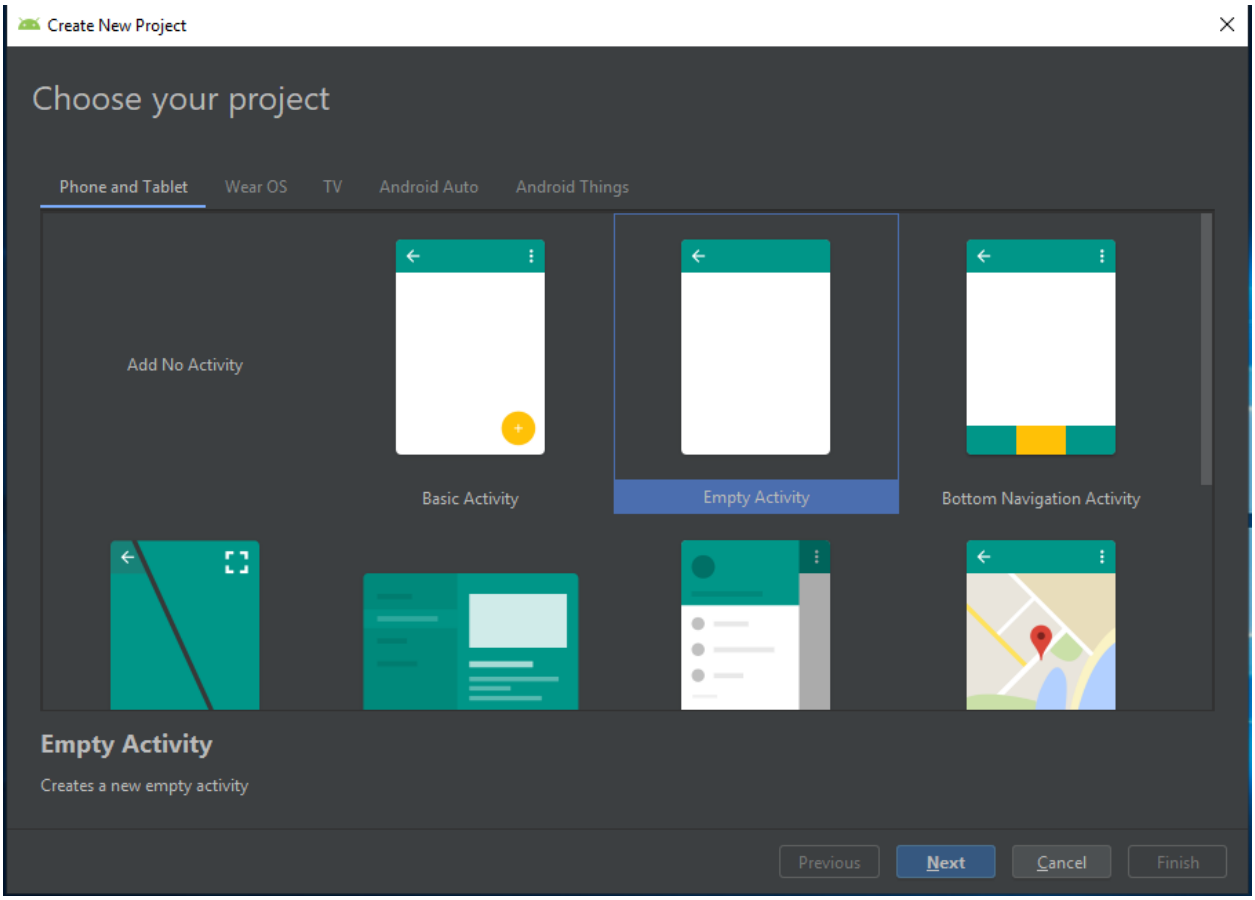
4.1. Uvod i stvaranje novog projekta u Android Studiju

Na slici 1.1. vidimo početno sučelje kada pokrenemo Android Studio. Zatim, kada odaberemo „*Start a new Android Studio project*" s centriranog popisa, otvara nam se slijed prozora za odabir i unos podataka o novom projektu (Slika 1.2., Slika 1.3. i Slika 1.4.). A na slici 1.5. možemo vidjeti naš projekt koji smo stvorili u okruženju – Android Studio.



Slika 1.1. Početno sučelje kada otvorimo Android Studio

Na slici 1.2. vidimo prozor za odabir vrste uređaja sa dodatnim mogućnostima početnog izgleda uređaja. Radi se o nekoliko predložaka sa implementiranim osnovnim funkcionalnostima.

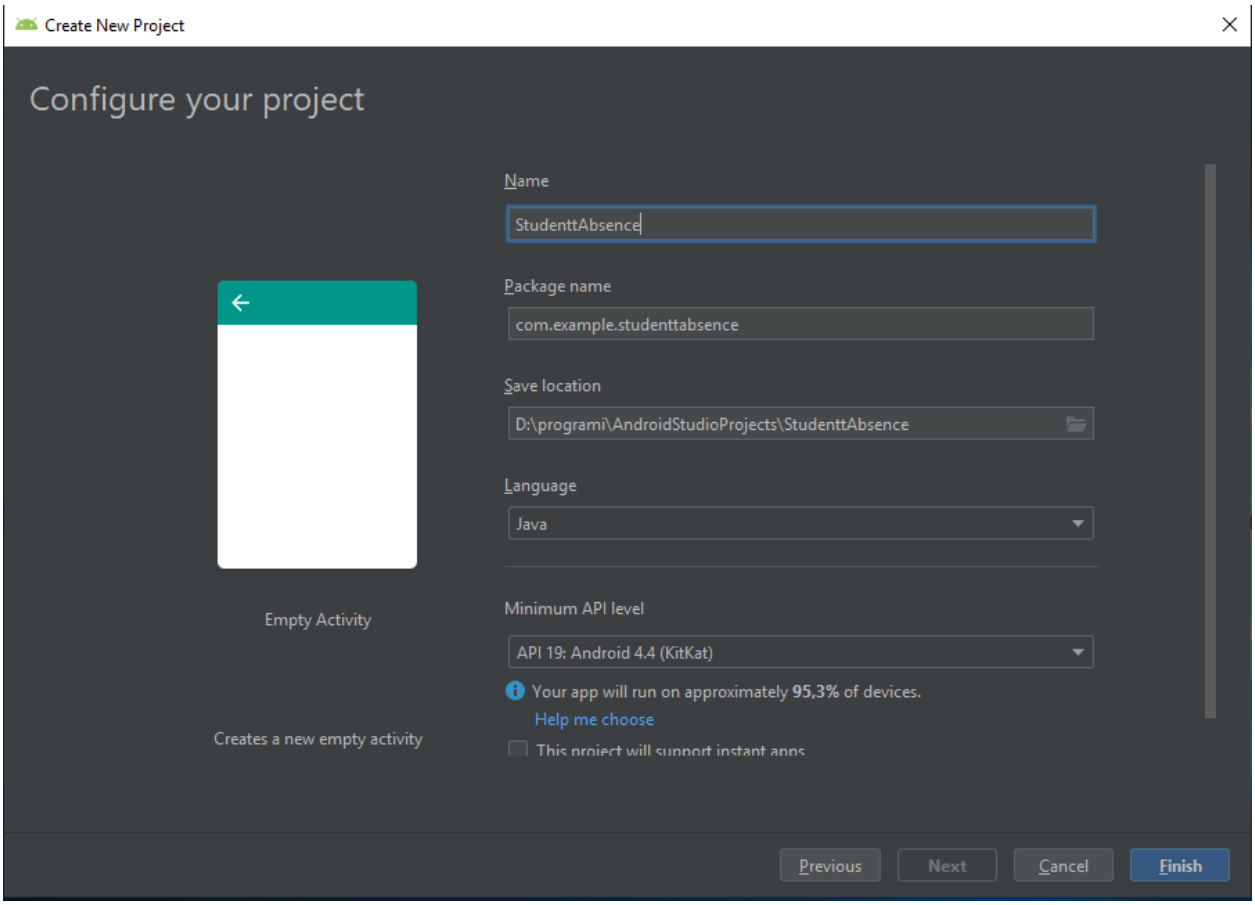


Slika 1.2. Odabir vrste uređaja

Označavamo „*Empty Activity*" i odabiremo „*Next*".

„*Activity"* predstavlja jedan zaslon u našoj aplikaciji. Izgled aplikacije (UI - „*User Interface"*) se sastoji od gradivnih objekata gdje je „*View*" osnovni gradivni objekt i iz njega su izvedeni svi ostali poput: „*TextView*" koji služi za prikaz teksta ili „*EditText*" koji služi za korisnički unos znakova, itd. [3]

Važno je napomenuti da ime aplikacije ne mora biti jedinstveno dok „*Company Domain*" („*CD*") mora biti. Kako su se razvijali uređaji i funkcionalnosti tako je rasla i verzija pa trebamo reći koju minimalnu verziju želimo podržati (Slika 1.3.). Neke funkcionalnosti također su izgubljene u najnovijim verzijama, stoga treba provjeriti pozadinu onoga što koristimo dok programiramo.



Slika 1.3. Unos detalja o aplikaciji

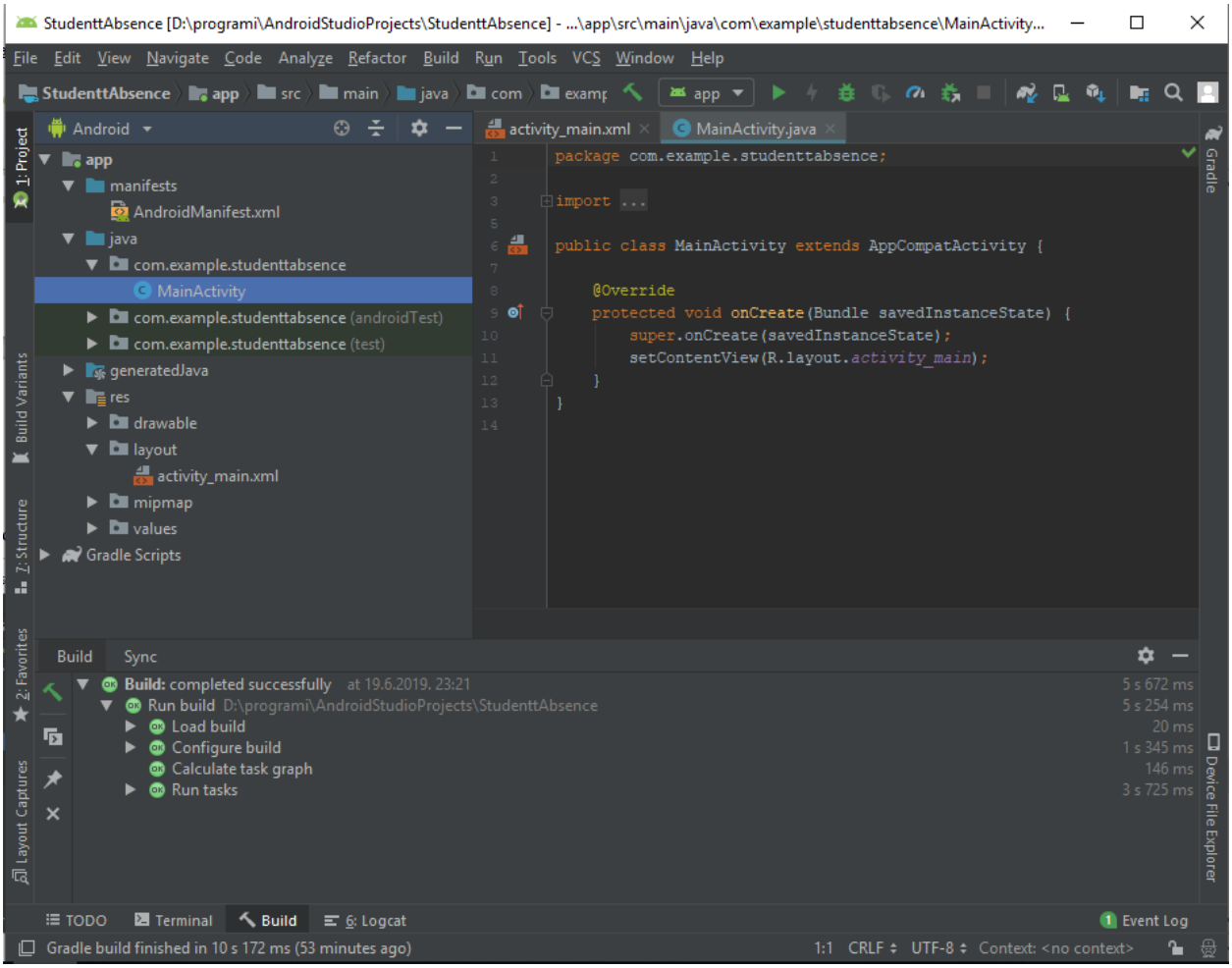
Unosimo naziv aplikacije, naziv paketa u kojem će se nalaziti „*source code*”, odabiremo mjesto gdje želimo spremiti projekt na lokalnom računalu te odabiremo programski jezik kojim ćemo implementirati logiku, te kliknemo „*Finish*”.

Na slici 1.4. vidimo početno sučelje nakon uspješnog stvaranja novog projekta u Android Studiju.

Dolje se nalazi prozor za obavijesti o rezultatima izvođenja aplikacije i prozor za pogreške koje nam okruženje ispiše kada se dogode prilikom pokretanja aplikacije. Desno vidimo radni prozor u koji unosimo programski kôd za logiku i *UI* dizajn.

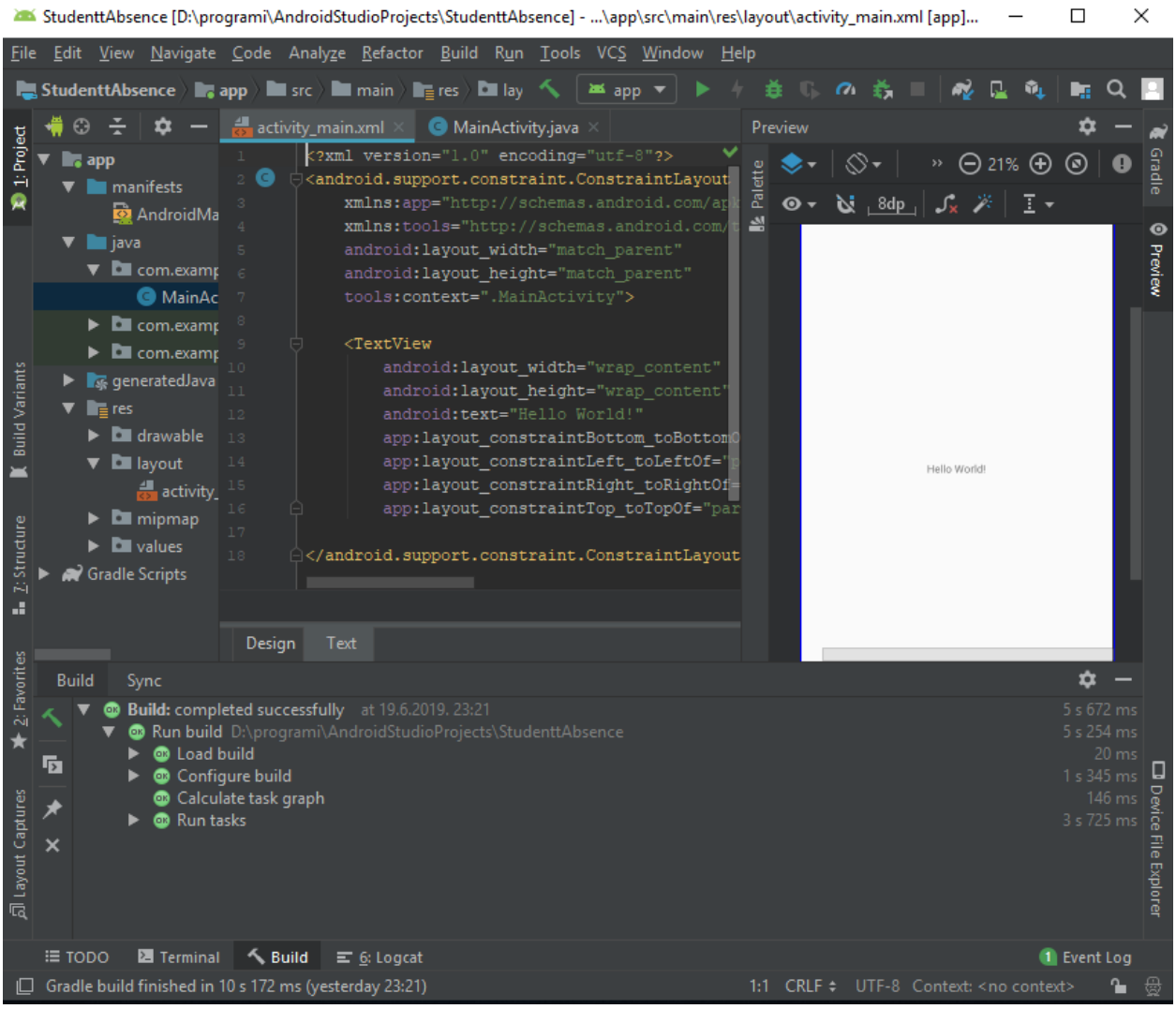
Lijevo se nalazi hijerarhija osnovnih mapa i dokumenata od kojih je najvažniji „*AndroidManifest.xml*” jer sadrži upute o tome kako otvoriti naš projekt u IDE.[4] Isto tako vidimo hijerarhiju i popis svih datoteka logike aplikacije (trenutno je to samo „*MainActivity”* datoteka unutar *app – java – com.example.studentAbsence* mape). Dok *UI* dizajn aplikacije

vidimo kao datoteku „activity_main.*xml*” u *app – res – layout* mapi. Desnim klikom na mapu dodajemo nove datoteke u naš projekt.



Slika 1.4. Početno sučelje novog projekta u Android Studiju

Kada je aktivno otvoren „*.xml*” dokument u radnom prozoru, na desnom rubu prozora se javlja opcija pretpregleda („*Preview*”) izgleda naše aplikacije a dolje lijevo u radnom prozoru se javljaju dvije metode programiranja dizajna: („*drag and drop*”) „*Design*” i „*Text*” (Slika 1.5.).



Slika 1.5. Aktivan ".xml" dokument u radnom prozoru

Naša aplikacija se pokreće i izvršava tako da se sinteziraju „*.xml*” dokumenati i klase unutar našeg paketa. Što znači da za svaki „*layout*” („*.xml*” dokument) moramo imati klasu koja će biti operativno sredstvo što se događa sa elementima na zaslonu, osluškuje događaje (klikove, povlačenja, tj. „*touch*” naredbe ali i neke druge poput glasovnih naredbi). Pa tako moramo napuhati („*inflate*”) naš „*layout*” unutar neke klase. Na Kôd 1.1. vidimo metodu koja se poziva pri stvaranju našeg početnog zaslona aplikacije „*onCreate()*” i poziv metode „*setContentView()*” koja napuhuje naš zaslon gradivnim elementima.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

Kôd 1.1. Metoda koja se poziva pri stvaranju početnog zaslona

4.2. Programski kôd dizajna zadatka (XML)

Glavni i početni zaslon aplikacije („*activity_main.xml*”) sastoji se od prostora za popis kolegija i gumba za dodavanje novog kolegija. Popis je realiziran korištenjem vanjske „*Recycler View*” komponente. Kôd 2.1. prikazuje dodavanje vanjske komponente u Gradle skriptu (koja je između ostaloga popis svih biblioteka koje se trebaju uključiti i koristiti za ispravno izvođenje aplikacije). A Kôd 2.2. prikazuje izgled „*activity_main.xml*” dokumenta s elementima sučelja.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'

    implementation 'com.android.support:recyclerview-v7:28.0.0'
}
```

Kôd 2.1. Gradle skripta

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".main.MainActivity"
    android:padding="20px">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerSubjects"
        android:layout_width="match_parent"
        android:layout_height="350dp"
        android:background="#aa44ff" />

    <Button
        android:id="@+id/btnAddNewSubject"
        android:layout_width="150dp"
        android:layout_height="50dp"
        android:text="@string/btnText_addSubject"
        android:layout_below="@+id/recyclerSubjects"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp" />
</RelativeLayout>
```

Kôd 2.2. „activity_main.xml”

Kôd 2.3. prikazuje „*layout*” za novi zaslon s kontejnerskim elementima („*FrameLayout*”) za fragmente. Unos novog kolegija se obavlja tako da se pokreće novi zaslon i u njemu se izmijene dva fragmenta (prvi za unos naziva kolegija a drugi za unos minimalne i maksimalne dolaznosti za konkretan kolegij). Kôd jednog dokumenta vrlo brzo može postati velik i dug, kao primjer priložen je „*.xml*” dokument za drugi fragment (vidi Prilog 1). Isto tako Java dokument može narasti dodavanjem novih funkcionalnosti stoga se treba pridržavati nekih pravila pisanja kôda, poput S. O. L. I. D. ili S. T. U. P. I. D. Principa. [5] [6]

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <FrameLayout
        android:id="@+id/fragmentContainer"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</RelativeLayout>
```

Kôd 2.3. „activity_input.xml”

4.3. Programski kôd logike zadatka (Java)

Kako ne bi ulazili u detalje i osnove pisanja Java kôda, prikazane su samo neke specifične logičke funkcionalnosti za aplikaciju, tj. rješenje zadatka.

Prvo se nameće baza podataka a korištena je Realm baza. Kreirana je pomoćna klasa za rad s bazom kako bi pojednostavili operacije s bazom a usput povećali preglednost samog kôda. Metode koje su definirane u ovoj klasi služe za reakciju na korisnikovu akciju. Stoga postoje metode za spremanje novog kolegija, brisanje kolegija i dodavanje izostanaka u postojećem kolegiju. Postoje i neke metode koje služe za internu komunikaciju same aplikacije poput dohvaćanja svih kolegija kako bi se ažurirao popis na početnom zaslonu, pronalazak određenog kolegija u bazi kako bi se dodali izostanci ili prikazali detalji te vrlo bitna metoda koja računa koliko sati smijemo izostati na osnovu podataka o kolegiju i trenutnom stanju dolaznosti. Kôd 3.1. prikazuje metodu za računanje maksimalnih sati koliko se smije izostati. Metoda prima kao parametre detalje o kolegiju koje smo unijeli pri stvaranju istog a vraća željenu vrijednost.

```
private int fetchMaxAbsence(int minPercent, int totalOfHours) {
    if (minPercent == 0){ return totalOfHours; }
    double temp = minPercent * 1.0 / 100 * totalOfHours;
    return (int) temp;
}
```

Kôd 3.1. Metoda koja vraća maksimalan broj sati koje smijemo izostati

Kako bi od korisnika dobili ispravan unos za naziv kolegija i detalje satnice kolegija, potrebno je provjeriti ima li smisla to što korisnik unosi te ga obavijestiti da je unio nešto pogrešno. Kôd 3.2. prikazuje metodu (prvog fragmenta) koja provjerava unos za naziv kolegija. Kôd koji prikazuje metodu (drugog fragmenta) koja provjerava unos satnice kolegija te ovisno o tôku izvršavanja ispisuje prikladnu poruku kao povratnu informaciju dan je u prilogu zbog dužine kôda (vidi Prilog 2).

```
private void goToStepTwo(){

    String name = etName.getText().toString().trim();
    if (name.isEmpty()) {
        Toast.makeText(getActivity(),
                getString(R.string.toast_invalid_subjectName),
                Toast.LENGTH_LONG).show();
     } else {
        IStepTwo.proceedToStepTwo(name.toUpperCase());
    }
}
```

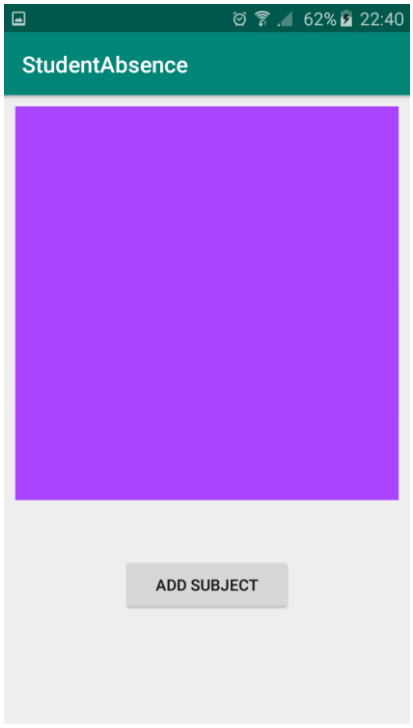
Kôd 3.2. Metoda koja provjerava unos naziva

Na kraju, tematika ovog rada je matematička obrada i implementacija dolazaka na nastavu konkretnog kolegija. To je realizirano tako da se nakon stvaranja kolegija omogući unos izostanaka, te se na osnovu podataka iz baze, tj. trenutnog stanja sati, ispiše poruka smije li korisnik izostati ili ne, te ako da, u kojoj količini. Prilog 3 prikazuje provjeru stanja i tôk programa sa opcijama koje će se ispisati na zaslonu na akciju korisnika na predviđeni gumb (vidi Prilog 3).

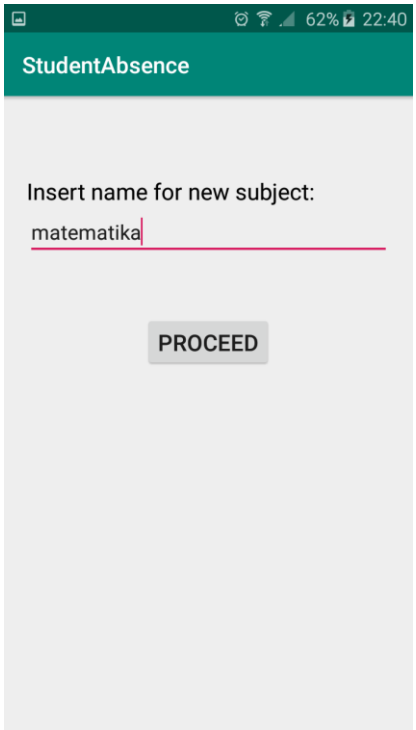
4.4. Instalacija aplikacije

Prvo treba povezati mobilni uređaj i računalo USB kabelom (pričekati eventualnu instalaciju driver-a). Odabrati „Postavke" na mobilnom uređaju. Zatim odabrati „O uređaju" pa odabrati „Podaci o softveru" te pritiskati „Broj verzije" dok se ne pojavi obavijest o uključenju programerskog načina rada. Nakon toga u Android Studiju pokrenuti aplikaciju na uređaju (pričekati dok se osvježi popis dostupnih uređaja te odabrati željeni). [7]

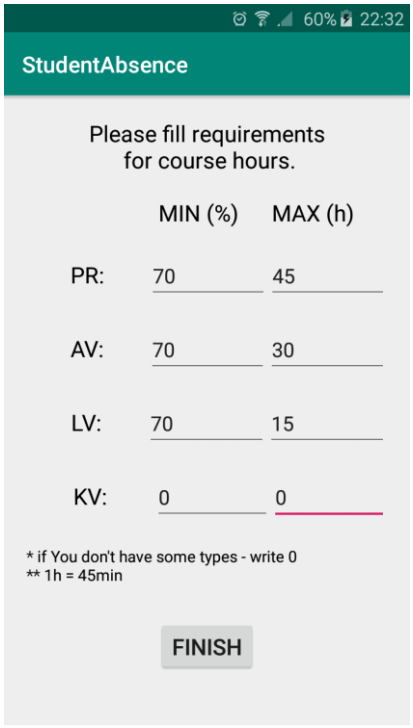
4.5. Simulacija na realnom uređaju



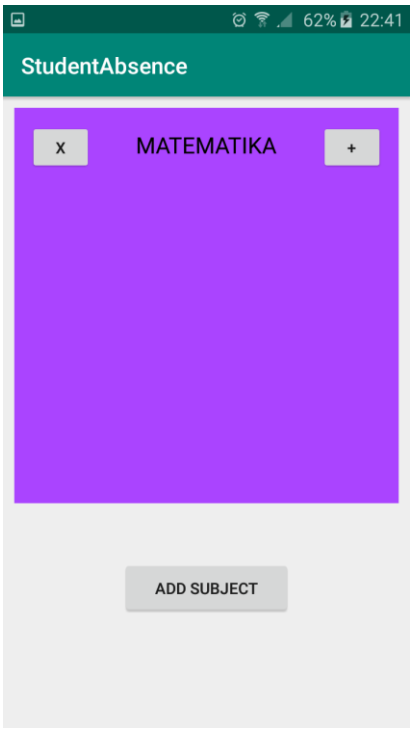
Slika 2.1. Početni zaslon



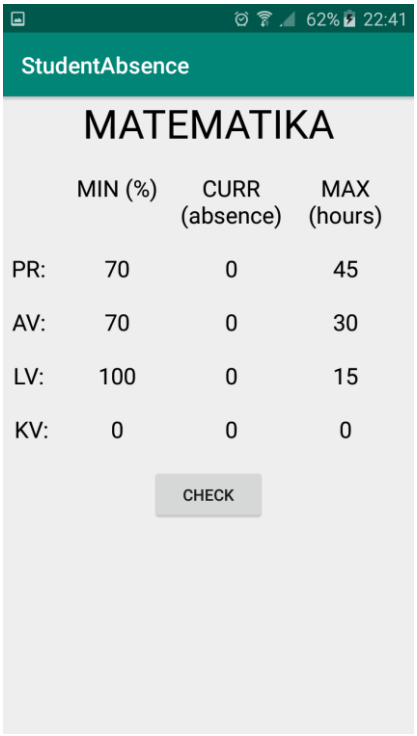
Slika 2.2. Unos naziva kolegija



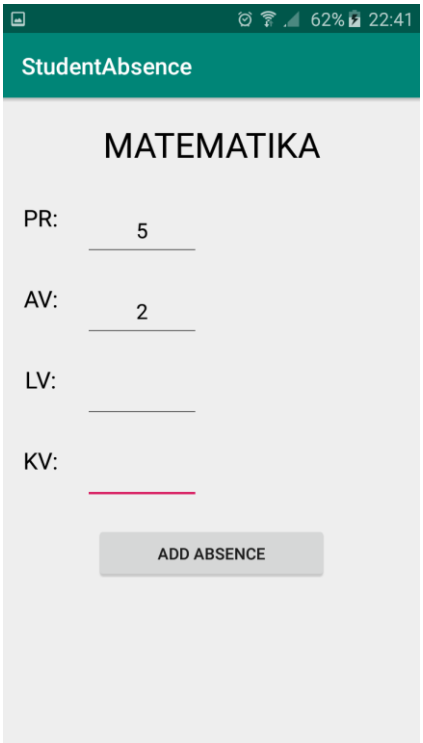
Slika 2.3. Unos minimalnog postotka i ukupnog broja sati



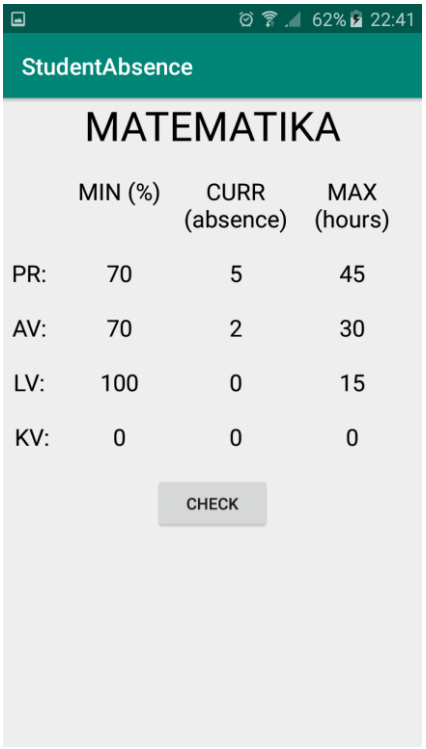
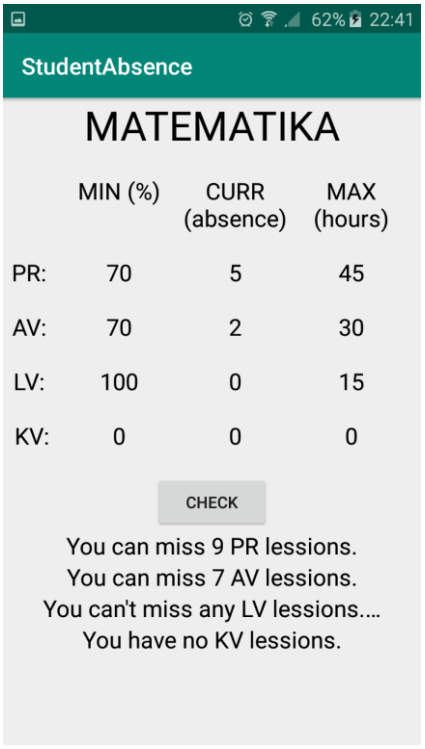
Slika 2.4. Izgled početnog zaslona nakon dodavanja



Slika 2.5. Detalji novog kolegija



Slika 2.6. Dodavanje izostanaka

nakon dodavanja	*sa početnog zaslona (+)*
	
Slika 2.7. Prikazana promjena nakon dodavanja izostanaka	*Slika 2.8. Prikazani sati koje možemo dodatno izostati*

5. Zaključak

Ovaj rad treba poslužiti kao uvid u materiju, konkretnije programiranje aplikacije. Odabran je Java objektno orijentirani jezik i okruženje Android Studio. Zadatak je bio jednostavan za iskusnije programere ali upravo dovoljno težak za programere početnike. Aplikacija je uspješno realizirana i prikazani su neki elementi poput rada s lokalnom bazom podataka, fragmentima, „*RecyclerView*" objektom, više zaslona, komunikacija i prosljeđivanje privremenih podataka u aplikaciji („*Intent*" objekt) te implementacija klikova korisnika pomoću sučelja („*interface*"), itd.. Sve su ovo svakodnevni elementi koje svaki programer mora znati i imati šira znanja o njima, kako bi ih mogao koristiti na raznim projektima. Postoji veliki spektar proširenja aplikacije. Kao prvo mogli bismo unaprijediti sam dizajn, oblikovati gumbe, izmijeniti boje i dodati prijelazne efekte pri izmjeni objekata na zaslonu. Zatim bismo mogli dodati nove funkcionalnosti poput datuma koji bi unosili i prikazivali za svaki izostanak. Dodati bodove koje skupljamo na kolegiju iz aktivnosti te krenuti u smjeru dnevnika za fakultetske obaveze ili nešto slično. Isto tako danas se sve više spominje *online* baza podataka za sve, stoga jednog dana možda unaprijediti aplikaciju da sama pomoću GPS lokatora unosi dolaznost ili izostanke ovisno o rasporedu predavanja i našoj lokaciji u vrijeme predavanja.

LITERATURA

[1] Link na prijedlog foruma, tj. vrlo velike baze rješenja za početničke probleme:

- https://stackoverflow.com

(23.06.2019.)

[2] Službeni link na web stranicu za razvojne programere android aplikacija:

- https://developer.android.com

(23.06.2019.)

[3] Link na službenu stranicu gradivnih blokova UI dizajna aplikacije

- https://developer.android.com/guide/topics/ui/declaring-layout

(23.06.2019.)

[4] Link na službenu stranicu s opisom „*AndroidManifest.xml*" datoteke

- https://developer.android.com/guide/topics/manifest/manifest-intro

(23.06.2019.)

[5] Link na stranicu na kojoj su objašnjeni S. O. L. I. D. principi u programiranju

- https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

(23.06.2019.)

[6] Link na stranicu na kojoj su objašnjeni S. T. U. P. I. D. principi sa usporedbom S. O. L. I. D. principa u programiranju

- https://williamdurand.fr/2013/07/30/from-stupid-to-solid-code/

(23.06.2019.)

[7] Link na službenu stranicu s uputama kako instalirati Android aplikaciju na mobitel

- https://developer.android.com/training/basics/firstapp/running-app

(23.06.2019.)

SAŽETAK

U ovom radu rješavan je problemski zadatak pisanja Android aplikacije. Tematika aplikacije je osobno praćenje dolaznosti na sve oblike i satnice nekog kolegija na fakultetu. Korišteno okruženje je Android Studio, izgled aplikacije pisan je u XML dokumentima a logika i funkcionalnosti u Java objektno orijentiranom programskom jeziku. Aplikacija je uspješno realizirana, broji izostanke i ispisuje detalje o određenom kolegiju, nakon stvaranja kolegija sa svim oblicima i satnicama. Uspješno je instalirana na stvarnom uređaju Samsung Galaxy S4.

Ključne riječi: aplikacija, android aplikacija, aplikacija za dolaznost, aplikacija za izostanke, fragmenti u aplikaciji, više zaslona u aplikaciji, recyclerview u aplikaciji, realm baza u aplikaciji

ABSTRACT

ANDROID APPLICATION FOR CLASS ATTENDANCE RECORDING

In this paper (in form of images or code screenshots) starting steps into writing an Android application are given and explained. Goal of this paper was writing an app which can record class attendance for user after creating new college course in database with all forms and hour rates. Developing environment was Android Studio, design was created with XML files and logic was writen in Java object oriented programming language. Application was created successfully - it counts absent hours and calculates and shows details if absence is possible or not for each course in database. Application was installed and tested successfully on mobile device Samsung Galaxy S4.

Keywords: application, android app, app for class attendance, app for class absence, fragments in android app, multiple activity in app, recyclerview in app, realm database in android app

ŽIVOTOPIS

Tomislav Gelešić rođen je u Daruvaru, 1996. godine. Završio je Opću Gimnaziju na na češkom jeziku, model C, 2016. godine. Trenutno je student 3. godine sveučilišnog preddiplomskog smjera računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. 2019. godine na proljeće prisustvovao je radionici za izradu iOS aplikacija (tvrtka Gauss Development, Osijek) te je savladao osnove Swift jezika i Xcode alata. Već treću godinu radi kao turistički animator te uz hrvatski govori: engleski, njemački, češki a služi se i talijanskim jezikom. U ljeto 2019. godine je postavljen za voditelja manjeg animacijskog tima. Vlada naučenim gradivom na dosadašnjem studiju uz poseban interes za: načela i principe pisanja programskog kôda, Java, Swift, CSS, HTML, Android Studio, Xcode. Solidno se služi Microsoft Office alatima.

PRILOZI

Prilog 1

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="20dp">
    <TextView
        android:id="@+id/tvInputTypes"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/tvTextStepTwo_lectureTypes"
        android:textColor="@color/inputTextColor"
        android:textSize="@dimen/inputTextSize"
        android:layout_below="@+id/etNameInput"
        android:textAlignment="center"/>
    <LinearLayout
        android:id="@+id/linLayoutInput_FirstRow"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="20dp"
        android:layout_below="@id/tvInputTypes">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/inputTextColor"
            android:textSize="@dimen/inputTextSize"
            android:layout_weight="1"
            android:textAlignment="center"/>
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/inputTextColor"
        android:textSize="@dimen/inputTextSize"
        android:text="MIN (%)"
        android:layout_weight="1"
        android:textAlignment="textEnd"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/inputTextColor"
        android:textSize="@dimen/inputTextSize"
        android:text="MAX (h)"
        android:layout_weight="1"
        android:textAlignment="center"/>
</LinearLayout>
<LinearLayout
    android:id="@+id/linLayoutInput_SecondRow"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="20dp"
    android:layout_below="@+id/linLayoutInput_FirstRow">
    <TextView
        android:id="@+id/tvPR"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/inputTextColor"
        android:textSize="@dimen/inputTextSize"
        android:text="PR:"
        android:layout_weight="1"
```

```
        android:textAlignment="center"/>
    <EditText
        android:id="@+id/etMinPR"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="phone"/>
    <EditText
        android:id="@+id/etMaxPR"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="phone"/>
</LinearLayout>
<LinearLayout
    android:id="@+id/linLayoutInput_ThirdRow"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="20dp"
    android:layout_below="@+id/linLayoutInput_SecondRow">
    <TextView
        android:id="@+id/tvAV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/inputTextColor"
        android:textSize="@dimen/inputTextSize"
        android:text="AV:"
        android:layout_weight="1"
        android:textAlignment="center"/>
    <EditText
        android:id="@+id/etMinAV"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="phone"/>
    <EditText
        android:id="@+id/etMaxAV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="phone"/>
</LinearLayout>
<LinearLayout
    android:id="@+id/linLayoutInput_FourthRow"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="20dp"
    android:layout_below="@+id/linLayoutInput_ThirdRow">
    <TextView
        android:id="@+id/tvLV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/inputTextColor"
        android:textSize="@dimen/inputTextSize"
        android:text="LV:"
        android:layout_weight="1"
        android:textAlignment="center"/>
    <EditText
        android:id="@+id/etMinLV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
```

```
        android:inputType="phone"/>
    <EditText
        android:id="@+id/etMaxLV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="phone"/>
</LinearLayout>
<LinearLayout
    android:id="@+id/linLayoutInput_FifthRow"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="20dp"
    android:layout_below="@+id/linLayoutInput_FourthRow">
    <TextView
        android:id="@+id/tvKV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/inputTextColor"
        android:textSize="@dimen/inputTextSize"
        android:text="KV:"
        android:layout_weight="1"
        android:textAlignment="center"/>
    <EditText
        android:id="@+id/etMinKV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="phone"/>
    <EditText
        android:id="@+id/etMaxKV"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="phone"/>
    </LinearLayout>
    <TextView
        android:id="@+id/tvNotification_step2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/tvTextStepTwo_notificationTypes"
        android:textColor="@color/inputTextColor"
        android:layout_below="@+id/linLayoutInput_FifthRow"
        android:layout_marginTop="20dp"/>
    <Button
        android:id="@+id/btnFinishInput"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btnText_finishInput"
        android:textSize="@dimen/inputTextSize"
        android:layout_below="@id/tvNotification_step2"
        android:layout_marginTop="30dp"
        android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

Prilog 2

```
private void finishInput() {
        String minPR_ = etMinPR.getText().toString();
        String minAV_ = etMinAV.getText().toString();
```

```
String minLV_ = etMinLV.getText().toString();
String minKV_ = etMinKV.getText().toString();

String maxPR_ = etMaxPR.getText().toString();
String maxAV_ = etMaxAV.getText().toString();
String maxLV_ = etMaxLV.getText().toString();
String maxKV_ = etMaxKV.getText().toString();

int minPR = -1, maxPR = -1, minAV = -1, maxAV = -1, minLV = -1, maxLV = -1, minKV = -1, maxKV = -1;

boolean ready = true;

if (minPR_.startsWith("0")) {
    minPR = 0;
} else {
    try {
        minPR = Integer.parseInt(minPR_);
    } catch (NumberFormatException e) {
        ready = false;
        Toast.makeText(getContext(), "Invalid 'MIN - PR' field input.", Toast.LENGTH_LONG).show();
    }
}

if (minAV_.startsWith("0")) {
    minAV = 0;
} else {
    try {
        minAV = Integer.parseInt(minAV_);
    } catch (NumberFormatException e) {
        ready = false;
```

```
            Toast.makeText(getActivity(), "Invalid 'MIN - AV' field input.",
Toast.LENGTH_LONG).show();
        }
    }

    if (minLV_.startsWith("0")) {
        minLV = 0;
    } else {
        try {
            minLV = Integer.parseInt(minLV_);
        } catch (NumberFormatException e) {
            ready = false;
            Toast.makeText(getActivity(), "Invalid 'MIN - LV' field input.",
Toast.LENGTH_LONG).show();
        }
    }

    if (minKV_.startsWith("0")) {
        minKV = 0;
    } else {
        try {
            minKV = Integer.parseInt(minKV_);
        } catch (NumberFormatException e) {
            ready = false;
            Toast.makeText(getActivity(), "Invalid 'MIN - KV' field input.",
Toast.LENGTH_LONG).show();
        }
    }

    if (maxPR_.startsWith("0")) {
        maxPR = 0;
    } else {
```

```
            try {
                maxPR = Integer.parseInt(maxPR_);
            } catch (NumberFormatException e) {
                ready = false;
                Toast.makeText(getActivity(), "Invalid 'MAX - PR' field input.",
Toast.LENGTH_LONG).show();
            }
        }

        if (maxAV_.startsWith("0")) {
            maxAV = 0;
        } else {
            try {
                maxAV = Integer.parseInt(maxAV_);
            } catch (NumberFormatException e) {
                ready = false;
                Toast.makeText(getActivity(), "Invalid 'MAX - AV' field input.",
Toast.LENGTH_LONG).show();
            }
        }

        if (maxLV_.startsWith("0")) {
            maxLV = 0;
        } else {
            try {
                maxLV = Integer.parseInt(maxLV_);
            } catch (NumberFormatException e) {
                ready = false;
                Toast.makeText(getActivity(), "Invalid 'MAX - LV' field input.",
Toast.LENGTH_LONG).show();
            }
        }
```

```
        if (maxKV_.startsWith("0")) {
            maxKV = 0;
        } else {
            try {
                maxKV = Integer.parseInt(maxKV_);
            } catch (NumberFormatException e) {
                ready = false;
                Toast.makeText(getActivity(), "Invalid 'MAX - KV' field input.",
Toast.LENGTH_LONG).show();
            }
        }

        if (minPR < 0 || minPR > 100) {
            ready = false;
            Toast.makeText(getContext(), "Invalid 'MIN - PR' field input.",
Toast.LENGTH_LONG).show();
        }
        if (minAV < 0 || minAV > 100) {
            ready = false;
            Toast.makeText(getContext(), "Invalid 'MIN - AV' field input.",
Toast.LENGTH_LONG).show();
        }
        if (minLV < 0 || minLV > 100) {
            ready = false;
            Toast.makeText(getContext(), "Invalid 'MIN - LV' field input.",
Toast.LENGTH_LONG).show();
        }
        if (minKV < 0 || minKV > 100) {
            ready = false;
            Toast.makeText(getContext(), "Invalid 'MIN - KV' field input.",
Toast.LENGTH_LONG).show();
```

```
        }
        if (maxPR < 0 || maxPR > 60 || ((maxPR == 0) && minPR != 0)) {
            ready = false;
            Toast.makeText(getContext(), "Invalid 'MAX - PR' field input.",
Toast.LENGTH_LONG).show();
        }
        if (maxAV < 0 || maxAV > 60 || ((maxAV == 0) && minAV != 0)) {
            ready = false;
            Toast.makeText(getContext(), "Invalid 'MAX - AV' field input.",
Toast.LENGTH_LONG).show();
        }
        if (maxLV < 0 || maxLV > 60 || ((maxLV == 0) && minLV != 0)) {
            ready = false;
            Toast.makeText(getContext(), "Invalid 'MAX - LV' field input.",
Toast.LENGTH_LONG).show();
        }
        if (maxKV < 0 || maxKV > 60 || ((maxKV == 0) && minKV != 0)) {
            ready = false;
            Toast.makeText(getContext(), "Invalid 'MAX - KV' field input.",
Toast.LENGTH_LONG).show();
        }
        if (ready) {
            IStepFinish.proceedToFinish(minPR, minAV, minLV, minKV, maxPR, maxAV, maxLV,
maxKV);
        }
    }
```

Prilog 3

```
this.btnCheck.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
        boolean additionalMessagePR = true, donePR = false;
        boolean additionalMessageAV = true, doneAV = false;
        boolean additionalMessageLV = true, doneLV = false;
        boolean additionalMessageKV = true, doneKV = false;

        int prMin = intent.getIntExtra("minPR", 0);
        int prCurr = intent.getIntExtra("currPR", 0);
        int prMax = intent.getIntExtra("maxPR", 0);

        int avMin = intent.getIntExtra("minAV", 0);
        int avCurr = intent.getIntExtra("currAV", 0);
        int avMax = intent.getIntExtra("maxAV", 0);

        int lvMin = intent.getIntExtra("minLV", 0);
        int lvCurr = intent.getIntExtra("currLV", 0);
        int lvMax = intent.getIntExtra("maxLV", 0);

        int kvMin = intent.getIntExtra("minKV", 0);
        int kvCurr = intent.getIntExtra("currKV", 0);
        int kvMax = intent.getIntExtra("maxKV", 0);

        int maxAbsence = 0;
// PR
        if (prMax == 0) {
            tvDescriptionPR.setText("You have no PR lessions.");
            additionalMessagePR = false;
            donePR = true;
        } else {
            if (prMin != 100 && !donePR) {
                maxAbsence = prMax - fetchMaxAbsence(prMin, prMax);
                if (prCurr >= maxAbsence) {
                    tvDescriptionPR.setText("You can't miss any PR lessions.");
```

```
                    additionalMessagePR = false;
                } else {
                    tvDescriptionPR.setText("You can miss " + (maxAbsence - prCurr) + " PR lessions.");
                    additionalMessagePR = false;
                }
            } else {
                if (!donePR) {
                    tvDescriptionPR.setText("You can't miss any PR lessions. (100%)");
                    additionalMessagePR = false;
                }
            }
        }
// AV
        if (avMax == 0) {
            tvDescriptionAV.setText("You have no AV lessions.");
            additionalMessageAV = false;
            doneAV = true;
        } else {
            if (avMin != 100 && !doneAV) {
                maxAbsence = avMax - fetchMaxAbsence(avMin, avMax);
                if (avCurr >= maxAbsence) {
                    tvDescriptionAV.setText("You can't miss any AV lessions.");
                    additionalMessageAV = false;
                } else {
                    tvDescriptionAV.setText("You can miss " + (maxAbsence - avCurr) + " AV lessions.");
                    additionalMessageAV = false;
                }
            } else {
                if (!doneAV) {
                    tvDescriptionAV.setText("You can't miss any AV lessions. (100%)");
```

```
                    additionalMessageAV = false;
                }
            }
        }
// LV
        if (lvMax == 0) {
            tvDescriptionLV.setText("You have no LV lessions.");
            additionalMessageLV = false;
            doneLV = true;
        } else {

            if (lvMin != 100 && !doneLV) {
                maxAbsence = lvMax - fetchMaxAbsence(lvMin, lvMax);
                if (lvCurr >= maxAbsence) {
                    tvDescriptionLV.setText("You can't miss any LV lessions.");
                    additionalMessageLV = false;
                } else {
                    tvDescriptionLV.setText("You can miss " + (maxAbsence - lvCurr) + " LV
lessions.");
                    additionalMessageLV = false;
                }
            } else {
                if (!doneLV) {
                    tvDescriptionLV.setText("You can't miss any LV lessions. (100%)");
                    additionalMessageLV = false;
                }
            }
        }
// KV
        if (kvMax == 0) {
            tvDescriptionKV.setText("You have no KV lessions.");
            additionalMessageKV = false;
```

```
                doneKV = true;
            } else {
                if (kvMin != 100 && !doneKV) {
                    maxAbsence = kvMax - fetchMaxAbsence(kvMin, kvMax);
                    if (kvCurr >= maxAbsence) {
                        tvDescriptionKV.setText("You can't miss any KV lessions.");
                        additionalMessageKV = false;
                    } else {
                        tvDescriptionKV.setText("You can miss " + (maxAbsence - kvCurr) + " KV lessions.");
                        additionalMessageKV = false;
                    }
                } else {
                    if (!doneKV) {
                        tvDescriptionKV.setText("You can't miss any KV lessions. (100%)");
                        additionalMessageKV = false;
                    }
                }
            }

            if (additionalMessagePR && additionalMessageAV && additionalMessageLV && additionalMessageKV) {
                tvDescriptionPR.setText("You can't miss anything anymore!");
                tvDescriptionAV.setText("");
                tvDescriptionLV.setText("");
                tvDescriptionKV.setText("");
            }
        }
    });
}
```