ANDROID KVIZ APLIKACIJA

Ivanković, Petar

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Preddiplomski sveučilišni studij

ANDROID KVIZ APLIKACIJA

Završni rad

Petar Ivanković

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 20.09.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Petar Ivanković
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3926, 26.07.2016.
OIB studenta:	08661211201
Mentor:	Izv. prof. dr. sc. Krešimir Nenadić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Android kviz aplikacija
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	20.09.2021.
Datum potvrde ocjene Odbora:	22.09.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 25.09.2021.

Ime i prezime studenta:	Petar Ivanković
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3926, 26.07.2016.
Turnitin podudaranje [%]:	12

Ovom izjavom izjavljujem da je rad pod nazivom**: Android kviz aplikacija**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Krešimir Nenadić

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:

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

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja __Petar Ivanković_____, OIB: __08661211201_____, student/ica Fakulteta elektrotehnike,
računarstva i informacijskih tehnologija Osijek na studiju _Preddiplomski sveučilišni studij Računarstvo_, kao autor/ica ocjenskog rada pod naslovom: _Android kviz aplikacija**,**
dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti

b) studentima/icama i djelatnicima/ama ustanove

c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

__Osijek, 25.09.2021._____ _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

SADRŽAJ

1. UVOD

U današnjem društvu svakodnevica je korištenje mobitela i interneta, a samim time i mobilnih aplikacija. Postoje razne vrste aplikacija, neke od njih služe za komunikaciju, a neke olakšavaju svakodnevni život, poput aplikacija za bankarske usluge. Među tim aplikacijama su se našle i kviz aplikacije koje su korisne, zanimljive i korisniku služe kao oblik razonode. U današnje vrijeme kvizovi su jako popularni i sudjelovanje u kvizovima potiče socijalizaciju, intelektualni napredak i razvoj, a u krajnjem slučaju sudionicima donosi i određene nagrade. Sve se to može prenijeti na pojedinca i preko mobilne aplikacije.

Tema ovog završnog rada je izrada Android kviz aplikacije. Rad sadrži pet poglavlja. U drugom poglavlju, iza uvoda, prikazano je pet sličnih rješenja, odnosno kviz aplikacija. Treće poglavlje sadrži opis postupka izrade aplikacije, koje su sve tehnologije korištene i kako, prikaz programskog kôda i strukturu baze podataka. Četvrto poglavlje opisuje strukturu aplikacije i sadrži slike zaslona svakog dijela aplikacije uz opis. U posljednjem, petom poglavlju, iznesen je zaključak cijelog rada.

1.1. Zadatak završnog rada

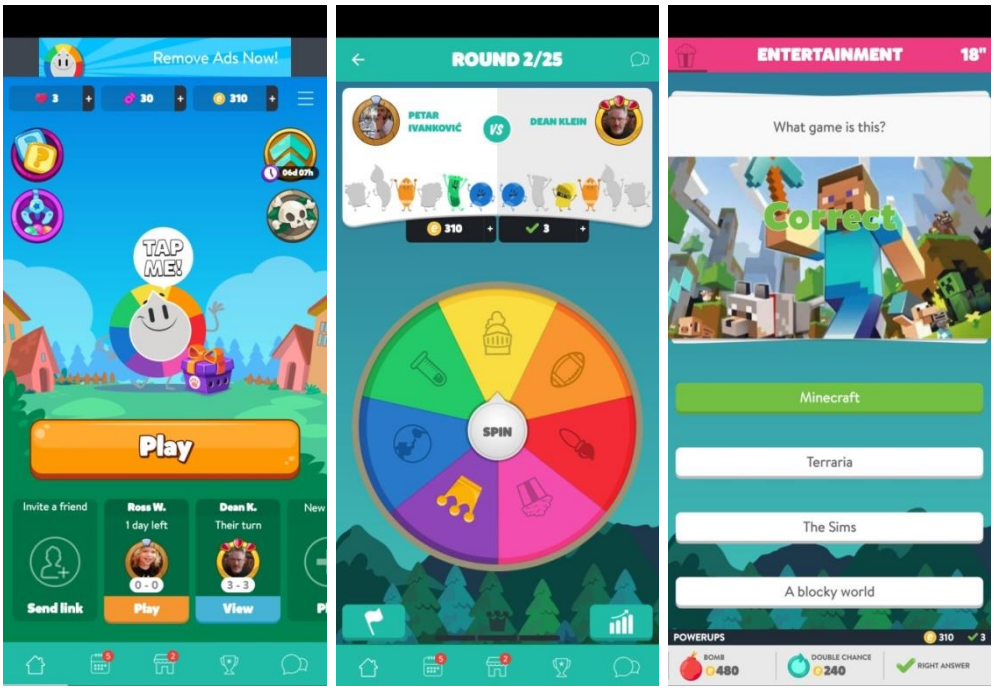
Izraditi Android aplikaciju pomoću koje korisnik može igrati kviz. Bazu pitanja potrebno je postaviti na neki online servis. Omogućiti unutar aplikacije predlaganje novih pitanja koji se administratoru šalju putem elektroničke pošte. Predvidjeti različite tipove igre: samostalna igra, igra protiv drugog igrača.

2. PREGLED SLIČNIH RJEŠENJA

U ovom poglavlju prikazana su slična rješenja koja su dostupna na trgovini Google Play. Sljedećih pet aplikacija koje su predstavljene pažljivo su odabrane, uzimajući u obzir popularnost, odnosno broj skidanja aplikacije, ocjene, dizajn i samu pristupačnost aplikacije. Rješenja su ukratko opisana te prikazana slikama.

2.1. Trivia Crack

Trivia Crack je daleko najpopularnija kviz aplikacija koja je dostupna na trgovini Google Play. Prema [1] Trivia Crack preuzeta je više od 100 milijuna puta, što ju čini najpopularnijom. Ukupna ocjena ove aplikacije je 4.3 od mogućih 5, a u ocjenjivanju je sudjelovalo više od 7 milijuna korisnika. Aplikacija je dostupna za više od 20 jezika. Trivia Crack ima vrlo zanimljiv princip kako igrač dolazi do pobjede. Za razliku od ostalih kviz aplikacija, u ovoj aplikaciji igrač, kako bi pobijedio, mora skupiti šest karikatura likova koji predstavljaju određene kategorije na način da pruži točan odgovor u danoj kategoriji, što je vidljivo na slici 2.1. Kategorija se bira na način da se zavrti (engl. *spin*) kotačić koji sadrži kategorije te se pitanja postavljaju iz one kategorije koja pokazuje strelicom.

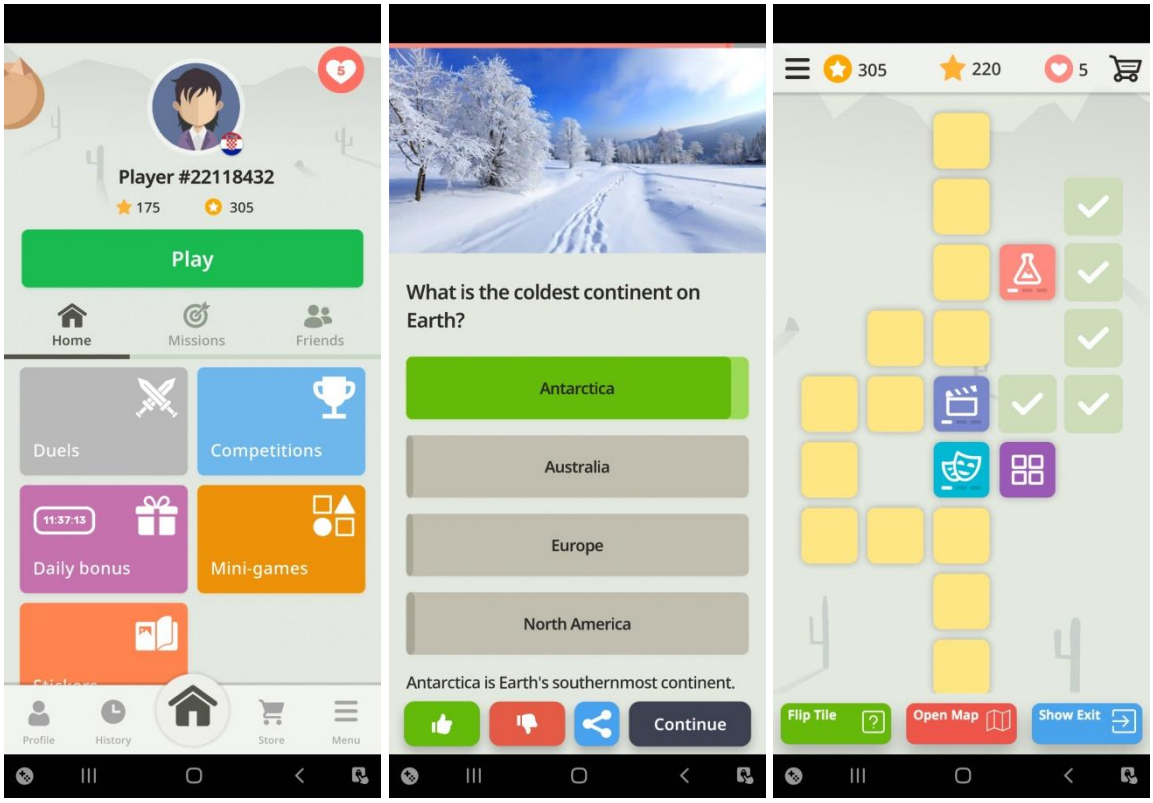


Slika 2.1. Trivia Crack – izgled korisničkog sučelja

(preuzeto s: https://play.google.com/store/apps/details?id=com.etermax.preguntados.lite&hl=hr&gl=US)

2.2. QuizzLand

Quizzland je jedna od popularnijih kviz aplikacija. Prema [2] instalirana je na više od 10 milijuna uređaja, a ima ukupnu ocjenu 4.7 od 5. Kao što je prikazano na slici 2.2., ova aplikacija je nešto drugačija od ostalih kviz aplikacija iz razloga što korisnici rješavaju razine na način da otvaraju nova pitanja koja se nalaze u kvadratićima koji skupa čine sustav nalik križaljci.

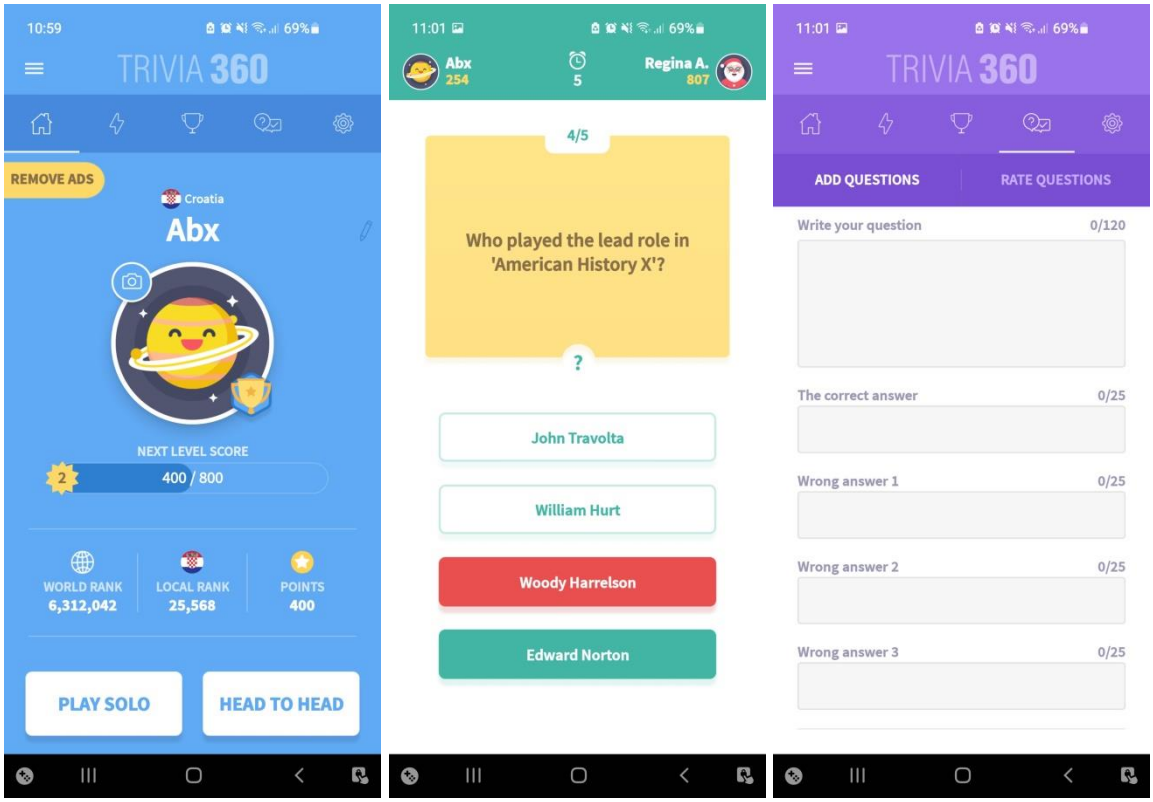


Slika 2.2. QuizzLand – izgled korisničkog sučelja

(preuzeto s: https://play.google.com/store/apps/details?id=com.xmonetize.quizzland&hl=hr&gl=US)

2.3. Trivia 360

Prema [3] ova aplikacija je instalirana na više od 5 milijuna uređaja i nosi ukupnu ocjenu 4.4 od mogućih 5. Trivia 360 je besplatna.



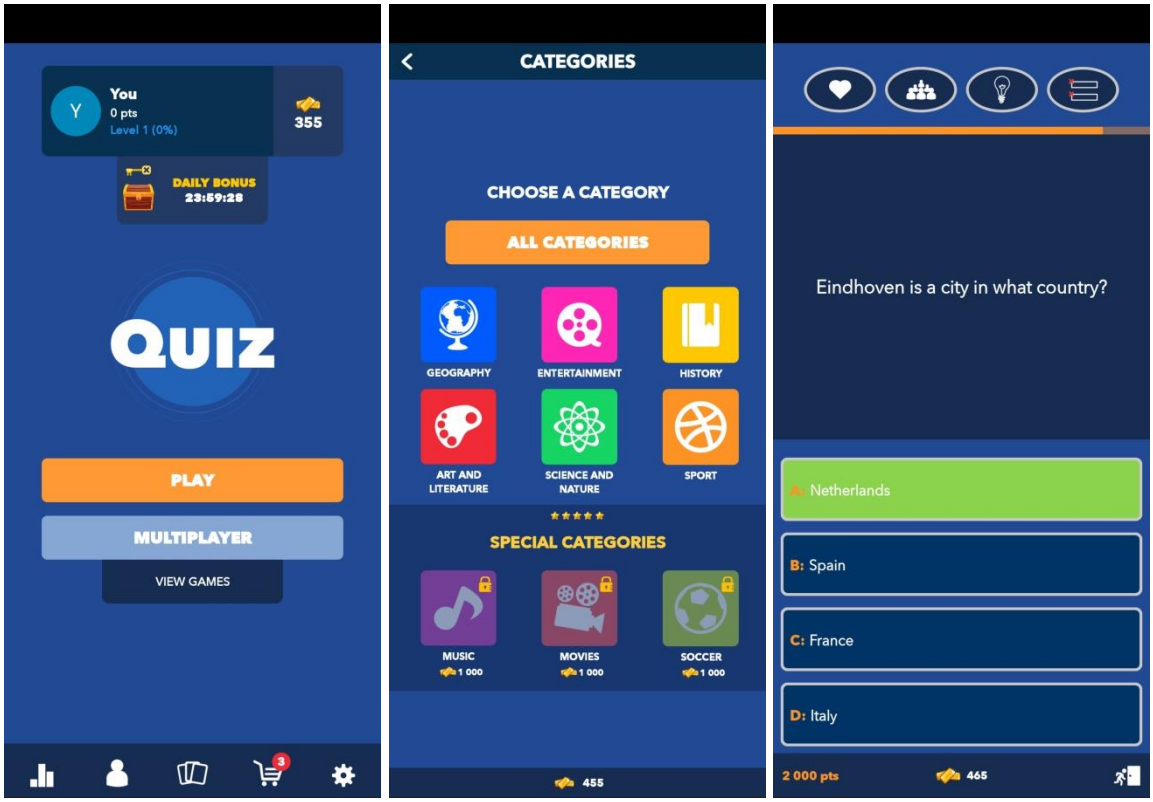
Slika 2.3. Trivia 360 – izgled korisničkog sučelja

(preuzeto s: https://play.google.com/store/apps/details?id=smartowlapps.com.quiz360&hl=hr&gl=US)

Na slici 2.3. može se vidjeti da korisnik može birati između dva tipa igre: samostalne i igre protiv ostalih korisnika. U tipu igre protiv ostalih korisnika, korisnici se ne natječu u isto vrijeme, već čekaju svoj red dok protivnik ne završi s igrom. Korisnici, osim postavljanja novih pitanja, imaju mogućnost i ocjenjivanja već postojećih pitanja kako bi ona bila što vjerodostojnija.

2.4. General Knowledge Quiz

Ova aplikacija napravljena je po uzoru na popularni TV kviz „Milijunaš", gdje korisnik odgovara na petnaest pitanja i online se natječe protiv drugog korisnika. Prema [4] ova aplikacija skinuta je više od 5 milijuna puta. U ocjenjivanju je sudjelovalo više od 140 tisuća korisnika, a aplikacija je skupila ocjenu 4.3 od 5. U aplikaciji korisnik može birati između šest popularnih kategorija, a pobjedom u kvizu ostvaruje nagrade korisne unutar igre.

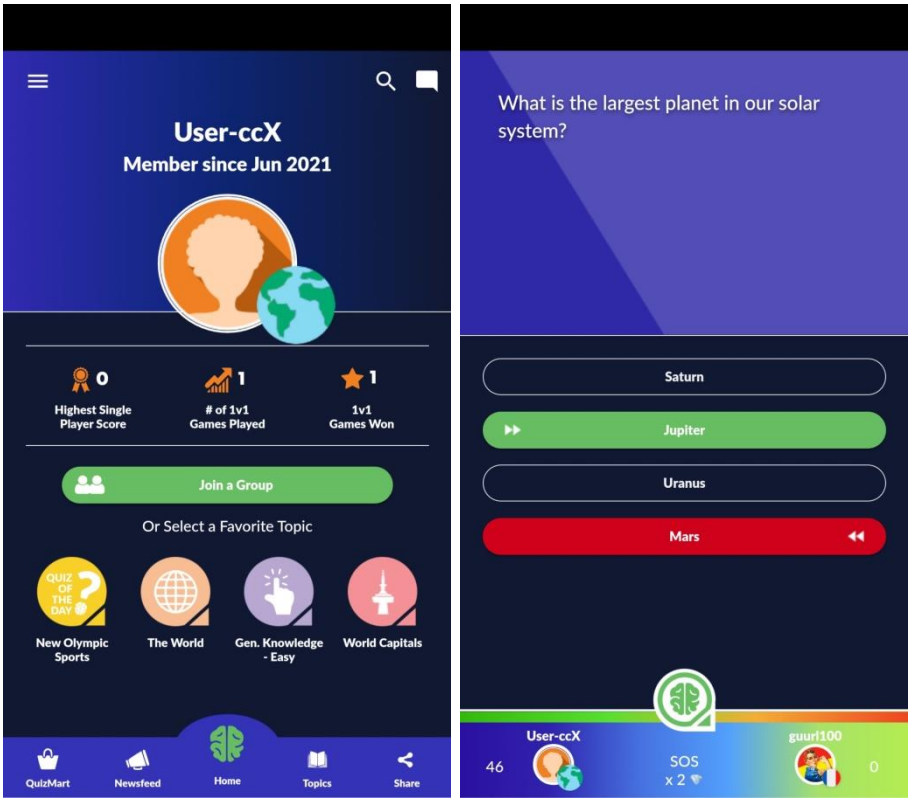


Slika 2.4. General Knowledge Quiz – izgled korisničkog sučelja

(preuzeto s: https://play.google.com/store/apps/details?id=com.creactive.quemquersermilionario.millonario.lite&hl=hr&gl=US)

2.5. Quizefy

Prema [5] ova aplikacija instalirana je na više od 50 tisuća uređaja, što ju čini najmanje popularnom od navedenih pet aplikacija. Korisnici u aplikaciji mogu birati između mnoštva kategorija te imaju izbor između samostalnog i online tipa igre protiv ostalih korisnika, što je prikazano na slici 2.5.



Slika 2.5. Quizefy – izgled korisničkog sučelja

(preuzeto s: https://play.google.com/store/apps/details?id=com.quizefy.game&hl=hr&gl=US)

3. OPIS POSTUPKA IZRADE APLIKACIJE

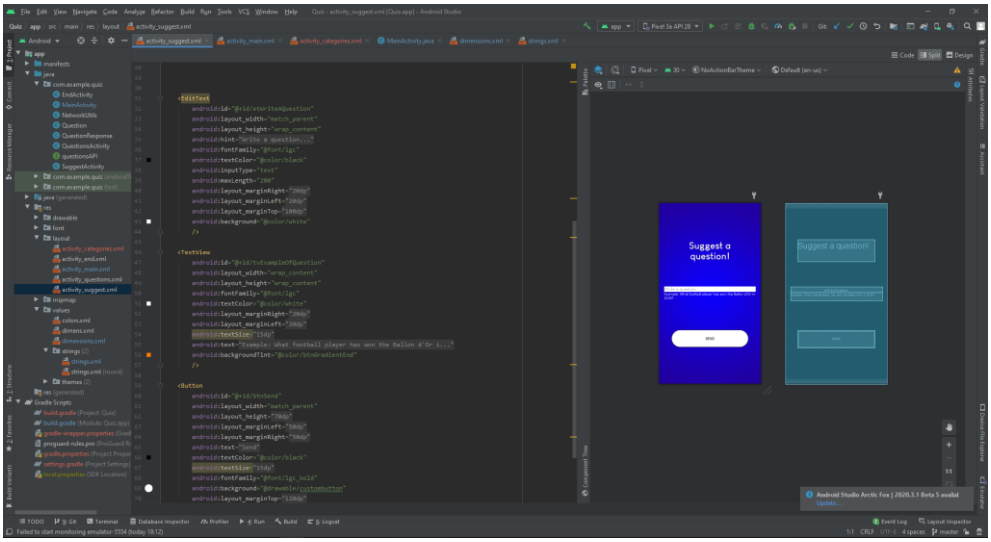
U ovom poglavlju opisane su korištene tehnologije, opisan je programski kôd uz slike važnih dijelova kôda te je opisana i struktura baze podataka.

3.1. Opis korištenih tehnologija

Za izradu aplikacije ovog završnog rada korišteno je razvojno okruženje Android Studio, Java programski jezik, XML opisni jezik, Firebase i Adobe Photoshop. Te tehnologije su kratko opisane u idućim potpoglavljima.

3.1.1. Android Studio

Android Studio je integrirano razvojno okruženje, koje je prema [6] službeno za Googleov Android operacijski sustav. Izgrađen je na JetBrains IntelliJ IDEA softveru i dizajniran isključivo za razvoj Android aplikacija. Android Studio podržava značajke kao što su pametno uređivanje, napredno refaktoriranje kôda i analize statičkog kôda. Android Studio dostupan je za četiri operacijska sustava: Windows, macOS, Linux i Chrome OS. U svibnju 2019. Kotlin je zamijenio Javu kao preferirani programski jezik za razvijanje Android aplikacija, no još uvijek su podržani i Java i C++. Na slici 3.1. može se vidjeti kako izgleda korisničko sučelje ovog razvojnog okruženja.



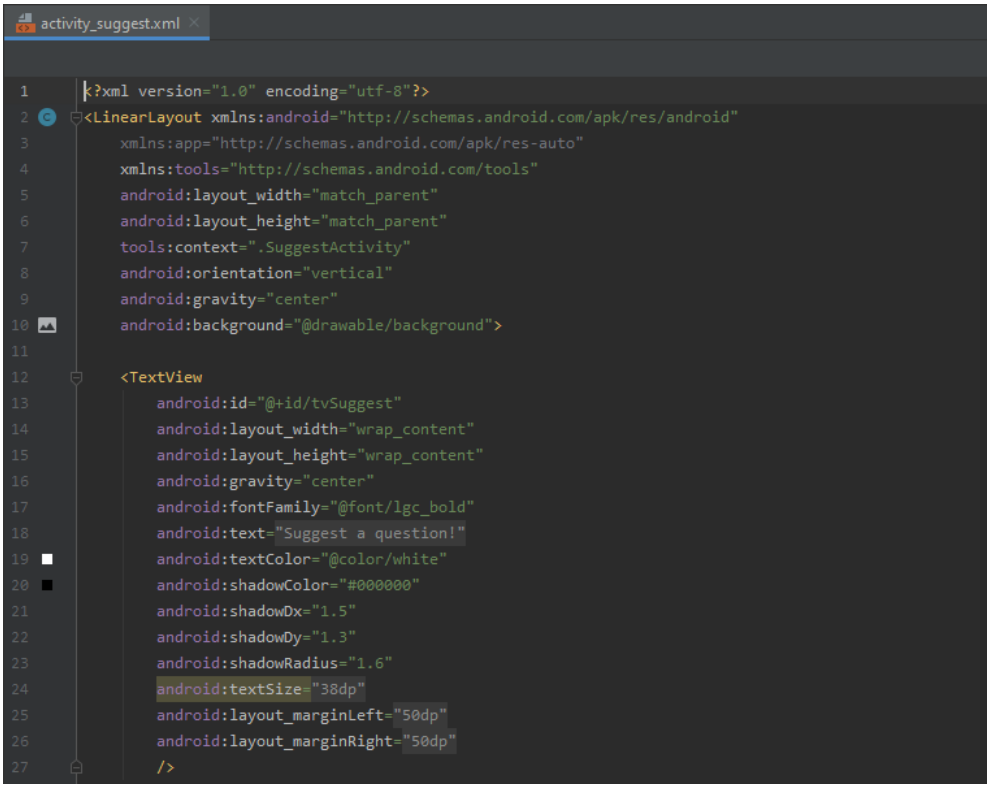
Slika 3.1. Android Studio – izgled korisničkog sučelja

3.1.2. Java programski jezik

Java je programski jezik i računalna platforma koju su razvili u tvrtci Sun Microsystems, a dizajnirali su ga James Gosling, Patrick Naughton i ostali, kako je navedeno u [7]. S razvojem jezika su započeli 1991. godine, a objavljen je 1995. godine. Java kao objektno-orijentirani jezik nastoji imati što manje moguće implementiranih zavisnosti. Sintaksa ovog jezika jako je slična jezicima poput C i C++. Java developerima dozvoljava rad po principu WORA (engl. *Write Once Run Anywhere*), odnosno „napiši jednom, pokreni svugdje“. Zadnja inačica Java 16 izdana je u Ožujku 2021., a Java 11 u Rujnu 2018. koja služi kao trenutno dugoročna podrška.

3.1.3. XML opisni jezik

XML opisni jezik (engl. *eXtensible Markup Language*) je standardizirani jezik za označavanje podataka, koji prema [8] definira komplet pravila za opisivanje, odnosno kodiranje dokumenata u formatu koji je čitljiv ljudima, a i računalu. Glavne karakteristike XML-a su jednostavnost, generalizacija i upotrebljivost preko interneta. Iako se XML fokusira na dokumente, ovaj jezik široko se koristi za reprezentaciju proizvoljnih struktura podataka, kako je navedeno u [9]. Sintaksa XML-a jako je slična sintaksi HTML-a (engl. *HyperText Markup Language*), iako su razvijeni s različitim namjenama. Na slici 3.2. prikazano je kako izgleda dio XML kôda u razvojnom okruženju Android Studio.



Slika 3.2. Primjer XML kôda u Android Studiu

3.1.4. Firebase

Firebase je platforma koja služi za ugradnju različitih funkcionalnosti u mobilne i web aplikacije. Firebase u isto vrijeme može pohraniti i sinkronizirati podatke. Ovu platformu razvila je tvrtka Firebase Inc. 2011. godine, a prema [10] od 2014. godine Firebase je u vlasništvu Google-a. Prvi proizvod Firebase-a bila je baza podataka u stvarnom vremenu, API koji sinkronizira podatke određene aplikacije i pohranjuje ih na Firebase-ov oblak. Danas se Firebase sastoji od mnoštva raznih usluga, od kojih se svaka može iskoristiti za potrebni oblik ili dio aplikacije. Neke od Firebase usluga su: Analytics, Authentication, Crash Reporting, Notifications, Realtime Database, Storage. Dodavanje Firebase usluga u projekt vrlo je jednostavno i zbog toga njegova popularnost raste. U ovom projektu korištene su dvije Firebase-ove usluge, Authentication i Realtime Database. Authentication usluga omogućava registraciju korisnika i prijavu korisnika u aplikaciju. Realtime Database usluga služi kako bi se pohranili podatci o korisnicima te sadrži sva pitanja kviza i ostale važne podatke.

3.1.5. Adobe Photoshop

Adobe Photoshop je grafički računalni program kojega su kreirala braća Knoll 1988. Godine, navodi se prema [11]. Iduće godine, 1989., John Knoll prodaje program firmi Adobe Systems, koja ga naziva *„Photoshop“*. Od tada Adobe Photoshop najpopularniji je program za obradu slike. Adobe Photoshop dostupan je za Windows i macOS operacijske sustave. Logo, pozadina aplikacije i slike kategorija izrađeni su pomoću nekih alata ovog programa, kao što su Rectangular Marque tool, Gradient tool i Horizontal Type tool.

Slika 3.3. Adobe Photoshop – izgled korisničkog sučelja

3.2. Razvoj aplikacije

U sljedećim potpoglavljima prikazani su koraci kreiranja aplikacije, najvažniji dijelovi kôda prikazani su slikom i opisani. Programski kôd je složen i sastoji se od više klasa i aktivnosti. Osim kôda prikazana je i struktura baze podataka korištene za ovaj projekt.

3.2.1. Gradle

Gradle je napredni alat za izgradnju koji se koristi kako bi se automatizirao i upravljao proces izgradnje te koji dopušta definiranje prilagođenih konfiguracija izgradnje, kako je navedeno u [12]. Svaka konfiguracija definira vlastiti set kôda i resursa, dok istodobno upotrebljava dijelove koji su zajednički svim verzijama aplikacije. Svaka zavisnost zahtjeva različitu vrstu biblioteke. Kako bi se aplikacija mogla razvijati potrebno je ubaciti određene implementacije. Na slici 3.4. prikazane su sve implementacije koje su potrebne kako bi

aplikacija funkcionirala. Dio implementacija odnosi se na uključivanje Firebase-ovih funkcija.

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'androidx.wear:wear:1.0.0'
    implementation 'com.google.firebase:firebase-auth:21.0.1'
    implementation 'com.google.firebase:firebase-database:20.0.1'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
    compileOnly 'com.google.android.wearable:wearable:2.6.0'

    implementation 'com.github.bumptech.glide:glide:4.12.0'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'
}
```

Slika 3.4. Prikaz gradle skripte

3.2.2. AndroidManifest datoteka

AndroidManifest.xml datoteka sastavni je dio svakog Android projekta, ta datoteka predstavlja sam korijen projekta. Sadrži informacije o paketima, o komponentama aplikacije kao što su aktivnosti, usluge i davatelji sadržaja. Ova datoteka brine i o zaštiti aplikacije, na način da je potrebno ubaciti određene dozvole kako bi se izvršile određene akcije. Ova datoteka deklarira API aplikacije koji će se koristiti. U ovoj aplikaciji pristupa se internetu pa je iz tog razloga implementirana dozvola za pristup internetu. Osim toga, u ovoj datoteci se nalazi i logo koji je korišten u aplikaciji te mogućnost micanja statusne trake sa zaslona, što je i izvedeno te je prikazano na slici 3.5.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.quiz">

    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher_round"
        android:label="Quiz"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Quiz"
        >
        <activity android:name=".SuggestActivity"
            android:screenOrientation="portrait"
            android:theme="@style/NoActionBarTheme"
            android:windowSoftInputMode="adjustResize"/>
        <activity
            android:name=".EndActivity"
            android:screenOrientation="portrait"
            android:theme="@style/NoActionBarTheme" />
        <activity
            android:name=".QuestionsActivity"
            android:screenOrientation="portrait"
            android:theme="@style/NoActionBarTheme" />
```

Slika 3.5. Prikaz dijela datoteke AndroidManifest.xml

3.2.3. Aktivnost *StartActivity*

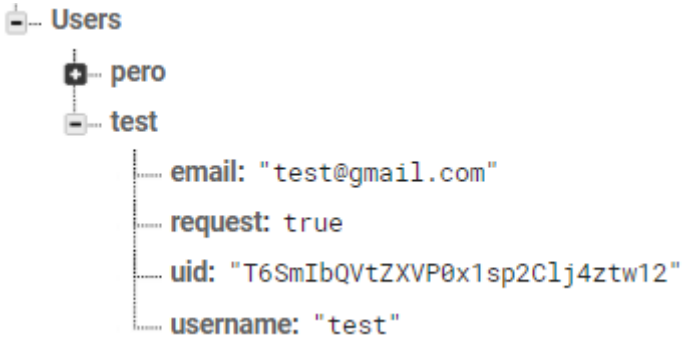
Ova aktivnost je prva pokrenuta aktivnost pri ulasku u aplikaciju, što je definirano unutar datoteke *AndroidManifest.xml*. U slučaju da korisnik nije registriran, prvo što treba napraviti je registrirati se, a zatim i prijaviti u aplikaciju. Ukoliko je korisnik već prijavljen, svaki put kada ponovno otvori aplikaciju, preskače početnu aktivnost i automatski se usmjerava na aktivnost glavnog izbornika, *MainActivity*. Prikaz tog kôda može se vidjeti na slici 3.6., u tom kôdu provjerava se postoji li objekt klase *FirebaseUser* koji označava korisnika, ukoliko postoji pomoću objekta klase *Intent* odmah se izvršava otvaranje željene aktivnosti.

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
if (user != null) {
    Intent intent = new Intent( packageContext: StartActivity.this, MainActivity.class);
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(intent);
}
```

Slika 3.6. Dio kôda koji se odnosi na preskakanje početne aktivnosti u slučaju već prijavljenog korisnika

3.2.4. Registracija i prijava korisnika

Kako bi se novi korisnik registrirao, u početnoj aktivnosti mora pritisnuti gumb *Register*. U aktivnosti registracije korisnika, od korisnika se zatražuje da ispuni tri polja, da unese nadimak, adresu elektroničke pošte i zaporku. Pomoću unesene adrese elektroničke pošte i zaporke korisnik se prijavljuje u aplikaciju. Za uspješnu registraciju postoje određeni kriteriji. Sva tri polja nikad ne smiju biti prazna. U drugom polju za unos adrese elektroničke pošte unesena adresa elektroničke pošte mora odgovarati pravom obliku adrese elektroničke pošte. Treće polje za unos koje se odnosi na zaporku ne smije sadržavati manje od 6 znakova. Ukoliko su sva tri polja pravilno popunjena, pritiskom na gumb *Register* korisnik se uspješno registrira, a to mu potvrđuje *Toast* poruka. Podatci koje je korisnik unio, osim zaporke, šalju se i spremaju u Firebase-ovu bazu podataka u stvarnom vremenu, unutar čvora *„Users“*. Za slanje podataka brinu se predefinirani protokoli kreirani od strane Firebase-a, koje uključujemo kroz gradle. Samo jedna registracija je moguća po adresi elektroničke pošte. Registriranog korisnika unutar baze podataka možemo pronaći po dijelu adrese elektroničke pošte prije znaka „@“. Na slici 3.7. prikazan je primjer registriranog korisnika u bazi podataka.



Slika 3.7. Primjer registriranog korisnika u bazi podataka

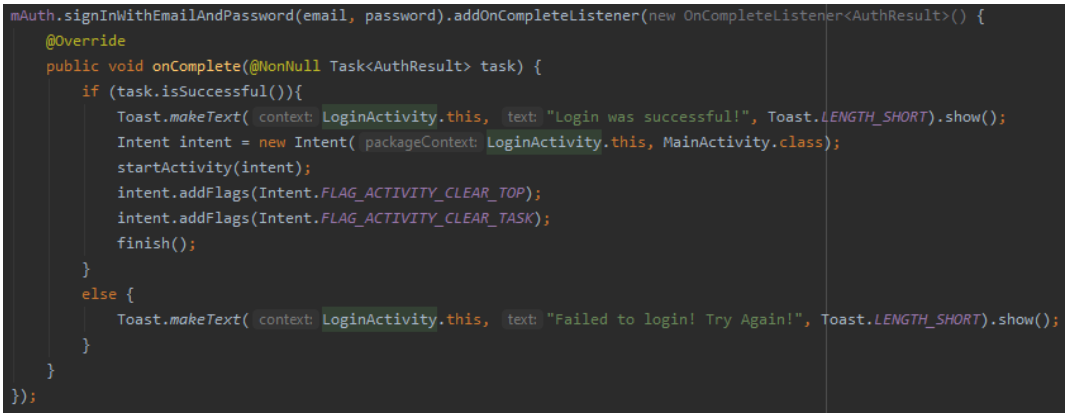
Registracija korisnika izvršava se već postojećom metodom *createUserWithEmailAndPassword()*, koja kao argumente prima adresu elektroničke pošte i zaporku korisnika. Ukoliko se dio kôda prikazanog na slici 3.8. uspješno izvršio, tada korisnik biva obaviješten putem *Toast* poruke kako je registracija uspješno izvršena, u

suprotnom dobiva poruku kako registracija nije uspješno izvršena te da pokuša ponovno. Pri uspješnoj registraciji korisnik se automatski prijavljuje u aplikaciju.

```
mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener( activity: RegisterActivity.this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if(task.isSuccessful()){
                    String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
                    User user = new User(username, email, uid);

                    FirebaseDatabase.getInstance().getReference( path: "Users")
                            .child(getOnlyEmailName(email))
                            .setValue(user).addOnCompleteListener( activity: RegisterActivity.this, new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if(task.isSuccessful()){
                                Toast.makeText( context: RegisterActivity.this,  text: "You have been registered successfully!", Toast.LENGTH_SHORT).show();
                                Intent intent = new Intent( packageContext: RegisterActivity.this, MainActivity.class);
                                startActivity(intent);
                                finish();
                            }
                            else {
                                Toast.makeText( context: RegisterActivity.this,  text: "Failed to register, try again!", Toast.LENGTH_SHORT).show();
                            }
                        }
                    });
                }
                else {
                    Toast.makeText( context: RegisterActivity.this,  text: "Something went wrong, try again!", Toast.LENGTH_SHORT).show();
                }
            }
        });
```

Slika 3.8. Prikaz dijela kôda koji se odnosi na registraciju korisnika

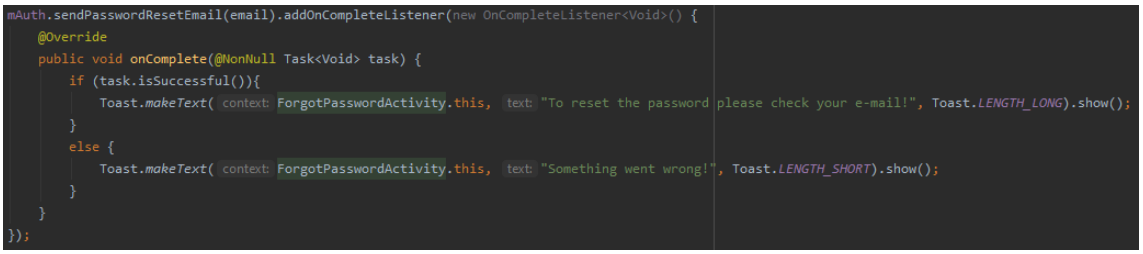
Prijava u aplikaciju izvršava se pritiskom na gumb *Login*, prije pritiska na gumb korisnik mora ispuniti dva polja, polje za adresu elektroničke pošte i polje za zaporku, to su podatci s kojima se korisnik registrirao. Podatci uneseni u polja moraju odgovarati podatcima spremljenim u bazu podataka. Prijava se izvršava već postojećom metodom *signInWithEmailAndPassword()*, koja isto kao i metoda za registraciju prima dva argumenta, adresu elektroničke pošte i zaporku korisnika. Pri uspješnom provođenju ove metode korisnik se prijavljuje u aplikaciju i otvara se aktivnost glavnog izbornika, u suprotnom korisnik dobiva *Toast* poruku iz koje može zaključiti kako je došlo do greške te da pokuša ponovno. Na slici 3.9. prikazan je kôd za prijavu korisnika.



Slika 3.9. Prikaz dijela kôda u kojemu se izvršava prijava korisnika u aplikaciju

3.2.4.1. Resetiranje zaporke

U aktivnosti *LoginActivity* nalazi se i tekst sadržaja *„Forgot password?“* koji se može pritisnuti. Pritiskom na taj tekst otvara se nova aktivnost *ForgotPasswordActivity* u kojoj korisnik može resetirati svoju zaporku. Sve što je potrebno je ispuniti polje koje zahtjeva adresu elektroničke pošte s kojom se korisnik registrirao. Protokoli Firebase-a brinu se o tome postoji li korisnik registriran s unesenom adresom elektroničke pošte. Ukoliko postoji, na tu adresu e-pošte šalje se URL link koji je potrebno otvoriti i unijeti novu zaporku. Korisnik dobiva obavijest da provjeri svoju e-poštu kako bi resetirao zaporku. Metoda koja se brine o ovom slučaju je već postojeća metoda imena *sendPasswordResetEmail()* koja kao argument prima adresu elektroničke pošte. Na slici 3.10. prikazana je metoda.



Slika 3.10. Prikaz metode *sendPasswordResetEmail()*

3.2.5. Glavni izbornik

MainActivitiy ili u XML layout nazivu *activity_main.xml* je glavni izbornik unutar kojeg korisnik može odabrati želi li igrati singleplayer offline tip igre, online tip igre, želi li

predložiti pitanje ili želi li se odjaviti iz aplikacije. Dizajn ove aktivnosti je jednostavan i korisniku jasno daje do znanja koji gumb se odnosi na koju radnju. Na dnu aplikacije nalazi se tekst u kojem se nalazi nadimak prijavljenog korisnika. Ukoliko se korisnik ne odjavljuje iz aplikacije, svaki put kada ju otvori automatski će se otvoriti aktivnost glavnog izbornika. Na slici 3.11. prikazan je dio kôda koji se odnosi na preuzimanje nadimka trenutno prijavljenog korisnika iz baze podataka i spremanje istog u TextView element predviđen za to.

```
reference.child(getOnlyEmailName(user.getEmail())).addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        User userProfile = snapshot.getValue(User.class);
        if (userProfile != null) {
            String username = userProfile.getUsername();
            tvLoggedIn.setText("You're logged in as: " + username);
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        Toast.makeText( context: MainActivity.this,  text: "Something went wrong!", Toast.LENGTH_SHORT).show();
    }
});
```

Slika 3.11. Prikaz dijela kôda iz aktivnosti *MainActivity*

3.2.6. Implementacija izbora kategorije

Aktivnost *CategoryActivity* se otvara nakon što je korisnik pritisnuo gumb *Play Singleplayer*. Na zaslonu korisnik može uočiti četiri slike, odnosno četiri kategorije između kojih može birati, a to su:

- Geografija
- Povijest
- Sport
- Filmovi i tv serije

Pritiskom na jednu od kategorija, pomoću objekta klase *Intent* otvara se aktivnost offline samostalnog tipa igre te uz taj objekt šalju se podatci u iduću pokrenutu aktivnost pomoću funkcije *putExtra()*. U iduću aktivnost šalju se dva podatka, referenca za put u bazi podataka i naziv kategorije koja je izabrana. Ta dva podatka programu su od velike važnosti kako bi se mogao sastaviti kviz iz odabrane kategorije. U trenutku kada korisnik pritisne na jednu od

kategorija aplikacija automatski kreće s igrom. Na slici 3.12. prikazan je primjer kôda s objektom klase *Intent* i funkcijama *putExtra()*.

```
ivGeography.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent( packageContext: CategoryActivity.this, QuestionsActivity.class);
        intent.putExtra( name: "categoryReference", value: "/GeographyQuestions/");
        intent.putExtra( name: "category", value: "Geography");
        startActivity(intent);
        finish();
    }
});
```

Slika 3.12. Dio kôda koji se izvršava pri odabiru jedne od kategorija

3.2.7. Samostalni tip igre

Ovo je jedna od glavnih funkcionalnosti ove aplikacije, mogućnost igranja offline samostalnog tipa igre. Ova aktivnost nosi ime *SingeplayerActivity*. Nakon što je korisnik odabrao kategoriju koju želi zaigrati, u glavnom *onCreate()* dijelu programa dohvaća se o kojoj se kategoriji radi i dobiva se naziv reference te kategorije u bazi podataka, pomoću metode *getReferencePath()*. Na slici 3.13. može se vidjeti ta metoda, ona vraća *String* objekt koji se dobije iz prošle aktivnosti pomoću funkcije *getIntent().getExtras()*.

```
private String getReferencePath() {
    Bundle extras = getIntent().getExtras();
    String data = new String();
    if (extras != null) {
        data = extras.getString( key: "categoryReference");
    }
    return data;
}
```

Slika 3.13. Metoda *getReferencePath()*

Nakon toga kreira se lista nasumičnih brojeva pomoću metode *generateRandomNumbers()*. U listu se sprema deset nasumičnih brojeva jer je kviz sastavljen od deset pitanja, maksimalan broj koji se može dobiti je onaj koliko ima pitanja u bazi podataka. Pitanja u bazi podataka poredana su brojevima, zato se koristi ova metoda kako

pitanja ne bi išla uvijek istim redom, već kako bi bila nasumična. S povećanjem broja pitanja u bazi podataka smanjuje se vjerojatnost pojavljivanja istih pitanja. Struktura pitanja u bazi podataka može se vidjetu u poglavlju *Struktura baze podataka*. Na slici 3.14. prikazana je metoda *generateRandomNumbers()*.

```
private ArrayList<Integer> generateRandomNumbers() {
    ArrayList<Integer> temp = new ArrayList<~>();
    for (int i = 1; i < 21; i++) {
        temp.add(i);
    }
    Collections.shuffle(temp);
    ArrayList<Integer> list = new ArrayList<>();
    for (int i = 0; i<10; i++){
        list.add(temp.get(i));
    }
    return list;
}
```

Slika 3.14. Metoda *generateRandomNumbers()*

Nakon što je program generirao nasumične brojeve poziva se metoda *playQuiz()* koja kao argument prima listu tih brojeva. Kako bi se dohvatila pitanja iz baze podataka mora se instancirati referenca na bazu podataka. Nakon toga sav ostali dio kôda nalazi se u metodi *addValueEventListener()* iz Firebase-ove knjižnice, koju poziva kreirana referenca baze podataka. Podatci iz baze podataka dohvaćaju se pomoću objekta klase *DataSnapshot*. Nakon što program dohvati podatke potrebne za sastavljanje pitanja, četiri moguća odgovora se spremaju u polje i miješaju se pomoću metode *shuffleAnswers()* koja kao argument prima ta četiri odgovora. To se izvodi iz tog razloga da odgovori ne budu uvijek na istim mjestima, već da budu na nasumičnim mjestima, odnosno gumbovima. Nakon izvođenja te metode izvršava se postavljanje pitanja, postavljanje odgovora i postavljanje slike koja je vezana uz pitanje.

Slika se prikazuje pomoću *Glide-a*. Glide je open-source framework za brzo učitavanje slika pomoću kojeg se u jednoj liniji kôda može prikazati slika u aplikaciji. Potrebno je imati URL slike kako bi slika bila prikazana. Svaki put kada se učita novo pitanje resetiraju se pozadine gumbova, tekst koji prikazuje na kojem se korisnik pitanju nalazi se poveća i pokrene se štoperica u trajnosti od 20 sekundi s naredbom *startTimer()*. Opisani procesi prikazani su na slici 3.15.

```
private void playQuiz(ArrayList<Integer> list) {
    int index = list.get(questionNumber);
    reference = database.getReference( path: referencePath + index + "/");
    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            String question = snapshot.child("Question").getValue(String.class);
            String correct_answer = snapshot.child("CorrectAnswer").getValue(String.class);
            String optionA = snapshot.child("OptionA").getValue(String.class);
            String optionB = snapshot.child("OptionB").getValue(String.class);
            String optionC = snapshot.child("OptionC").getValue(String.class);
            String optionD = snapshot.child("OptionD").getValue(String.class);
            String url = snapshot.child("QuestionImage").getValue(String.class);
            ArrayList<String> all_options = shuffleAnswers(optionA, optionB, optionC, optionD);
            tvQuestion.setText(question);
            btnFirstAnswer.setText(all_options.get(0));
            btnSecondAnswer.setText(all_options.get(1));
            btnThirdAnswer.setText(all_options.get(2));
            btnFourthAnswer.setText(all_options.get(3));
            Glide.with( activity: SingleplayerActivity.this).load(url).into(ivQuestionImage);

            tvQuestionCount.setText("Question: " + (questionNumber + 1) + "/10");
            resetButtonBackground(btnFirstAnswer);
            resetButtonBackground(btnSecondAnswer);
            resetButtonBackground(btnThirdAnswer);
            resetButtonBackground(btnFourthAnswer);

            startTimer();
```

Slika 3.15. Prikaz dijela kôda metode *playQuiz()*

Pritiskom na jedan od gumbova, odnosno odgovora, zaustavlja se štoperica. Zatim se provjerava sadrži li pritisnuti gumb točan odgovor. Ukoliko sadrži, gumb tada poprimi zelenu boju koja asocira na to da je odgovor točan, korisniku se također ispisuje *Toast* poruka da je točno odgovorio i povećava se int varijabla koja se odnosi na broj točnih odgovora koji se kasnije ispisuju u završnoj aktivnosti. Ukoliko gumb na koji je korisnik pritisnuo ne sadrži točan odgovor, taj gumb poprima crvenu boju koja asocira na to da je korisnik netočno odgovorio. Zatim se provjeravaju ostali gumbovi kako bi se ustanovilo koji od njih sadrži točan odgovor, nakon što se ustanovi koji gumb sadrži točan odgovor tada taj gumb poprima zelenu boju, a korisniku se ispisuje u obliku *Toast* poruke kako je netočno odgovorio i koji je odgovor zapravo točan. Ukoliko korisnik nije pritisnuo niti jedan gumb, a vrijeme je isteklo, tada kviz ide dalje, a korisniku se ispisuje *Toast* poruka kako nije na vrijeme pružio odgovor. Taj proces definiran je u metodi *onFinish()* metode *startTimer()*. Nakon provjere, pomoću *Handler-a* u trajanju od dvije sekunde se provjerava da li je kviz došao do kraja ili nije.

Ukoliko nije došao do kraja tada se varijabla koja se odnosi na broj pitanja povećava za jedan i poziva se metoda *playQuiz()*. Ako je kviz došao do kraja tada pomoću objekta klase *Intent* otvara se završna aktivnost te se u tu aktivnost šalju podatci na koliko je korisnik pitanja točno odgovorio i u kojoj se kategoriji kviz održao. Opisani procesi mogu se vidjeti na slici 3.16.

```
btnFirstAnswer.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        countDownTimer.cancel();
        if (btnFirstAnswer.getText().equals(correct_answer)) {
            btnFirstAnswer.setBackgroundResource(R.drawable.custombutton_answers_correct);
            Toast.makeText( context: SingleplayerActivity.this, text: "Correct!", Toast.LENGTH_SHORT).show();
            correctAnswerCount++;
        } else {
            btnFirstAnswer.setBackgroundResource(R.drawable.custombutton_answers_wrong);
            Toast.makeText( context: SingleplayerActivity.this, text: "Wrong! Correct answer is: " + correct_answer, Toast.LENGTH_SHORT).show();
            if (btnSecondAnswer.getText().equals(correct_answer)) btnSecondAnswer.setBackgroundResource(R.drawable.custombutton_answers_correct);
            if (btnThirdAnswer.getText().equals(correct_answer)) btnThirdAnswer.setBackgroundResource(R.drawable.custombutton_answers_correct);
            if (btnFourthAnswer.getText().equals(correct_answer)) btnFourthAnswer.setBackgroundResource(R.drawable.custombutton_answers_correct);
        }
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                if (questionNumber < 9) {
                    questionNumber++;
                    playQuiz(list);
                } else {
                    Intent intent = new Intent(getApplicationContext(), EndActivity.class);
                    intent.putExtra( name: "correctAnswers", correctAnswerCount);
                    intent.putExtra( name: "categoryName", categoryName);
                    startActivity(intent);
                    finish();
                }
            }
        }, delayMillis: 2000);
    }
});
```

Slika 3.16. Prikaz dijela kôda za jedan *Button* element metode *playQuiz()*

Unutar ove aktivnosti definirane su i metode *onPause()* i *onDestroy()*. U *onPause()* metodi stvara se novi *Intent* koji otvara aktivnost glavnog izbornika i ovu aktivnost zatvara. To je definirano iz razloga da kada korisnik izađe iz aplikacije da nije u mogućnosti vratiti se u već započeti kviz. U *onDestroy()* metodi zaustavlja se štoperica.

3.2.8. Implementacija predlaganja pitanja

Svrha aktivnosti *SuggestActivity* je uključivanje korisnika u proširivanje kviza na način da mu se omogući predlaganje novih pitanja. Na zaslonu aktivnosti korisnik može uočiti kako pravilno treba predložiti pitanje. Korisnik u polje unosi pitanje i točan odgovor. Tekst unesen u polje ne smije sadržavati manje od 30 ili više od 200 znakova. Nakon što je ispunio polje

tada se pritiskom na gumb *Suggest A Question* pokreće metoda *sendEmail()* koja pomoću objekta klase *Intent* šalje e-poštu. Naslov e-pošte je *„Question“*, a tekst koji je korisnik unio sadržan je u tekstu e-pošte. Korisnik može izabrati pomoću kojeg klijenta elektroničke pošte želi poslati e-poštu. Opisani proces može se vidjeti na slici 3.17.

```
void sendEmail(){
    String question = etWriteAQuestion.getText().toString();
    if (question.length() < 30){
        Context context = getApplicationContext();
        Toast.makeText(context, text: "Your question is too small.", Toast.LENGTH_SHORT).show();
        return;
    }
    String[] emailTo = {"ivankovicpetar2@gmail.com"};
    String subject = "Question";
    Intent intent = new Intent(Intent.ACTION_SENDTO, Uri.parse("mailto:"));
    intent.putExtra(Intent.EXTRA_EMAIL, emailTo);
    intent.putExtra(Intent.EXTRA_SUBJECT, subject);
    intent.putExtra(Intent.EXTRA_TEXT, question);
    startActivity(Intent.createChooser(intent, title: "Choose an e-mail client"));
}
```

Slika 3.17. Prikaz metode *sendEmail()*

3.2.9. Online multiplayer tip igre

Izrada ove funkcionalnosti najteži je i najkompleksniji dio ovog rada. Kako bi se omogućilo korisnicima da igraju jedan protiv drugoga bilo je potrebno smisliti na koji to način ostvariti. To je omogućeno putem Firebase-ove baze podataka u stvarnom vremenu koja savršeno dobro služi za igre bazirane na potezima. Ova funkcionalnost sadrži se od šest aktivnosti. Nakon što je korisnik pritisnuo gumb *Play Online* aplikacija ga usmjerava u drugu aktivnost gdje korisnik ili čeka da ga netko pozove ili kako bi pozvao nekoga mora u za to predviđeno polje unijeti adresu elektroničke pošte korisnika kojeg želi pozvati. U slučaju da korisnik očekuje poziv za igru od nekoga tada se pokreće metoda *incomingCalls()*. Ta metoda osluškuje postoji li u bazi podataka poziv od nekog korisnika. Ukoliko postoji tada se čvor *„request“* postavlja u *true* stanje i pomoću *Intent* objekta otvara se aktivnost *AcceptInviteActivity* gdje korisnik treba pritisnuti gumb kako bi prihvatio poziv. U slučaju da korisnik poziva nekoga u igru tada se izvodi metoda *sendRequest()* koja je prethodno okinuta pritiskom na gumb *Send Request*. U čvoru korisnika kojeg se poziva kreira se čvor imena *„request“* u koji se postavlja čvor jedinstvene vrijednosti pomoću metode *push()* i adresa elektroničke pošte korisnika koji poziva. Program obrača pozornost na to postoji li uopće

korisnik registriran pod pozvanom adresom elektroničke pošte, a u slučaju poziva samog sebe javlja se greška. Ukoliko je pozivnica uspješno poslana, tada se korisniku koji je pozvao drugog korisnika otvara aktivnost imena *InviterGameAwaitingActivity* gdje on čeka da pozvani korisnik prihvati pozivnicu. Metode *sendRequest()* i *incomingCalls()* mogu se vidjeti u slijedećim slikama.

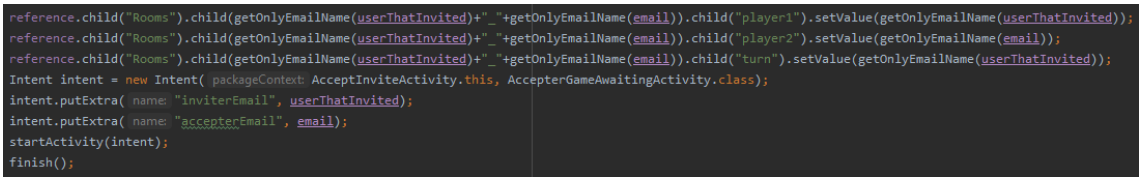
```
if (!email.equals(currentUserEmail)) {
    reference.child("Users").addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            if (snapshot.hasChild(getOnlyEmailName(email))) {
                reference.child("Users").child(getOnlyEmailName(email)).child("request").push().setValue(currentUserEmail).addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if (task.isSuccessful()) {
                            Toast.makeText( context: InviteActivity.this,  text: "Invite successfully sent!", Toast.LENGTH_SHORT).show();
                            //reference.child("Users").child(getOnlyEmailName(email)).child("request").setValue(true);
                            Intent intent = new Intent( packageContext: InviteActivity.this, InviterGameAwaitingActivity.class);
                            intent.putExtra( name: "accepterEmail", email);
                            intent.putExtra( name: "inviterEmail", currentUserEmail);
                            startActivity(intent);
                        } else {
                            Toast.makeText( context: InviteActivity.this,  text: "Error, invite hasn't been sent!", Toast.LENGTH_SHORT).show();
                        }
                    }
                });
            } else {
                etPlayerInvite.requestFocus();
                etPlayerInvite.setError("No user registered under this e-mail!");
            }
```

Slika 3.18. Prikaz dijela metode *sendRequest()*

```
private void incomingCalls(){
    reference.child("Users").child(getOnlyEmailName(mAuth.getCurrentUser().getEmail()))
            .child("request").addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            for (DataSnapshot child : snapshot.getChildren()){
                String value = (String) child.getValue();
                reference.child("Users").child(getOnlyEmailName(mAuth.getCurrentUser().getEmail())).child("request").setValue(true);
                Intent intent = new Intent( packageContext: InviteActivity.this, AcceptInviteActivity.class);
                intent.putExtra( name: "firstUser", value);
                intent.putExtra( name: "firstUserEmail", etPlayerInvite.getText().toString());
                startActivity(intent);
                finish();
                break;
            }
```

Slika 3.19. Prikaz dijela metode *incomingCalls()*

Ukoliko pozvani korisnik prihvati pozivnicu pritiskom na gumb *Accept Invite*, unutar aktivnosti *AcceptInviteActivity*, tada se unutar čvora *„Rooms“* kreira čvor pod kombiniranim dijelovima adresa e-pošte oba korisnika te se u taj čvor spremaju vrijednosti nadimaka igrača i u čvor *„turn“* postavlja se nadimak korisnika koji je pozvao i označava to da taj korisnik prvi kreće sa igrom. Nakon toga otvara se aktivnost *AccepterGameAwaitingActivity* gdje korisnik čeka da igra započne.



Slika 3.20. Dio kôda iz aktivnosti *AcceptInviteActivity*

Unutar aktivnosti *InviterGameAwaitingActivity* pomoću metode *checkIfRoomExists()* provjerava se postoji li soba odnosno da li je drugi korisnik prihvatio pozivnicu. Metoda se rekurzivno poziva svakih 100 milisekundi kako drugi korisnik ne bi dugo čekao, to se odvija pomoću *Handler* objekta. Ukoliko je drugi korisnik prihvatio pozivnicu tada korisnik dobiva *Toast* poruku koja mu poručuje kako je soba pronađena i tada se pokreće aktivnost *PreGameActivity*. Metoda *checkIfRoomExists()* pokreće se i u aktivnosti drugog korisnika koji je pozvan. Ova metoda može se vidjeti na slici 3.21.

Slika 3.21. Prikaz metode *checkIfRoomExists()*

Nakon što su oba korisnika prebačena u aktivnost *PreGameActivity* pomoću objekta klase *Intent* provjerava se koji je korisnik pozvan, a koji je pozivao.

Podatke poslane preko *Intent-a* koristi se kako bi se znalo koji korisnik je koji, kako bi multiplayer tip igre uopće bio moguć. Dok jedan korisnik odgovara drugi čeka na red. U

programu se to provjerava dobivajući podatak *„turn“* iz baze podataka. Estetski aktivnost multiplayer tipa igre ista je aktivnosti samostalnog tipa igre. Sva provjera za točne odgovore gotovo je identična provjerama iz samostalnog tipa igre. U multiplayer tipu igre korisnici ne dobivaju 10 već 5 pitanja kako igra ne bi predugo trajala. Kategorija u kojoj će korisnici igrati bira se nasumično sa metodom *getRandomCategoryReference()*. Ova metoda može se vidjeti na slici 3.22.

```
private String getRandomCategoryReference(){
    ArrayList<String> categories = new ArrayList<>();
    categories.add("GeographyQuestions");
    categories.add("HistoryQuestions");
    categories.add("SportQuestions");
    categories.add("MoviesQuestions");
    Collections.shuffle(categories);
    Random random = new Random();
    int randomNumber = random.nextInt( bound: 4);
    return categories.get(randomNumber);
}
```

Slika 3.22. Prikaz metode *getRandomCategoryReference()*

Na kraju igre ili u slučaju izlaženja iz igre u bazi podataka briše se stvorena soba kako pri idućem pozivu ne bi došlo do greške.

3.2.10. Aktivnost *EndActivity*

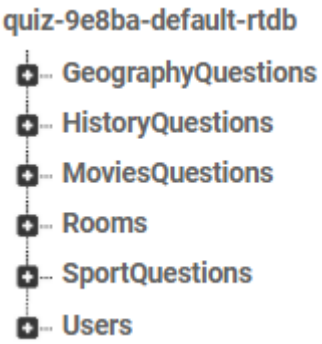
Ova aktivnost se pokreće kada se završi samostalni tip igre kviza. Pomoću klase *Intent* u ovu aktivnost šalju se dva podatka: na koliko pitanja je korisnik točno odgovorio i u kojoj kategoriji je igrao. Ti podatci se ubacuju u dva TextView XML elementa, što je prikazano na slici 3.23. U ovoj aktivnosti postoje i dva gumba, jedan korisnika vodi u glavni izbornik, a drugi ga vodi u ponovni izbor kategorije. Dizajn tih gumbova može se vidjeti u potpoglavlju 4.10.

```
int correctAnswerCount = getIntent().getIntExtra( name: "correctAnswers",  defaultValue: 0);

String category = getIntent().getStringExtra( name: "categoryName");

tvQuizResults.setText("Correct Answers: " + correctAnswerCount + "/10");

tvQuizFinished.setText("You have finished the quiz in the category of: " + category);
```

Slika 3.23. Dio kôda iz aktivnosti *EndActivity*

3.2.11. Struktura baze podataka

Baza podataka strukturirana je od šest čvorova. Četiri čvora odnose se na kategorije kviza, peti se odnosi na korisnike, imena *„Users“*, a šesti imena *„Rooms“* odnosi se na multiplayer sobe. Glavna struktura prikazana je na slici 3.24.



Slika 3.24. Prikaz glavne strukture baze podataka

Unutar svake od kategorija nalaze se čvorovi koji su smješteni po rednom broju. Čvor označava pitanje. Tako svaka kategorija sadržava određen broj pitanja. Nadalje, kada se otvori čvor nekog pitanja tada se mogu vidjeti svi elementi pitanja, a to su: pitanje, četiri opcije, točan odgovor i URL slike. Primjer je prikazan na slici 3.25.

Slika 3.25. Prikaz strukture jednog pitanja u bazi podataka

Baza podataka se ažurira uvozom datoteke formata JSON (engl. *JavaScript Object Notation*). JSON je tekstualni format dizajniran za razumljiv prijenos strukture podataka. Administrator ažurira bazu podataka na način da skine trenutnu JSON datoteku, ažurira je i zatim ju učita nazad na platformi baze podataka.

4. STRUKTURA APLIKACIJE

Kroz ovo poglavlje može se vidjeti dizajn na samom uređaju, odnosno strukturu aplikacije svake aktivnosti. Aplikacija se sastoji od 16 aktivnosti, a to su: početna aktivnost, aktivnost registracije korisnika, aktivnost prijave korisnika, aktivnost zaboravljene zaporke, aktivnost glavnog izbornika, aktivnost izbora kategorije, aktivnost offline samostalnog tipa igre, aktivnost prijedloga pitanja, aktivnosti online tipa igre kojih ima sedam i završna aktivnost. Elementi XML-a koji se nalaze u aktivnostima su: Button, ImageView, TextView, EditText i ProgressBar.

4.1. Početna aktivnost

Ova aktivnost sastoji se od slike loga aplikacije, teksta koji poručuje korisniku dobrodošlicu u aplikaciju i tri gumba *Login, Register* i *Exit*. Prilikom pritiska na gumb *Login* otvara se nova aktivnost gdje korisnik treba unijeti svoje podatke kako bi se uspješno prijavio. Pri pritisku gumba *Register* otvara se nova aktivnost koja od korisnika traži da unese svoje podatke kako bi se reigstrirao. Pritiskom na gumb *Exit* aplikacija se zatvara.

Slika 4.1. Prikaz početne aktivnosti

4.2. Aktivnost registracije korisnika

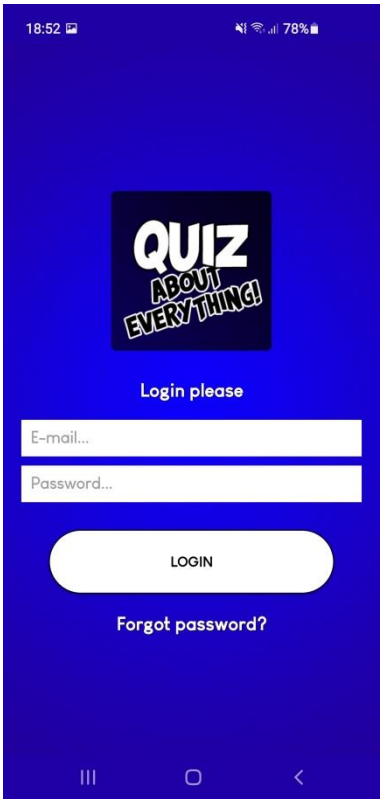
Korisnik u ovoj aktivnosti mora unijeti svoj nadimak (engl. *Username*), svoju adresu elektroničke pošte i željenu zaporku s kojom će se prijavljivati u aplikaciju. Na ekranu se nalazi slika loga aplikacije i tekst koji korisnika asocira da se radi o registraciji. Nakon što je korisnik ispunio sva polja potrebna za registraciju, a sva polja moraju biti popunjena, tada pritiskom na gumb *Register* završava registraciju. Korisnik dobiva obavijest u obliku *Toast* poruke da li je registracija bila uspješna ili nije. Prilikom uspješne registracije podatci korisnika se spremaju u bazu podataka. Nakon uspješne registracije, korisnik se automatski prijavljuje u aplikaciju i otvara se aktivnost glavnog izbornika.

Slika 4.2. Prikaz aktivnosti registracije korisnika

4.3. Aktivnost prijave korisnika

Ova aktivnost sastoji se od dva EditText XML elementa u koje korisnik treba unijeti adresu elektroničke pošte i zaporku s kojima se registrirao, pomoću kojih se prijavljuje u sustav. Prilikom pritiska na gumb *Login*, ukoliko je korisnik uspješno unio svoje podatke s

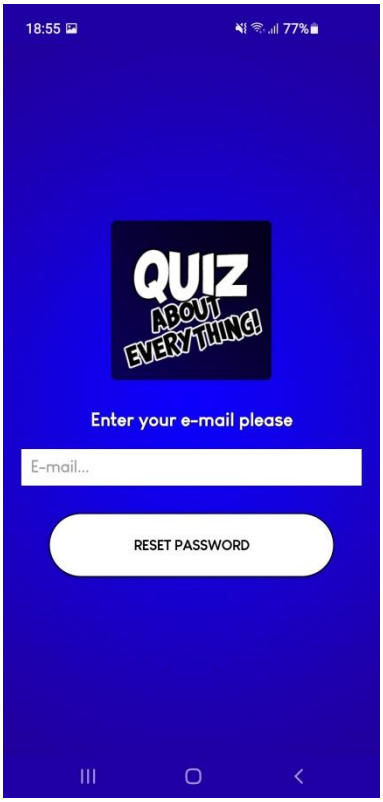
kojima se regstrirao tada dobiva obavijest u obliku *Toast* poruke kako je prijava uspješno izvršena i tada se ova aktivnost zatvara i otvara se aktivnost glavnog izbornika, u kojoj pri dnu korisnik može vidjeti pod kojim nadimkom je prijavljen. Ukoliko prijava nije uspješno izvršena, tada korisnik dobiva *Toast* poruku kako je došlo do greške te da pokuša ponovno.



Slika 4.3. Prikaz aktivnosti prijave korisnika u aplikaciju

4.4. Aktivnost zaboravljene zaporke

Aktivnost zaboravljene zaporke strukturirana je od jednog ImageView XML elementa koji sadrži logo aplikacije, od jednog TextView XML elementa koji korisnika asocira na to da treba unijeti svoju adresu elektroničke pošte u polje EditText XML elementa i od gumba *Reset Password*. Prilikom pritiska gumba korisniku na e-poštu stiže link na kojem može resetirati zaporku.



Slika 4.4. Prikaz aktivnosti zaboravljene zaporke

4.5. Aktivnost glavnog izbornika

Glavni izbornik aplikacije sastoji se od slike loga aplikacije, četiri gumba i teksta na dnu aktivnosti. Prilikom pritiska na gumb *Play Singleplayer* korisnik se prebacuje u aktivnost izbora kategorije gdje bira jednu od kategorija u kojoj želi igrati. Pritiskom na gumb *Play Online* otvara se aktivnost gdje korisnik mora pozvati nekog drugog korisnika u igru. Gumb *Suggest A Question* korisnika prebacuje u aktivnost prijedloga pitanja. Gumb *Logout* korisnika odjavljuje iz aplikacije te otvara početnu aktivnost, prilikom uspješnog odjavljivanja korisnik dobiva *Toast* poruku kako se uspješno odjavio iz aplikacije. Na dnu aplikacije nalazi se tekst koji označava pod kojim nadimkom je korisnik prijavljen.

Slika 4.5. Prikaz aktivnosti glavnog izbornika

4.6. Aktivnost izbora kategorije

Ova aktivnost sadrži četiri kategorije koje su predstavljene slikama i tekstom unutar slike. Kategorije između kojih korisnik može birati su: geografija (engl. *Geography*), povijest (engl. *History*), sport (engl. *Sports*) te filmovi i tv serije (engl. *Movies and TV shows*). Pritiskom na sliku izabire se kategorija. Izborom određene kategorije sastavlja se kviz od pitanja samo iz birane kategorije.

Slika 4.6. Prikaz aktivnosti izbora kategorije

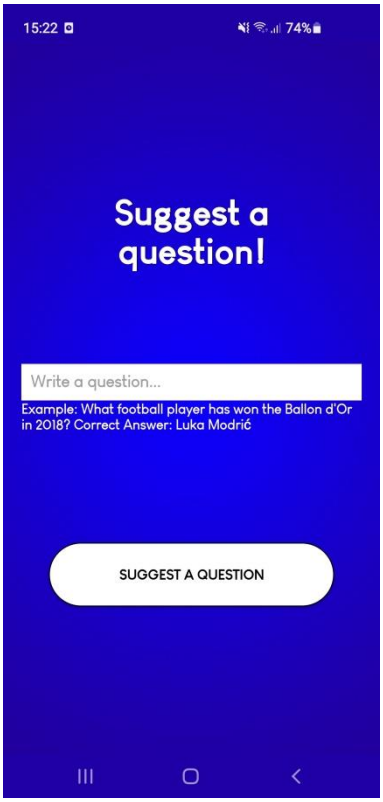
4.7. Aktivnost offline samostalnog tipa igre

Ova aktivnost se sastoji od slike koja korisnika asocira na postavljeno pitanje, TextView XML elementa koji sadrži trenutno postavljeno pitanje, TextView elementa u kojem se nalazi brojni red pitanja koje je trenutno postavljeno i još jednog TextView elementa koji prikazuje koliko korisnik ima vremena da odgovori na pitanje. Pri postavljanju svakog novog pitanja korisnik ima 20 sekundi da odgovori na pitanje, ukoliko ne odgovori na pitanje unutar 20 sekundi tada korisnik dobiva *Toast* poruku da nije pružio odgovor i zatim program učitava iduće pitanje. Pitanja su nasumična, kao i odgovori koji su uvijek izmiješani. Također postoje i četiri gumba koji označavaju četiri moguća odgovora na dano pitanje. Prilikom pritiska na neki od gumbova, ovisno o točnosti odgovora koji se nalazi u pritisnutom gumbu on će poprimiti zelenu boju radi li se o točnom odgovoru ili crvenu boju radi li se o krivom odgovoru, a gumb koji sadrži točan odgovor poprimiti će zelenu boju. Korisnik će dobiti i *Toast* poruku koja će mu dati do znanja da li je odgovor na čiji je gumb pritisnuo točan ili nije, ukoliko nije ispisat će koji je odgovor točan. Nakon dvije sekunde aplikacija učitava novo pitanje i nove odgovore.

Slika 4.7. Prikaz aktivnosti offline samostalnog tipa igre

4.8. Aktivnost prijedloga pitanja

Ova aktivnost sastoji se od jednog EditText XML elementa u koji se unosi pitanje. Pitanja su ograničena po broju simbola, najmanji broj simbola je 30, a najveći 200, ukoliko se unese više ili manje korisnik dobiva *Toast* poruku da je izvan okvira broja simbola. Ispod toga nalazi se tekst koji sadrži primjer kako pitanje treba biti pravilno postavljeno. Na kraju se nalazi gumb naziva *Suggest A Question*, prilikom čijeg pritiska se pitanje šalje administratoru na adresu e-pošte. Prije samog slanja e-pošte, korisniku aplikacija nudi na izbor pomoću kojeg klijenta da pošalje e-poštu.



Slika 4.8. Prikaz aktivnosti prijedloga pitanja

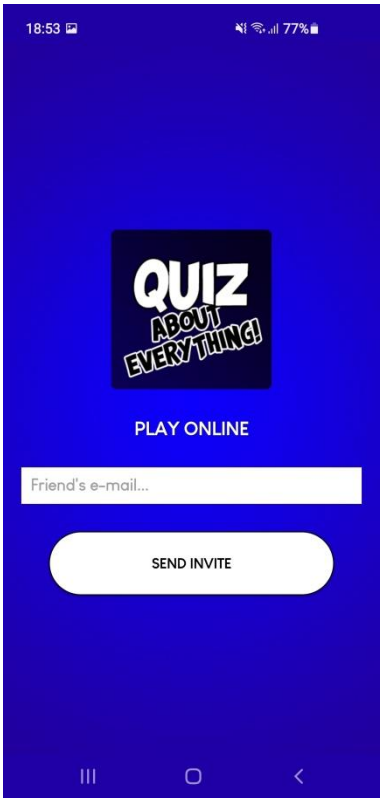
4.9. Aktivnosti online tipa igre

Aktivnosti online tipa igre uključuju dva korisnika koji se natječu jedan protiv drugoga. Postoji sedam aktivnosti:

- Aktivnost slanja pozivnice
- Aktivnost čekanja pozivatelja
- Aktivnost čekanja pozvanog igrača
- Aktivnost prihvaćanja pozivnice
- Aktivnost prije pokretanja igre
- Multiplayer aktivnost
- Multiplayer završna aktivnost

U aktivnosti slanja pozivnice kako bi korisnik izazvao svog prijatelja mora mu poslati pozivnicu. Pozivnica se šalje na način da se u EditText element prikazan na slici 4.9. unese adresa elektroničke pošte korisnika kojeg se želi pozvati na dvoboj. Zatim prilikom pritiska na gumb *Send Invite* šalje se pozivnica korisniku unesene adrese elektroničke pošte. Nakon

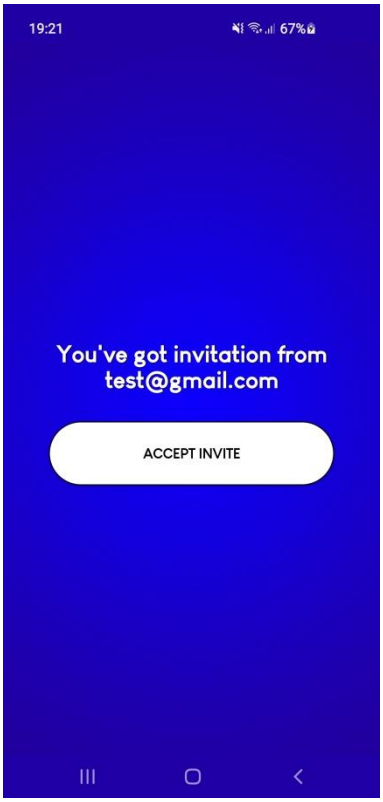
što igrač pošalje pozivnicu on čeka drugog igrača u aktivnosti *„aktivnost čekanja pozivatelja“*. Kako ta aktivnost izgleda može se vidjeti na slici 4.10.



Slika 4.9. Prikaz aktivnosti slanja pozivnice – slanje pozivnice drugom korisniku

Slika 4.10. Prikaz aktivnosti čekanja pozivatelja

Prilikom poziva korisnik kojeg se poziva dobiva obavijest koju može prihvatiti. Kako bi dvoboj započeo korisnik kojeg se poziva mora prihvatiti pozivnicu. To se odvija u aktivnosti *„aktivnost prihvaćanja pozivnice“*, kako ta aktivnost izgleda može se vidjeti na slici 4.11.

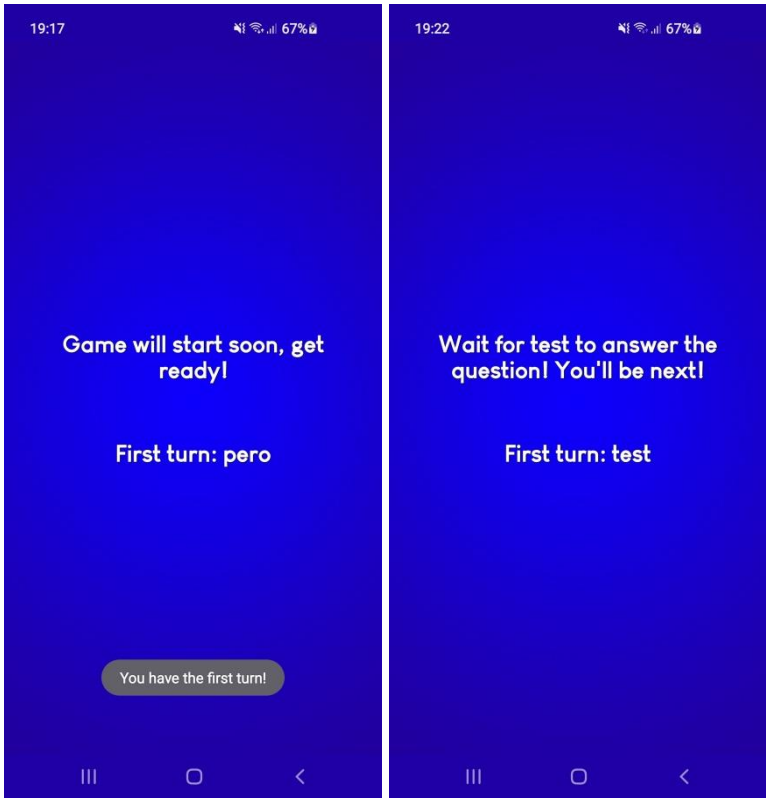


Slika 4.11. Prikaz aktivnosti prihvaćanja pozivnice

Nakon što korisnik prihvati pozivnicu tada se otvara aktivnost „aktivnost čekanja pozvanog igrača“.

Slika 4.12. Prikaz aktivnosti čekanja pozvanog igrača

Prije no što započne igra *„aktivnost prije pokretanja igre“* se pokreće kod oba korisnika.



Slika 4.13. i Slika 4.14. Prikaz aktivnosti *„aktivnost prije pokretanja igre“* u oba slučaja

Kao što je i prikazano većina aktivnosti sadrži pokoji TextView XML element, ProgressBar XML element i eventualno Button XML element.

Multiplayer aktivnost identična je aktivnosti samostalnog tipa igre. U multiplayer aktivnosti prije postavljanja novog pitanja nakon kratkog vremena korisnici dobivaju obavijest da li je njihov protivnik odgovorio točno ili netočno. Dodatnu tenziju igri daje vremensko ograničenje od 20 sekundi na svako pitanje.

Pri završetku igre korisnici saznaju koji je pobjedio, to je vidljivo u završnoj multiplayer aktivnosti.

4.10. Završna aktivnost

Završna aktivnost sastoji se od slike loga aplikacije, teksta koji korisniku na ekran ispisuje kako je uspješno priveo kviz kraju i iz koje su kategorije bila pitanja. Ispod toga se nalazi drugi tekst koji prikazuje na koliko je pitanja korisnik točno odgovorio. Također

postoje i dva gumba. Prvi naziva *Exit To Main Menu* korisnika prebacuje na aktivnost glavnog izbornika, a drugi naziva *Play Again* korisnika prebacuje u aktivnost izbora kategorije gdje može ponovno zaigrati.

Slika 4.15. Prikaz završne aktivnosti

5. ZAKLJUČAK

U sadašnjosti kvizovi kao oblik razonode sve više jačaju. Upravo iz te popularizacije kvizova rađa se ideja za kreacijom jedne takve aplikacije, koja bi uzela samo ono najbolje i svojom jednostavnošću i kvalitetom zasigurno zaintrigirala mnoge kvizoljupce da ju iskušaju. Aplikacija korisnicima služi za zabavu, a nepobitno je da korisnici kroz aplikaciju uče nove stvari. Primjerena je za sve uzraste, pa se može koristiti u mnoštvu situacija.

Cilj ovog završnog rada bio je realizacija kviz aplikacije u Android operacijskom sustavu. Aplikacija je u potpunosti razvijena u Android Studio okruženju. Realizirana su dva tipa igre, offline samostalni i online protiv drugih korisnika. Aplikacija je napisana u Java programskom jeziku, a dizajnirana je pomoću XML opisnog jezika. Korištena baza podataka u stvarnom vremenu nalazi se na Firebase platformi. Tokom rada na aplikaciji bilo je susreta sa mnogim poteškoćama i preprekama koje su izazvane nedovoljnim poznavanjem koncepta potrebnih za realizaciju ovog projekta.

Daljnji razvoj aplikacije moguć je ugradnjom dodatnih funkcionalnosti, mogućnosti su razne, kao što su na primjer: bodovna ljestvica korisnika, mogućnost stavljanja slika profila korisnika, mogućnost dopisivanja između korisnika, dodavanje još kategorija i pitanja, prijava u aplikaciju preko Facebook-a ili Google-a.

LITERATURA

[1] Trivia Crack, Trgovina Google Play, dostupno na: https://play.google.com/store/apps/details?id=com.etermax.preguntados.lite&hl=hr&gl=US (Datum zadnjeg pregleda: 22.6.2021.)

[2] QuizzLand, Trgovina Google Play, dostupno na: https://play.google.com/store/apps/details?id=com.xmonetize.quizzland&hl=hr&gl=US (Datum zadnjeg pregleda: 22.6.2021.)

[3] Trivia 360, Trgovina Google Play, dostupno na: https://play.google.com/store/apps/details?id=smartowlapps.com.quiz360&hl=hr&gl=US (Datum zadnjeg pregleda: 23.6.2021.)

[4] General Knowledge Quiz, Trgovina Google Play, dostupno na: https://play.google.com/store/apps/details?id=com.creactive.quemquersermilionario.millonario.lite&hl=hr&gl=US (Datum zadnjeg pregleda: 23.6.2021.)

[5] Quizefy, Trgovina Google Play, dostupno na: https://play.google.com/store/apps/details?id=com.quizefy.game&hl=hr&gl=US (Datum zadnjeg pregleda: 23.6.2021.)

[6] X. Ducrohet, T. Norbye, K. Chou, Android Studio: An IDE built for Android, srpanj 2013., dostupno na: https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html (Datum zadnjeg pregleda: 25.6.2021.)

[7] Službena stranica, Java, dostupno na: https://java.com/en/download/help/index.html (Datum zadnjeg pregleda: 28.6.2021.)

[8] Skupina autora, XML 1.0 Specification, World Wide Web Consortium, dostupno na: https://www.w3.org/TR/REC-xml/ (Datum zadnjeg pregleda: 29.6.2021.)

[9] P. Fennell, Extremes of XML, XML London 2013, lipanj 2013., dostupno na: https://xmllondon.com/2013/presentations/fennell/ (Datum zadnjeg pregleda: 29.6.2021.)

[10] J. Tamplin, Firebase is Joining Google!, Firebase Inc., listopad 2014., dostupno na: https://techcrunch.com/2014/10/21/google-acquires-firebase-to-help-developers-build-better-realtime-apps/ (Datum zadnjeg pregleda: 29.6.2021.)

[11] What is Adobe Photoshop?, Techopedia, dostupno na: https://www.techopedia.com/definition/32364/adobe-photoshop (Datum zadnjeg pregleda: 14.7.2021.)

[12] Configure your build, Android Studio, dostupno na: https://developer.android.com/studio/build (Datum zadnjeg pregleda: 14.7.2021.)

SAŽETAK

U ovom završnom radu prikazan je razvoj Android kviz aplikacije, koja ima dva moguća tipa igre: offline samostalni i online protiv drugih korisnika. Aplikacija je potpuno razvijena u Android Studio okruženju te je napisana u Java programskom jeziku. U izradi aplikacije korišten je Firebase kako bi se realizirala baza podataka u stvarnom vremenu i autentifikacija korisnika. Dizajn aplikacije kreiran je pomoću XML opisnog jezika. Logo aplikacije te dizajn pozadine izrađeni su u grafičkom računalnom programu Adobe Photoshop. Nakon uvoda u radu se nalazi poglavlje koje sadrži pet sličnih rješenja aplikacije koja se mogu usporediti sa ovom. Kroz ovaj rad opisane su glavne funkcionalnosti aplikacije i prikazani su glavni dijelovi kôda.

Ključne riječi: Android, Java, kviz, online igra

ABSTRACT

Android quiz application

This final paper shows the development of an Android quiz application. The application has two types of gameplays: offline singleplayer and online multiplayer. The application was completely developed in Android Studio environment, and it was written in Java programming language. Firebase was used in the development of the application to achieve user authentication and realtime database. Design of the application was created with XML markup language. The logo and the background of the application were designed in a raster graphics editor called Adobe Photoshop. After the introduction of this final paper comes the chapter that contains five similar solutions which can be compared to this project. Main functionalities are described within this paper and main parts of code are shown.

Keywords: Android, Java, online game, quiz

ŽIVOTOPIS

Petar Ivanković rođen je 14.05.1997. godine u gradu Požegi, Hrvatska. U gradu Požegi pohađao je i završio Osnovnu Školu Julija Kempfa. Tijekom svog osnovnoškolskog obrazovanja sudjeluje na školskim natjecanjima iz raznih predmeta, a kroz sve četiri godine sudjeluje na županijskim natjecanjima iz geografije. Nakon završene osnovne škole, 2012. godine upisuje opći smjer Gimnazije u Požegi. 2016. godine završava svoje srednjoškolsko obrazovanje te upisuje preddiplomski sveučilišni studij Računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.