Android aplikacija za ograniziranje neformalnih sportskih susreta

Lovreković, Tomislav

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

Sveučilišni studij

Android aplikacija za organiziranje neformalnih sportskih susreta

Završni rad

Tomislav Lovreković

Osijek, 2022.

SADRŽAJ

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 19.09.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Tomislav Lovreković
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R 4384, 22.07.2019.
OIB Pristupnika:	02443134203
Mentor:	Izv.prof.dr.sc. Zdravko Krpić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Android aplikacija za ograniziranje neformalnih sportskih susreta
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	Zadatak završnog rada je napraviti mobilnu aplikaciju za Android OS koja omogućuje organiziranje neformalnih sportskih susreta u svrhu druženja i socijaliziranja. Aplikacija treba omogućiti prijavu na istu te unos i pregled događaja. Za svaki događaj inicijator treba moći unijeti potrebne podatke, dok ostali korisnici mogu potvrditi dolazak na isti ili odbiti ga. Aplikacija treba omogućiti prikaz događaja na karti, gdje klikom na isti se dobiju sve potrebne informacije o događaju kao i o popisu ljudi koji su najavili dolazak. U teorijskom dijelu rada treba istražiti ista ili slična rješenja, predstaviti ih i na osnovu njihovih karakteristika definirati zahjeve na programsko rješenje (aplikaciju). Na osnovu zahtjeva odabrati tehnologije za izradu navedenog rješenja, konstruirati arhitekturalni model aplikacije te opisati osnovne značajke potrebne za implementaciju iste. Naposljetku istražiti i ocijeniti korisničko iskustvo aplikacije na skupini korisnika iste. Tema rezervirana za: Tomislav Lovreković
Prijedlog ocjene završnog rada:	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene od strane mentora:	19.09.2022.
Datum potvrde ocjene od strane Odbora:	22.09.2022.

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

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 22.09.2022.

Ime i prezime studenta:	Tomislav Lovreković
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R 4384, 22.07.2019.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija za ograniziranje neformalnih sportskih susreta**

izrađen pod vodstvom mentora Izv.prof.dr.sc. Zdravko Krpić

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:

1. UVOD

Socijaliziranje je proces kojim osoba uči postati dijelom grupe, zajednice ili društva. Svrha socijaliziranja je upoznavanje i prihvaćanje novih članova u društvo ali služi i poticanju osobe na daljnje druženje i napredak u zajednici. U različitim razmjerima ljudi su iskusili usamljenost i društvenu izolaciju tijekom pandemije, sve to je također utjecalo i na sportske aktivnosti. U sadašnjosti prilika i mogućnost jednostavnog stupanja u kontakt s ljudima i organizacija druženja lako je dostupna. Svemu tome su zaslužne društvene mreže i mobilni uređaji, prvenstveno zbog mogućnosti kretanja i ostajanja u dodiru uz razvoj interesa i pronalazak ljudi koji dijele te iste interese.

Cilj ovog završnog rada je definirati, dizajnirati i izraditi mobilnu aplikaciju za organiziranje neformalnih sportskih susreta, s obzirom da postojeće aplikacije ne omogućuju jednostavno stvaranje i pregled sportskih susreta, kao ni stvaranje korisničkog računa bez novčane naknade. Osim toga, rijetke postojeće aplikacije su prilagođene našem području. S obzirom na navedeno u aplikaciji izrađenoj u okviru ovog rada omogućiti će se stvaranje i uređivanje korisničkog računa te stvaranje i pregled sportskih susreta. Provođenjem ankete biti će ispitano korisničko iskustvo s funkcionalnostima navedene aplikacije.

U drugom poglavlju prikazana su detaljno analizirana programska rješenja te njihove prednosti i nedostatci na osnovu kojih je izrađena aplikacija u ovom radu. U trećem poglavlju razrađeni su zahtjevi na programsko rješenje, te su na osnovu funkcionalnosti postojećih rješenja opisane funkcionalnosti registracije i prijave te stvaranja i pregleda događaja. Zatim su, u četvrtom poglavlju, na osnovu zahtjeva na programsko rješenje predstavljene tehnologije koje su upotrijebljene za izradu programskog rješenja te arhitekturni dizajn aplikacije temeljen na jednom od arhitekturnih obrazaca. U posljednjem poglavlju prikazana je implementacija programskog rješenja te korisničko iskustvo, njihovi utisci te potencijalne nadogradnje i nedostatci.

1.1. Zadatak završnog rada

Zadatak završnog rada je napraviti mobilnu aplikaciju za Android OS koja omogućuje organiziranje neformalnih sportskih susreta u svrhu druženja i socijaliziranja. Aplikacija treba omogućiti prijavu na istu te unos i pregled događaja. Potrebni podaci za svaki događaj unose se od strane inicijatora, a potvrdu dolaska na određeni događaj šalju ostali korisnici. Aplikacija treba omogućiti prikaz događaja na karti, gdje se klikom na isti dobiju sve potrebne informacije o

događaju kao i o popisu ljudi koji najavljuju dolazak. U teorijskom dijelu rada istražena su ista ili slična rješenja, predstavljena su, te su na osnovu njihovih karakteristika definirani zahtjevi na programsko rješenje (aplikaciju). Na osnovu zahtjeva odabrane su tehnologije za izradu navedenog rješenja, konstruiran arhitekturni model aplikacije te opisane osnovne značajke potrebne za implementaciju iste. Naposljetku potrebno je istražiti i ocijeniti korisničko iskustvo aplikacije na skupini korisnika.

2. PREGLED POSTOJEĆIH PROGRAMSKIH RJEŠENJA

U ovom dijelu završnog rada će biti prikazano područje rada koje je služilo kao inspiracija u stvaranju programskog rješenja. Odabrano je pet sličnih programskih rješenja i predstavljene njihove karakteristike i osobne preferencije na osnovu kojih su postavljeni zahtjevi na programsko rješenje. Karakteristike ključne za ostvarivanje programskog rješenja mogu se podijeliti u sljedeće kategorije:

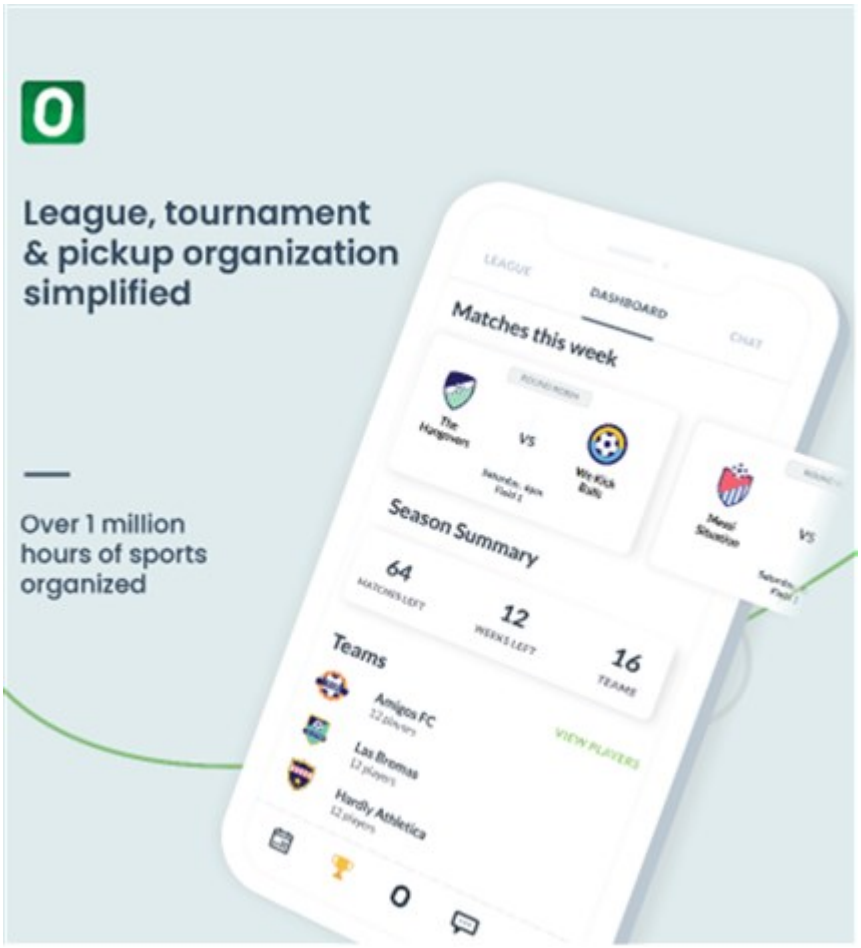
- Interakcija među korisnicima – kao što su chat, to jest izmjena poruka te stvaranje grupa ljudi za igru.
- Personalizacija – registracija korisničkog računa, dodavanje korisničkog imena te slike.
- Uređivanje i izmjena spremljenih podataka – nadzor prisutnosti, broj ljudi koji će se pojaviti na susretu te njihova imena, zatim uređivanje prisutnosti, mogućnost dodavanja i uklanjanja nepoželjnih korisnika.
- Preglednost i jednostavnost – kalendar/podsjetnik, jednostavna pretraga stvorenih susreta, karta s markerima te opis lokacije (točna adresa).

2.1. OpenSports

OpenSports aplikacija omogućuje vođenje tjednih, pa čak i dnevnih događaja bez dugotrajnog organizacijskog opterećenja ručnog slanja pozivnica putem više platformi, prikupljanja plaćanja, lista čekanja, itd. OpenSports smanjuje pojavu nesuglasica oko plaćanja termina i situacija kao što su nedolazak na termin. Neke od značajki aplikacije su obavijesti o nadolazećim igrama te jednostavno pridruživanje postojećim igrama putem aplikacije ili web-mjesta.

Aplikacija OpenSports pruža značajke kao što su:

- obavijesti
- nadzor prisutnosti
- uređivanje prisutnosti
- kalendar/podsjetnik
- registracija korisničkog računa
- chat/dopisivanje
- stvaranje ekipe/klana
- rezervacija i plaćanje terena/dvorane
- stvaranje liga i detaljno praćenje svih rezultata



Sl. 2.2.1. prikaz korisničkog sučelja OpenSports

Aplikaciju OpenSports ističe čist dizajn i najveću korisnost predstavlja vlasnicima sportskih klubova i grupa, administratorima i upraviteljima nekih sportskih objekata te organizatorima sportskih organizacija. Neki od nedostataka ove aplikacije su mjesečna naknada od dvadeset dolara te loša korisnička podrška. Unatoč tome ovo programsko rješenje veoma dobro rješava problem prisutnosti igrača na terenu, to jest daje na uvid koliko ljudi dolazi na susret. Inspirirano tim, u ovom se radu koristi sličan pristup popisu igrača.

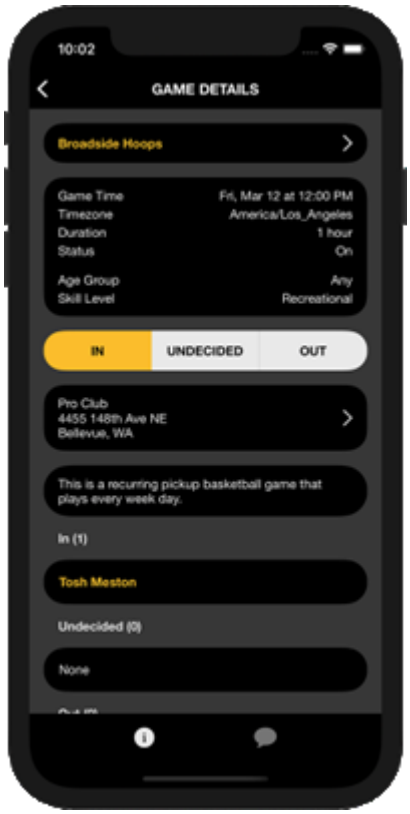
2.2. InfiniteHoops

Aplikacija InfiniteHoops locira obližnje košarkaške utakmice i olakšava pretragu skupina korisnika koji su spremni za igru uz prikaz mjesta i popunjenosti košarkaškog terena.

Aplikacija InfiniteHoops pruža značajke kao što su:

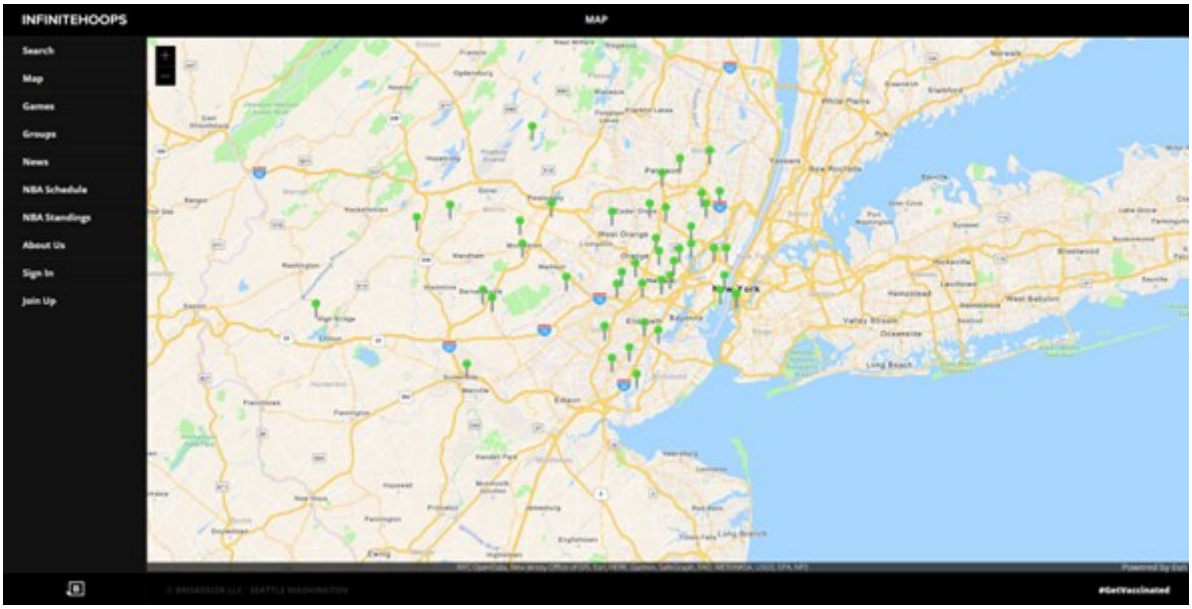
- pretraga obližnjih košarkaških utakmica
- pretraga grupa za igranje košarke

- pregled nadolazećih dogovorenih susreta
- mogućnost oznake dolazim/ne dolazim
- pregled tko dolazi na organizirane susrete
- pregled susreta na karti
- registracija korisničkog računa
- chat
- pregled vijesti u svijetu sporta



Sl. 2.2.2. prikaz korisničkog sučelja InfiniteHoops

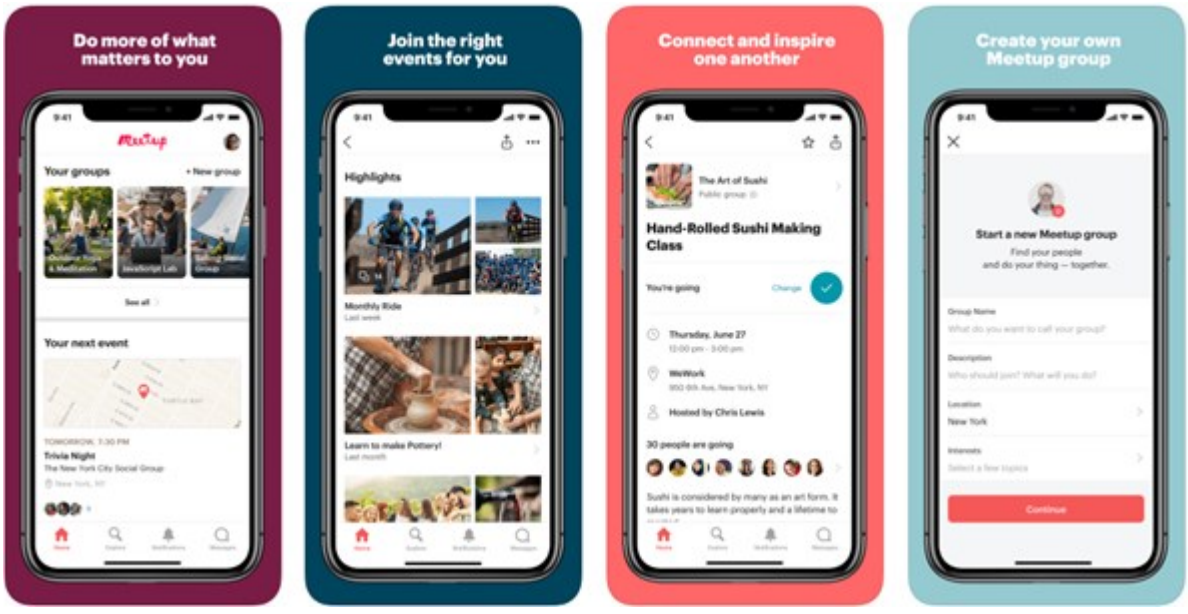
Aplikacija InfiniteHoops pristupačna je i laka za korištenje, sadrži prikaz susreta na karti. Susreti su označeni markerima i klikom na marker korisnik dobiva informaciju o broju igrača na terenu. Dodana je mogućnost stvaranja grupa kojima korisnik može poslati pozivnicu na susret bez da ga napravi javnim. Uz prikaz i iniciranje susreta ova aplikacija sadrži novosti te košarkaške rezultate. Aplikaciju nedostaju neke mogućnosti: obavijesti, ne sadrži točne podatke o lokaciji te nema traku za pretraživanje (engl. *search bar*) za pretragu sportskih susreta. Ova aplikacija izabrana je isključivo zbog svog sučelja i prikaza događaja na karti, u vlastitom programskom rješenju implementirati će se karta te markeri slični kao i u ovoj aplikaciji.



Sl. 2.2.2. prikaz web stranice InfiniteHoops i značajke karte

2.3. Meetup

Meetup je aplikacija za pronalaženje i. povezivanje ljudi. Sadrži značajke slične društvenim mrežama i osim za sport koristi se za druge oblike susreta. Aplikacija koristi preciznu lokaciju uređaja (na temelju GPS-a i mreže) za preporuku događaja Meetup-a koji su u blizini.



Sl. 2.3.1. prikaz dizajna Meetup aplikacije

Aplikacija Meetup pruža značajke kao što su:

- registracija i prijava
- prikaz događaja na karti

- stvaranje i upravljanje događajima
- grupna i pojedinačna komunikacija
- postavljanje slika s događaja
- dodavanje oznaka na fotografije
- personalizirani osobni računi
- različiti filteri interesa i tema
- pretraživanje i dodavanje prijatelja

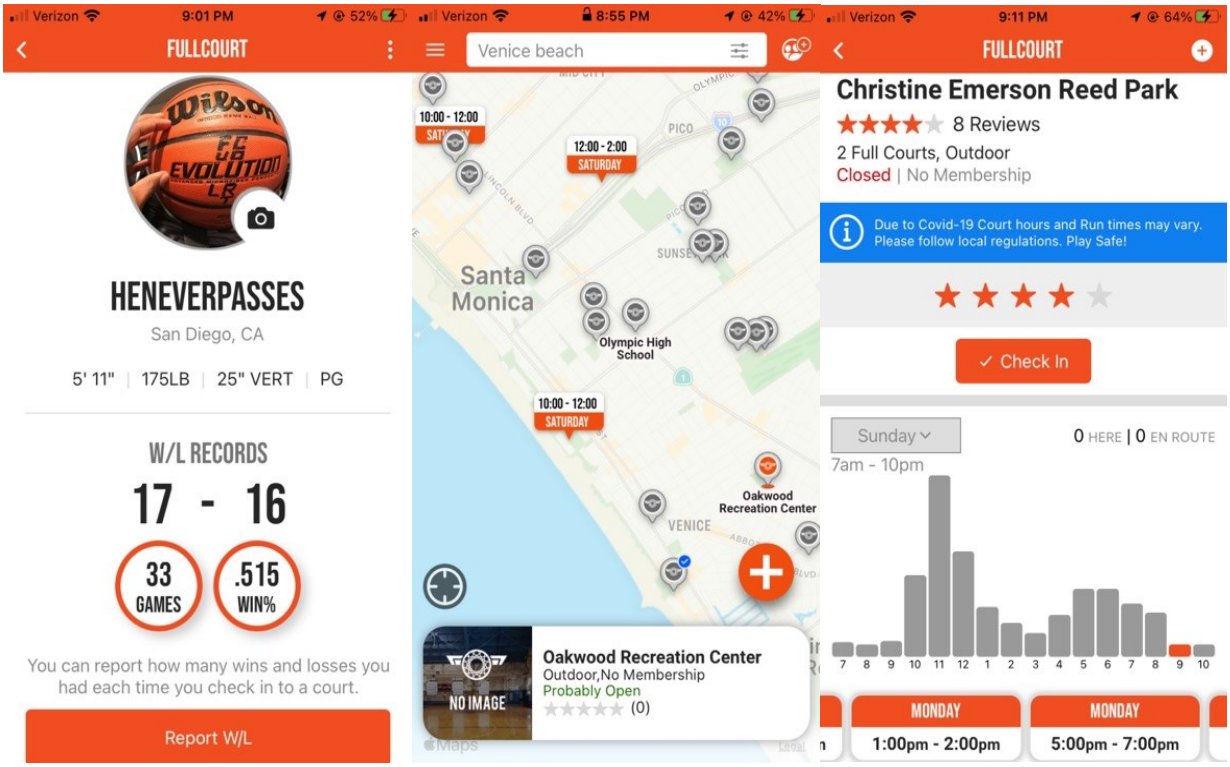
Aplikacija pruža pronalazak lokalnih događaja i grupa ljudi. Moguće je pretraživanje događaja i grupa pretraživanjem određene ključne riječi, kategorije ili vidjeti što je popularno u određenim područjima. Ako korisnik putuje s jednog mjesta na drugo, aplikacija ima opciju promjene lokacije što je prijeko potrebno i korisno. Iako Meetup pruža puno funkcionalnosti, često zamrzavanje ekrana i nedostatak tehničke podrške uz beskonačno mnogo reklama kvari cjelokupno iskustvo. Aplikacija Meetup spomenuta je zbog svoje mogućnosti uređivanja osobnog računa, jer prikaz osobnih podataka i slike čini veliku prednost kod aplikacija za socijaliziranje.

2.4. Fullcourt

Fullcourt je aplikacija koja omogućuje korisnicima prikaz korisnih informacija vezanih uz košarkaške susrete. Ona predstavlja mjesto za dijeljenje podataka potrebnih za organiziranje susreta.

Aplikacija Fullcourt pruža značajke kao što su:

- registracija i prijava
- prikaz lokacija na karti
- opis lokacija
- prikaz broja igrača na terenu
- grafovi kojima se prikazuje promet na terenu
- mogućnost postavljanja slike terena
- tjedni prikaz najčešće posjećenosti
- mogućnost pozivanja na teren
- prikaz svih tekućih utakmica u popisu



Sl. 2.4.1. prikaz dizajna Fullcourt aplikacije

Ovo programsko rješenje na najjednostavniji način ispunjava sve potrebe za dogovaranje i odabir igrališta za košarkaški susret. Grafovima daje uvid u prosječnu posjećenost terena po satu, što pomaže pri odabiru ovisno želi li korisnik igrati sam ili u društvu. Korisnički računi sadrže informacije o korisniku te broj dobivenih i izgubljenih utakmica te postotak pobjeda. Za označena mjesta susreta može se objaviti recenzija. Kod Fullcourt aplikacije najviše se ističe prikaz svih susreta u tijeku, uz nedostatak popisa već zakazanih susreta. Dodavanjem dijela „zakazani susreti“ u aplikaciji izrađenoj u okviru ovog rada bit će nadopunjena ova karakteristika.

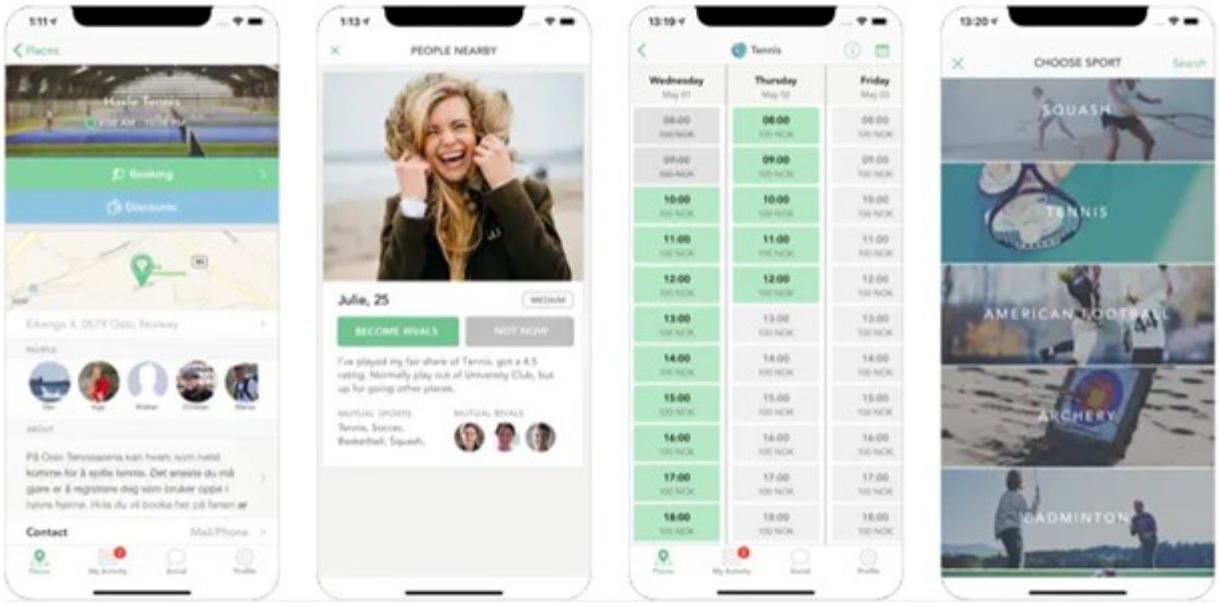
2.5. Sporty

Aplikacija Sporty pruža istraživanje aktivnosti i pretraživanje ljudi u okolici, također predstavlja sportsku platformu za ustanove i prostore nudeći pretplate, rezervacije i automatizirano plaćanje. Detaljnom statistikom pomaže korisniku pri vizualnom uspoređivanju podataka.

Aplikacija Sporty pruža značajke kao što su:

- registracija i prijava
- prikaz događaja na karti
- rezervacija karata
- rezervacija terena

- unajmljivanje trenera
- pametni organizator tečajeva
- sučelje za trenera
- razgovori
- lista čekanja
- automatizirano plaćanje
- detaljna statistika



Sl. 2.5.1. prikaz dizajna Sporty aplikacije

Sporty nudi širok raspon sportskih aktivnosti, ne samo rekreativni događaji nego i tečajevi te treninzi. Jedna od zanimljivijih značajka je lista čekanja koja pravi popis i naknadno obavještava ako je mjesto oslobođeno. Aplikacija nudi organizaciju događaja od stvaranja do rezervacije i plaćanja, široka ponuda i čista izvedba. Nedostatak ovog programskog rješenja je učestali pad servera te nedostupnost u određenim regijama, korisnici su većinom iz Skandinavskih država. Sporty programsko rješenje spomenuto je zbog svog čistog dizajna i sučelja koje dodaje važnosti aplikacije. Dobra praksa je imati čisto korisničko sučelje (engl. *user interface*, UI)i smanjiti prikaz suvišnih informacija.

3. ZAHTJEVI NA PROGRAMSKO RJEŠENJE

U ovom poglavlju završnog rada na osnovu postojećih rješenja objašnjeno je što će programsko rješenje imati od funkcionalnosti. Jasno definirani zahtjevi prvi su korak do uspješnog projekta. Zahtjevi na programsko rješenje opisuju funkcionalnosti koje proizvod mora imati da bi ispunio očekivanja korisnika.

3.1. Registracija i prijava

Stranica za prijavu trebala bi biti prva stranica koju korisnici vide u aplikaciji. Trebala bi se sastojati od dva tekstualna polja, jedno za unos imena za prijavu i jedno za unos lozinke. Osim toga, trebala bi imati naredbeni gumb koji pokreće radnju provjere lozinke. Ako je bilo koje od tekstualnih polja ostavljeno praznim, to je greška koja se mora prijaviti korisniku. Ako su oba polja popunjena, ali nema zapisa o korisničkom imenu ili je lozinka netočna, to se također mora prijaviti korisniku. Korisnici koji se još nisu registrirali ne mogu se prijaviti. Prvo se moraju registrirati klikom na naredbeni gumb za registraciju. Oni bi to trebali moći učiniti bez dobivanja pogreške za prazno polje imena ili lozinke. Registracija korisničkog računa znači da se korisnici mogu jedinstveno identificirati unutar aplikacije. Za korisnike to znači da mogu vidjeti vlastiti profil i podatke i promijeniti te iste podatke.

Zahtjev za registraciju korisnika trebao bi sadržavati:

1. Pojedinosti o identifikaciji (ID, e-mail adresa)
2. Podaci o provjeri identiteta (zaporka)
3. Činjenice o korisniku (spol)

3.2. Pregled događaja

Korisnik nakon uspješne prijave mora moći imati pregled svih stvorenih događaja. Kod izabiranja specifičnog događaja može odabrati prikaz na karti, koji je intuitivniji i pruža lakši pregled udaljenosti. Također korisnik može odabrati prikaz događaja u popisu te otkriti više informacija bez otvaranja samog događaja. Također korisnik ima dostupnu traku za pretraživanje u koju može upisati lokaciju te dobiti popis svih sportskih događaja organiziranih na toj lokaciji. Pregled događaja omogućuje lakše izmjenjivanje informacija, u samo nekoliko dodira korisnik može pronaći što traži. Ovo je bitan dio aplikacije jer sadrži sve sastavne dijelove susreta, kao što su adresa, broj igrača itd.

3.3. Stvaranje događaja i povratna informacija

Korisnik mora moći stvoriti sportski događaj kao i potvrditi svoj dolazak na isti. Inicijator uz priložene informacije stvara događaj koji se unosi u bazu podataka te postaje vidljiv svim korisnicima. Korisnici tada izabiru koje događaje žele posjetiti te potvrđuju svoj dolazak. Korisnik mora moći potvrditi više događaja i pri dolasku naznačiti da je tamo. Informacije koje inicijator unosi prilikom stvaranja sportskog događaja su adresa susreta te koliko igrača je potrebno. Za aplikaciju je bitna tvorba događaja jer korisnici stvaranjem različitih susreta dijele i za uzvrat primaju potrebne informacije za ostvarivanje susreta. Ovaj dio aplikacije mora biti najintuitivniji.

4. ALATI I TEHNOLOGIJE

Programsko rješenje biti će napisano u programskom jeziku Kotlin te razvijeno u razvojnom okruženju Android Studio. Neki od razloga korištenja navedenog programskog jezika su značajke kao što su: podatkovne klase te funkcionalni koncepti. Nadalje za stvaranje baze podataka te provjeru identiteta biti će korišteni Firebase Authentication te Cloud Firestore. Razlog korištenja ovih tehnologija je olakšana integracija i postavljanje te povećana sigurnost sustava. Uz navedeno implementirati će se ubrizgavanje ovisnosti (engl. *dependency injection*) korištenjem Koin programskog okvira i suspendiranje zadataka koji bi blokirali glavnu nit (engl. *thread*) korištenjem Kotlin Coroutines.

4.1. Android Studio

Android Studio službeno je integrirano razvojno okruženje (engl. *Integrated Development Envirnoment*, IDE) za razvoj Android aplikacija, temeljeno na IntelliJ IDEA integriranom razvojnom okruženju. Povrh IntelliJ-ovog uređivača koda i alata za razvojne programere, Android Studio nudi još više značajki koje poboljšavaju produktivnost pri izradi Android aplikacija, kao što su navedene u [1]:

- Fleksibilan sustav gradnje zasnovan na Gradle-u
- Brz i značajkama bogat emulator
- Jedinstveno okruženje u kojem se mogu razvijati aplikacije za sve Android uređaje
- Predlošci koda i GitHub integracija koji omogućuju izradu uobičajenih značajki aplikacija i uvoz uzoraka koda
- Opsežni alati i okviri za testiranje
- Lint alati za analiziranje performansi, upotrebljivosti, kompatibilnosti verzija i drugih problema
- Podrška za C++ i Android izvornog razvojnog kompleta (engl. Android Native Development Kit, NDK)
- Ugrađena podrška za Google Cloud Platform, što olakšava integraciju Google Cloud Messaging-a i App Engine-a

4.2. Kotlin

Prema [2] Kotlin je objektno orijentirani jezik za razvoj Android aplikacija koji ima puno funkcionalnih programskih elemenata, to jest pruža općenita rješenja za širok raspon problema (rekurzije, kompozicije funkcija itd.). Kotlinov sustav opisa očekivanog ponašanja i moguće tipove podatka (engl. *types*) u vrijeme prevođenja razlikuje tipove podatka s mogućnošću nulte vrijednosti i one koji nisu nulti, postižući *null-safety*, tj. jamčeći odsutnost pogrešaka uzrokovanih nedostatkom vrijednosti. Sa strane funkcionalnog programiranja, ima podršku za funkcije višeg reda i lambda izraze. Lambda izrazi omogućuju kreiranje anonimnih metoda te su definirani korištenjem operatora =>. Kod izbora programskog jezika izabran je Kotlin zbog njegovog funkcionalnog programiranja i prijašnjeg iskustva na kolegijima.

Prednosti pisanja koda u programskom jeziku Kotlin su:

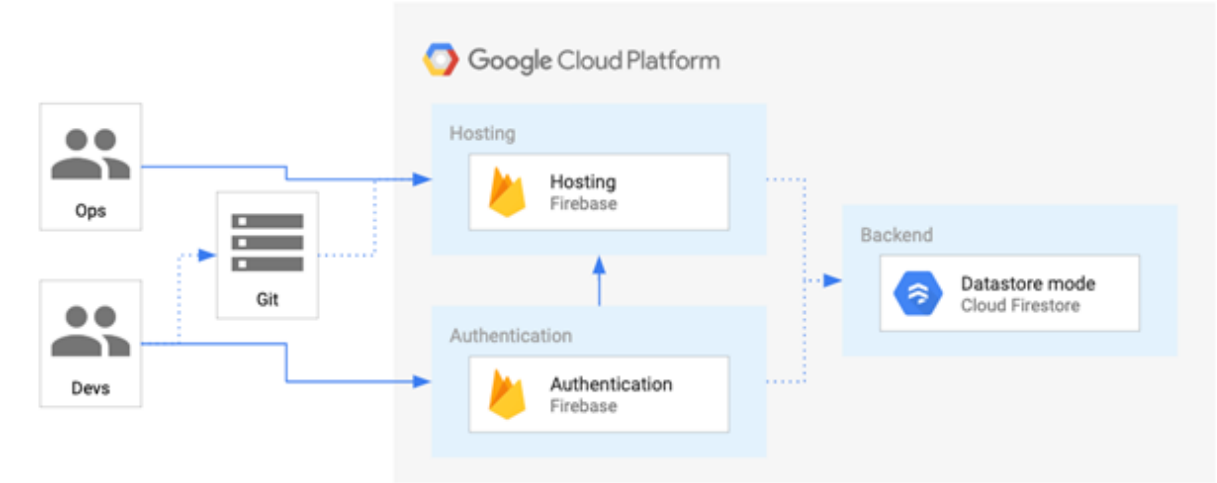
- Jednostavno usvajanje od strane razvojnih programera
- Potpuno je kompatibilan s Javom
- Ne nameće troškove izvođenja
- Nudi veću sigurnost u razvoju
- Pogodan je za razvoj na više platformi

4.3. Firebase

Firebase je pozadinski dio programa (engl. *backend*) kao usluga (engl. *Backend-as-a-Service* Baas), pruža programerima razne alate i usluge koji im pomažu u razvoju. Usluge Firebasea potrebne su u ovom radu kako bi se omogućila provjera identiteta korisnika te kako bi se izradila baza podataka. Razlog korištenja Firebase alata je mogućnost integracije jedne ili više načina prijave u aplikaciju te sinkroniziranje podataka u stvarnom vremenu i dostupnost izvan mreže.

4.3.1. Firebase Authentication

Većina aplikacija mora znati identitet korisnika. Poznavanje identiteta korisnika omogućuje aplikaciji da sigurno sprema korisničke podatke u oblak i pruža isto personalizirano iskustvo na svim uređajima korisnika. Firebase Authentication pruža pozadinske usluge, SDK-ove jednostavne za korištenje i gotove biblioteke korisničkog sučelja za provjeru identiteta korisnika u nekoj aplikaciji. Podržava provjeru identiteta pomoću lozinki, telefonskih brojeva, popularnih pružatelja kao što su Google, Facebook i Twitter i još mnogo toga. Firebase provjera identiteta se usko integrira s drugim Firebase uslugama i koristi industrijske standarde kao što su OAuth *2.0* i OpenID Connect, kao što je navedeno u [3]. Firebase Authentication koristi se zbog uštede vremena na razvijanju metoda za provjeru identiteta. Za pohranu korisničkih podataka nakon što se korisnik provjerava s Firebaseom potreban je poziv samo jedne ugrađene metode. Također, znatno se štedi vrijeme razvoja baze podataka budući da se može izbjeći razvoj metoda na strani poslužitelja za različite vrste verifikacije znakova (engl. *token*) u slučaju da je potrebno dodati prijavu putem društvenih mreža kao što su Facebook i Google.



Sl. 4.3.1.1. prikaz servisa koje nudi Google Cloud

4.3.2. Cloud Firestore

Cloud Firestore je NoSQL (engl. *not only Structured Query* Language) baza podataka smještena u oblaku kojoj Apple, Android i web aplikacije mogu pristupiti izravno putem izvornih SDK-ova. Cloud Firestore je također dostupan u izvornim Node.js, Java, Python, Unity, C++ i Go SDK-ovima, uz REST i RPC API-je (engl. *Application Programming Interface)*. Slijedeći NoSQL podatkovni model Cloud Firestorea, podaci se pohranjuju u dokumente koji sadrže polja koja se preslikavaju na vrijednosti. Ti dokumenti pohranjeni su u zbirkama, koji su spremnici korišteni za organiziranje podataka i izradu upita. Podržane su različite vrste podataka, od jednostavnih nizova i brojeva do složenih, ugniježđenih objekata kao što je opisano u [3]. Također omogućeno je stvaranje pod zbirki unutar dokumenata i izgradnja hijerarhijske strukture podataka koje se povećavaju kako baza podataka raste. Podatkovni model Cloud Firestore podržava bilo koju strukturu podataka koja najbolje funkcionira za aplikaciju. Ključne sposobnosti:

- Fleksibilnost podatkovnih struktura
- Precizno postavljanje upita
- Ažuriranje podataka u stvarnom vremenu
- Izvan mrežna podrška
- Može podnijeti velik promet

Prednost korištenja Cloud Firestorea (kao i Firebasea u cjelini) je činjenica da je veći dio posla obavljen bez delegiranja zahtjeva na dodatne usluge. Uklanja prisustvo dodatne usluge kako bi se osigurao sloj između klijenta i poslužitelja. Održavanje pozadine programa košta vremena i novca a to Cloud Firestore eliminira u većini slučajeva.

4.4. RecyclerView

RecyclerView je sučelni program (engl. *widget*) koji je fleksibilnija i naprednija verzija GridViewa i ListViewa, grafičkih elemenata koji predstavljaju tablični i prikaz podataka u listi. RecyclerView olakšava učinkovito prikazivanje velikih skupova podataka. Podaci su dostavljeni i izgled stavke je definiran, a biblioteka RecyclerView dinamički stvara elemente kada su potrebni. Kao što naziv implicira, RecyclerView reciklira te pojedinačne elemente. Kada se stavka pomakne sa zaslona, RecyclerView ne uništava njezin prikaz. Umjesto toga, RecyclerView ponovno koristi prikaz za nove stavke koje su se pomicale na zaslonu. Ova ponovna upotreba uvelike poboljšava performanse, poboljšava odziv aplikacije i smanjuje potrošnju energije, kao što je spomenuto u [4]. Pravilno postavljanje RecyclerViewa uvelike ovisi o implementaciji Adaptera i ViewHoldera.

Ove dvije klase rade zajedno kako bi prikaz podataka bio definiran. ViewHolder je omot oko pogleda koji sadrži izgled za pojedinačnu stavku na popisu. Adapter stvara ViewHolder objekte prema potrebi, a također postavlja podatke za te poglede. Proces povezivanja pogleda s njihovim podacima naziva se obvezivanje.

Postoje tri važne metode koje je potrebno prepisati (engl. *override*):

- onCreateViewHolder()*:* RecyclerView poziva ovu metodu kad god treba stvoriti novi ViewHolder. Metoda stvara i inicijalizira ViewHolder i pridruženi View, ali ne ispunjava sadržaj pogleda - ViewHolder još nije vezan za određene podatke.
- onBindViewHolder(): RecyclerView poziva ovu metodu da poveže ViewHolder s podacima. Metoda dohvaća odgovarajuće podatke i koristi podatke za popunjavanje izgleda nositelja pogleda. Na primjer, ako RecyclerView prikazuje popis imena, metoda bi mogla pronaći odgovarajuće ime na popisu i ispuniti *widget* TextView vlasnika pogleda.
- getItemCount(): RecyclerView poziva ovu metodu da dobije veličinu skupa podataka. Na primjer, u aplikaciji adresara to može biti ukupan broj adresa. RecyclerView to koristi da odredi kada više nema stavki koje se mogu prikazati.

4.5. Koin

Koin je okvir za ubrizgavanje ovisnosti u programskom jeziku Kotlin. Napisan je u potpunosti u Kotlinu, zbog čega je učinkovit i lagan te ima vrlo dobru podršku za Android. Injekcija ovisnosti je dizajn koji se koristi za implementaciju inverzne kontrole, tijek aplikacije je obrnut, što znači da umjesto korištenja fiksnog redoslijeda događaja, redoslijed kojim se upravlja podacima je kontroliran. Ovisni objekt može biti stvoren izvan klase i pružiti te objekte klasi na različite načine na način koji je opisan u [5].

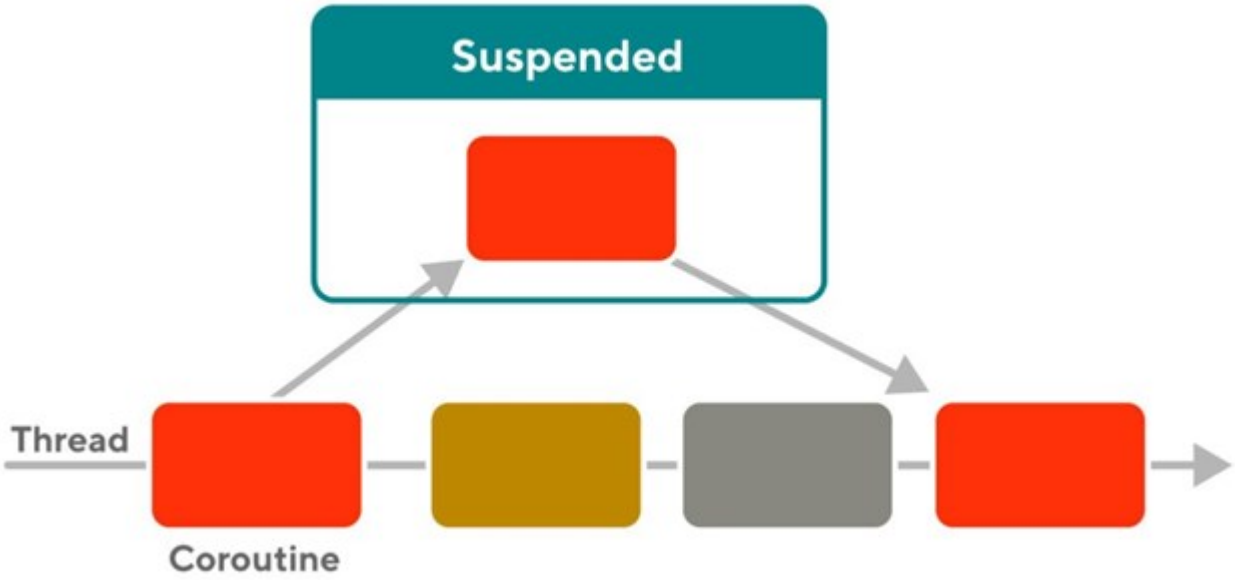
Ubrizgavanje ovisnosti povoljno je zbog sljedećih razloga:

- Smanjene uske povezanosti klasa
- Čitljiviji kod
- Lakša provjera koda
- Više koda koji se može „reciklirati“

4.6. Kotlin Coroutines

Asinkrono programiranje važan je dio razvojne okoline. Prilikom izrade aplikacija na strani poslužitelja, desktopa ili mobilnih aplikacija, ono pruža iskustvo koje nije samo fluidno iz

perspektive korisnika, već i skalabilno kada je to potrebno. Kotlin rješava ovaj problem na fleksibilan način pružanjem podrške za korutine (engl. *coroutines*) na razini jezika i delegiranjem većine funkcionalnosti bibliotekama kao što je navedeno u [6]. Na Androidu korutine pomažu u upravljanju dugotrajnim zadacima koji bi inače mogli blokirati glavnu nit i uzrokovati da aplikacija ne reagira.



Sl. 4.6.1. prikaz djelovanja Kotlin *schedulera* i raspoređivanje *korutina*

Na Androidu, korutine su rješenje za dva problema:

- Dugotrajni zadaci kojima je potrebno predugo da blokiraju glavnu nit.
- Main-safety omogućuje da bilo koja funkcija obustave (engl. *suspend function*) bude pozvana s glavne niti, funkcija obustave je funkcija koja se može pauzirati i nastaviti kasnije

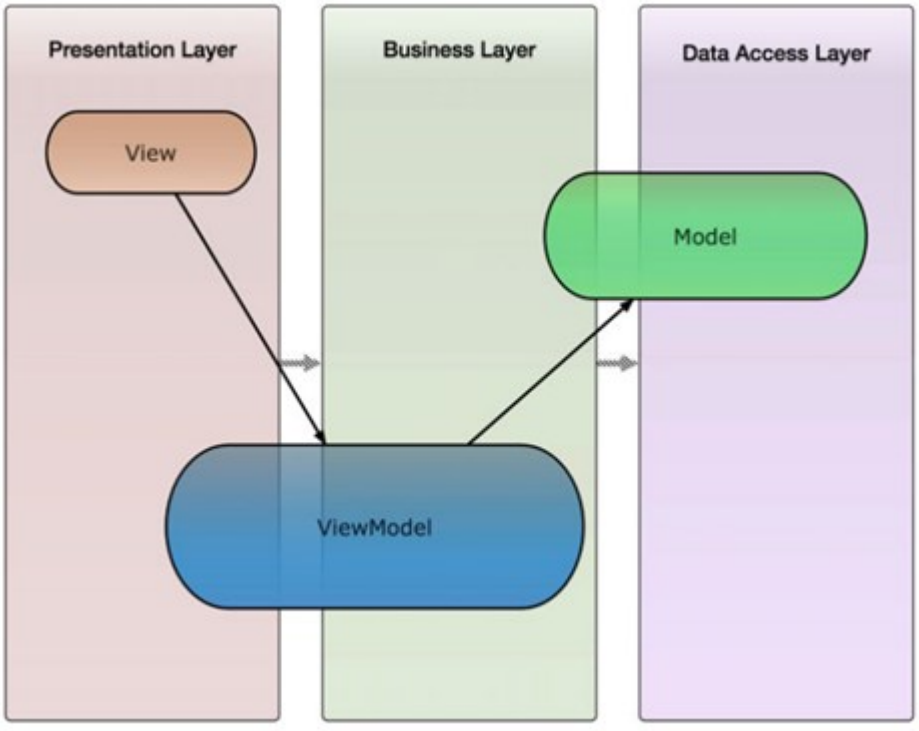
4.7. Dizajn visoke razine

Arhitektura softvera je vrlo važna. Bez implementacije softverske arhitekture, UI kod može biti čvrsto povezan s pozadinskim kodom. To povećava troškove održavanja i kod se pretvara u „špageti kod“, jer se redovi koda počinju povećavati s vremenom. Kako bi se riješio ovaj problem, postoje mnogi arhitekturni obrasci koji su dostupni. Arhitekturni obrasci su opća rješenja za ponovnu upotrebu problema koji se često pojavljuje. Svaki obrazac ima svoje prednosti. Arhitekturni obrasci s vremenom se razvijaju i počinju se koristiti moderniji uzorci prema vještinama i veličini tima. Uglavnom se poslovne aplikacije razvijaju pomoću arhitekture koja čini kod dobro održavanim i lakim za rad. Arhitektura programskog rješenja temelji se na MVVM

(engl. Model-View-ViewModel) arhitekturnom obrascu zbog lakšeg paralelnog razvoja korisničkog sučelja te poslovne logike.

4.7.1. MVVM arhitekturni obrazac

MVVM pomaže u rješavanju problema pružanjem jasnijeg skupa pravila i raščlanjivanjem logike aplikacije na manje, lako upravljive dijelove. Uobičajeno je da se MVVM uzorak predstavlja na linearan način. Razlog iza ovog je naglasak na promjenu zadataka koji se izvode iz svakog dijela obrasca te ukazati na tijek podataka i informacija. Model je odgovoran za pristup različitim izvorima podataka (npr. baze podataka, datoteke ili poslužitelji). Općenito, model ima tendenciju da bude vrlo tanak u MVVM implementaciji. View predstavlja podatke u odgovarajućem formatu (grafički/ne grafički), te odražava stanje podataka i prikuplja interakciju korisnika i događaja. Kao i kod modela, View u MVVM uključuje minimalni implementacijski kod, samo ono što je potrebno da bi View radio i omogućio radnje korisnika. U MVVM-u, najveći dio koda nalazi se u ViewModelu. Koncept iza ViewModela je da ova komponenta predstavlja ono što se očekuje da će biti View (View stanje) i očekuje se da će se ponašati prema interakcijama korisnika (View logika). To je model Viewa u smislu da opisuje skup principa i struktura koje predstavljaju specifične podatke dohvaćene putem Modela. ViewModel upravlja komunikacijom između Viewa i Modela prosljeđivanjem svih potrebnih podataka iz Viewa u Model u obliku koji Model može probaviti.

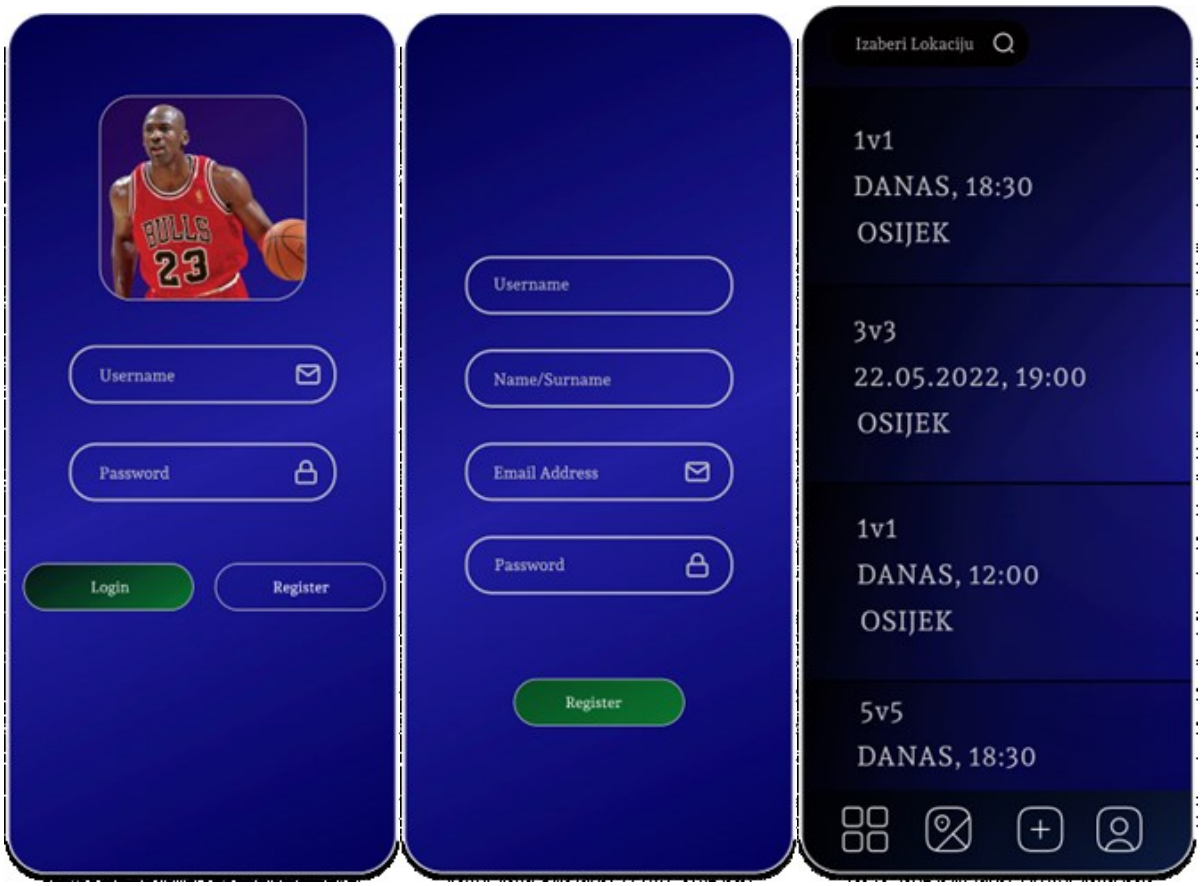


Sl. 4.7.1. Odnos između dizajna i MVVM

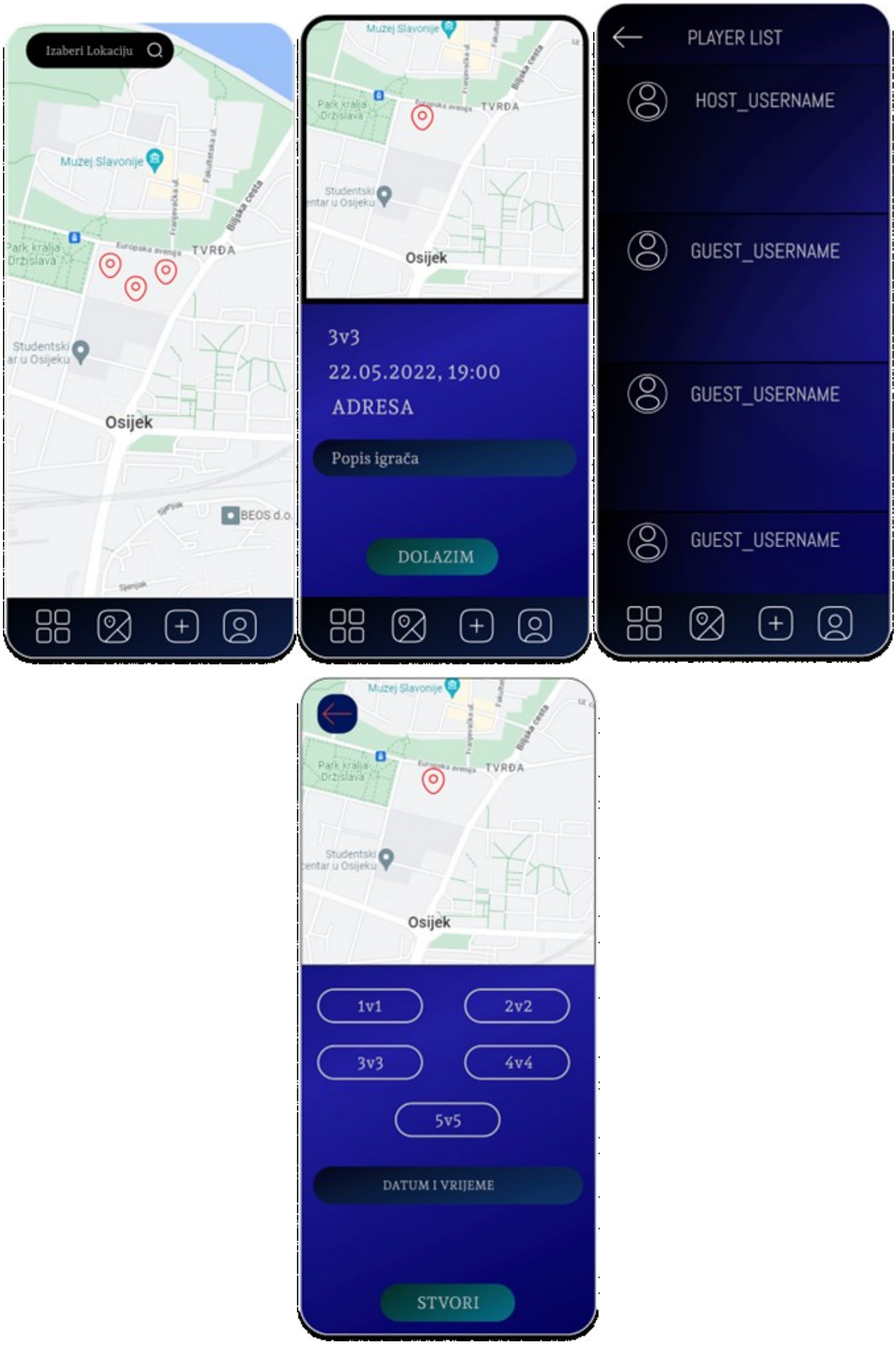
5. IMPLEMENTACIJA PROGRAMSKOG RJEŠENJA

5.1. Figma *mockup* dizajn

Mockup predlaže kako će izgledati konačni dizajn, a obično se dijeli s klijentima i suradnicima. *Mockup* se gradi i dijeli kako bi komunicirali strukturu i funkcionalne zahtjeve dizajna. Modeli su sheme s dodanim površinskim slojem koji predstavlja vizualni dizajn (boje, slike, tipografija).Za razliku od prototipa, *mockup* je statičan i ne uključuje nikakve interakcije.



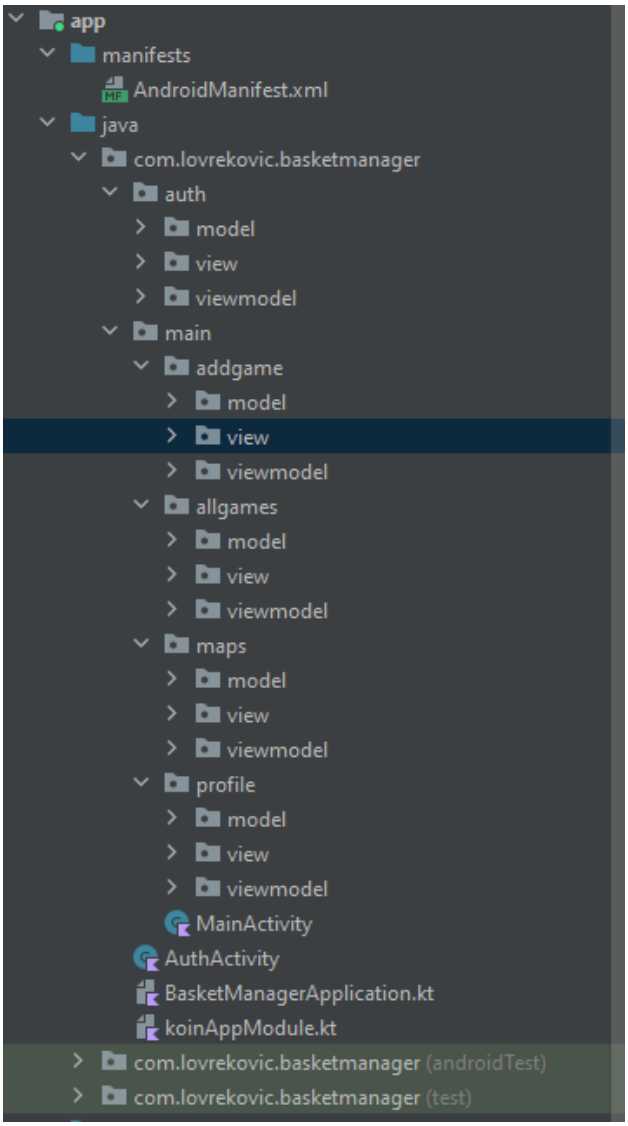
Sl. 5.1.1. prikaz *mockupa*



Sl. 5.1.1.(nastavak) prikaz *mockupa*

5.2. Izrada aplikacije

Kao što je navedeno, programsko rješenje razvijeno je u razvojnom okruženju Android Studio, a koristi se programski jezik Kotlin. Stvorena je prijava i registracija te dohvaćanje podataka iz baze povezivanjem s Firebase platformom. Izgled aplikacije izrađen je po uzoru na Figma *mockup* u .xml datotekama, te su obuhvaćeni prikaz stvorenih događaja i stvaranje novih događaja. Programsko rješenje sastoji se od četiri ključne značajke: registracije i prijave na račun, stvaranja novih događaja, pregleda događaja i pregleda događaja na karti. Izgled aplikacije pisan je po uzoru na Figma *mockup*, a cijela arhitektura aplikacije temelji se na MVVM arhitekturnom obrascu kao što je prikazano na slici 5.2.1. U ovom poglavlju biti će objašnjeno nastajanje svake od ključnih značajki te neke dijelove koda koji su bitni za razumijevanje implementacije.



Sl. 5.2.1 MVVM arhitektura.

5.2.1. Baza podataka

Prvi korak povezivanja Android aplikacije s Firebaseom je pokretanje Firebase pomoćnika i dodavanje svih potrebnih programskih knjižnica i *plugina* u app.gradle dokument. Jedan od uvjeta je prijava na Google račun te registracija na Firebase. Nakon provjere da su ovisnosti za Firebase Firestore dodane u Gradle kao što je prikazano na slici 5.2.2. projekt se sinkronizira i sve je spremno za daljnji rad.

```
implementation 'com.google.firebase:firebase-auth-ktx:21.0.6'
implementation 'com.google.firebase:firebase-firestore-ktx:24.2.1'
```

Sl. 5.2.2. Firebase Dependency

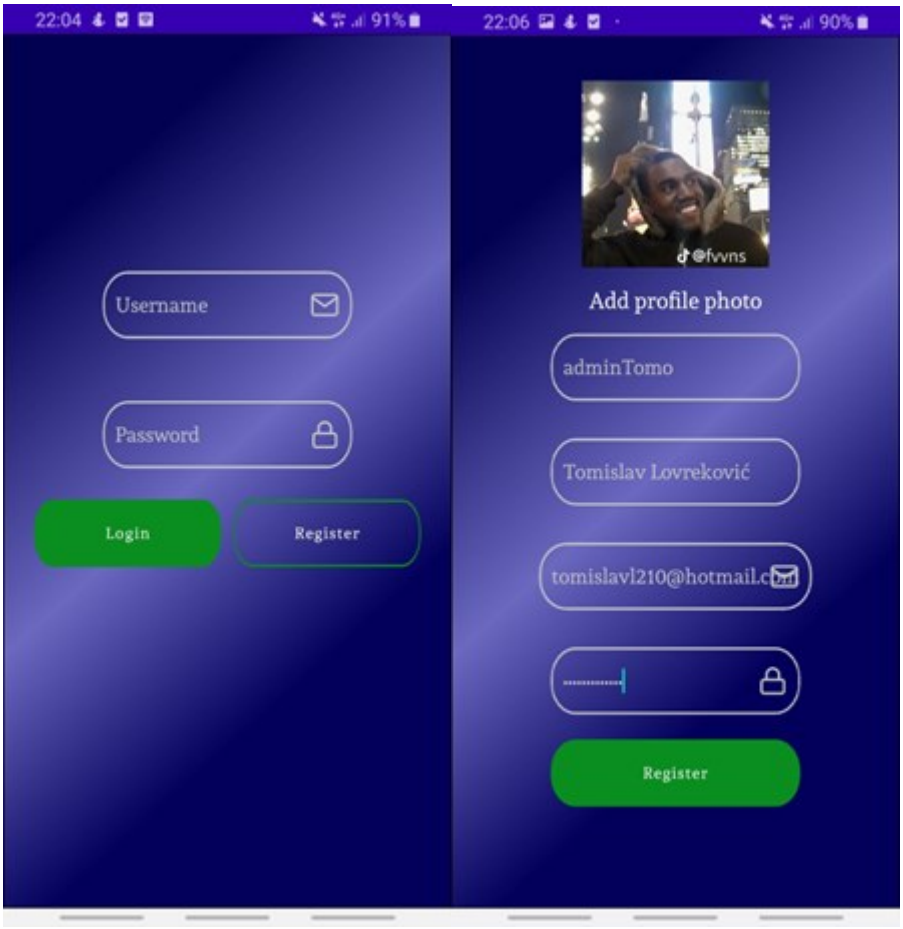
Većina konfiguracije može se automatski postaviti kroz Android Studio tako da oni ne predstavljaju prepreku. Automatski se stvara i google-services.json datoteka koja je poveznica između Firebasea i programskog rješenja. Firebase Firestore pohranjuje datoteke u Google Cloud Storage spremnik i to ih čini dostupnima putem Firebasea i Google Clouda. Ta stavka omogućuje fleksibilnost kod preuzimanja i postavljanja podataka u bazu. Kako bi se omogućilo izvođenje ovog procesa i pristup internetu u manifest se mora dodati programski kod sa slike 5.2.3. Nakon povezivanja s Firebaseom aplikacija ima pristup serveru i bazi podataka.

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

Sl. 5.2.3. Manifest

5.2.2. Registracija i prijava

Kod prijave i registracije koristi se modul Firebase Authentication za provjeru identiteta korisnika i proces spremanja korisnika u bazu podataka. Kao što je ranije navedeno, programsko rješenje je povezano s Firebase platformom i dodana je ovisnost o knjižnici kao što se vidi u programskom kodu sa slike 5.2.2. Prilikom registracije unose se osnovni podatci te postoji mogućnost dodavanja slike. Ako je korisnički račun već izrađen te je korisnik prijavljen, pri ulasku u aplikaciju odmah se prebacuje na glavni izbornik.



Sl. 5.2.4. Izgled prozora prijave i registracije

Na slici 5.2.4 vidi se sučelje prijave i registracije, prvi zaslon koji se otvara kada se aplikacija pokrene je zaslon za prijavu, pritiskom gumba „Register“ aplikacija otvara zaslona za registraciju. Primjećuje se da su sučelja veoma slična *mockupu* napravljenom prije izrade aplikacije.

Stvorena je *register()* metoda koja provjerava email adresu i lozinku te stvara novog korisnika metodom *storeUser()* kao što se vidi u programskom kodu na slici 5.2.6. Metoda *storeUser()* pomoću repozitorija i ugrađenih metoda iz modula Firebase Authentication na Cloud sprema potrebne informacije o korisniku. Varijable *error* i *success* dobivaju vrijednost ovisno o tome je li registracija uspješna ili neuspješna. Te se vrijednosti promatraju te kada dođe do promjene metoda *postValue()* dodaje zadatak u glavnu nit aplikacije. Takva logika korištena je u oba primjera koda kao što je prikazano na slikama programskog koda 5.2.5 i 5.2.6. Način programiranja koji koristi navedene principe naziva se asinkrono programiranje, što je ideja

```
fun login(){
    try {
        authRepository.login(email.value!!, password.value!!)
            .addOnCompleteListener { loginTask ->
                if(loginTask.isSuccessful){
                    success.postValue( value: true)
                } else {
                    loginTask.exception?.printStackTrace()
                    error.postValue( value: "Something went wrong! Check your credentials.")
                }
            }
    } catch (e: Exception){
        e.printStackTrace()
        error.postValue( value: "Something went wrong! Check your internet connection and try again")
    }
}
```

Sl. 5.2.5. Metoda *login()*

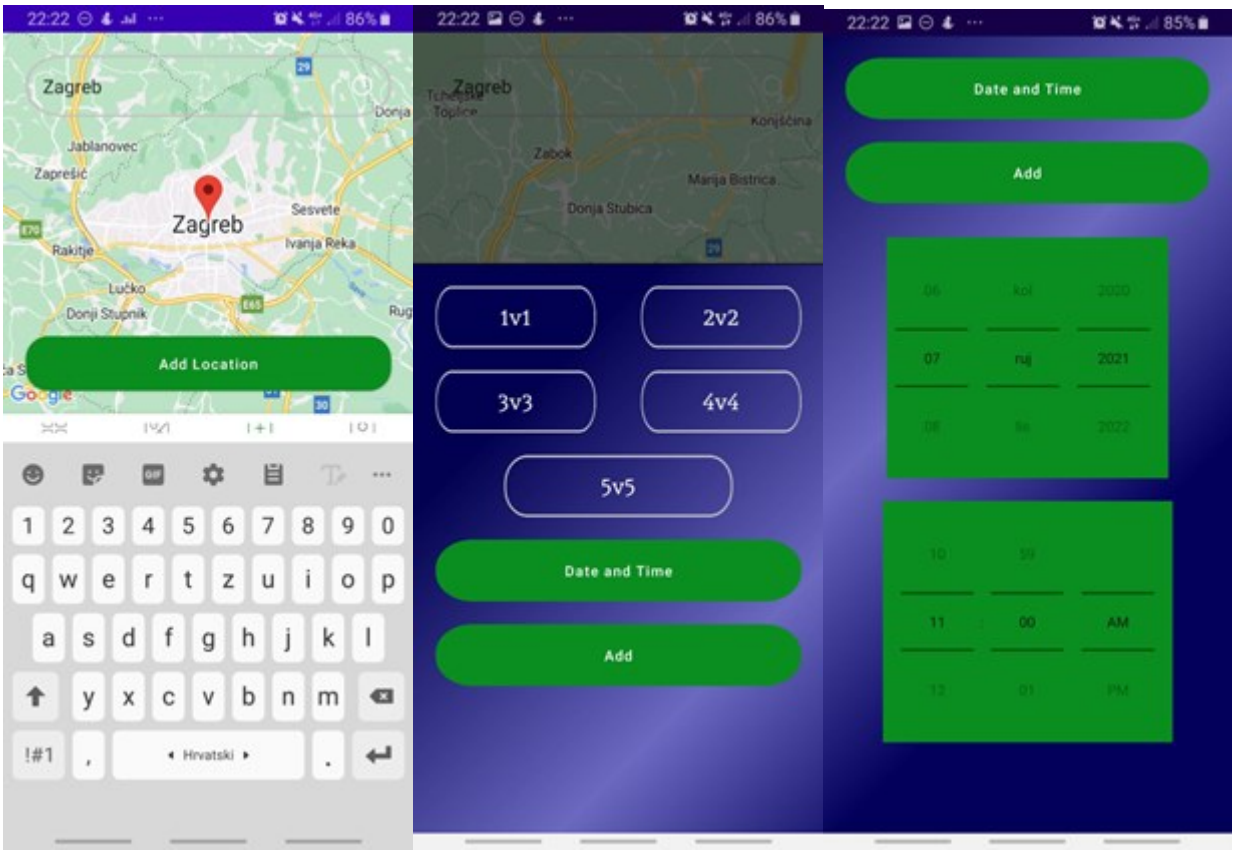
```
fun register() {
    try {
        if (email.value != null && password.value != null && nameSurname.value != null && username.value != null) {
            authRepository.registerUser(email.value!!, password.value!!)
                .addOnCompleteListener { loginTask ->
                    if (loginTask.isSuccessful) {
                        storeUser()
                    } else {
                        loginTask.exception?.printStackTrace()
                        error.postValue( value: "Something went wrong! Check your credentials.")
                    }
                }

        } else {
            error.postValue( value: "Please fill all the required fields!")
        }
    } catch (e: Exception) {
        e.printStackTrace()
        error.postValue( value: "Something went wrong! Check your internet connection and try again")
    }
}
```

Sl. 5.2.6. Metoda *register()*

obustavljenog izvršavanja programa, tj. kada funkcija može obustaviti svoje izvršavanje u nekom trenutku i nastaviti kasnije.

5.2.3. Stvaranje novih događaja

Prilikom stvaranja novog događaja korisnik odabire detaljne informacije mjesta, vremena i tipa igre. Sa priloženim informacijama stvara se događaj koji se unosi u bazu podataka te postaje vidljiv svim korisnicima.



Sl. 5.2.7. Proces stvaranja novog događaja

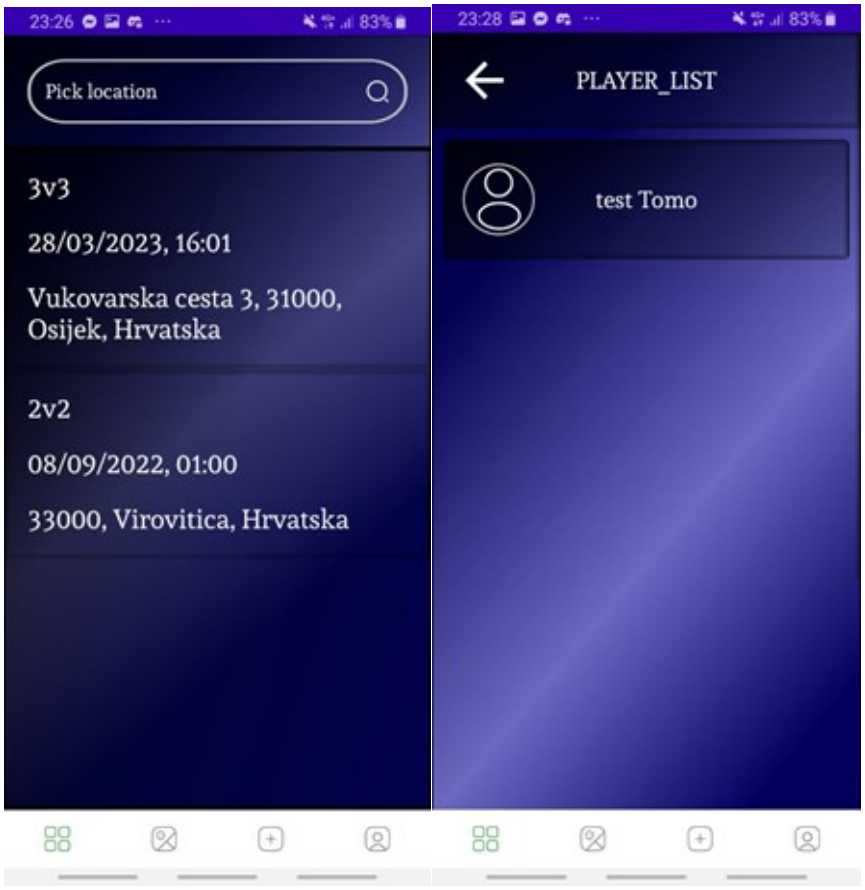
Da bi se događaj dodao u bazu podataka nakon pritiska na gumb *„Add“* poziva se funkcija *addGame()* (slika 5.2.8.). Funkcija *addGame()* nakon provjere jesu li svi parametri popunjeni (mjesto te datum i vrijeme pomoću view komponente *TimePicker*) šalje sve potrebne informacije o događaju kao parametre metodi *addGame()*.

```
fun addGame(address: Address, numOfPlayers: Int) {
    val date = getDateFromPicker()
    val time = getTimeFromTimePicker()
    if (date.isEmpty()) {
        return
    }
    if (time.isEmpty()) {
        return
    }
    addGameRepository.addGame(
        Game(
            numOfPlayers = numOfPlayers,
            cityName = address.getAddressLine( index: 0),
            latitude = address.latitude,
            longitude = address.longitude,
            date = date,
            time = time,
            playerIDs = emptyList()
            )
    )
        .addOnCompleteListener { addGameTask ->
            if(addGameTask.isSuccessful) {
                success.postValue( value: true)
            } else {
                addGameTask.exception?.printStackTrace()
                error.postValue( value: "Something went wrong. Check your internet connection and try again!")
            }
        }
}
```

Sl. 5.2.8. Metoda *addGame()*

5.2.4. Prikaz stvorenih događaja

Fragment koji prikazuje stvorene događaje pristupa podacima tako što pozove metodu u ViewModelu, te putem njega uspostavlja komunikaciju s repozitorijem koji dohvaća podatke iz baze podataka. Korisnik ima mogućnost pregleda stvorenih sportskih događaja na dva načina: prikaz popisom i prikaz pomoću Google karte. Prikaz popisom omogućen je putem RecyclerViewa za koji je napisan adapter *AllGamesRecyclerAdapter*. Korisnik ima mogućnost klikom na događaj pregledati koji korisnici dolaze te je dodana mogućnost pretraga igara po lokaciji, kao što je prikazano na slici 5.2.9.



Sl. 5.2.9.. Izgled prozora prikaza u listi

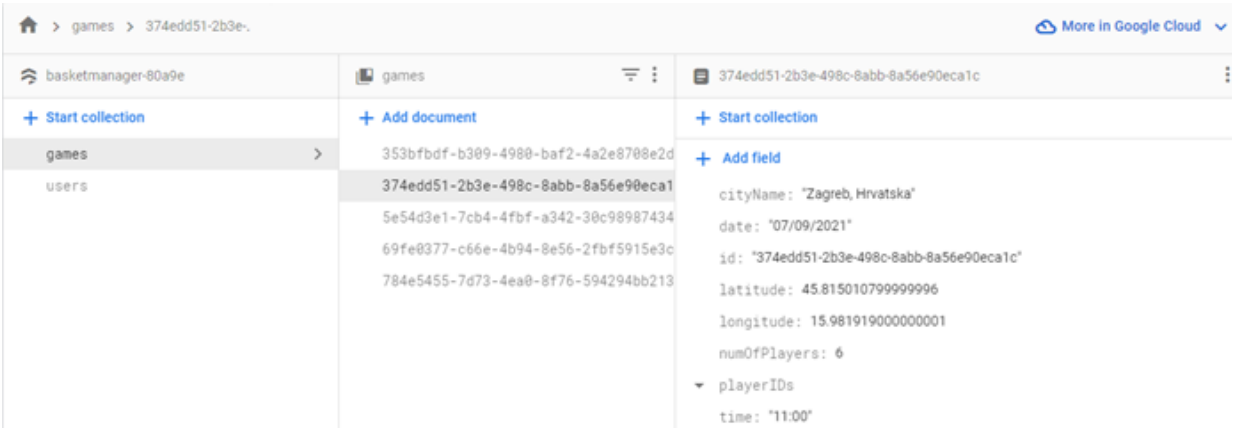
U prikazu pomoću karte svaki tekući događaj označen je markerom što poboljšava preglednost. Klikom na marker omogućen je odabir prikaza igraća te gumb za najavu dolaska na susret. Za dodavanje markera na kartu koristi se metoda *setMarkers()* (slika 5.2.10.) koja dohvaća geografsku širinu i geografsku duljinu te je njom omogućeno da se klikom na marker pojave detalji.

```
private fun setMarkers(games: List<Game>) {
    games.forEach { game ->
        val latLng = LatLng( latitude: game.latitude ?: 0.0,  longitude: game.longitude ?: 0.0)
        val options: MarkerOptions = MarkerOptions().position(latLng).title( title: "")
        val marker = mMap?.addMarker(options)
        marker?.tag = game.id
        mMap?.setOnMarkerClickListener {  it: Marker
            showBottomSheet(game)
            true  ^setOnMarkerClickListener
        }
    }
}
```

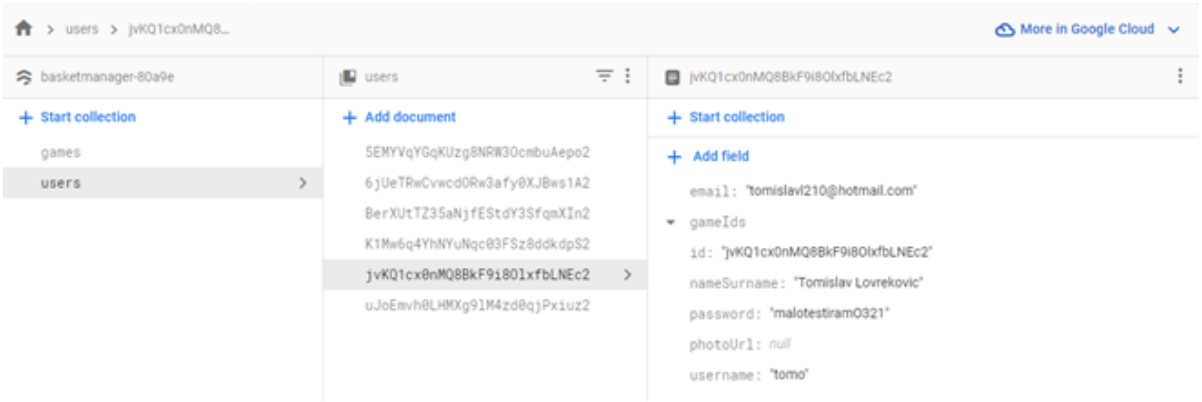
Sl. 5.2.10. Metoda *setMarkers()*

5.3. Korisničko iskustvo

U ovom poglavlju programsko rješenje dano je na testiranje korisnicima kako bi se pronašle eventualne pogreške te kako bi korisnici iznijeli mišljenje o mogućem napretku te popravcima. Korisnici će se prvo susresti s prozorom za prijavu i registraciju, kako bi se prijavili ili registrirali bitna je komunikacija programskog rješenja s bazom podataka. Neki od korisničkih računa te stvorenih susreta pohranjeni su u bazu podataka kako je prikazano na slikama 5.3.1 i 5.3.2. Podatci su spremljeni u dvije kolekcije: *games* i *users*. Podatcima o događajima i korisnicima ostali korisnici mogu pristupiti samo kod prikaza događaja kao što je prikazano na slici 5.2.9.



Sl. 5.3.1. Događaji u bazi podataka



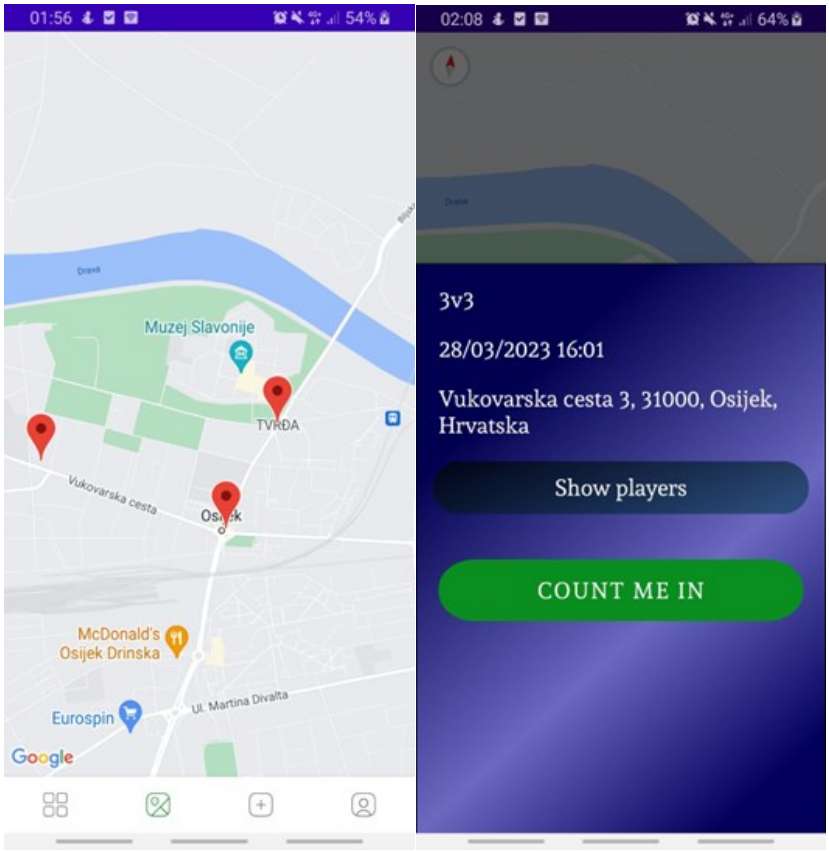
Sl. 5.3.2. Korisnici u bazi podataka

Korisnik prilikom stvaranja novog sportskog susreta upisivanjem lokacije u traku za pretraživanje mora na karti dobiti markerom prikazanu tu istu lokaciju. Korisnici su napomenuli kako je traka za pretraživanje osjetljiva na velika i mala slova (engl. *case sensitive*) te kako nisu odmah mogli

Sl. 5.3.3. Tražilica lokacije

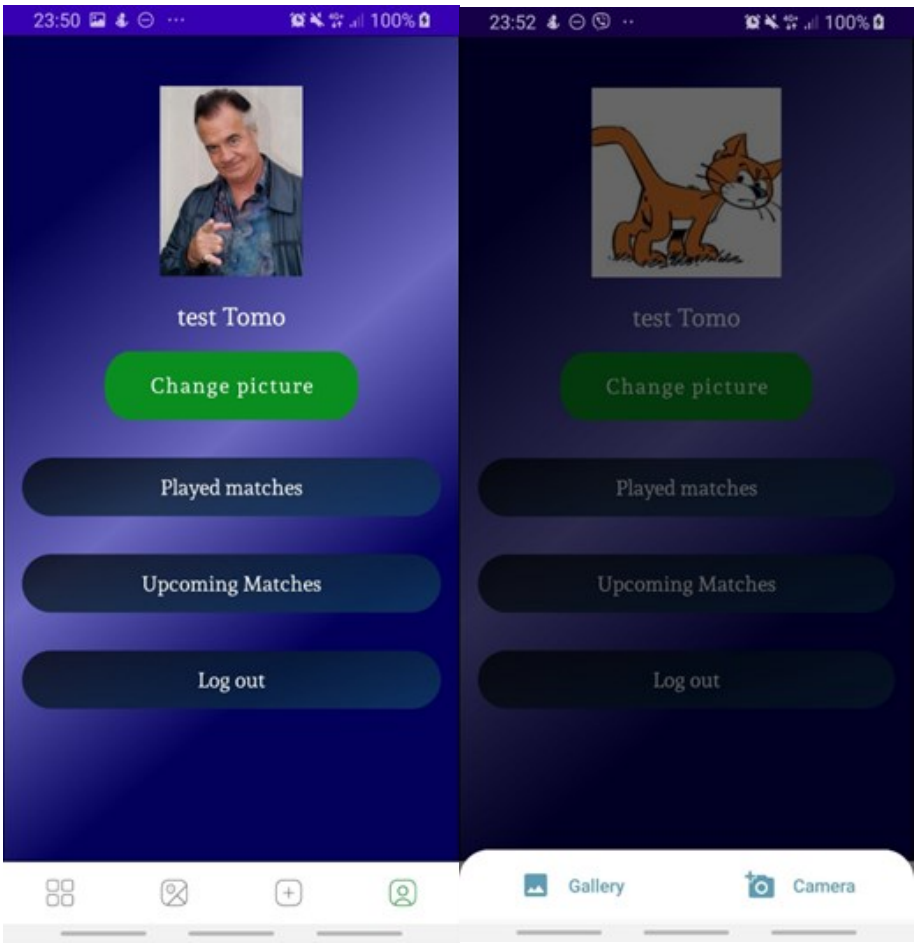
postaviti lokaciju.

Kao dodatak ,korisnik provjerava pokazuje li se ista lokacija na karti stvorenih događaja nakon dodavanja te prikazuju li se ostali tekući događaji u obliku markera na karti. Nadalje klikom na jedan od markera korisnik dobiva detalje o susretu te može najaviti svoj dolazak.



Sl. 5.3.4. Prikaz događaja na karti

Na kraju korisnik mora moći promijeniti fotografiju korisničkog računa te pristupiti kameri i galeriji slika. Prilikom klika na gumb *„Change picture*“ otvara se sučelje za odabir slike, i ako korisnik dozvoli pristup galeriji i kameri, moći će odabrati sliku po želji ili uslikati novu sliku kamerom te ju postaviti za profilnu sliku.



Sl. 5.3.4. Izmjena profilne slike

5.3.1. Ispitivanje na korisnicima

Prikupljeno je mišljenje i recenzija dvadeset različitih korisnika slične dobi, studenata FERIT-a i korisnika koji su često aktivni u sportskih aktivnostima te redovno organiziraju sportske susrete. Razlog odabira ciljne grupe korisnika je dokazani interes za programsko rješenje. To usklađuje vrijednost programskog rješenja s korisnicima koji mogu iskoristiti sve funkcionalnosti aplikacije. Interes korisnika također treba uzeti u obzir prilikom nadogradnje programskog rješenja. Recenzije su bile pozitivne te je većina korisnika izjavila da bi u slobodno vrijeme dogovarali susrete putem aplikacije. Najviše im se svidjelo to što su stvorene događaje mogli pregledati putem karte te lako mogli najaviti svoj dolazak. Ukazano je na moguću nadogradnju dodavanjem funkcionalnosti *chata,* koji bi uvelike poboljšao komunikaciju i dogovore. Nekolicina korisnika predložila je

dodavanje mogućnosti stvaranja grupe u kojoj bi nakon stvaranja događaja samo korisnici koji su članovi te grupe bili pozvani.

6. ZAKLJUČAK

U završnom radu istražene su i opisane tehnologije za izradu Android mobilnih aplikacija te je definirana, dizajnirana i izrađena Android mobilna aplikacija za organiziranje neformalnih sportskih susreta. Uzimajući u obzir da postojeće aplikacije ne pružaju neke od funkcionalnosti, bilo je potrebno razviti programsko rješenje koje omogućuje funkcionalnosti kao što su registracija i prijava te stvaranje i pregled događaja. Alati i tehnologije koji su korišteni za izradu aplikacije: Android Studio razvojno okruženje, programski jezik Kotlin, sučelni program RecyclerView, okvir za ubrizgavanje Koin, pozadinski dio programa Firebase koji omogućuje bazu podataka i sustav za provjeru identiteta te se arhitektura programskog rješenja temelji na MVVM programskom obrascu. Rješenje je uspješno implementirano te ispunjava sve definirane zahtjeve. Ispitivanjem ciljane grupe korisnika dobivena je povratna informacija o postojećim problemima i mogućim nadogradnjama. Funkcionalnosti koje su navedene kao moguća nadogradnja su chat i stvaranje grupe u kojoj se nalaze bliski prijatelji. Programsko rješenje omogućuje jasan pregled sportskih događaja na lokacijama širom svijeta te olakšava organiziranje sportskih susreta.

LITERATURA

[1] Skupina autora, Meet Android Studio [online], AndroidDevelopers, 2022., dostupno na: https://developer.android.com/studio/intro [08.06.2022]

[2] M.Akhin i M.Belyaev, Kotlin docs [online], Kotlin, 2020., dostupno na:

https://kotlinlang.org/docs/home.html [09.06.2022]

[3] Skupina autora, Firebase Authentication [online], Firebase, 2022., dostupno na:

https://firebase.google.com/docs/auth [05.06.2022.]

[4] Skupina autora, Create dynamic lists with RecyclerView [online], AndroidDevelopers, 2022., dostupno na: https://developer.android.com/guide/topics/ui/layout/recyclerview [05.06.2022.]

[5] J.L:Aasenden, Inversion of control, dependency Injection, service oriented programming? [online], WordPress, 2015., dostupno na: https://jonlennartaasenden.wordpress.com/2015/01/13/inversion-of-control-dependency-injection-service-oriented-programming/ [12.06.2022.]

[6] Skupina autora, Corutines guide [online], Kotlin, 2021., dostupno na: https://kotlinlang.org/docs/coroutines-guide.html [16.06.2022.]

SAŽETAK

Sportske aktivnosti i socijaliziranje su ključni elementi u čovjekovoj svakodnevici, olakšavanje dogovora i dijeljenja informacija o sportskom susretu sve je potrebnije. U ovom radu izrađena je mobilna aplikacija za Android OS koja omogućuje organiziranje neformalnih sportskih susreta u svrhu druženja i socijaliziranja. Neki od zahtjeva implementiranih u programskom rješenju su: registracija korisničkog računa, stvaranje događaja, pregled i pristup događaju te uređivanje korisničkog računa. Kako se radi o programskom rješenju za Android operacijski sustav, izrađeno je u programskom jeziku Kotlin u Android Studio programskom okruženju. Provjera identiteta korisnika i baza podataka izrađena je uporabom modula Firebase Authentication i Cloud Firestore usluga. Iz analize rezultata ankete provedene nad ciljnom grupom korisnika može se zaključiti da je najbolje ocijenjena funkcionalnost prikaz događaja na karti, te su predložene neke preinake i nadogradnje.

Ključne riječi: Android aplikacija, Android Studio, Firebase, Kotlin, socijalizacija

ABSTRACT

Android application for organizing informal sports events

Sports activities and socialization are key elements in a person's everyday life, facilitating agreements and sharing information about sports events is increasingly necessary. In this work, a mobile application for Android OS was created, which enables the organization of informal sports meetings for the purpose of socializing. Some of the requirements implemented in the application are: user account registration, event creation, event viewing and access, and user account editing. Being the Android applicaiton, it was created in the Kotlin programming language in the Android Studio programming environment. User and database identity verification is done using the Firebase Authentication module and Cloud Firestore services. The analysis of the results of the survey conducted on the target group of users, it can be concluded that the best rated functionality is the display of events on the map, and some changes and upgrades have been proposed.

Keywords: Android application, Android Studio, Firebase, Kotlin, socialization

ŽIVOTOPIS

Tomislav Lovreković rođen je 02.10.2000. godine u Virovitici. Završava Osnovnu školu Ivane Brlić Mažuranić u Virovitici 2015.godine te upisuje prirodoslovno matematički smjer Gimnazije Petra Preradovića Virovitica. Maturira 2019. te iste godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, preddiplomski sveučilišni studij računarstva. Od osnovne škole razvija znanje o programskim i stilskim jezicima kao što su: Python, C, C#, Kotlin, HTML, CSS te VHDL.

_____

Potpis autora

PRILOZI

Programsko rješenje ovog završnog rada nalazi se na github repozitoriju: https://github.com/tomislavl210/BasketManager.git