LOKACIJSKI KVIZ

Dadić, Toni

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Stručni diplomski studij Informacijske tehnologije

TONI DADIĆ

Z A V R Š N I R A D

LOKACIJSKI KVIZ

Split, rujan 2024.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Stručni diplomski studij Informacijske tehnologije

Predmet: Programsko inženjerstvo

Z A V R Š N I R A D

Kandidat: Toni Dadić

Naslov rada: Lokacijski kviz

Mentor: mr. sc. Karmen Klarin, viši predavač

Split, rujan 2024.

Sažetak

U ovom radu je opisana izrada mobilne aplikacije koja korisniku daje mogućnost igranja kviza isključivo na lokaciji na kojoj se korisnik sa svojim mobilnim uređajem nalazi. Svaka lokacija nudi korisniku nekoliko tema i razina težine za igru. Rezultati se pohranjuju u bazu podataka u realnom vremenu te su dostupni svim korisnicima. Za izradu mobilne aplikacije korišten je Android Studio, dok je za dodatne usluge potrebne za izradu aplikacije korištena platforma Google Firebase.

Ključne riječi: Android studio, baza podataka, Google Firebase, kviz, lokacija

Summary

Location quiz

This paper presents the development of a mobile application designed to allow users to play location-based quizzes directly on their mobile devices. Each location provides users with various themes and difficulty levels to choose from. Quiz results are stored in real time in a database, making them accessible to all users. The mobile application was developed using Android Studio, with additional services implemented through the Google Firebase platform.

Key word: Android studio, database, Google Firebase, kuiz, location

1. Uvod

U modernom tehnološkom okruženju mobilne aplikacije igraju ključnu ulogu u svakodnevnim životima korisnika, nudeći im razne oblike interakcije i personaliziranih iskustava. Jedan od uzbudljivih trendova u razvoju mobilnih aplikacija je primjena geolokacijskih tehnologija, koje omogućavaju aplikacijama da isporuče sadržaj prilagođen trenutnoj fizičkoj lokaciji korisnika. Ova inovacija otvara vrata za nove načine interakcije, posebno u područjima igara, obrazovanja i društvenih mreža.

U ovom radu je predstavljena mobilna aplikacija, razvijena u Android Studiju, koja koristi Firebase za pohranu i upravljanje podacima. Specifičnost ove aplikacije leži u njezinoj integraciji geolokacijskih podataka, čime se omogućava korisnicima sudjelovanje u kvizu samo kada se nalaze na određenoj fizičkoj lokaciji. Tako kvizovi nisu samo intelektualni izazovi, već postaju i fizičke avanture koje korisnike potiču na istraživanje svojih okolina. Ova funkcionalnost dodaje novu dimenziju tradicionalnim kvizovima, pružajući korisnicima priliku da nauče više o lokacijama koje posjećuju kroz interaktivne i edukativne sadržaje.

Povezivanjem kviza s lokacijom korisnici se potiču da istražuju stvarni svijet oko sebe, dok istovremeno sudjeluju u natjecanju znanja. Na primjer, kviz se može aktivirati samo kada se korisnik nalazi u blizini određene znamenitosti, čime se stvara povezanost između lokacije i teme kviza. Ova funkcionalnost ne samo da povećava angažman korisnika već omogućava i prilagodbu sadržaja na temelju kulturnih, povijesnih ili geografskih značajki određene lokacije.

Jedna od ključnih značajki ove aplikacije je mogućnost odabira različitih tema i razina težine za svaki kviz, što dodatno personalizira korisničko iskustvo. Na svakoj lokaciji korisnici mogu birati između različitih kvizova koji su tematski vezani za specifičnu lokaciju. Osim toga, korisnici mogu odabrati razinu težine, uz mogućnost prilagodbe kviza svojim preferencijama i razini znanja. Ova fleksibilnost osigurava da aplikacija bude privlačna širokom spektru korisnika, od onih koji traže lagani izazov do iskusnih kvizaša koji žele testirati svoje znanje na najvišim razinama.

Android Studio, kao jedno od najpopularnijih razvojnih okruženja za Androidove aplikacije, pruža sve potrebne alate za razvoj složenih i funkcionalnih mobilnih aplikacija.

Kombinacija Java ili Kotlin programskih jezika, Androidovog kompleta za razvoj softvera (eng. *Software Development Kita, SDK*), te bogatog seta biblioteka omogućuje izradu aplikacija koje su ne samo tehnološki napredne, već i prilagođene korisnicima. Android Studio je u ovom projektu omogućio učinkovitu integraciju geolokacijskih funkcionalnosti, kao i razvoj intuitivnog korisničkog sučelja koje omogućava jednostavnu navigaciju i interakciju.

Firebase, kao moćna platforma za *backend* usluge, igra ključnu ulogu u razvoju i funkcionalnosti ove aplikacije. Firebase nudi širok spektar alata koji omogućuju sigurno pohranjivanje podataka, autentifikaciju korisnika, analitiku i upravljanje stvarnim vremenom. Korištenje Firebasea omogućilo je razvoj kviza koji se dinamički prilagođava potrebama korisnika, s brzim i sigurnim pristupom podacima te mogućnošću praćenja i analize korisničkih interakcija. Ove funkcionalnosti čine aplikaciju robusnom i pouzdanom, osiguravajući da korisnici mogu uživati u neprekinutom i sigurnom iskustvu igranja kviza.

Cilj ove aplikacije je kombinirati edukaciju i zabavu kroz interaktivno iskustvo koje se temelji na lokaciji korisnika. Uvođenjem lokalno specifičnih kvizova aplikacija ne samo da pruža korisnicima priliku da testiraju svoje znanje, već ih također potiče na istraživanje i učenje o svijetu oko sebe na interaktivan način. Na svakoj lokaciji korisnici se mogu suočiti s izazovima koji su specifično osmišljeni za tu lokaciju, čime se kvizovi pretvaraju u edukativne pustolovine koje povezuju korisnike s njihovim fizičkim okruženjem.

Nakon uvodnog dijela, rad se sastoji od nekoliko poglavlja. U poglavlju o korištenim tehnologijama i alatima opisane su tehnologije korištene pri izradi aplikacije, s posebnim naglaskom na ulogu Android Studija kao glavnog razvojnog okruženja te Firebase platforme koja pruža backend podršku. U dijelu o funkcionalnostima obrađuju se ključne značajke aplikacije poput geolokacijske provjere za pokretanje kviza, izbora različitih tema i razina težine te dinamičkog prilagođavanja sadržaja prema lokaciji korisnika, uz dodatne funkcionalnosti koje poboljšavaju korisničko iskustvo, kao što su praćenje napretka i pohranjivanje rezultata. Poglavlje o implementaciji detaljno opisuje razvoj aplikacije, uključujući korake od dizajna korisničkog sučelja do integracije backend servisa, pri čemu su objašnjeni tehnički aspekti poput autentifikacije korisnika, postavljanja geolokacijskih provjera i upravljanja bazom podataka u Firebaseu. Na kraju, u zaključku se pruža sažeti pregled rada uz preporuke za budući razvoj i moguća poboljšanja funkcionalnosti.

2. Korištene tehnologije i alati

Za izradu mobilne aplikacije lokacijskog kviza korišten je višenamjenski operacijski sustav otvorenog koda (engl. *open-source*) Android. Prvi uređaj koji se temelji na Androidovom operacijskom sustavu predstavila je 2008. godine tvrtka Google koja je Android kupila 2005. godine od tvrtke Android Inc. Danas je Android najrasprostranjeniji operacijski sustav na svijetu s oko 2 milijarde aktivnih korisnika mjesečno, dok trgovina za aplikacije Play sadrži preko 3,5 milijuna aplikacija [1].

Android je prvenstveno dizajniran za mobilne uređaje, ali također ima široku primjenu i na ostale elektroničke uređaje kao što su tableti, televizori i slično. Kao platforma otvorenog koda, Android omogućava proizvođačima uređaja i developerima aplikacija da prilagode operativni sustav svojim potrebama. To je rezultiralo razvojem raznih vrsta aplikacija, uključujući standardne Androidove aplikacije, privilegirane aplikacije i aplikacije proizvođača uređaja. Ova fleksibilnost čini Android popularnim izborom za mnoge proizvođače.

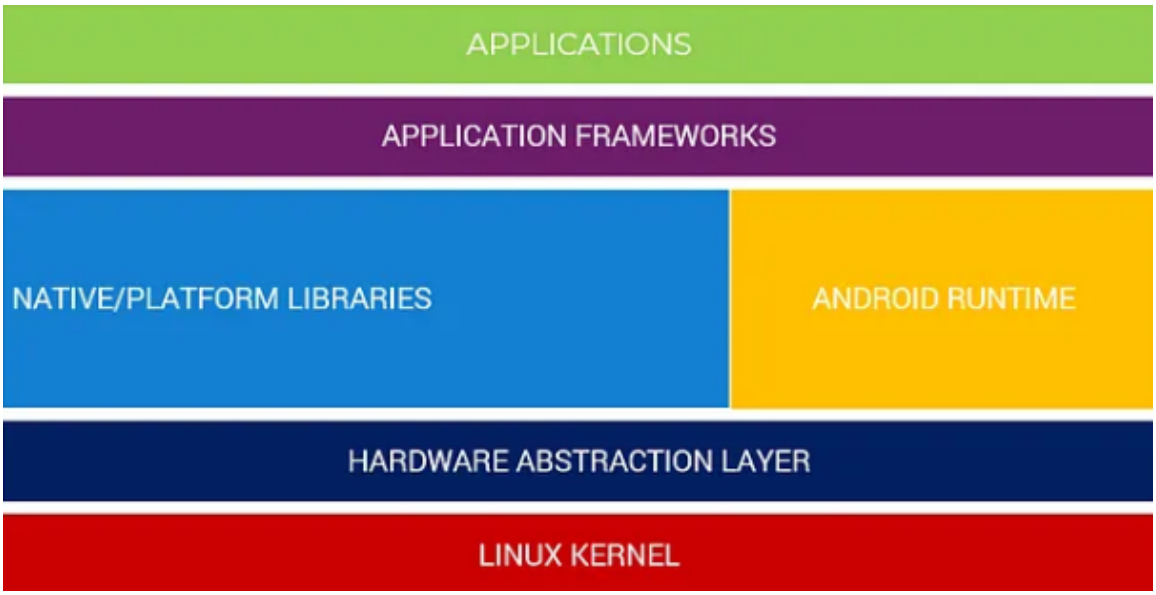
Arhitektura Androidovog sustava (Slika 1.) sastoji se od nekoliko slojeva, od kojih svaki ima specifičnu ulogu u funkcioniranju sustava [2]. Ovi slojevi su podijeljeni na različite komponente koje zajedno omogućavaju rad Androidove platforme. Osnovni slojevi Androidove arhitekture su sljedeći:

- **Jezgra (eng. *kernel*) Linuxa** – ovaj osnovni sloj upravlja svim hardverskim upravljačkim programima kao što su zaslon, tipkovnica, kamera, Bluetooth, audio i memorijski upravljački programi, koji su ključni za radnu funkcionalnost Androidovih uređaja.
- **Sloj apstrakcije hardvera (eng. *Hardware Abstraction Layer, HAL*)** – hardverski sloj apstrakcije nalazi se iznad jezgre Linuxa i pruža standardizirano sučelje za dobavljače hardvera za implementaciju funkcionalnosti specifičnih za uređaj bez modificiranja jezgre Androidovog sustava. HAL omogućuje Androidu interakciju s hardverskim komponentama na ujednačen način, bez obzira na osnovne hardverske razlike.
- **Nativne/platformske biblioteke (eng. *Native/Platform Libraries*)** – ovaj sloj obuhvaća različite biblioteke sustava napisane u programskim jezicima C i C++,

one pružaju bitne funkcije sustavu i aplikacijama kao što su grafika, medija, baza podataka i druge osnovne funkcije.

- **Android Runtime, ART** – sloj koji služi za pokretanje Androidovih aplikacija. Svaka aplikacija u Androidu se izvršava u svom procesu s vlastitom instancom ART-a. Ovo okruženje također uključuje jezgrene (engl. *core*) biblioteke koje omogućavaju aplikacijama pristup standardnim Androidovim API funkcijama.
- **Razvojni okvir (eng. *Framework*) Android**– ovaj sloj se može nazvati i srcem operacijskog sustava Android, pruža sučelja za programiranje aplikacija (eng. *Application Programming Interface, API*) i druge bitne usluge za razvoj aplikacija.
- **Aplikacijski sloj (eng. *Application Layer*)** – na vrhu Androidove arhitekture nalazi se aplikacijski sloj u kojem korisnici izravno komuniciraju s uređajem. Ovo uključuje osnovne aplikacije koje dolaze sa sustavom, kao i aplikacije koje korisnici preuzimaju s trgovine i drugih izvora.

Ova arhitektura omogućava Androidu da bude fleksibilan, prilagodljiv i sposoban raditi na širokom rasponu uređaja. Svaki sloj ima jasno definiranu ulogu i omogućava modulacijski dizajn sustava koji se lako može proširivati i unaprjeđivati.



Slika 1. Arhitektura Androidovog operacijskog sustava

2.1 Android Studio

Nakon razumijevanja arhitekture Androida, važno je upoznati i Android Studio, službeno razvojno okruženje (engl. *Integrated development environment IDE*) za razvoj Androidovih aplikacija koje je također korišteno kao alat za izradu mobilne aplikacije lokacijskog kviza [3].

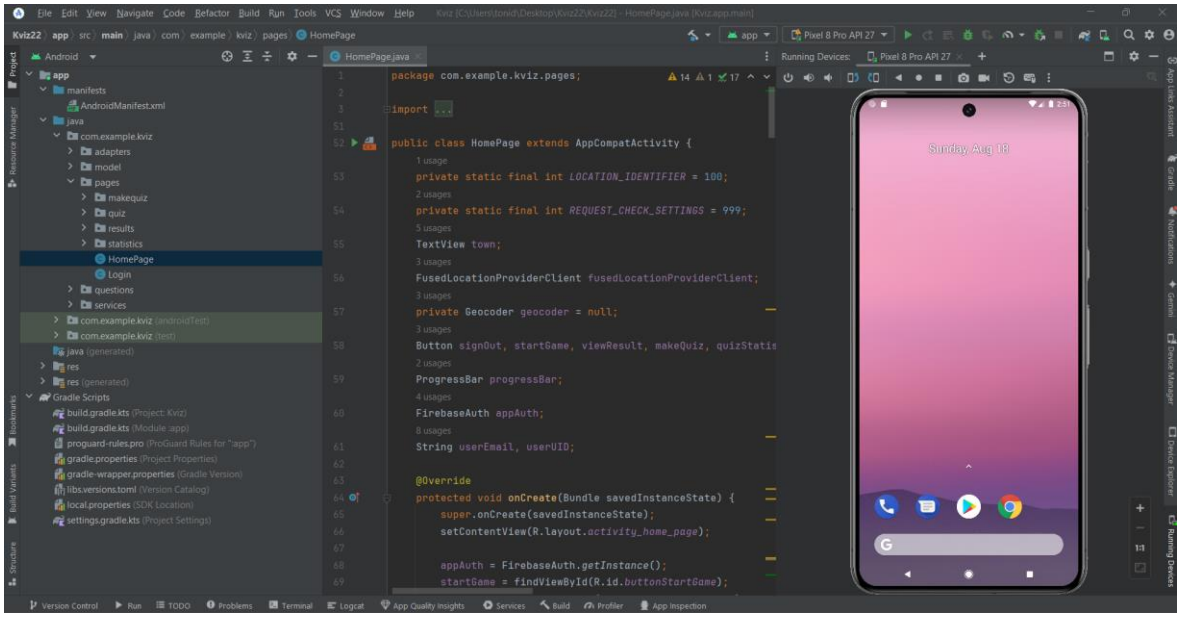
Android Studio je razvila tvrtka Google i predstavlja alat za programere koji žele stvarati aplikacije za Androidovu platformu. Predstavljen je 2013. godine kao nasljednik Eclipse Android razvojnih alata (E-ADT) i ubrzo je postao standard u industriji za razvoj Androidovih aplikacija.

Android Studio temelji se na softveru JetBrains IntelliJ IDEA, koji pruža sveobuhvatan skup alata za pisanje, testiranje, otklanjanje pogrešaka (engl. *debugging*) i optimiziranje Androidovih aplikacija. Jedan od ključnih aspekata Android Studija je njegova sposobnost integracije sa svim slojevima Androidove arhitekture, omogućujući developerima lak pristup funkcionalnostima poput upravljanja resursima, baze podataka, mrežnih operacija i sučelja.

Android Studio nudi još više značajki koje povećavaju produktivnost pri izradi Androidovih aplikacija, neke od njih su [3], [4]:

- fleksibilan sustav izrade temeljen na Gradleu
- brz emulator bogat značajkama
- jedinstveno okruženje u kojem se može razvijati za sve Androidove uređaje kao što je prikazano na Slici 2.
- uređivanje u stvarnom vremenu (eng. *Live Edit*) za ažuriranje komponenata u emulatorima i fizičkim uređajima u stvarnom vremenu
- predlošci koda i GitHub integracija koji će pomoći izgraditi česte značajke aplikacije i uvođenje jednostavnih dijelova koda
- opsežni alati i okviri za testiranje
- Lint alati za hvatanje performansi, upotrebljivosti, kompatibilnosti verzija i drugih problema
- C++ i Nativni razvojni komplet podrške (eng. Native Development Kit, NDK)

- ugrađena podrška za Google Cloud platformu, što olakšava integraciju Google Cloud Messaging i App Engine funkcionalnosti.

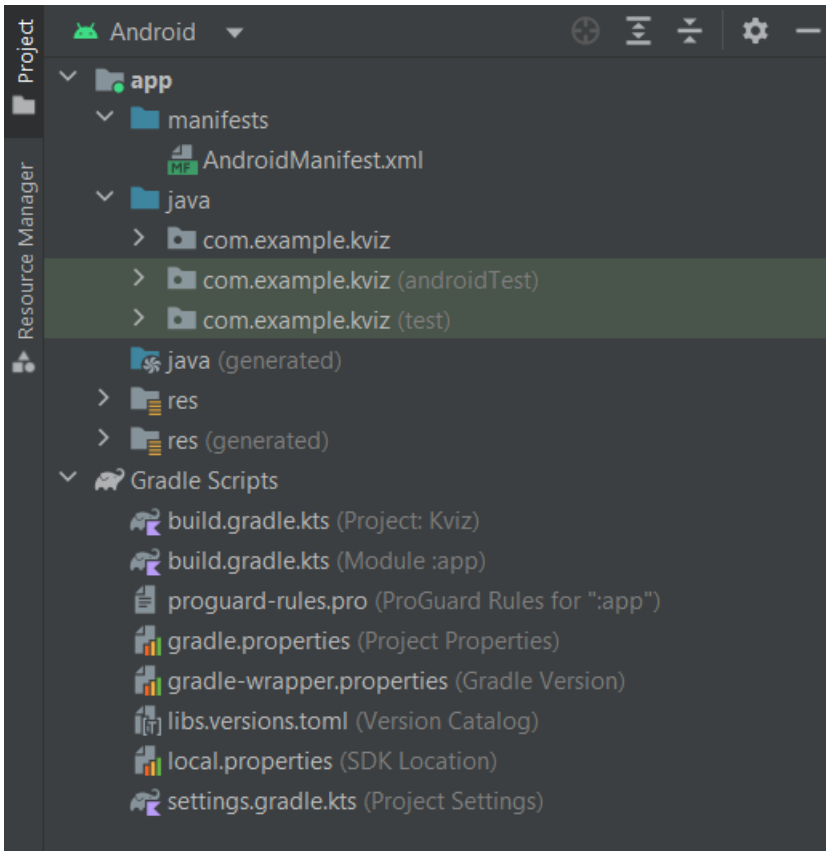


Slika 2. Radno okruženje u Android Studiju

Kod izrade nove aplikacije u Android Studiju zapravo se kreira novi projekt koji ima svoju strukturu. Svaki projekt sadrži jedan ili više modula s datotekama izvornog kôda i datotekama resursa [3], [4]. Vrste modula uključuju:

- module Androidove aplikacije
- knjižničke (engl. *library*) module
- Google App Engine module.

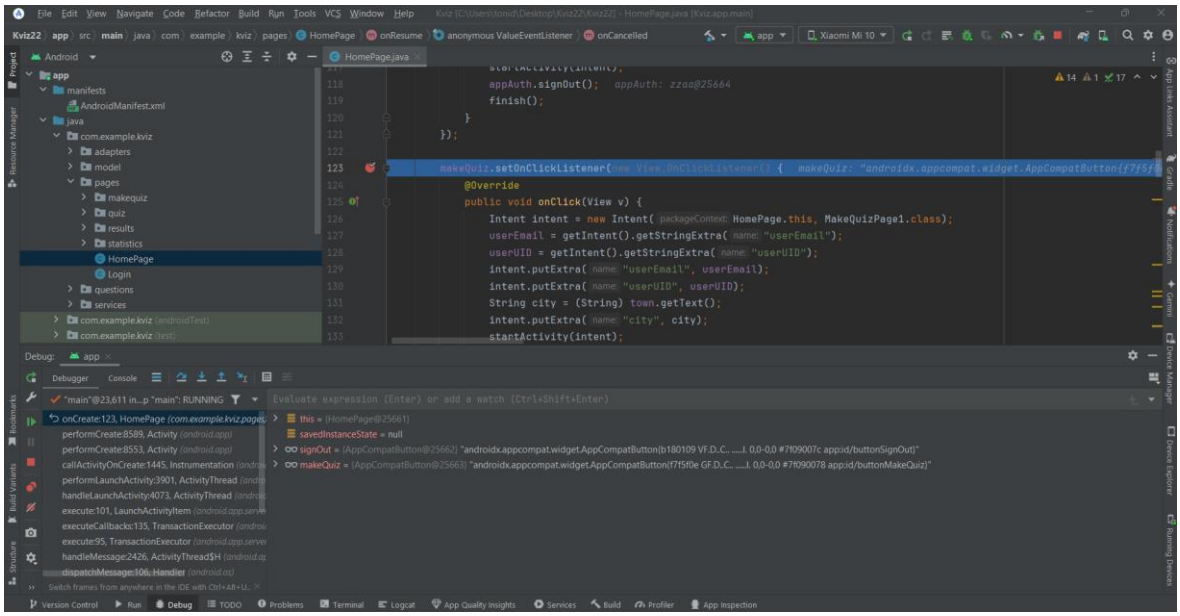
Prema zadanim postavkama, Android Studio prikazuje projektne datoteke u prikazu Androidovog projekta, kao što je prikazano na Slici 3. Ovaj je prikaz organiziran po modulima kako bi se omogućio brz pristup ključnim izvornim datotekama projekta.



Slika 3. Projektne datoteke u prikazu Androidovog projekta

Na Slici 3. su prikazane sve projektne datoteke, ali vrijedno je izdvojiti neke koje se smatraju najbitnijima za cijeli projekt. Datoteka `AndroidManifest.xml` ključna je komponenta svake Androidove aplikacije. Pruža bitne informacije o aplikaciji operativnom sustavu Android, uključujući naziv paketa aplikacije, verziju, dopuštenja, aktivnosti, usluge i primatelje. `AndroidManifest.xml` datoteka je potrebna sustavu Android za pokretanje aplikacije i određivanje njezine funkcionalnosti. Sve datoteke za izgradnju vidljive su na najvišoj razini, pod Gradle skriptama. `Build.gradle` je konfiguracijska datoteka koja se koristi u Android Studiju za definiranje postavki izgradnje za Androidov projekt. Napisan je u programskom jeziku Groovy i koristi se za konfiguriranje procesa izgradnje za projekt. Također je vrijedno spomenuti da se aplikacije mogu razvijati u Android Studiju korištenjem programskog jezika Java ili Kotlin. Kao što je prikazano na Slici 3., za razvoj lokacijske aplikacije korišten je programski jezik Java, naravno uz XML (eng. *eXtensible Markup Language*) za definiranje korisničkog sučelja [5].

Još jedna značajka koju je potrebno istaknuti je otklanjanje pogrešaka. Otklanjanje pogrešaka kritičan je dio razvoja softvera, a Android Studio pruža robustan skup alata za otklanjanje pogrešaka koji pomažu programerima da identificiraju i poprave probleme u svojim aplikacijama. Korištenjem ovih alata programeri mogu optimizirati performanse, testirati i potvrditi svoj kôd te poboljšati kvalitetu svojih aplikacija. Da bi koristili alate za otklanjanje pogrešaka u Android Studiju, programeri prvo moraju konfigurirati svoj projekt za otklanjanje pogrešaka dodavanjem prijelomnih točaka svom kodu (Slika 4.). Prijelomne točke su markeri koji govore *debuggeru* da pauzira izvršenje na određenoj točki kôda. Nakon što su prijelomne točke postavljene, programeri mogu pokrenuti svoju aplikaciju u *debug* modu i koračati kroz kôd redak po redak, provjeravajući varijable i procjenjujući izraze u hodu [5].



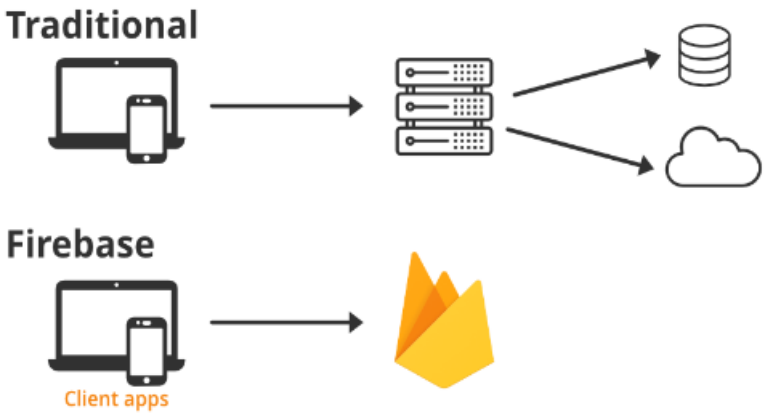
Slika 4. Primjer otklanjanja pogrešaka u Android Studiju

Na kraju se može zaključiti da Android Studio nije samo alat za razvoj aplikacija, već cijelo razvojno okruženje koje podržava sve faze razvoja, od pisanja kôda do testiranja i optimizacije. Njegova integracija s Androidovim ekosustavom, zajedno s moćnim alatima koje nudi, čini ga neophodnim za svakog developera koji želi razvijati visokokvalitetne Androidove aplikacije. Korištenjem Android Studija programeri mogu iskoristiti sve

prednosti Androidove platforme i stvoriti aplikacije koje će se izdvojiti na tržištu, osiguravajući vrhunsko korisničko iskustvo.

2.2 Google Firebase platforma

Firebase je sveobuhvatna platforma koja programerima pomaže da jednostavno izgrade, upravljaju i razvijaju svoje aplikacije korištenjem niza alata i usluga (Slika 5.). Pomaže razvojnim programerima da svoje aplikacije izgrade brže i sigurnije. Na Firebase strani nije potrebno programiranje što olakšava učinkovitiju upotrebu njegovih značajki.



Slika 5. Firebase koncept

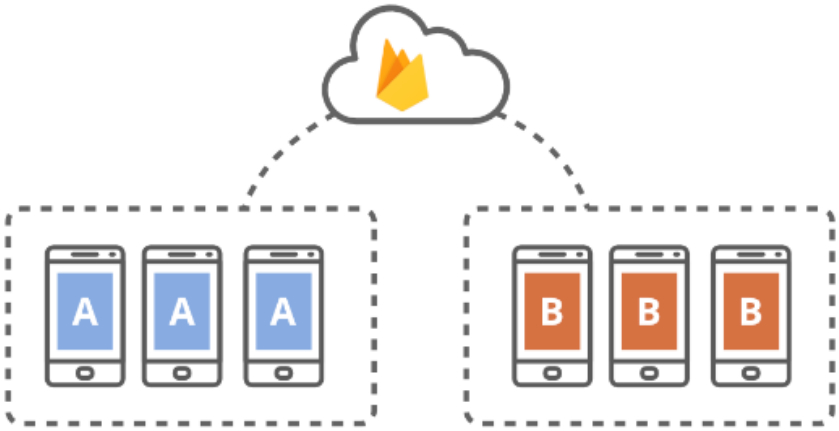
Firebase je tvrtka koja je 2011. godine počela razvijati *backend* softver. Prvotno je predstavljen kao *backend-as-a-service* (BaaS) platforma, međutim Firebase je brzo evoluirao u potpuni ekosustav koji podržava cijeli životni ciklus aplikacija, od razvoja i testiranja do distribucije i angažmana korisnika. Njegovo ime nastavlja se kao skup pozadinskih usluga računalstva u oblaku (engl. *cloud*) i platformi za razvoj aplikacija koje pruža Google. U njemu se nalaze baze podataka, razne usluge, autentifikacija i integracija za razne aplikacije, uključujući Android, iOS, JavaScript, Node.js, Java, Unity, PHP i C++ [6] ,[7].

Firebase nudi razne značajke koje pomažu programerima u razvoju aplikacija, ovo su neke od najčešće korištenih [7], [8]:

- **Baza podataka u stvarnom vremenu (engl. *realtime database*)** – radi se o NoSQL (eng. *Not Only SQL*) bazi podatka, što znači da ne koristi tradicionalni relacijski model kao što su tablice i stupci, umjesto toga koriste razne modele za pohranu i upravljanje podacima kao što su na primjer grafovi, ključ-vrijednost (engl. *key-value*) parovi i slično. Ova baza podataka smještena u oblaku organizacijama omogućuje pohranu i sinkronizaciju podataka u stvarnom vremenu na svim uređajima svojih korisnika. To olakšava izradu aplikacija koje su uvijek ažurne, čak i kada su korisnici izvan mreže. Baza podataka u stvarnom vremenu sprema podatke u JSON (eng. *JavaScript Object Notation*) formatu pa se ujedno može smatrati i velikom JSON datotekom.
- **Firestore** – radi se o još jednoj vrsti baze podataka (također NoSQL) koju Firebase nudi, a to je modernija verzija baze podataka koja nudi napredne mogućnosti kao što su pohrane, sinkronizacije i upiti putem aplikacije na globalnoj razini. Pohranjuje podatke u obliku objekata poznatih i kao dokumenti (engl. *Documents*). Ima par ključ-vrijednost i može pohraniti sve vrste podataka kao što su nizovi, binarni podaci, pa čak i JSON stabla.
- **Firebase autentifikacija** – ova značajka omogućava jednostavno upravljanje autentifikacijom korisnika, nudeći podršku za razne metode prijave, uključujući *e-mail* i lozinku, prijavu putem Googlea, Facebooka i drugih društvenih mreža, kao i anonimnu prijavu (Slika 6.). Smanjuje radnu snagu i napor koji je potreban za razvoj i održavanje usluge autentifikacije korisnika. Čak se nosi sa zadacima poput spajanja računa, koji ako se rade ručno, mogu biti naporni [9].

Slika 6. Firebase autentifikacija

- **Firebase Cloud Messaging (FCM)** – ova usluga pruža vezu između poslužitelja i krajnjih korisnika aplikacije, koja se može koristiti za primanje i slanje poruka i obavijesti kao što je prikazano na Slici 7. FCM omogućava besplatno slanje *push*-obavijesti i poruka korisnicima na Androidovim, iOS i *web*-platformama. Programeri mogu koristiti FCM za slanje personaliziranih obavijesti, marketinških poruka, ili za obavještavanje korisnika o važnim događajima unutar aplikacije, čime se povećava angažman korisnika.



Slika 7. Firebase Cloud Messaging (FCM)

- **Firebase Crashlytics** – alat za praćenje i analizu rušenja aplikacija. Ovaj alat pomaže programerima brzo identificirati i popraviti greške koje uzrokuju padove aplikacija, omogućujući stabilnije i pouzdanije aplikacije. Crashlytics pruža detaljne izvještaje o uzrocima rušenja, uključujući informacije o uređaju, verziji operacijskog sustava i kontekstu u kojem je došlo do rušenja.
- **Firebase laboratorijsko testiranje** – usluga je temeljena na oblaku koja programerima omogućuje testiranje svoje aplikacije na različitim uređajima i konfiguracijama. Ovo značajno smanjuje vrijeme i resurse potrebne za testiranje aplikacija te programerima pomaže osigurati da aplikacija dobro radi na raznim uređajima i u različitim mrežnim uvjetima.

Uz navedene značajke Firebase nudi i brojne druge brojne prednosti za razvoj aplikacija. Jedna od prednosti je sposobnost integracije s ostatkom Googleovog ekosustava, uključujući Google analitiku, Google oglase, i druge alate koji pomažu u unaprjeđenju aplikacija i poslovnih performansi. Firebase također podržava *cross-platform* razvoj, omogućujući programerima da razvijaju aplikacije koje se mogu koristiti na više platformi, uključujući Android, iOS i *web*.

Firebase je idealan za *startupove* i mala poduzeća koja žele brzo izgraditi i skalirati svoje aplikacije, ali ga također koriste i velike kompanije zbog njegove pouzdanosti, sigurnosti i jednostavnosti integracije. Još jedna prednost je cijena, Firebase se može početi koristiti besplatno, a cijena se povećava ovisno o broju korisnika, količini korištene memorije i slično. Ova platforma predstavlja sveobuhvatan alat za sve faze razvoja aplikacija, od ideje do implementacije, čineći ga neophodnim za moderan razvoj aplikacija.

3. Funkcionalnosti

U ovom dijelu će se detaljno razraditi sve funkcionalnosti koje mobilna aplikacija lokacijski kviz sadrži. Ukratko, radi se o geolokacijskoj aplikaciji kviza koja korisniku daje mogućnost igranja kviza isključivo na lokaciji na kojoj se nalazi. Da bi korisnik pristupio kvizu, prvo se mora registrirati svojom *e-mail* adresom i lozinkom te potom prijaviti. Nakon što se korisnik uspješno prijavi u aplikaciji, prvo se očitava trenutna lokacija korisnika odnosno mobilnog uređaja. Dok se lokacija ne učita, u aplikaciji nije moguća nijedna radnja, također ako korisnik ne dopusti korištenje lokacije uređaja, korisnika će se vratiti na stranicu za prijavu i registraciju.

Ako je prijava i očitavanje lokacije uspješno, korisnik ima nekoliko opcija na početnom zaslonu. Korisnik može započeti igru za očitanu, tj. trenutnu lokaciju u nekoliko kategorija (glazba, sport, povijest i hrana), kao i u nekoliko razina težine (lako, srednje, teško i miks). Sva pitanja se odnose na trenutnu lokaciju, odnosno grad u kojem se korisnik nalazi. Nakon odigranog kviza rezultati se spremaju u bazu podataka u stvarnom vremenu. Ako ne postoji kreiran kviz za trenutnu lokaciju ili pojedina tema nije kreirana, korisnik će dobiti poruku da odabrani kviz još nije kreiran i da se ne može pokrenuti.

Druga opcija koju svaki korisnik ima je pregled rezultata. Rezultati se prikazuju u kombinaciji korisnikove *e-mail* adrese, lokacije, kategorije, razine težine i ukupnim bodovima. Iz baze se povlače u aplikaciju samo najbolji rezultati za svaku kombinaciju igranja. Opcija pregleda rezultata nudi i filtriranje rezultata u bilo kojoj kombinaciji vezanih za korisnikovu *e-mail* adresu, lokaciju, kategoriju i razinu težine.

Postoje i dvije opcije kojima pristup ima samo administrator. Prva je opcija kreiranja kviza kroz mobilnu aplikaciju. U navedenoj opciji administrator unosi lokaciju (ne mora biti trenutna lokacija), kategoriju, razinu težine i na kraju unosi novo pitanje koje se odmah automatski sprema u bazu podataka i pojavit će se kada se idući put pokrene kviz na toj lokaciji. Prilikom kreiranja novog pitanja kroz aplikaciju, svi korisnici aplikacije će na svoje uređaje dobiti notifikaciju o novo kreiranom pitanju.

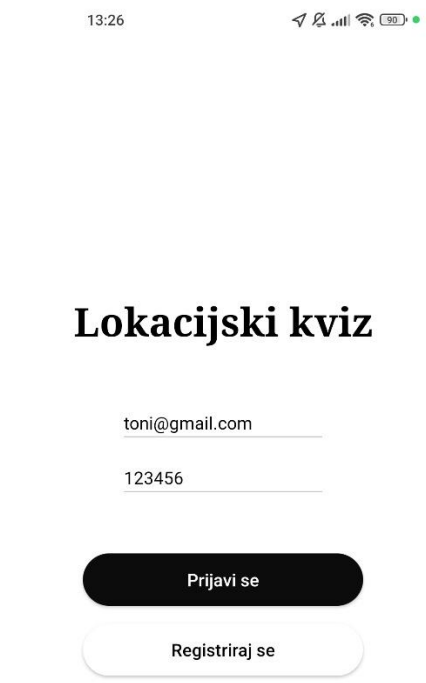
Druga opcija kojoj pristup ima samo administrator je pregled statistike kvizova – opcija koja daje administratoru informaciju o tome koji se kvizovi igraju najčešće, koliki je prosječni rezultat za svaki kombinaciju igranja i slično. Statistika je prikazana kao

kombinacija lokacije, kategorije, razine težine, ukupnog broja igranja i prosječnog rezultata. S ovim informacijama administrator može pratiti statistiku i po potrebi modificirati pitanja, dodavati nova, brisati stara i slično.

U nastavku će se detaljno opisati sve navedene funkcionalnosti koje sadrži mobilna aplikacija lokacijskog kviza.

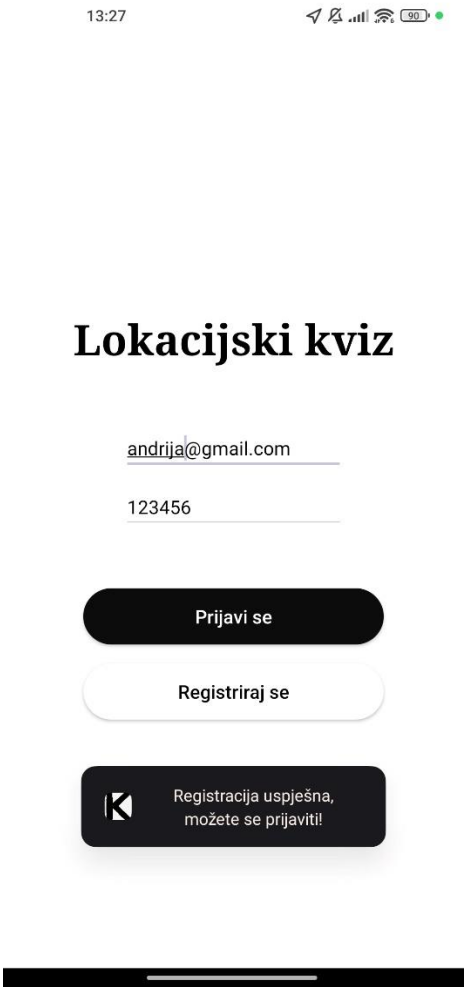
3.1 Autentifikacija korisnika i pristup lokaciji uređaja

Da bi korisnik pristupio aplikaciji, prvo mora odraditi registraciju, kod pokretanja aplikacije početni zaslon prikazan na Slici 8. daje mogućnost registracije za nove korisnike te prijavu za postojeće. U aplikaciju se prijavljuje s registriranom korisnikovom *e-mail* adresom i lozinkom.

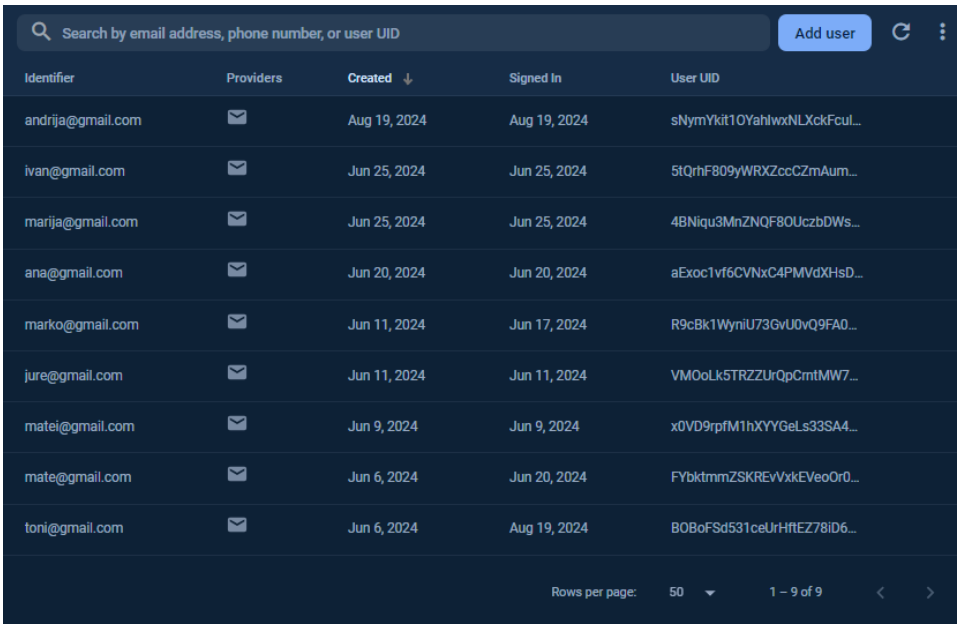


Slika 8. Početni zaslon aplikacije

Je li korisnik uspješno registriran odmah javi *toast* poruka u aplikaciji (Slika 9.), ali administrator može isto provjeriti sa strane Firebasea pod značajkom „autentifikacija“. Nakon uspješne registracije korisnika u aplikaciji, u stvarnom vremenu će se novi korisnik sa svojim podacima dodati listi svih aktivnih korisnika aplikacije (Slika 10.).

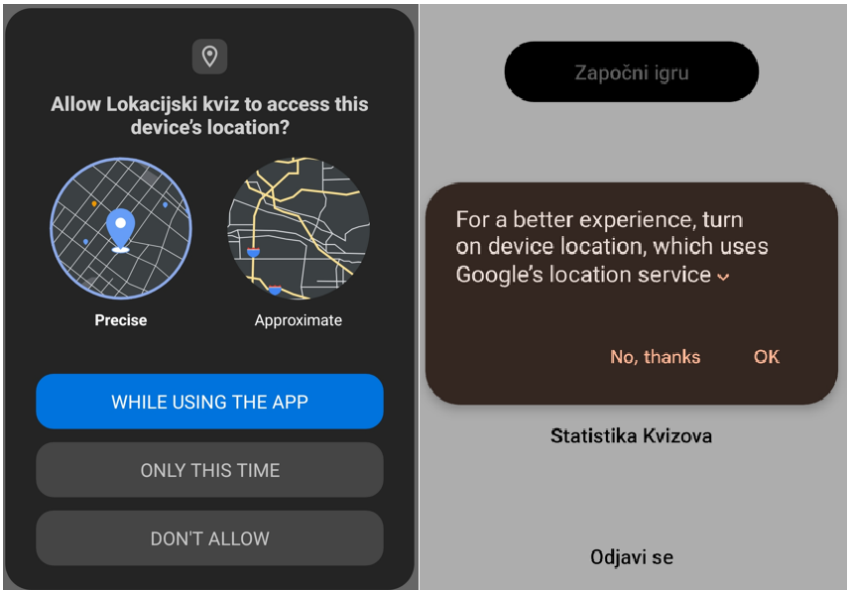


Slika 9. Primjer uspješne registracije korisnika u aplikaciji



Slika 10. Primjer uspješne registracije korisnika u Firebaseu

Svi korisnici koji se nalaze u Firebase autentifikacijskoj listi mogu se prijaviti u aplikaciju svojom *e-mail* adresom i lozinkom. Prilikom prijave korisnika automatski se izvršava provjera je li dopušteno korištenje lokacije unutar aplikaciji, kao i je li opcija lokacije uključena na uređaju (Slika 11.). Kao što je već spomenuto, ako se ne dopusti korištenje i pristup lokaciji, korisnika se vraća na početni zaslon za prijavu.



Slika 11. *Pop-up* prozori za dopuštanje aplikaciji korištenje lokacije i uključivanje lokacije

Ako je prijava uspješna i lokacijske postavke uključene, očitava se trenutna lokacija uređaja, tj. ime grada i ispisuje na početnom zaslonu aplikacije kao što je prikazano na Slici 12. Za očitavanje lokacije koristi se Google Play Service Location API koji omogućava aplikacijama precizno i energetski učinkovito dobivanje korisnikove trenutne lokacije. Ovaj API koristi kombinaciju GPS-a, Wi-Fi mreža, mobilnih tornjeva i senzora uređaja kako bi brzo i pouzdano odredio lokaciju. Programerima pruža mogućnost biranja između različitih razina preciznosti i učestalosti ažuriranja lokacije, što omogućuje optimizaciju za različite potrebe aplikacija, poput praćenja u stvarnom vremenu ili povremenog očitavanja lokacije.

Slika 12. Uspješna prijava u aplikaciju

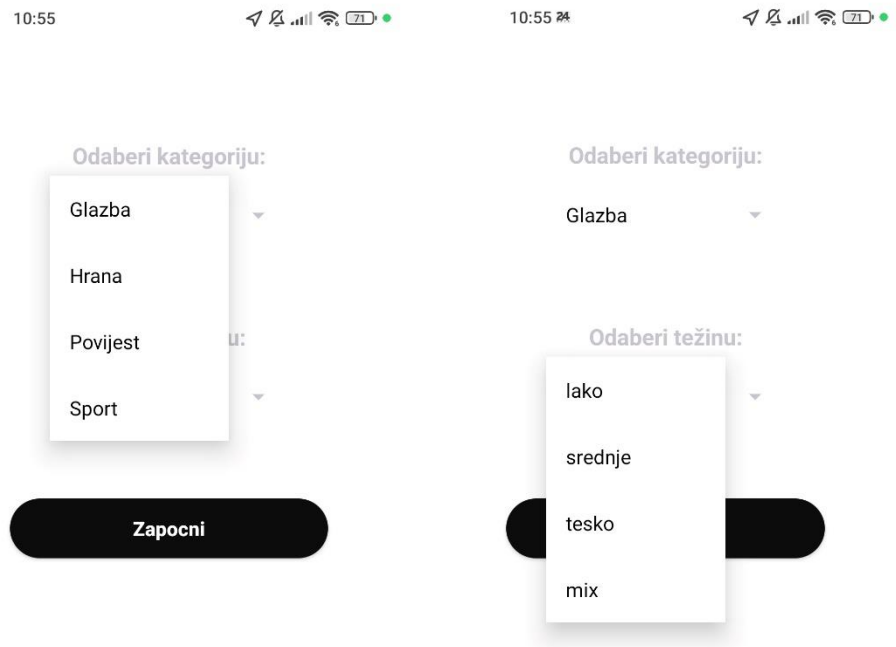
U prethodnom primjeru je prikazana uspješna prijava korisnika, ali postoji razlika ako se u aplikaciju prijavljuje administrator. Određivanje koji korisnik ili korisnici će imati administratorska prava definirano je u samom kôdu. Kada se administrator prijavi u

aplikaciju, na početnom zaslonu su vidljive još neke opcije (Slika 13.) koje ostali korisnici ne vide. Ostatak procedure vezane za prijavu i postavke lokacije vrijede za administratora kao i za sve ostale korisnike.

Slika 13. Uspješna prijava administratora u aplikaciji

3.2 Igra i pravila lokacijskog kviza

Nakon što je korisnik uspješno prijavljen i trenutna lokacija prikazana na početnom zaslonu, korisnik odabirom opcije da započne igru otvara novi zaslon. Novootvoreni zaslon nudi korisniku odabir kategorije koju želi igrati, kao i razinu težine kao na Slici 14.



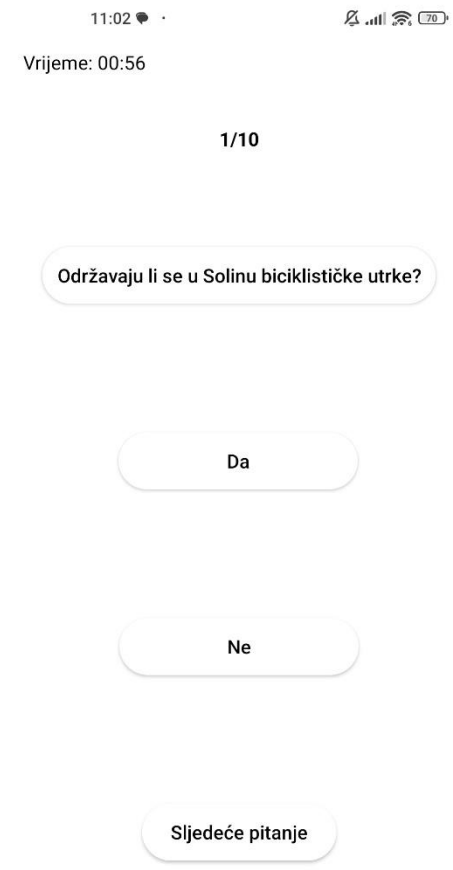
Slika 14. Odabir kategorije i razine težine kviza

Ako neku od kombinacija kategorije i razine težine administrator nije još kreirao ili nema dovoljan broj pitanja u bazi podataka (svaka razina težine za određenu kategoriju mora imati kreirano minimalno 10 pitanja da bi se pokrenuo kviz), korisnik nije u mogućnosti pokrenuti igru te mu se na ekranu ispisuje poruka „Ova kategorija se ne može trenutno igrati!“.

Nakon uspješnog odabira željene kategorije i razine težine korisnik započinje s igrom, primjer čega je prikazan na Slici 15. Važno je naglasiti da se svaki kviz sastoji od 10 pitanja nasumično dohvaćenih iz baze podataka za željenu kategoriju i razinu težine. Po pokretanju igre mjerač vremena (engl. *timer*) počinje s odbrojavanjem te za svaku igru korisnik ima maksimalno 1 minutu. Ako ne odgovori na sva pitanja unutar minute, igra se prekida. Kada korisnik završi s igrom, na ekranu se ispisuje rezultat koji se odmah sprema u bazu podataka.

Postoje 3 različita modela pitanja ovisno o odabranoj razini težine:

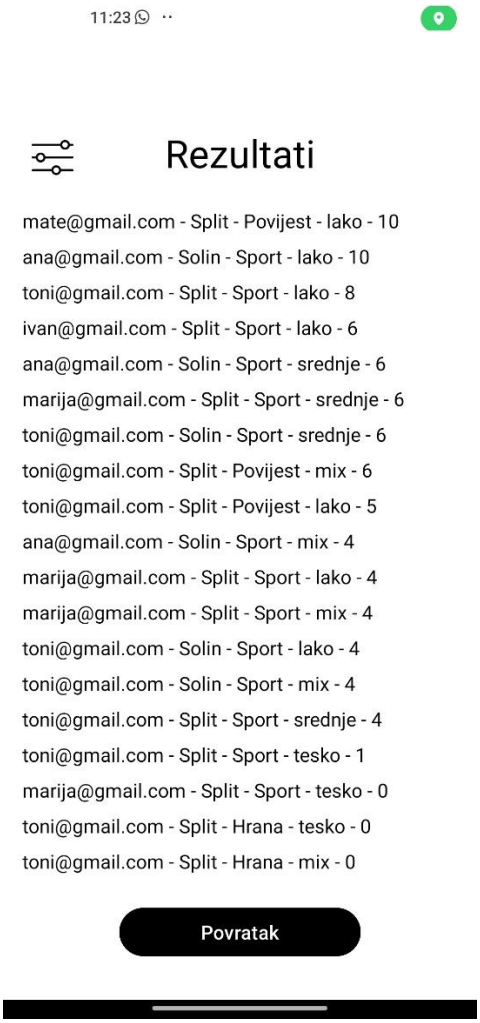
- **lako** – sva pitanja nude dogovore „Da“ i „Ne“ (Slika 15.)
- **srednje** – sva pitanja imaju 3 ponuđena odgovora od kojih je jedan točan
- **teško** – na pitanja se ne nude odgovori nego je potrebno upisati točan odgovor
- **miks** – kombinacija pitanja iz razina težine „lako“, „srednje“ i „teško“.



Slika 15. Uspješno pokrenut kviz

3.3 Pregled rezultata

Svaki korisnik ima opciju „Rezultati“ na početnom zaslonu. U bazu podataka upisuju se svi rezultati koji su ostvareni neovisno o kojem se korisniku radi, lokaciji, kategoriji ili razini težine. Međutim, u aplikaciji su vidljivi samo najbolji rezultati za pojedinu kombinaciju korisnika, lokacije, kategorije i razine težine (Slika 16.). Ako su dva različita korisnika ostvarila isti, a ujedno i najbolji rezultat za isti kviz, oba rezultata se prikazuju u aplikaciji.



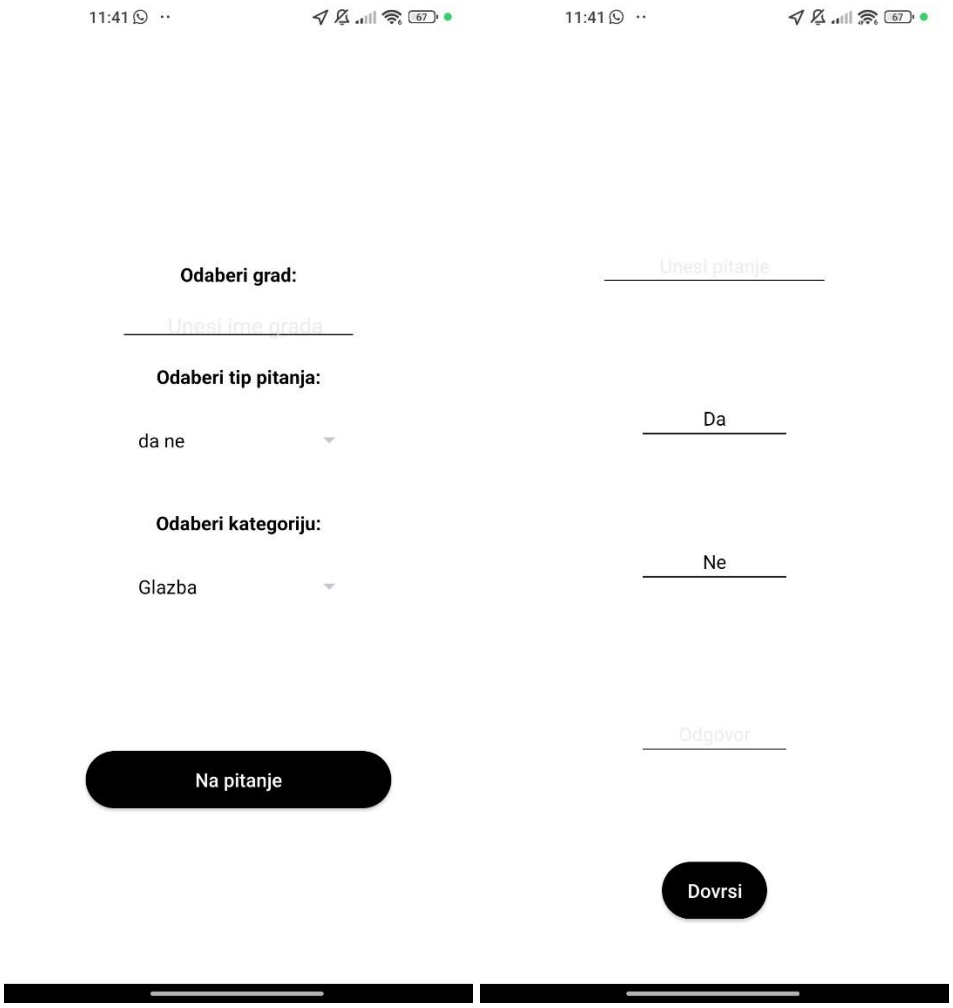
Slika 16. Prikaz rezultata

Prikaz rezultata moguće je filtrirati po želji, klikom na dugme za filtriranje prikazuju se opcije za filtriranje. Korisnik može filtrirati rezultate po *e-mail* adresi korisnika (ali je izbor samo trenutno prijavljeni korisnik ili svi korisnici), lokaciji, kategoriji i razini težine, kao i bilo koju kombinaciju od navedenog.

3.4 Administratorske funkcije

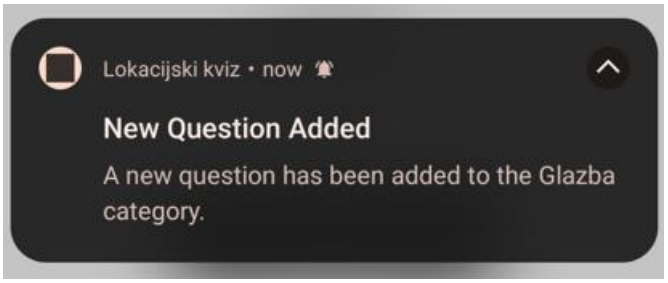
Jedna od opcija u aplikaciji koju može koristiti samo administrator je kreiranje kviza, to jest dodavanje novog pitanja u bazu podataka kroz aplikaciju. Na početnom zaslonu aplikacije administratoru je vidljiva opcija za kreiranje kviza pomoću koje se dodaje novo pitanje u bazu podataka. Prvo administrator mora upisati lokaciju za koju želi dodati pitanje (za dodavanje pitanja u bazu podataka nije bitna trenutna lokacija uređaja), kategoriju i tip pitanja kao na Slici 17.

Nakon odabranih opcija u prvom koraku potrebno je upisati željeno pitanje, ovisno o odabranom tipu pitanja u prvom koraku, otvara se zaslon za unos novog pitanja prilagođen željenom tipu pitanja, također vidljivo na Slici 17. Po završetku unosa novog pitanja, ono se automatski sprema u bazu podataka u stvarnom vremenu.



Slika 17. Unos novog pitanja u bazu podataka kroz aplikaciju

Nakon uspješnog dodavanja novog pitanja u bazu podataka kroz aplikaciju, svaki korisnik na svoj uređaj prima notifikaciju o novododanom pitanju (Slika 18.).



Slika 18. Primjer notifikacije

Druga opcija koju može samo administrator koristiti je pregled statistike. U pregledu statistike vidljive su sve kombinacije kvizova barem jednom odigrane, a prikazuje se svaka kombinacija kviza s lokacijom, kategorijom i razinom težine, ali s dodatkom ukupnog broja igranja pojedinog kviza te prosječni rezultat (Slika 19.). Ova opcija administratoru pomaže kako bi što bolje optimizirao pojedine kvizove.

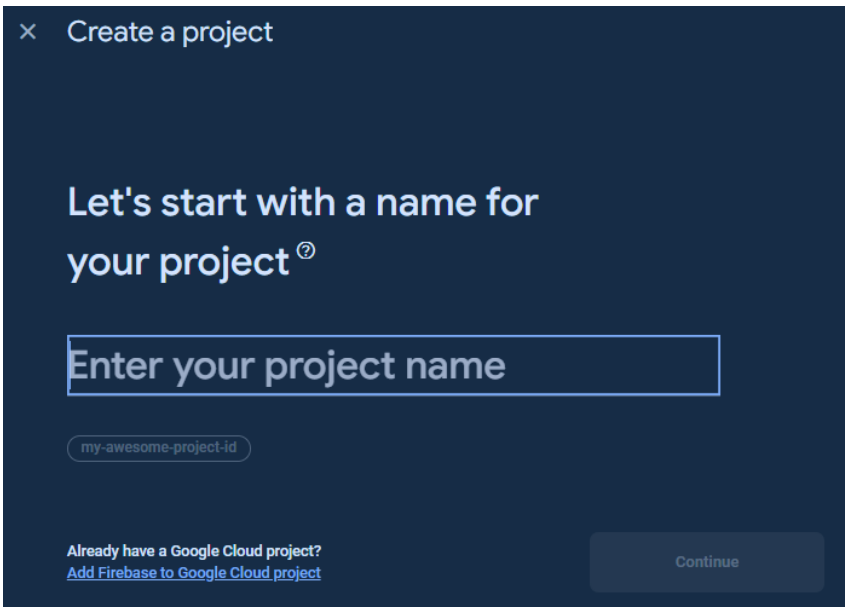
Slika 19. Prikaz statistike

4. Implementacija

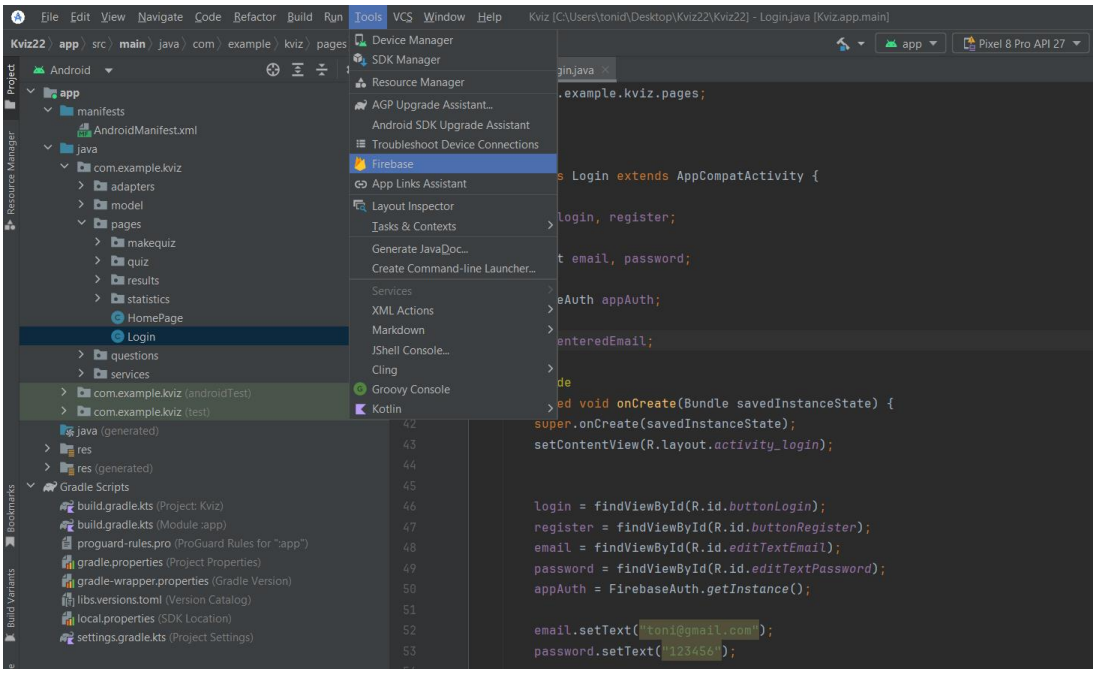
Ovo poglavlje pruža detaljan pregled procesa izrade i integracije svih bitnijih komponenti unutar aplikacije. U ovom dijelu opisuje se konkretna realizacija dizajniranih rješenja, uključujući tehnologije, alate i metode korištene za razvoj aplikacije. Fokus je na nekim od ključnih koraka implementacije, kao što su:

- autentifikacija korisnika
- Google Play Service Location API za očitavanje trenutne lokacije
- baza podataka u stvarnom vremenu
- notifikacija
- reciklirajući prikaz u Android Studiju (eng. *RecyclerView*).

Kao što je već u prethodnom tekstu spomenuto, za razvoj i implementaciju aplikacije korišten je Android Studio u programskom jeziku Java te Google Firebase platforma. Da bi ova dva alata međusobno funkcionirala, potrebno ih je povezati. Ukratko, prvo je potrebno kreirati novi Firebase projekt (Slika 20.), a zatim kroz Android Studio povezati kreirani Firebase projekt koristeći Firebase asistenta (engl. *Firebase Assistant*). Firebase asistent je već integriran alat u Android Studiju za jednostavno povezivanje s Firebase projektima (Slika 21.).



Slika 20. Primjer kreiranja novog Firebase projekta



Slika 21. Lokacija Firebase asistenta u Android Studiju

4.1 Autentifikacija korisnika

Da bi se korisnik registrirao i prijavio u aplikaciju, korištena je značajka Firebasea za autentifikaciju. Ova značajka nudi nekoliko opcija, ali u aplikaciji lokacijskog kviza korištena je opcija za registraciju i prijavu s korisničkom *e-mail* adresom i lozinkom. U aplikaciju se mogu prijaviti svi uspješno registrirani korisnici čija je lista vidljiva u Firebaseu kao što je već prikazano na Slici 10. Dio kôda kojim je implementirana autentifikacija korisnika može se vidjeti u Ispisima 1 i 2.

```
void createAccount(String userEmail, String userPassword){
        appAuth.createUserWithEmailAndPassword(userEmail,
userPassword).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if(task.isSuccessful()){
                    String userId = task.getResult().getUser().getUid();
                    DatabaseReference database =
FirebaseDatabase.getInstance().getReference();
                    Map<String, Object> userData = new HashMap<>();
```

```
                    String email = task.getResult().getUser().getEmail();
                    if(Objects.equals(email, "toni@gmail.com")){
                        userData.put("role", "admin");
                    } else {
                        userData.put("role", "user");
                    }
                    database.child("users").child(userId).setValue(userData)
                            .addOnSuccessListener(new OnSuccessListener<Void>()
{
                                @Override
                                public void onSuccess(Void aVoid) {
                                    Toast.makeText(Login.this, "Registracija
uspješna, možete se prijaviti!", Toast.LENGTH_SHORT).show();
                                }
                            })
                            .addOnFailureListener(new OnFailureListener() {
                                @Override
                                public void onFailure(@NonNull Exception e) {
                                    Toast.makeText(Login.this, "Pogreska
prilikom registracije: " + e.getMessage(), Toast.LENGTH_SHORT).show();
                                }
                            });
                } else {
                    Toast.makeText(Login.this, "Registracija neuspješna ili
korisnik već postoji!",Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
```

Ispis 1. Funkcija za registraciju korisnika

```
void loginUser(String userEmail, String userPassword){
        appAuth.signInWithEmailAndPassword(userEmail,
userPassword).addOnCompleteListener(this, new OnCompleteListener<AuthResult>()
{
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if(task.isSuccessful()){
                    String userUID = task.getResult().getUser().getUid();
                    Intent intent = new Intent(Login.this, HomePage.class);
                    intent.putExtra("userEmail", userEmail);
                    intent.putExtra("userUID", userUID);
                    startActivity(intent);
                    Toast.makeText(Login.this, "Uspješno ste se prijavili!",
Toast.LENGTH_SHORT).show();
                }
                else{
                    Toast.makeText(Login.this, "Prijava neuspješna, niste
registrirani!", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
```

Ispis 2. Funkcija za prijavu korisnika u aplikaciju

Unutar funkcije za registraciju korisnika u Ispisu 1 vidljiva je i dodjela prava korisnicima. Korisnici mogu dobiti administratorska prava `admin` ili prava običnih korisnika `user`. Korisniku ili korisnicima s dodijeljenim `admin` pravima dostupne su dodatne funkcionalnosti unutar aplikacije s obzirom na korisnike s `user` pravima.

4.2 Google Play Service Location API za očitavanje trenutne lokacije

Za očitavanje trenutne lokacije uređaja korišten je Google Play Service Location API. Ovaj API omogućava programerima jednostavan i učinkovit način za integraciju funkcionalnosti lokacije u Androidove aplikacije. Isto tako kombinira podatke iz različitih izvora, uključujući GPS, Wi-Fi mreže i mobilne tornjeve, kako bi pružio precizne i energetski učinkovite informacije o lokaciji korisnika. Korištenjem ovog API-ja aplikacije mogu pružiti usluge poput praćenja u stvarnom vremenu, *geofencinga* i prilagođenih lokacijskih obavijesti, bez potrebe za upravljanjem složenim pozadinskim procesima za očitavanje lokacije.

U poglavlju funkcionalnosti opisane su potrebne postavke uređaja vezane za lokaciju da bi korisnik uspješno mogao pristupiti aplikaciji (Ispis 3.). Ako su na uređaju sve lokacijske postavke podešene, aplikacija po uspješnoj prijavi korisnika očitava trenutnu lokaciju te je ispisuje na zaslon.

```
if (ActivityCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
        ActivityCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
    return;
}
```

Ispis 3. Primjer provjere ima li uređaj pristup lokaciji

U kodu je očitavanje trenutne lokacije podijeljeno na 3 funkcije:

- **`getLastKnownLocation`** – dohvaćanje zadnje zabilježene lokacije uređaja (Ispis 4.)
- **`setLocation`** – iz dohvaćene lokacije izdvajanje grada u kojem se uređaj nalazi i ispis istog na ekranu (Ispis 5.)
- **`setupLocationTracking`** – postavljanje praćenja lokacije, postavljajući interval i preciznost ažuriranja te definira kako će aplikacija obraditi podatke o lokaciji kada postanu dostupni (Ispis 6.).

```
public void getLastKnownLocation() {

    fusedLocationProviderClient.getLastLocation()
            .addOnSuccessListener(this, new OnSuccessListener<Location>() {
                @Override
                public void onSuccess(Location location) {
                    if (location != null) {
                        setLocation(location);
                    }
                }
            })
```

Ispis 4. Dohvaćanje zadnje zabilježene lokacije uređaja

```
public void setLocation(Location location) {

        if (location != null) {

            double latitudeValue = location.getLatitude();

            double longitudeValue = location.getLongitude();

            if (geocoder != null) {

                try {

                    List<Address> addresses =
geocoder.getFromLocation(latitudeValue, longitudeValue, 1);

                    if (addresses != null && addresses.size() > 0) {

                        Address address = addresses.get(0);

                        town.setText(address.getLocality());

                        enableUIButtons();

                    }

                }

            }

        }

    }
```

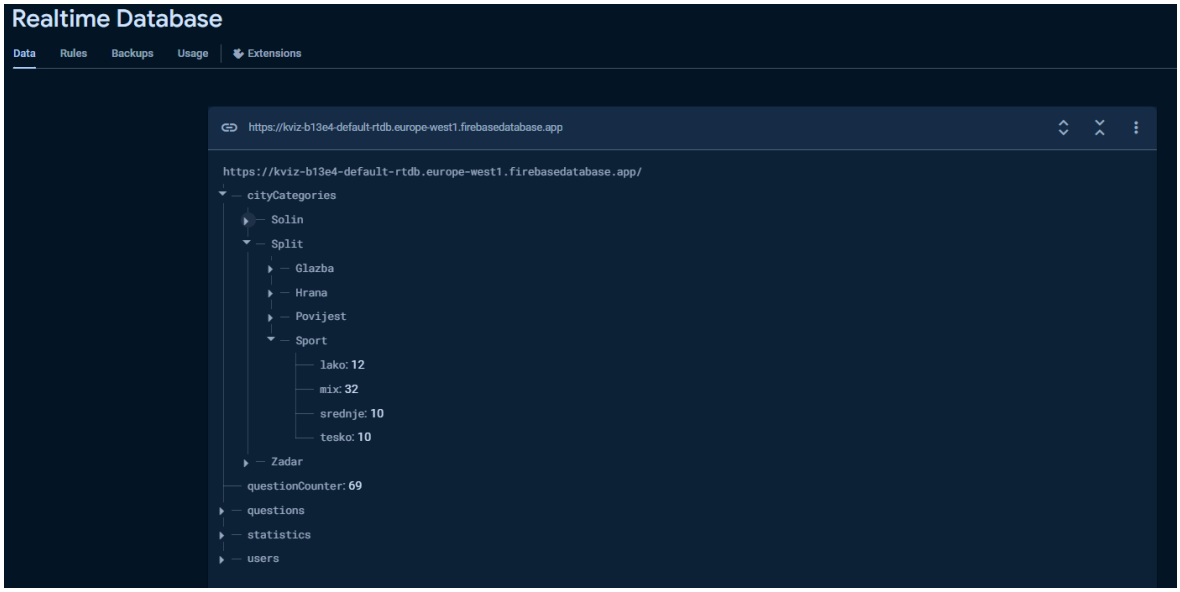
Ispis 5. Izdvajanje grada iz dohvaćene lokacije i ispis istog na zaslon

```
public void setupLocationTracking() {

        if (ActivityCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&

                ActivityCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

            return;

        }

        LocationRequest locationRequest = new LocationRequest.Builder(10000)

                .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY)

                .build();

        LocationCallback locationCallback = new LocationCallback() {

            @Override

            public void onLocationResult(LocationResult locationResult) {

                if (locationResult != null) {

                    Location location = locationResult.getLastLocation();

                    if (location != null) {

                        setLocation(location);

                    }

                }

            }

            @Override

            public void onLocationAvailability(LocationAvailability
locationAvailability) {

                boolean isLocationAvailable =
locationAvailability.isLocationAvailable();

            }

        };

        fusedLocationProviderClient.requestLocationUpdates(locationRequest,
locationCallback, Looper.getMainLooper());

    }
```

Ispis 6. Postavljanje praćenja lokacije, tj. ažuriranje trenutne lokacije uređaja

4.3 Baza podataka u stvarnom vremenu

Aplikacija lokacijskog kviza za pohranu i obradu podataka koristi Firebase bazu podataka u stvarnom vremenu. Radi se o bazi podataka baziranoj na tehnologiji oblaka koja pohranjuje i sinkronizira podatke u stvarnom vremenu među korisnicima. Još jednom je važno istaknuti da se radi o NoSQL bazi podataka te umjesto tradicionalnog relacijskog modela kao što su tablice i stupci koriste se različite metode pohrane i obrade podataka kao što su na primjer ključ-vrijednost parovi koji se koriste u ovom projektu. Podaci se u ovaj tip baze podataka spremaju u obliku JSON stabla, što omogućava fleksibilno i hijerarhijsko strukturiranje podataka, primjer za lokacijski kviz prikazan je na Slici 22.



Slika 22. Struktura podataka u Firebase bazi podataka u stvarnom vremenu

Tijekom korištenja aplikacije više se puta aplikacija povezuje s bazom podataka i pohranjuje ili dohvaća podatke. Neki od primjera rada aplikacije s bazom podataka su:

- dohvaćanje pitanja kod pokretanja nove igre
- pohrana svakog rezultata po završetku igre (Ispis 7.)
- unos novog pitanja u bazu podataka putem aplikacije (Ispis 8.)
- dohvaćanje rezultata i statistike kvizova.

```
public void addNewStatistics() {

        UserStatistic newData = new UserStatistic(userEmail, correctAnswers,
category, city, difficulty);



        String newKey = databaseReferenceStatistics.push().getKey();



        databaseReferenceStatistics.child(newKey).setValue(newData)

                .addOnSuccessListener(aVoid -> System.out.println("Data added
successfully"))

                .addOnFailureListener(e -> System.out.println("Error adding
data: " + e.getMessage()));

    }
```

Ispis 7. Pohrana rezultata u bazu podataka po završetku igre

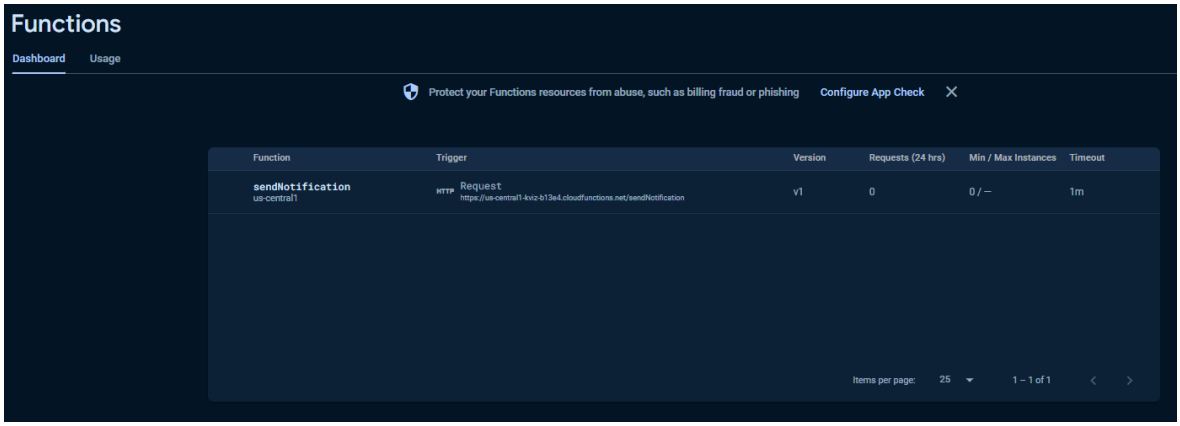
```
private void saveQuestionToFirebase(String question, List<String> options,
String answer, String type, String category, String city) {
    DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference();

    String difficultyTypeMapping = mappings.get(type);

    checkAndCreateCityCategory(city, category, databaseReference, () -> {
        incrementDifficultyMapping(question, options, answer, type, category,
city, databaseReference, difficultyTypeMapping);
        incrementMix(category, city, databaseReference);
    });
}

private void checkAndCreateCityCategory(String city, String category,
DatabaseReference databaseReference, Runnable callback) {
    DatabaseReference cityRef =
databaseReference.child("cityCategories").child(city);

    cityRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot snapshot) {
            if (!snapshot.hasChild(category)) {
                cityRef.child(category).setValue(0)
                        .addOnSuccessListener(aVoid -> {
                            Log.d("Firebase", "Category initialized
successfully!");
                            callback.run();
                        })
                        .addOnFailureListener(e -> Log.e("Firebase", "Failed to
initialize category", e));
            } else {
                callback.run();
            }
        }
```

```
        @Override
        public void onCancelled(DatabaseError databaseError) {
            Log.e("Firebase", "Failed to fetch data!",
databaseError.toException());
        }
    });
}
```

Ispis 8. Pohranjivanje novog pitanja u bazu podataka putem aplikacije

4.4 Notifikacija

U kôdu je definirano da se notifikacija šalje svim korisnicima aplikacije kada se novo pitanje doda u bazu podataka kroz aplikaciju. Za dinamičko slanje notifikacija korištena je Firebase značajka Cloud Functions (Slika 23.).



Slika 23. Firebase Cloud Functions za slanje notifikacija

Princip rada ove funkcije je takav da prvo aplikacija skuplja potrebne podatke te ih šalje prema Firebase Cloud Functionu. Ova funkcija se poziva asinkrono, što znači da se aplikacija ne blokira dok čeka odgovor od poslužitelja. Podaci se prosljeđuju ovoj funkciji koja obrađuje zahtjev na poslužitelju. Zatim funkcija na poslužitelju obrađuje podatke i generira obavijest koju zatim šalje putem Firebase Cloud Messaginga (FCM) na ciljanog korisnika, u ovom slučaju sve korisnike aplikacije. Ako je proces uspješno završen,

aplikacija može prikazati obavijest ili obraditi povratne podatke. Kod za navedene radnje prikazan je na Ispisu 9.

```
private Task<String> addMessage(String text, String category) {
        Map<String, Object> data = new HashMap<>();
        data.put("text", text);
        data.put("push", true);
        data.put("category", category);
        return mFunctions
                .getHttpsCallable("sendNotification")
                .call(data)
                .continueWith(new Continuation<HttpsCallableResult, String>() {
                    @Override
                    public String then(@NonNull Task<HttpsCallableResult> task)
throws Exception {
                        if (!task.isSuccessful()) {
                            Exception e = task.getException();
                            if (e != null) {
                                Log.e("Firebase", "sendNotification failed",
e);
                                throw e;
                            }
                        }
                        HttpsCallableResult result = task.getResult();
                        if (result != null) {
                            return (String) result.getData();
                        } else {
                            Log.e("Firebase", "sendNotification result is
null");
                            return null;
                        }
                    }
                });
    }
```

Ispis 9. Primjer koda za aktiviranje procesa slanja notifikacije

4.5 Reciklirajući prikaz u Android Studiju

Reciklirajući prikaz je napredni i fleksibilni *widget* u Androidu koji se koristi za prikazivanje velikih skupova podataka u obliku liste ili mreže. Budući da se samo vidljivi pogledi drže u memoriji, performanse su bolje čak i kada se radi o tisućama stavki. Nasljednik je zastarjelog ListViewa i GridViewa te donosi mnoge prednosti u performansama i fleksibilnosti [10].

Da bi se u aplikaciji dobilo dinamičku listu s podacima, potrebno je definirati nekoliko stvari u kôdu [11]:

- **`ViewHolder`** – svaki element u listi definiran je kao jedan objekt koji nije vezan ni za jedan podatak (Ispis 10.)
- **`Adapter`** – klasa koja povezuje podatke s ViewHolderom (Ispis 11.)
- **`Layout Manager`** – u glavnoj aktivnosti (eng. *Main Activity*) definirati prikaz podataka, a on može biti jednodimenzionalan i dvodimenzionalan (Ispis 12.)
- Aktiviranje ViewHoldera u glavnoj aktivnosti (Ispis 12.)

Reciklirajući prikaz se u ovoj aplikaciji koristi 2 puta:

- za ispis rezultata
- za ispis statistike.

```
public class ResultsViewHolder extends RecyclerView.ViewHolder {
    TextView textView;
    public ResultsViewHolder(@NonNull View itemView) {
        super(itemView);
        textView = itemView.findViewById(R.id.textView5);
    }
}
```

Ispis 10. ViewHolder klasa

```
public class ResultsAdapter extends RecyclerView.Adapter<ResultsViewHolder> {
    List<UserStatistic> userResults;
    Context context;

    public ResultsAdapter(List<UserStatistic> userResults, Context context) {
        this.userResults = userResults;
        this.context = context;
    }


    @NonNull
    @Override
    public ResultsViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        return new
ResultsViewHolder(LayoutInflater.from(context).inflate(R.layout.view_holder,par
ent,false));
    }

    @Override
    public void onBindViewHolder(@NonNull ResultsViewHolder holder, int
position) {
        String email = userResults.get(position).getEmail();
        String result =
String.valueOf(userResults.get(position).getUserResult());
        String city = String.valueOf(userResults.get(position).getCity());
        String category =
String.valueOf(userResults.get(position).getCategory());
        String difficulty =
String.valueOf(userResults.get(position).getDifficulty());

        holder.textView.setText(String.valueOf(email + " - " + city + " - " +
category + " - " + difficulty + " - " + result));
    }

    @Override
    public int getItemCount() {
        return userResults.size();
    }
}
```

Ispis 11. Adapter klasa

```
recyclerView.setLayoutManager(new LinearLayoutManager(Leaderboard.this));
initializeRecyclerView();

private void initializeRecyclerView() {
    ResultsAdapter adapter = new ResultsAdapter(new ArrayList<>(),
getApplicationContext());
    recyclerView.setAdapter(adapter);
}
```

Ispis 12. Definiranje prikaza i aktiviranje reciklirajućeg prikaza

Reciklirajući prikaz ima još neke prednosti kao što je podrška za animacije i dekoracije koristeći `ItemDecoration` klase. Ova klasa omogućava dodavanje prilagođenih animacija za dodavanje, uklanjanje i pomicanje stavki. Također je vrlo prilagodljiv i može se proširiti kako bi podržao različite vrste interakcija kao što su povlačenje stavki (eng. *drag & drop*), povlačenje za osvježavanje (eng. *swipe to refresh*), i povlačenje za brisanje (eng. *swipe to delete*) [12].

Za reciklirajući prikaz može se reći da je postao standard u prikazu lista podataka za moderne Androidove aplikacije zbog svoje već spomenute fleksibilnosti, učinkovitosti i mogućnosti prilagodbe.

5. Zaključak

Lokacijska aplikacija kviza predstavlja inovativan način angažiranja korisnika kroz interaktivnu kombinaciju geolokacijskih usluga i izazovnih pitanja. Korištenjem tehnologija za određivanje lokacije, aplikacija omogućava korisnicima da istražuju okolinu, sudjeluju u kvizovima na specifičnim lokacijama te time povezuju učenje i zabavu na dinamičan i prilagođen način. Ova aplikacija ne samo da potiče korisnike na fizičku aktivnost i otkrivanje novih mjesta, već također obogaćuje njihovo znanje na zanimljiv i interaktivan način, čineći iskustvo učenja zanimljivijim i dostupnijim.

Android Studio se pokazao kao vrlo učinkovito razvojno okruženje, omogućujući jednostavnu integraciju različitih funkcionalnosti te brzu iteraciju kroz faze razvoja korisničkog sučelja. S druge strane, Firebase je bio ključan za backend podršku, pružajući pouzdano rješenje za autentifikaciju korisnika, upravljanje bazom podataka i pohranu rezultata kviza. Kombinacija ovih alata omogućila je bržu i jednostavniju implementaciju složenih funkcionalnosti poput geolokacije i dinamičkog prilagođavanja sadržaja, što je značajno unaprijedilo korisničko iskustvo.

Kako bi se aplikacija dodatno unaprijedila, neke od ideja za daljnji razvoj su uvođenje pomoći tijekom igre (eng. *jokera*), dodavanje novih tema koje ne moraju biti vezane isključivo za grad u kojem se korisnik nalazi, sustav nagrađivanja za ostvarene rezultat poput znački i slično.

Literatura

[1] Wikipedija, „Android (operacijski sustav),“ Android (operating system) - Wikipedia (posjećeno 11.8.2024.)

[2] Medium, „Understanding the Android Operating System Architecture: A Detailed Breakdown,“ Understanding the Android Operating System Architecture: A Detailed Breakdown | by Rashik | Medium (posjećeno 11.8.2024.)

[3] Java Point, „Android studio,“ Android Studio - javatpoint (posjećeno 14.8.2024)

[4] Developers, „Meet Android Studio,“ Meet Android Studio | Android Developers (posjećeno 14.8.2024.)

[5] Geeks for geeks, „What is Android Studio?,“ What is Android Studio? - GeeksforGeeks (posjećeno 14.8.2024.)

[6] Wikipedija, „Firebase,“ Firebase - Wikipedia (posjećeno 18.8.2024.)

[7] Geeks for geeks, „Firebase - Introduction,“ Firebase - Introduction - GeeksforGeeks (posjećeno 18.8.2024.)

[8] Tech Target, „Google Firebase,“ What is Google Firebase? (techtarget.com) (posjećeno 18.8.2024.)

[9] Firebase, „Firebase Authentication,“ Firebase Authentication (google.com) (posjećeno 19.8.2024.)

[10] Geeks for geeks, „RecyclerView in Android with Example,“ RecyclerView in Android with Example - GeeksforGeeks (posjećeno 23.8.2024.)

[11] Developers, „Create dynamic lists with RecyclerView,“ Create dynamic lists with RecyclerView | Views | Android Developers (posjećeno 23.8.2024.)

[12] Medium, „RecyclerView.ItemDecoration: Making the Most of It,“ https://medium.com/surf-dev/recyclerview-itemdecoration-making-the-most-of-it-41d34f74c948 (posjećeno 24.8.2024.).