RAZVOJ MOBILNE APLIKACIJE ZA UMREŽAVANJE LJUDI ZAJEDNIČKIH INTERESA

Čavala, Šime

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

RAZVOJ MOBILNE APLIKACIJE ZA UMREŽAVANJE LJUDI ZAJEDNIČKIH INTERESA

Šime Čavala

Zagreb, veljača 2023.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, 11. 2. 2023.

Predgovor

Želim se zahvaliti Danielu Beleu na savjetima, motivaciji i znanju koje mi je prenio u prethodne tri godine preddiplomskog studija na Visokom učilištu Algebra. Također, zahvaljujem se tvrtki Tehnozavod Marušić i Visokom učilištu Algebra koji su mi svojom stipendijom omogućili studiranje.

A Gradišćanska 24, HR-10000 Zagreb
T (01) 2222 182
E student@algebra.hr
www.algebra.hr

ALGEBRA

Temeljem članka 8. Pravilnika o završnom radu i završnom ispitu na preddiplomskom studiju Visokog učilišta Algebra sačinjena je ova

Potvrda o dodjeli završnog rada

kojom se potvrđuje da student Šime Čavala, JMBAG 0321011737, OIB 82498422633 u šk. godini 2021./2022., studij: Primjenjeno računarstvo - Preddiplomski studij, smjer: Programsko inženjerstvo, od strane povjerenstva za provedbu završnog ispita, dana 25.01.2022. godine, ima odobrenu izradu završnog rada

s temom: **Razvoj mobilne aplikacije za umrežavanje ljudi zajedničkih interesa**

i sažetkom rada: Student će svojim radom prikazati napredno korištenje inovativnih tehnologija u kontekstu razvoja aplikacije za umrežavanje. Rezultat ovog završnog rada je sustav koji omogućuje interaktvino i napredno koristenje tehnologije u svrhu upravljanja aktivnostima i uslugama umrežavanja ljudi sličnih interesa.

Mentor je: Daniel Bele.

Odobrenjem završnog rada studentu je omogućen upis kolegija "Izrada završnog projekta/Praksa" te je sukladno članku 8. Pravilnika o završnom radu i završnom ispitu dužan najkasnije do početka nastave ljetnog semestra u sljedećoj školskoj godini, uspješno obraniti završni rad uspješnim polaganjem završnog ispita.

U protivnom student može zatražiti novog mentora/icu i temu te ponovo upisati kolegij "Izrada završnog projekta/Praksa" budući da rad koji nije predan i obranjen na završnom ispitu u roku određenom Pravilnikom završnom radu i završnom ispitu prestaje vrijediti. Izrada novog završnog rada se izvodi sukladno rokovima određenima za školsku godinu u kojoj je studentu određen novi mentor/ica i dodijeljen novi završni rad.

Potpis studenta:	Potpis mentora:	Potpis predsjednika povjerenstva:

Ova potvrda izdaje se u 4 (četiri) primjerka od kojih 3 (tri) idu kao prilog završnom radu.

Sažetak

Ovaj završni rad bavi se izradom troslojnoga programskog rješenja čija je svrha povezivanje ljudi zajedničkih interesa. Programsko će rješenje korisnicima omogućiti pregledavanje, stvaranje i filtriranje događaja, kao i komunikaciju s drugim korisnicima te pretplaćivanje na obavijesti o novim objavama. Prije same implementacije, provedena je anketa nad ciljanom skupinom korisnika te su prepoznati nedostaci i prednosti sličnih rješenja na tržištu. Nakon utvrđivanja problema koji ovaj rad rješava, objašnjene su tehnologije i programski jezici korišteni prilikom razvoja. Glavni je dio ovoga programskog rješenja klijentski sloj koji predstavlja mobilna android aplikacija. U radu su također objašnjene funkcionalnosti aplikacije i prikazani su ključni dijelovi koda. Nakon razvoja programskog rješenja, prepoznati su nedostaci i predložena su moguća poboljšanja.

Ključne riječi: android, mobilna aplikacija, zajednički interesi, događaji

Summary

This final thesis deals with the creation of a three-layer software solution whose purpose is to connect people with common interests. The software solution will enable users to browse, create and filter events, communicate with other users and subscribe to notifications about new events. Before the actual implementation, a survey of the target group of users was conducted, and the shortcomings and advantages of similar solutions on the market were recognized. After determining the problem this work solves, the technologies and programming languages used during development are explained. The main part of this software solution is the client layer represented by the mobile android application. The paper also presents the application's functionality and shows the code's key parts. After the development of the software solution, shortcomings were recognized, and possible improvements were proposed.

Key words: android, mobile application, common interests, events

Sadržaj

1. Uvod

Tehnologija se ubrzano razvija iz dana u dan, što rezultira učinkovitijim i olakšanim provođenjem poslovnih procesa u tvrtkama. Tvrtke koriste značajan broj resursa kako bi se digitalizirale i povećale svoju skalabilnost. No razvoj tehnologije nije stao tu. Čovjek se u svakodnevnom životu znatno oslanja na tehnologiju kako bi mu olakšala svakodnevni život. Gotovo sve informacije, koje su mu potrebne u nekom trenutku, dostupne su javno na internetu. Ljudi se međusobno umrežavaju i međusobnim dijeljenjem informacija stvaraju golemu bazu podataka. Osim toga, koriste internet da pronađu ljude istih svjetonazora kako bi razmijenili mišljenja i gledišta te raspravljali o raznim temama. Društvene mreže u današnje vrijeme imaju i do nekoliko milijardi mjesečno aktivnih korisnika. Najpopularnije društvene mreže imaju najviše korisnika, što dovodi do velikog broja različitih vrsta sadržaja na jednome mjestu. Zbog toga se sve više pojavljuju nove društvene mreže koje se specijaliziraju za određenu vrstu sadržaja. To rezultira brzim povezivanjem ljudi i pronalaženjem željenog sadržaja. Upravo je to osnovna svrha ovog rada, čija je tema razvoj mobilne aplikacije za umrežavanje ljudi zajedničkih interesa. Cilj je rada razviti programsko rješenje koje se sastoji od podatkovnog, servisnog i klijentskog sloja pa će korisnicima omogućiti da učinkovito ispune svoje slobodno vrijeme, steknu nova iskustva i poznanstva.

Rad započinje analizom koja će istaknuti prednosti i nedostatke trenutno dostupnih rješenja. Nakon toga, slijedi obrada provedene ankete medu korisnicima društvenih mreža s namjerom ispitivanja potrebe za mobilnom aplikacijom za umrežavanje ljudi sličnih interesa. Također, analiza ispituje navike i aktivnosti ispitanika na društvenim mrežama. Zatim slijedi skica i opis arhitekture programskog rješenja. Opisana je implementacija i funkcionalnosti servisnog, kao i klijentskog sloja, uz prikaz zaslona mobilne aplikacije. Rad završava identifikacijom nedostataka i predlaganjem mogućih poboljšanja za daljnji razvoj programskog rješenja.

2. Problematika i prijedlog rješenja

Važan nulti korak prije implementacije programskog rješenja je analiza trenutno dostupnih rješenja, kao i ispitivanje potreba ciljne grupe korisnika. Uz to, važan faktor je i utvrđivanje činjenice ima li dovoljno potencijalnih korisnika i hoće li se programsko rješenje, koje želimo razviti, uspješno probiti na današnjem tržištu. U cilju ostvarenja što većeg broja funkcionalnosti koje su korisnicima potrebne, a koje nedostaju u trenutnim rješenjima, provedena je komparacija postojećih rješenja. Uzevši u obzir rezultate ankete, kao i vlastita iskustva s postojećim rješenjima slične tematike, predloženo je novo rješenje.

2.1. Ograničenja trenutnih rješenja

U zadnjih nekoliko godina pogodila nas je pandemija koronavirusa, koja je potaknula socijalno distanciranje i znatno utjecala na mentalno zdravlje pojedinaca. Stručni rad Vesne Antičević, koji se bavi učincima pandemije na mentalno zdravlje, pokazao je da su neki od najčešćih problema mentalnoga zdravlja, koji su u porastu otpočetka pandemije, anksioznost i depresija [1]. Veliku ulogu u stvaranju ovih problema kod ljudi odigrale su epidemiološke mjere, kao što su ograničavanje okupljanja i fizičko distanciranje. U vrijeme pisanja ovog rada, većina mjera donesena od strane Stožera civilne zaštite Republike Hrvatske su ukinute. Mnogo ljudi osjeća se zakinuto jer nisu proveli planiranu količinu vremena socijalizirajući se, baveći se omiljenim hobijem i slično. Zbog toga se žele povezati i pronaći druge, slične sebi. Nažalost, mogućnosti za povezivanje ljudi koji žele nadoknaditi propušteno vrijeme i učinkovito ispuniti svoje slobodno vrijeme su ograničene. Glavni je problem što ne postoji centralno mjesto čija bi jedina svrha bila okupiti i povezati ljude zajedničkih interesa. Internet omogućuje pristup gotovo svakoj informaciji i kontaktiranje gotovo svake osobe na svijetu, ali su podaci raspršeni. Čak se i kod društvenih mreža pojavljuje sličan nedostatak. Postoje brojne virtualne zajednice u koje su ljudi grupirani pa je potrebno utrošiti određeno vrijeme kako bi se pronašla odgovarajuća zajednica. Također, sadržaj koji se objavljuje nije strukturiran i u većini ga je slučajeva nemoguće filtrirati prema željenoj vrsti sadržaja.

2.2. Prijedlog rješenja

Društvene mreže, čije su glavne uloge umrežavanje ljudi i komunikacija, znatno utječu na život pojedinca u današnjem svijetu. Ubrzani ritam života modernog čovjeka, kao i pandemija koronavirusa, koja je potaknula socijalno distanciranje, neki su od mnogih čimbenika koji su potaknuli rast društvenih mreža i sličnih aplikacija. Uzevši u obzir kako su ljudi danas ovisni o tehnologiji i kako su navikli „odraditi“ određene aktivnosti iz udobnosti svojeg doma u nekoliko klikova miša, aplikacija sa svojstvima društvenih mreža bila bi idealno rješenje. Ideja je razviti mobilnu aplikaciju za povezivanje ljudi zajedničkih interesa kako bi umreženi korisnici učinkovito pronašli zanimaciju u svoje slobodno vrijeme. Aplikacija će biti prilagođena ciljnim korisnicima i omogućit će im pretraživanje i filtriranje sadržaja, kontaktiranje drugih korisnika, dobivanje obavijesti za željenu vrstu sadržaja, kao i stvaranje sadržaja. Cilj je aplikacije vrlo jednostavan – da bude centralno mjesto s raznim vrstama strukturiranog sadržaja koji je moguće prilagoditi potrebama korisnika. Ime aplikacije bi bilo „tipar“, a odabrano je zato što nije generičko, kratko je i lako pamtljivo. Uz to, ime aplikacije potječe iz urbanog žargona, što bi moglo u određenom dijelu pozitivno utjecati na probijanje aplikacije na tržište.

2.3. Usporedba s postojećim rješenjima

Nakon provedenog istraživanja tržišta, utvrđeno je da trenutno ne postoji rješenje kojem je cilj umrežiti korisnike koji žele pronaći slične sebi na temelju zajedničkih interesa, a da je pritom sadržaj pregledan i strukturiran. U nastavku slijede prednosti i nedostaci sličnih rješenja u usporedbi s programskim rješenjem „tipar“.

Facebook

Jedna od glavnih funkcionalnosti društvene mreže *Facebook* je to što bilo koji korisnik ima mogućnosti kreirati grupu koja će povezivati ljude na temelju određene tematike. Ovdje se može primijetiti prvi nedostatak *Facebook* grupa. Postoji velik broj grupa različitih tematika pa korisnik mora prije svega uložiti vrijeme da pronađe odgovarajuću grupu za sebe. Najveća zajednica koja okuplja studente je grupa „Studentski dom Stjepan Radić – Sava“ koja trenutno broji gotovo 48 000 članova. Dnevno se objavljuje nekoliko desetaka objava vezano uz aktualne događaje u Gradu Zagrebu. To je, naravno, samo jedna od mnogih vrsta objava

koje se pojavljuju u grupi. Nedostatak ove i sličnih grupa je to što sadržaj nije strukturiran pa je iz gomile teško izdojiti objave određene tematike. Prednost programskog rješenja „tipar“ u odnosu na *Facebook* grupe je to što bi svaka objava bila kreirana s ciljem da autor događaja pronađe sudionike i proširi vijest o svojem događaju. Ako korisnik ima namjeru pratiti sadržaj određene grupe, a da ne koristi ostale funkcionalnosti *Facebooka*, svejedno mora otvoriti korisnički račun i unijeti svoje osobne podatke iako možda ima nešto protiv toga. Još jedna prednost programskog rješenja „tipar“ je to što bi se objave mogle filtrirati prema tipu i kategoriji događaja. Također, mobilna bi aplikacija preuzela i neke funkcionalnosti *Facebooka,* kao što su „čavrljanje“ te označavanje objava sa „Sviđa mi se“. Za razliku od *Facebooka,* gdje se moguće pretplatiti za dobivanje obavijesti o svim objavama određene grupe, mobilna aplikacija programskog rješenja „tipar“ imat će funkcionalnost da se korisnik pretplati samo na dobivanje obavijesti željene kategorije događaja.

Instagram

Postoji mnogo društvenih mreža koje nisu fokusirane na jednu funkcionalnost, već pružaju svojim korisnicima niz sličnih funkcionalnosti. Instagram se ističe u moru generičkih društvenih mreža zbog toga što pridaje veću pozornost na mogućnost objave slika i videozapisa. Nije rijetka pojava na *Instagramu* da korisnici objave sliku uz opis nekog društvenog događaja na kojem se trenutno nalaze. Također, česta je pojava da neki ugostiteljski objekt ima svoj račun na *Instagramu* pa objavama obavještava svoje pratitelje o relevantnim događajima koje organizira. Glavni nedostatak *Instagrama* je to što bilo tko može otvoriti *Instagram* račun te je potrebno utrošiti vrijeme da se prate svi računi koji zanimaju pojedinog korisnika. *Instagramu*, kao i *Facebooku*, nije glavna ciljna grupa korisnici koji žele učinkovito ispuniti svoje slobodno vrijeme. Objave su razbacane, kao i kod *Facebooka,* i nije ih moguće filtrirati. Prednosti mobilne aplikacije programskog rješenja „tipar“ u odnosu na *Instagram* su što bi se uz sliku i opis mogla dodati lokacija te vrijeme pojedinog događaja.

Drink&Pick – Playful&Fun app

Drink&Pick mobilna aplikacija ima prije svega svrhu da se korisnici brzo povežu na temelju želje za socijaliziranjem. Ova aplikacija omogućuje i pretraživanje aktualnih događaja. Nedostatak aplikacije je to što korisnik ne može stvoriti događaj, već se za to brinu autori

aplikacije, a to dovodi do znatno manje liste aktualnih događaja. Također, nedostatak je što korisnici imaju samo jednu mogućnost povezivanja: na temelju namjere da upoznaju tu osobu u stvarnom životu i potom se socijaliziraju. Programsko rješenje „tipar“ bi ovo proširilo na različite kategorije događaja kao što su hobiji, sport, zabava i slično.

U Tablica 2.1 Usporedba aplikacija prikazana je usporedba po najvažnijim karakteristikama i funkcionalnostima.

Tablica 2.1 Usporedba aplikacija

	tipar	*Facebook*	*Instagram*	*Drink&Pick*
Stvaranje objava	✓	✓	✓	✗
Filtriranje objava	✓	✗	✗	✗
Komunikacija s drugim korisnicima	✓	✓	✓	✓
Web-stranica	✗	✓	✓	✓
Mobilna aplikacija	✓	✓	✓	✓
Dodavanje lokacije na objavu	✓	✗	✗	✓
Personalizirane obavijesti o objavama	✓	✗	✗	✗

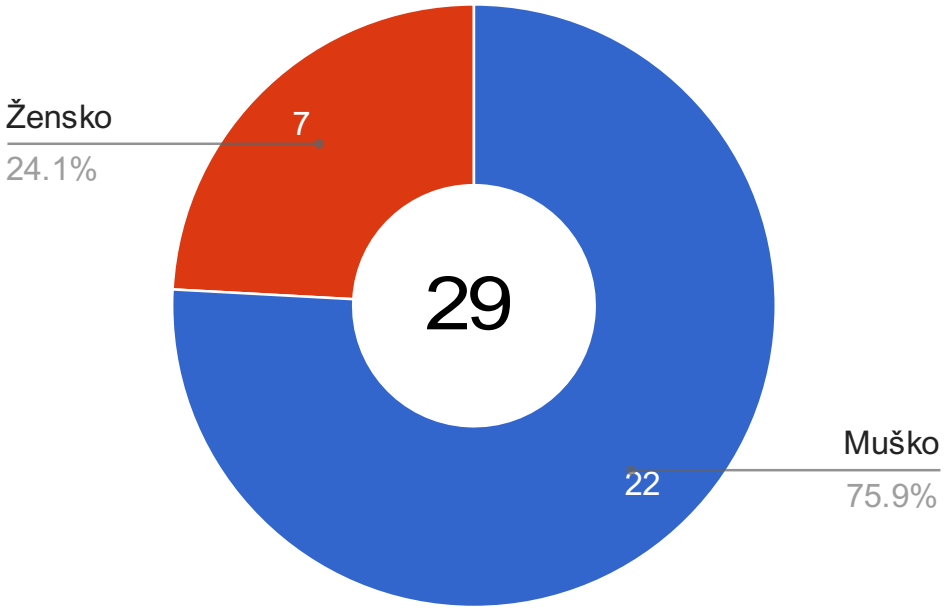
Iz usporedbe aplikacija po karakteristikama možemo vidjeti da se programsko rješenje „tipar“ ističe po jedinstvenim karakteristikama: personalizirane obavijesti o objavama i filtriranje objava. Također, iz usporedbe se vidi da programsko rješenje „tipar“ ne posjeduje jednu karakteristiku koju imaju sva druga rješenja, a to je posjedovanje *web*-stranice. U sklopu ovog rada, bit će implementirano samo programsko rješenje mobilne aplikacije dok se mogućnost razvoja još jedne klijentske aplikacije ostavlja za budućnost.

2.4. Provođenje ankete među ispitanicima i njezini rezultati

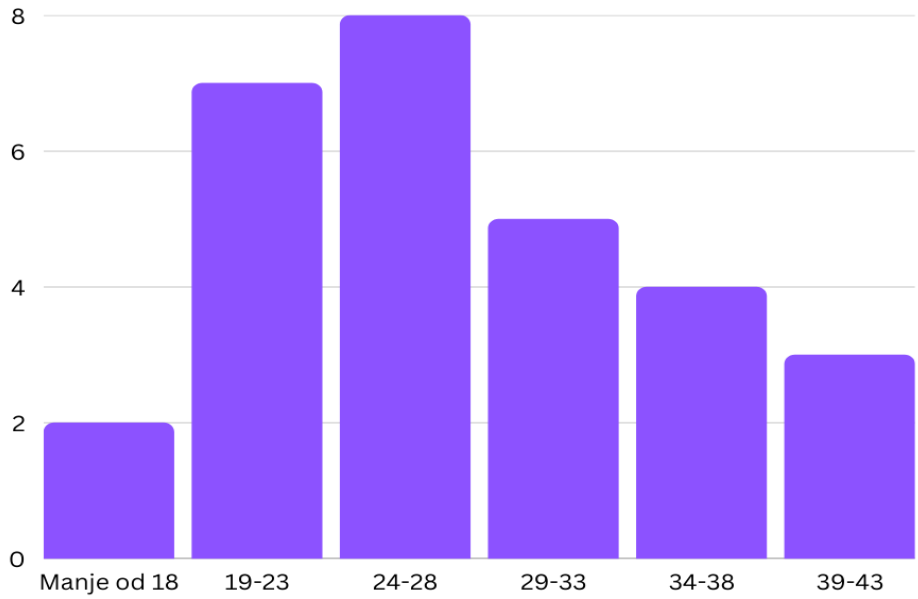
Anketa je provedena među ciljnom skupinom potencijalnih korisnika aplikacije „tipar“ koji koriste slična programska rješenja kako bi učinkovito popunili svoje slobodno vrijeme. Cilj

je ankete utvrditi koliki je interes za aplikaciju „tipar“. Za provođenje ankete korištena je programska podrška za administraciju anketa *Google Forms*.

2.4.1. Demografija ispitanika



Slika 2.1 Spol ispitanika

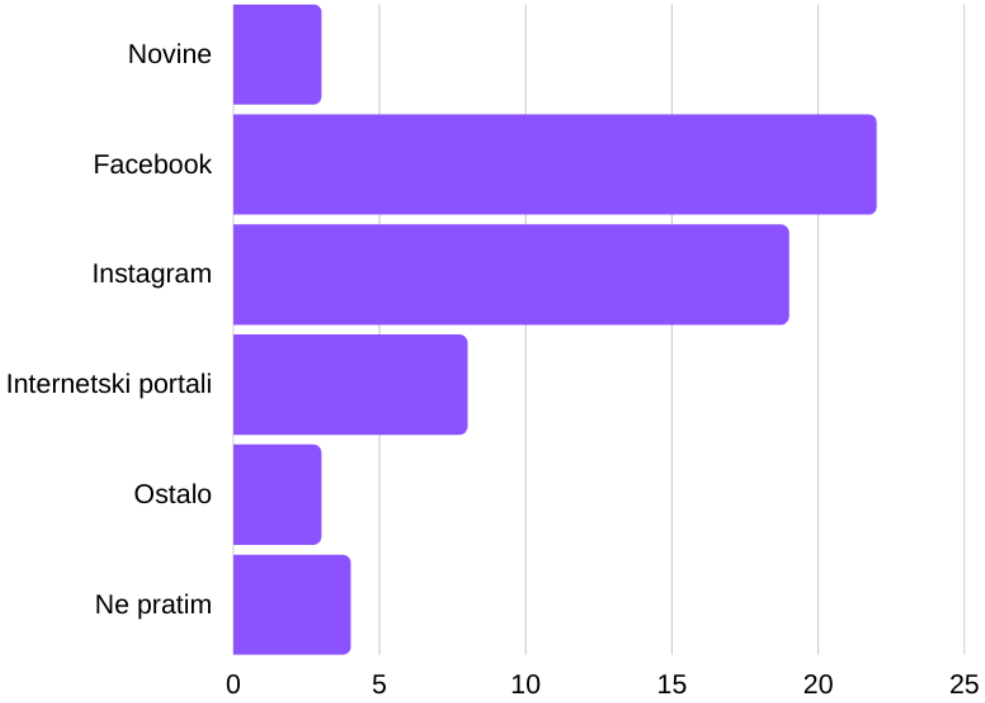


Slika 2.2 Dob ispitanika

Anketa započinje pitanjima o demografskim značajkama ispitanika. Iz tortnog grafikona (Slika 2.1 Spol ispitanika) može se zaključiti da je anketu ispunilo ukupno 29 ispitanika od kojih je 7 (24,1%) bilo ženskog te 22 (75,9%) muškog spola.

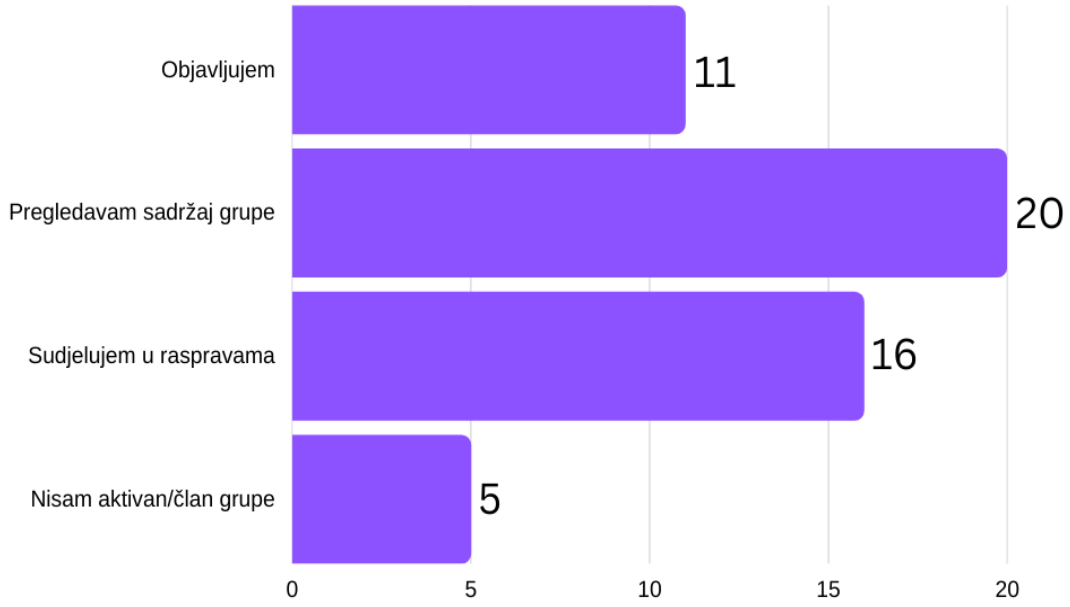
Na temelju prikupljenih podataka o dobi ispitanika stvorene su dobne skupine prikazane u stupčastom grafikonu (Slika 2.2 Dob ispitanika). Najviše je ispitanika u dobi između 19 i 23 godine, a zatim u dobi između 23 i 28 godina. Najmanje ispitanika, koji su ispunili anketu, pripadaju dobnoj skupini mlađoj od 18 godina, dok nema osoba starijih od 43 godine koje su ispunile anketu.

2.4.2. Navike ispitanika na društvenim mrežama



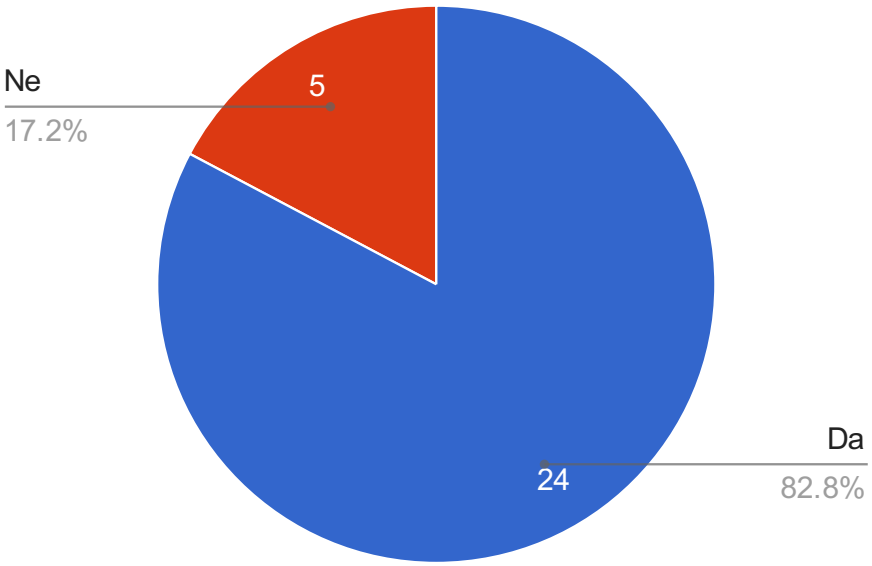
Slika 2.3 Načini praćenja aktualnih događaja

Slika 2.3 Načini praćenja aktualnih događaja) prikazuje koja sredstva i u kojoj mjeri koriste korisnici kako bi došli do informacija o događajima s vremenom radnje u bliskoj budućnosti. Na temelju prikazanog stupčastoga grafikona možemo zaključiti da ispitanici najviše koriste društvene mreže *Instagram* i *Facebook*, kao glavni izvor informacija o aktualnim događajima. Treći najpopularniji način praćenja su internetski portali, dok su se samo četiri ispitanika izjasnila da ne prate aktualne događaje.



Slika 2.4 Aktivnost ispitanika u Facebook grupama

Slika 2.4 Aktivnost ispitanika u Facebook grupama) prikazuje kojim se radnjama korisnici najviše koriste u *Facebook* grupama gdje je cilj svih ili dijela objava informiranje o aktualnim događajima bliske budućnosti. Rezultati su ankete pokazali da je najčešća aktivnost ispitanika pregledavanje sadržaja grupe, a potom sudjelovanje u raspravama. Pet ispitanika izjasnilo se da nije aktivno ili da nisu članovi grupe.



Slika 2.5 Interes ispitanika za aplikaciju

Anketa završava ispitivanjem ispitanika o interesu korištenja aplikacije za učinkovito iskorištavanje slobodnog vremena. Njih 24 (82,8%) izjasnilo se da bi koristilo navedenu aplikaciju (Slika 2.5 Interes ispitanika za aplikaciju). Rezultati ankete pokazali su da je velik dio ispitanika aktivan na društvenim mrežama i da koristi *Facebook* grupe kako bi sudjelovali u raspravama, pregledavali i objavljivali sadržaj. Uzevši sve u obzir, zaključeno je da postoji potreba za aplikacijom ovakve tematike.

2.5. SWOT analiza

U današnje vrijeme, prije važnih strateških odluka poduzeća provode brojne analize koje će pokazati hoće li se odlučiti za određeni potez ili ne. Jedna od najpoznatijih i najosnovnijih poslovnih analiza je SWOT analiza. SWOT je skraćenica koju predstavljaju četiri engleske riječi: snage (engl. *strengths*), slabosti (engl. *weaknesses*), prilike (engl. *opportunities*) i prijetnje (engl. *threats*) [2]. Ove četiri riječi ujedno predstavljaju faktore koji se koriste prilikom analize za usporedbu vanjskih prilika i prijetnji te unutarnjih snaga i slabosti. Cilj je ove analize, osim da se donese odluka hoće li poduzeće krenuti u realizaciju projekta, i da se odabere najpovoljnija strategija za ostvarivanje rezultata [2].

Prije realizacije ovog projekta, provedena je SWOT analiza, kao što prikazuje Slika 2.6 SWOT analiza) u kojoj su prepoznate najvažnije snage, slabosti, prilike i prijetnje.

Slika 2.6 SWOT analiza

Nakon provedene analize, zaključeno je da su najveće snage iskustvo s razvojem aplikacija te znanje o temi i poznavanje sličnih aplikacija. Slabosti su manjak iskustva u dizajniranju sučelja aplikacija i zastarjela oprema na kojoj će se provoditi implementacija programskog dijela. Prilike su inovativna ideja i rast društvenih mreža te aplikacija za upoznavanje, ali prijetnje su to što aplikacija ima samo jednu vrstu klijentske aplikacije te što bi neočekivani val koronavirusa mogao uvelike utjecati na učestalost korištenja aplikacije.

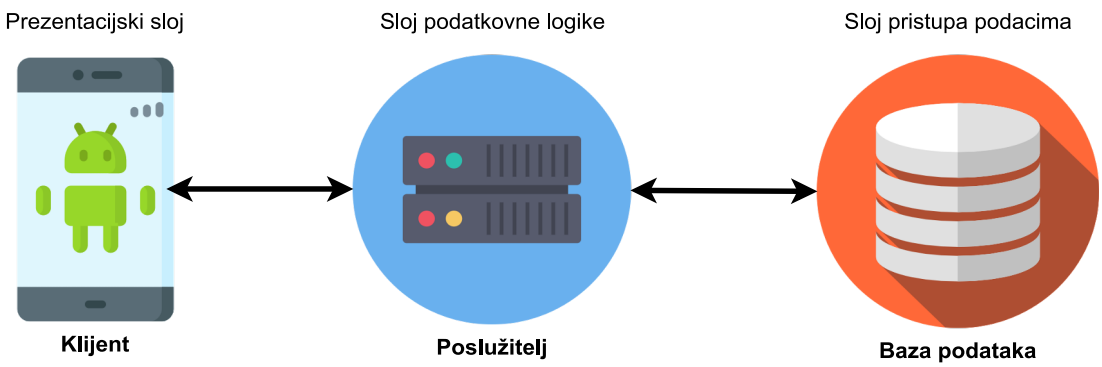
3. Arhitektura sustava

Prije implementacije samoga programskog rješenja, potrebno je na temelju funkcionalnih i nefunkcionalnih zahtjeva odabrati odgovarajuću arhitekturu sustava. Funkcionalni zahtjevi odnose se na to kako sustav treba raditi, dok se nefunkcionalni odnose na to koliko dobro sustav treba raditi [3]. Naravno, aplikacija mora biti održiva i skalabilna te ovi faktori također utječu na odabir arhitekture aplikacije. Uzevši u obzir sve faktore, donesena je odluka da će programsko rješenje „tipar“ koristiti troslojnu arhitekturu sustava koja se sastoji od tri sloja [4]:

1. prezentacijski sloj (engl. *presentation layer*)
2. sloj podatkovne logike (engl. *business logic layer*)
3. sloj pristupa podacima (engl. *data access layer*)

Ova je arhitektura odabrana jer su navedeni slojevi neovisni jedan o drugome i svaki ima vlastitu infrastrukturu. Znači, slojevi mogu biti ažurirani ili održavani neovisno o tome utječu li na ostale slojeve. Također, ova arhitektura omogućuje brži razvoj programskog rješenja, poboljšanu skalabilnost i sigurnost [5].

3.1. Skica arhitekture sustava



Slika 3.1 Skica arhitekture sustava

Slika 3.1 Skica arhitekture sustava) prikazuje glavne elemente pojedinog sloja arhitekture programskog rješenja „tipar“ :

1. **Prezentacijski sloj (klijent)** – Glavna je zadaća prezentacijskog sloja interakcija s korisnikom. Kao što se može zaključiti iz imena, u ovom sloju nalazi se korisničko sučelje čiji je glavni zadatak prikaz i prikupljanje podataka o korisnicima [5]. Klijent može biti *desktop*, mobilna ili *web*-aplikacija. U sklopu ovoga završnog rada razvijat će se samo mobilna aplikacija, ali moguća je i jednostavna implementacija neke od drugih klijentskih aplikacija zbog odabrane skalabilne troslojne arhitekture.
2. **Sloj podatkovne logike (poslužitelj)** – Riječ je o središnjem sloju sustava koji komunicira s klijentom i bazom podataka. Zadatak mu je koristiti podatkovnu logiku kako bi upravljao podacima. Kada poslužitelj dobije zahtjev od klijenta, obavi odgovarajuće radnje nad bazom podataka i klijentu vraća odgovor, ovisno o ishodu. Radnje nad bazom podataka obuhvaćaju brisanje, ažuriranje, čitanje i stvaranje novih podataka [6]. Poslužitelj koristi API (engl. *API – Application Programming Interface*) pristupne točke kako bi se otvorio prema klijentskim aplikacijama. Ovo omogućuje klijentu jednostavnu komunikaciju s poslužiteljem pomoću dokumentiranog sučelja [6].
3. **Sloj pristupa podacima (baza podataka)** – Zadaća ovog sloja je pohrana i upravljanje podacima. Kao što Slika 3.1 Skica arhitekture sustava) prikazuje, klijent ne može direktno komunicirati s bazom podataka, već se sva komunikacija odvija putem poslužitelja [5]. Ovo je jedna od prednosti troslojne arhitekture: poboljšana sigurnost.

3.2. Korištene tehnologije i programski jezici

S obzirom na to da je jedna od prednosti odabrane troslojne arhitekture mogućnost razvijanja različitih slojeva neovisno jedan o drugom, programsko rješenje „tipar“ koristi nekoliko različitih programskih jezika, razvojnih okruženja, radnih okvira (engl. *framework*) i tehnologija. U nastavku su objašnjene:

- temeljne komponente mobilne aplikacije, programski jezik i razvojno okruženje korišteno u klijentskom sloju
- radni okviri, programski jezik i razvojno okruženje korišteno u servisnom sloju
- modeli baze podataka i sustavi za pohranu podataka korišteni u podatkovnom sloju.

3.2.1. Klijentski sloj

Temeljne komponente android aplikacije

Postoje četiri glavne komponente android aplikacije, gdje svaka od njih može biti ulazna točka, a to su:

1. **Aktivnosti (engl. *Activities*)**[1] – Riječ je o ulaznim točkama koje predstavljaju korisničko sučelje i služe za interakciju s korisnikom. U pravilu aktivnost predstavlja jedan zaslon u aplikaciji. Većina mobilnih aplikacija ima više zaslona pa se može zaključiti da se sastoji i od više aktivnosti. Obično svaka aktivnost ima svoj odgovarajući izgled (engl. *layout*) koji određuje od kojih se elemenata sastoji sučelje i kako su raspoređeni. Vrsta izgled datoteke je XML (engl. *Extensible Markup Language*) i sastoji se od hijerarhije *View* i *ViewGroup* objekata.[2]
2. **Servisi (engl. *Services*)**[1] – Glavna karakteristika servisa je što nemaju korisničko sučelje. Koriste se da drže aplikaciju pokrenutu u pozadini kako bi obavila određeni posao. Primjer servisa je slušanje glazbe prilikom pretraživanja interneta. Servis za reprodukciju glazbe radi u pozadini, dok se druga aplikacija koristi u prvom planu.
3. ***Broadcast receivers***[1] – Ova je komponenta iznimno važna za integraciju s android operacijskim sustavom, kao i s ostalim aplikacijama. Također, kao i servisi, ova komponenta nema korisničko sučelje. *Broadcast receiveri* imaju ulogu osluškivati obavijesti o određenim događajima. Primjer sistemske obavijesti na koju se može pretplatiti je slaba baterija.
4. **Pružatelji sadržaja (engl. *Content providers*)**[1] – Služe za upravljanje pristupa podacima. Glavna im je zadaća omogućiti dijeljenje podataka između aplikacija. Pomoću ove komponente aplikacija, koja dohvaća podatke, ne mora znati gdje su podaci ili na koji način su spremljeni. Kao i *broadcast receiveri*, pružatelji usluga imaju značajnu ulogu u integraciji s ostalim aplikacijama.

[1] https://developer.android.com/guide/components/fundamentals
[2] https://developer.android.com/develop/ui/views/layout/declaring-layout

Svaka navedena komponenta ima svoj životni ciklus i specifičnu ulogu u izvođenju određene android aplikacije. Naravno, android aplikacija ne mora sadržavati sve ove komponente, već samo one koje su joj potrebne.

Ostale komponente koje su važne za programsko rješenje „tipar“ uključuju:

- **Fragmenti**[3] – Komponente koje su najsličnije aktivnostima. Kao i aktivnosti, imaju svoj izgled i predstavljaju grafičko sučelje za interakciju s korisnikom. Fragmenti se uvijek nalaze u aktivnosti, a njihove su najvažnije karakteristike modularnost i ponovno korištenje u nekoj drugoj aktivnosti. Sastavni su dio većine današnjih aplikacija zbog svoje fleksibilnosti.
- **Viewmodel**[4] – Razred (engl. *class*) s dva zadatka: enkapsulacija relevantne poslovne logike i opskrbljivanje korisničkog sučelja podacima. Najveća prednost *viewmodela* je što omogućava aplikaciji da radi besprijekorno ako dođe do promjene konfiguracije kao što je, na primjer, rotiranje ekrana.
- **Adapters**[5] – Služe za povezivanje izvora podataka s korisničkim sučeljem. Imaju zadaću uzimati podatke i odrediti na koji će se način prikazati. Ovo rezultira time da je korisničko sučelje intuitivno i lako za korištenje.
- **Intenti**[6] – Može ih se shvatiti kao način komuniciranja između komponenata. Intentima se obično šalje poruka koja je namijenjena specificiranoj komponenti s akcijom koju je potrebno izvršiti. Najčešće se koriste za pokretanje aktivnosti i servisa.

Klijentsku aplikaciju programskog rješenja „tipar“ predstavlja android mobilna aplikacija. Glavne komponente korištene za razvoj klijentske aplikacije bile su:

- programski jezik *Kotlin*
- razvojno okruženje Android Studio

[3] https://developer.android.com/guide/fragments
[4] https://developer.android.com/topic/libraries/architecture/viewmodel
[5] https://developer.android.com/reference/android/widget/Adapter
[6] https://developer.android.com/guide/components/intents-filters

Kotlin

Dugi niz godina glavni programski jezik za razvoj nativnih android aplikacija je bila Java. Dva važna događaja vezana uz stupanje *Kotlina* na scenu su konferencije Google I/O 2017 i Google I/O 2019. Google je 2017. godine objavio da razvoj android aplikacija službeno podržava *Kotlin* kao programski jezik [7]. Dvije godine kasnije, na istoj konferenciji, *Kotlin* postaje glavni programski jezik za razvoj mobilnih aplikacija [8]. Kada govorimo o razvoju mobilnih aplikacija, neke od prednosti *Kotlina* su:

- **Preglednost koda**[7] – Kod je jednostavniji nego u Javi, što dovodi do lakšeg razumijevanja tuđeg koda i trošenja manje vremena tipkajući vlastiti.
- **Interoperabilnost**[7] – *Kotlin* se može koristiti uz Javu u aplikacijama bez potrebe za migracijama.
- **Razvoj na više platformi**[7] – *Kotlin*, osim androida, podržava i razvoj iOS, *backend* i *web*-aplikacija. To omogućava korištenje istog koda, koji je neovisan o platformi, na više mjesta.
- **Sigurnost**[7] – Preglednost koda čini kod čitljivijim, što dovodi do manjeg broja pogrešaka. Osim toga, sofisticiranost *Kotlina* dolazi do izražaja kod eliminiranja opasnosti nultih (engl. *null*) vrijednosti. Prilikom deklariranja varijable mora se odrediti može li varijabla sadržavati nulte vrijednost.[8]
- **Velika zajednica**[7] **–** *Kotlin* je programski jezik otvorenog koda s visokom razinom podrške i doprinosima od strane zajednice. Prema Googleu, više od 60% najpopularnijih aplikacija na Google Play trgovini koristi *Kotlin*.

[7] https://kotlinlang.org/docs/android-overview.html
[8] https://kotlinlang.org/docs/null-safety.html

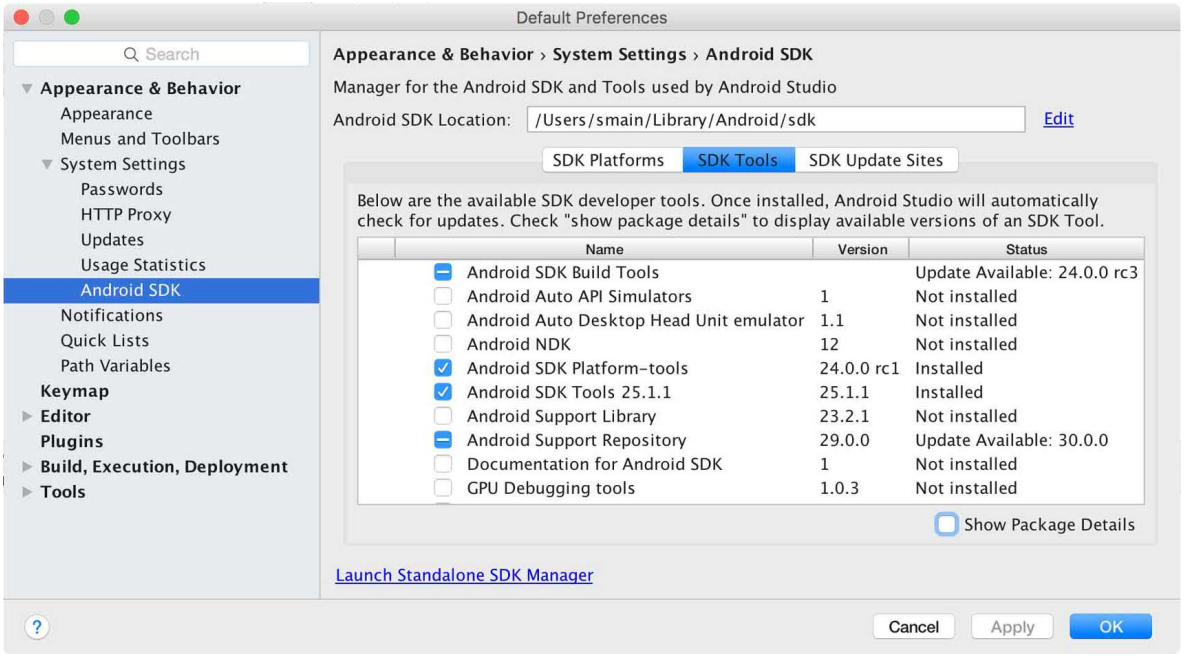
Android Studio

Android Studio[9] je službeno integrirano razvojno okruženje (engl. *Integrated Development Environment*, skraćeno IDE) za razvoj android aplikacija temeljen na JetBrainsovu IntelliJ IDEA softveru. Osim IntelliJovog moćnog uređivača koda i brojnih alata za razvoj, Android Studio pruža i dodatne značajke, posebno za razvoj android aplikacija kao što su: [9]

- Fleksibilan sustav za izgradnju projekta Gradle – Omogućuje brojne mogućnosti vezane uz verzioniranje i izgradnju projekta. Gradle je iznimno olakšao stvaranje aplikacija različitih verzija na temelju istog koda [9].
- Brz emulator sa svim potrebnim značajkama za testiranje.
- Podrška za razvoj aplikacija za sve vrste android uređaja.
- Dodatni radni okviri i alati za testiranje aplikacija.
- Podrška za NDK (engl. *NDK – native development kit*) – Skup alata koji omogućuje da dio projekta koristi nativni kod programskih jezika C i C++.
- Ugrađena podrška za Google Cloud Platform – laka integracija s Google servisima.
- Lint – Alat za pregledavanje koda koji pomaže identificiranju i popravljanju mogućih grešaka.

Uz Android Studio, potrebno je i spomenuti Android SDK (engl. *SDK – software development kit*) koji ima važnu ulogu u razvoju android aplikacija te uključuje emulatore, alate za ispravljanje grešaka (engl. *debugger*), dokumentaciju i ostale komponente važne za razvoj android aplikacija.[10] Slika 3.2 SDK Manager) prikazuje grafičko sučelje za upravljanje SDK komponentama (engl. *SDK Manager*) koji, osim dohvaćanja novih razvojnih alata, omogućava i ažuriranje trenutnih.

[9] https://developer.android.com/studio/intro
[10] https://www.techopedia.com/definition/4220/android-sdk



Slika 3.2 SDK Manager

3.2.2. Servisni sloj

Servisni sloj programskog rješenja „tipar“ predstavlja server koji pomoću API-ja opskrbljuje podacima klijentsku aplikaciju. Glavne komponente korištene za razvoj servisnog sloja bile su:

- programski jezik *C#*
- razvojno okruženje Visual Studio
- radni okvir *Entity Framework Core*
- radni okvir .NET Core.

C#

C# je moderan, objektno orijentiran programski jezik, stvoren od strane tvrtke Microsoft. Ima širok opseg primjene, koristi se u izradi video igara, *web*-aplikacija pa čak i mobilnih aplikacija. Nastao je na temelju jezika C, C++, Jave i Javascripta s namjerom da preuzme najbolje osobine spomenutih programskih jezika. Neke od najvažnijih značajki C# koje pomažu u izradi robusnih i izdržljivih aplikacija su:[11]

[11] https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

- Automatsko oslobađanje memorije (engl. *garbage collection*) – Služi za automatsko oslobađanje memorije koja je dodijeljena programu, ali je postala zauzeta nedostupnim nekorištenim objektima.
- Prilikom deklariranja varijabli može ih se označiti kao vrijednosti koje mogu biti nulte (engl. *null*).
- Rukovanje pogreškama (engl. *exceptions handling*) – Pruža strukturiran pristup otkrivanju i odgovaranju na pogreške.
- Lambda izrazi – Podrška za tehnike funkcionalnog programiranja.
- LINQ (engl. *Language Integrated Query*) – Sintaksa koja stvara obrazac za rad s podacima iz bilo kojeg izvora.
- Podrška za asinkrone operacije – Pruža sintaksu za izgradnju distribuiranih sustava.

Visual Studio

Visual Studio ima već ugrađenu podršku za brojne programske jezike kao što su C, C++ i C#, ali pruža i podršku za mnoge druge programske jezike instalacijom dodatnih komponenti i alata. Ovo razvojno okruženje može se koristiti za razvoj android, iOS i Windows aplikacija, kao i za razvoj *web*-aplikacija te servisa u oblaku (engl. *cloud services*). Kao i spomenuti programski jezik C#, također je stvoren od strane tvrtke Microsoft. Najvažnije karakteristike ovoga razvojnog okruženja je što podržava alat za dovršavanje koda (engl. *IntelliSense*) i što sadrži integrirani alat za ispravljanje pogrešaka (engl. *debugger*). Visual Studio je važan za razvoj programskog rješenja „tipar“ zbog svojeg integriranoga razvojnog okruženja za programski jezik SQL (engl. *Structured Query Language*) koji ima važnu ulogu u razvoju podatkovnog sloja. Ovime će se za razvoj koristiti manji broj različitih razvojnih okruženja, što će rezultirati manjom razinom kompleksnosti i većom učinkovitošću pri razvoju. [12]

Entity Framework Core

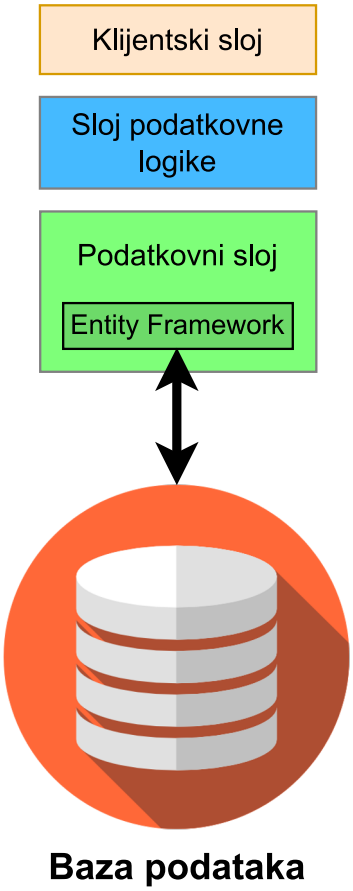
EF (engl. *Entity Framework*) Core je radni okvir otvorenog koda, stvoren od strane tvrtke Microsoft pa će u sklopu razvoja programskog rješenja „tipar“ imati ulogu ORM-a (engl. *ORM – Object – relational mapping*).[13] Ovo znači da:

[12] https://visualstudio.microsoft.com/vs/
[13] https://learn.microsoft.com/en-us/ef/core/

- omogućuje rad s bazom podataka pomoću .NET objekata
- uklanja potrebu za kodom pristupa podacima – čini projekt jednostavnijim.

S EF Core, pristup podacima je realiziran pomoću modela koji se sastoji od entiteta razreda i konteksta. Kontekst u ovom slučaju predstavlja sesiju s bazom podataka.

Ovaj radni okvir podržava mnogo baza podataka, od kojih je jedna SQL baza podataka koja je korištena za razvoj programskog rješenja „tipar“. Također, upiti prema bazi podataka su realizirani pomoću LINQ, što čini najkorištenije radnje nad bazom podataka jednostavnijima. Slika 3.3 Položaj EF u arhitekturi sustava) prikazuje kako se EF uklapa u aplikaciju. Svojim postojanjem pruža veću razinu apstrakcije kada je riječ o radu s podacima.[14]



Slika 3.3 Položaj EF u arhitekturi sustava

[14] https://www.entityframeworktutorial.net/what-is-entityframework.aspx

.NET Core

.NET je platforma namijenjena programerima koja se sastoji od alata, programskih jezika i biblioteka za razvoj brojnih različitih vrsta aplikacija.[15] .NET Core je besplatni radni okvir otvorenog koda sa širokim opsegom primjene, stvoren od tvrtke Microsoft. Stupio je na scenu kao nasljednik i poboljšana verzija .NET Frameworka. .NET Framework je platforma za razvoj aplikacija koje su tipizirane za operacijski sustav Windows dok .NET Core, osim Windowsa, podržava i Linux te Mac OSX. Također, .NET Core je optimiziran za potrebe modernih aplikacija, a zbog svoje arhitekture pruža bolje performanse i veću skalabilnost od .NET Frameworka.[16] Jedna od važnijih obilježja .NET Corea je modularnost. Dakle, sastoji se od niza osnovnih NuGet paketa, dok se svi ostali potrebni paketi mogu dodati neovisno jedan o drugome [10]. NuGet je esencijalan alat za razvoj aplikacija na .NET platformi koji služi za dijeljenje koda, odnosno biblioteka.[17]

U programskom rješenju „tipar“ .NET Core služi za realizaciju *web* API-ja pomoću REST (engl. *REST – Representational state transfer*). REST predstavlja stil arhitekture za izgradnju *web*-servisa, odnosno standard koji omogućuje lakšu komunikaciju između računalnih sustava na internetu.[18] API predstavlja sučelje, odnosno niz radnji koje se mogu okinuti, ali za programsko rješenje „tipar“ je važniji *web* API koji predstavlja niz radnji koje se mogu okinuti koristeći HTTP (engl. *HTTP – Hypertext Transfer Protocol*).[19] U programskom rješenju „tipar“ *web* API izvršava okinute radnje nad SQL bazom podataka. *Web* API radi na način da izloži svoje pristupne točke, gdje svaka predstavlja određenu radnju. Osim putanje pristupne točke, potreban je i prikladan HTTP glagol (engl. *HTTP verb*), ovisno o vrsti radnje koja se želi izvršiti.[20] Programsko rješenje „tipar“ koriste sljedeće HTTP glagole za:

- stvaranje – POST
- brisanje – DELETE
- ažuriranje – PUT
- dohvaćanje – GET

---

[15] https://dotnet.microsoft.com/en-us/apps/aspnet
[16] https://www.interviewbit.com/blog/net-core-vs-net-framework/
[17] https://learn.microsoft.com/en-us/nuget/what-is-nuget
[18] https://www.codecademy.com/article/what-is-rest
[19] https://dotnet.microsoft.com/en-us/apps/aspnet/apis
[20] https://learn.microsoft.com/en-us/training/modules/build-web-api-aspnet-core/2-what-is-rest-in-aspnet

3.2.2.1 Firebase

Firebase je platforma za razvoj aplikacija, stvorena od strane tvrtke Google. Omogućuje laku integraciju s android, iOS i *web* aplikacijama i pruža brojne alate, servise i usluge za razvoj i održavanje aplikacija. Programsko rješenje „tipar“ koristi sljedeće Firebase usluge:

- **Cloud Firestore**[21] – Cloud Firestore je fleksibilna i skalabilna NoSQL baza podataka. NoSQL predstavlja nerelacijsku bazu podataka. Stvorena je i održavana od strane tvrtke Google, a realizirana je pomoću računalstva u oblaku (engl. *cloud computing*). Podaci su organizirani u dokumente (engl. *documents*) koji se sastoje od polja (engl. *fields*) mapiranih na vrijednosti (engl. *values*). Organizacija podataka je omogućena pomoću kolekcija (engl. *collection*) koje se sastoje od više dokumenata. Koristeći Firestore, ne treba se brinuti o tome na kojem su fizičkom poslužitelju podaci spremljeni ili o implementaciji radnji za upravljanje podacima. Programsko rješenje „tipar“ koristi Firestore za pohranu podataka o „sobama za čavrljanje“ (engl. *chat*) te za pohranu podataka o korisnicima.
- **Firebase Authentication**[22] – Programsko rješenje „tipar“ koristi ovu uslugu za prijavu u sustav te stvaranje i upravljanje korisničkim računom. Isti autentifikacijski sustav može se koristiti na više platformi, što omogućuje programskom rješenju „tipar“ laku implementaciju nove klijentske aplikacije.
- **Firebase Cloud Messaging**[23] – Servis za slanje i primanje obavijesti koji se može koristiti za ciljanje jednog uređaja ili grupe uređaja. Programsko rješenje „tipar“ koristi ovaj servis kako bi obavijestilo korisnike o novim događajima pomoću *push* obavijesti (engl. *push notification*).

3.2.3. Podatkovni sloj

Za pohranu i pristup podacima u programskom rješenju „tipar“ korišteni su sljedeći sustavi:

- Microsoft SQL Server
- Firebase Firestore.

[21] https://firebase.google.com/docs/firestore
[22] https://firebase.google.com/docs/auth
[23] https://firebase.google.com/docs/cloud-messaging

Microsoft SQL Server

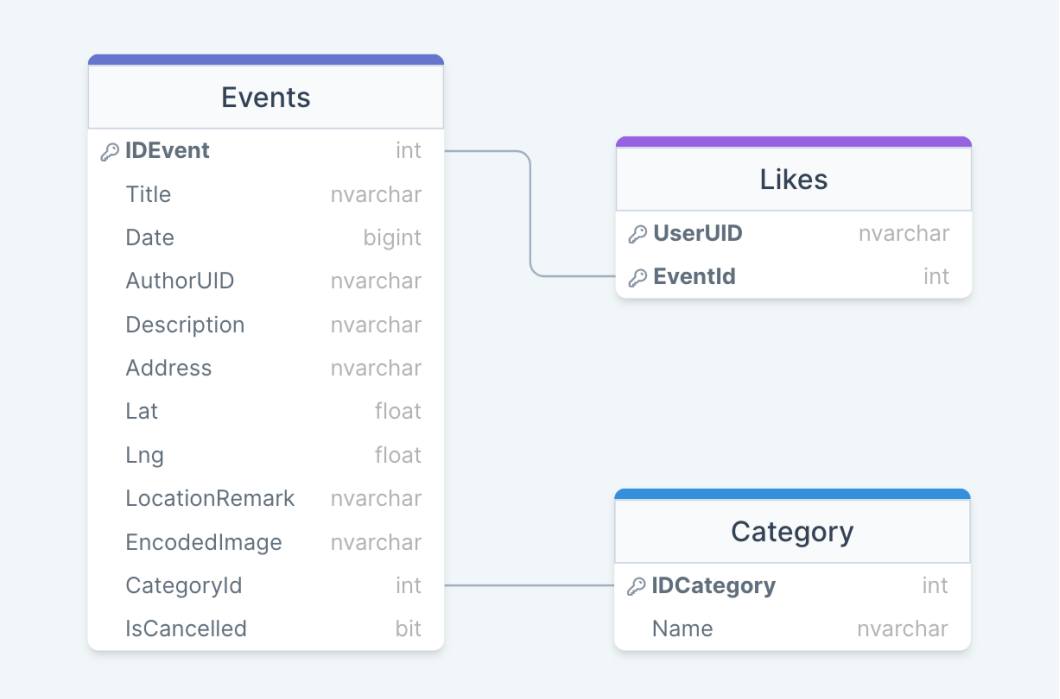
Microsoft SQL Server je relacijski sustav za pohranu podataka. Glavna karakteristika ovog sustava je način na koji su podaci organizirani. Dakle, podaci su organizirani u tablice koje se sastoje od redova. Tablice se mogu međusobno povezati, što dovodi do izbjegavanja redundancije podataka. Glavni jezik za upravljanje podacima SQL Servera je T-SQL (engl. *Transact – Structured Query Language*) koji, osim funkcionalnosti SQL-a, pruža i neke dodatne mogućnosti [11]. Uz SQL Server, osim alata za upravljanje podacima, dolaze i alati za analitiku te alati za poslovnu inteligenciju (engl. *Business Intelligence*, skraćeno: *BI*) [12].

3.2.3.1 Model baze podataka

Programsko rješenje „tipar“ koristi dva već spomenuta sustava za pohranu podataka. Podaci o događajima spremaju se u Microsoft SQL Serveru, dok se podaci o korisnicima spremaju u Firebase Firestore bazi podataka.

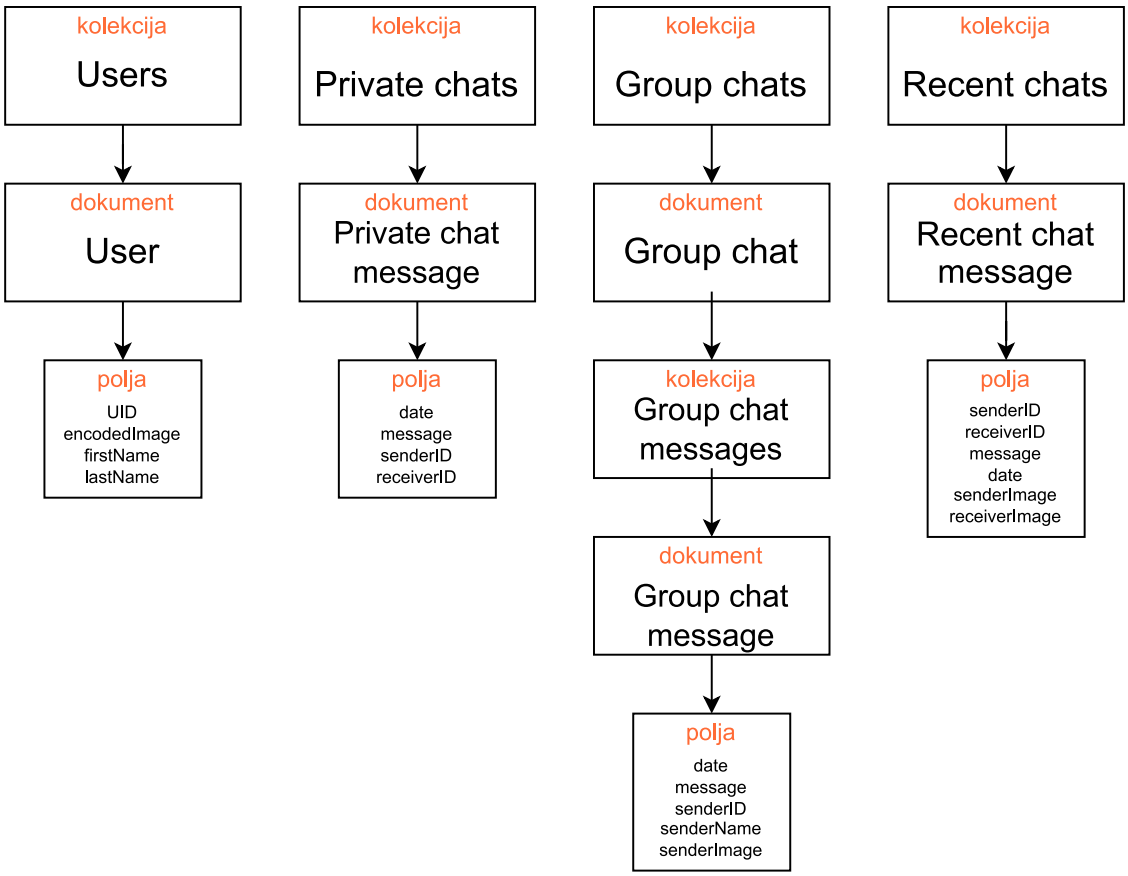
Relacijska baza, koja se nalazi na SQL Serveru, napravljena je pomoću EF Core. Za stvaranje baze, tablica i relacija korišten je prvo-kod (engl. *code-first*) pristup. U ovom pristupu prvo se određuju razredi, njihova obilježja i odnosi između razreda. EF Core, na temelju razreda, stvara tablice koje se sastoje od redaka i stupaca. Red u tablici predstavlja jednu instancu razreda, odnosno entitet, dok stupac predstavlja obilježje, odnosno atribut.[24] Odnose između entiteta i njihove atribute može se najbolje prikazati ER dijagramom (engl. *Entity relationship diagram*, skraćeno *ER* dijagram). ER dijagram prikazuje odnose između tablica i logičku strukturu baze podataka.[25] Kako prikazuje Slika 3.4 ER dijagram baze podataka), relacijska baza podataka sastoji se od ukupno tri tablice; „Events“, „Category“ i „Likes“. Glavna tablica ove baze je „Events“ koja služi za pohranu svih podataka o događajima. Ova tablica ima jedan strani ključ koji služi za povezivanje s tablicom „Category“, koja pohranjuje podatke o kategorijama događaja. Za pohranjivanje podataka o označavanju događaja sa „Sviđa mi se“ služi tablica „Likes“.

[24] https://binaryterms.com/difference-between-entity-and-attribute-in-database.html
[25] https://www.guru99.com/er-diagram-tutorial-dbms.html



Slika 3.4 ER dijagram baze podataka

Nerelacijska baza podataka Firebase Firestore služi za pohranu podataka o korisnicima i „sobama za čavrljanje“. Kao što Slika 3.5 Organizacija Firebase Firestore kolekcija) prikazuje, Firebase Firestore pohranjuje podatke organizirane u četiri kolekcije: „Users“, „Private chats“, „Group chats“ i „Recent chats“. Kolekcija „Users“ čuva podatke registriranih korisnika, kao što su njihovo ime, prezime i slika korisničkog računa. Ostale tri kolekcije služe za pohranjivanje podataka o „čavrljanju“. „Private chats“ čuva podatke o „privatnim čavrljanjima“ između korisnika, gdje su isključivo dva sudionika, dok „Group chats“ čuva podatke o porukama između grupe korisnika. Kolekcija „Recent chats“ čuva podatke o posljednjim porukama „soba za čavrljanje“ te ima važnu ulogu u poboljšavanju korisničkog iskustva kada je riječ o funkcionalnosti „čavrljanja“.



Slika 3.5 Organizacija Firebase Firestore kolekcija

4. Implementacija programskog rješenja

Implementacija programskog rješenja počinje izradom baze podataka i servisnoj sloja. Dizajn baze podataka i svojstva servisnog sloja stvoreni su na temelju funkcionalnosti programskog rješenja „tipar“. Prvi je korak generiranje baze podataka na temelju definiranih razreda pomoću radnog okvira EF. Nakon toga, uslijedio je razvoj servisnog i klijentskog sloja.

4.1. Klijentski dio

Klijentski dio programskog rješenja „tipar“ obuhvaća android mobilnu aplikaciju. Mobilna aplikacija ima ulogu predstavnika ovoga programskog rješenja jer služi za komunikaciju s korisnikom. Važno je da aplikacija dobro prezentira programsko rješenje dobrim korisničkim iskustvom i lakom navigacijom.

4.1.1. Korisničko sučelje

Dizajn korisničkog sučelja predstavlja važan korak u implementaciji programskog rješenja. Korisničko sučelje treba biti intuitivno te se korisnicima mora omogućiti da brzo pronađu traženu funkcionalnost aplikacije. Također, važno je da aplikacija ima smisleno pozicionirane elemente na ekranu i da ne koristi velik broj boja. Kod stvaranja korisničkog sučelja android aplikacija, Google preporučuje služenje sustavom za dizajniranje zvanim *Material Design* [13]. Ovaj sustav za dizajniranje sastoji se od skupa smjernica, komponenti i alata koji podržavaju nabolje praksu dizajna korisničkog sučelja.[26] Za izradu klijentske aplikacije programskog rješenja „tipar“ korišten je *Material Design*.

Prilikom pokretanja aplikacije, prvo što korisnik vidi je pozdravni zaslon (engl. *splash screen*). Kao što Slika 4.1 Pozdravni zaslon mobilne aplikacije) prikazuje, na ovom zaslonu prikazan je logo i ime aplikacije te animacija koja signalizira korisniku da se aplikacija učitava. Aplikacija koristi tamnu temu u kombinaciji s varijantama boje cijan (engl. *cyan*) za isticanje elemenata.

[26] https://m3.material.io/

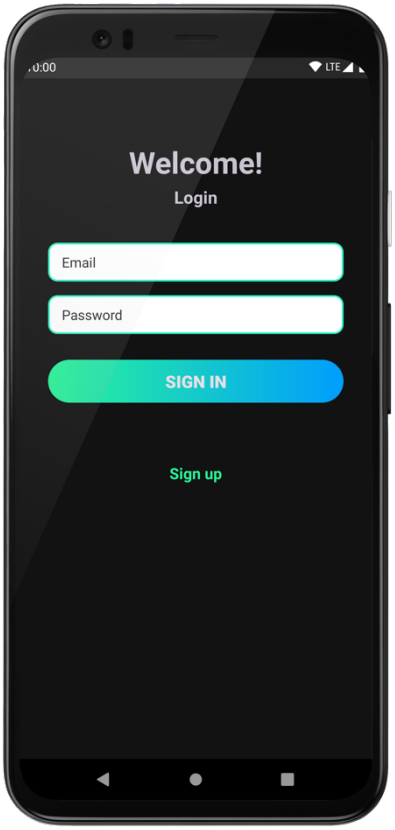
Slika 4.1 Pozdravni zaslon mobilne aplikacije

4.1.2. Funkcionalnosti aplikacije

Funkcionalnosti mobilne aplikacije programskog rješenja „tipar“ odnose se na upravljanje korisničkom računom i događajima te pretplaćivanje za obavijesti određene kategorije događaja. Također, funkcionalnosti uključuju i „čavrljanje“ između dvoje korisnika, kao i „čavrljanje“ s grupom korisnika.

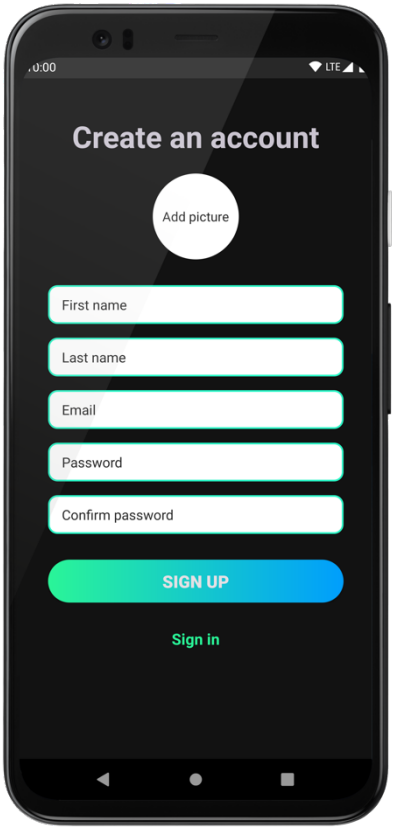
4.1.2.1 Kreiranje korisničkog računa i prijava

Nakon pozdravnog zaslona, ako korisnik nije prijavljen u sustav, prikazuje se zaslon za prijavu u sustav. Kao što Slika 4.2 Prijava u sustav) prikazuje, glavni elementi zaslona su pozdravna poruka, polja za unos elektroničke pošte i lozinke te gumb „Prijavi se“ (engl. *Sign in*). Korisnik mora unijeti točnu adresu elektroničke pošte i lozinku korisničkog računa kako bi se prijavio u sustav. Ako korisnik unese podatke na temelju kojih ga nije moguće autentificirati, dobit će odgovarajuću informacijsku poruku.



Slika 4.2 Prijava u sustav

Zaslon za prijavu također sadrži tekst „Registracija“ (engl. *Sign up*) pa ako korisnik klikne na njega, prikazuje se zaslon za kreiranje računa kao što se vidi na Slika 4.3 Kreiranje korisničkog računa). Korisnik prilikom kreiranja korisničkog računa mora unijeti sljedeće podatke: ime, prezime, adresu elektroničke pošte, lozinku i ponovljenu lozinku. Također, korisnik treba postaviti i sliku korisničkog računa. Za sliku korisničkog računa odabire sliku koja je spremljena na mobilnom uređaju. Ako korisnik ne unese sve potrebne podatke, ne postavi sliku korisničkog računa ili unese pogrešno ponovljenu lozinku, prikazat će mu se odgovarajuća informacijska poruka.



Slika 4.3 Kreiranje korisničkog računa

Zaslon za prijavu u sustav te stvaranje korisničkog računa su dva fragmenta koja se nalaze unutar aktivnosti `EntryActivity`. Navigacija između fragmenata je realizirana pomoću navigacijske XML datoteke koju prikazuje Kôd 4.1 Konfiguracija fragmenata). U ovoj datoteci navedeni su fragmenti vezani uz stvaranje računa i prijavu u sustav te navigacijske radnje koje služe za transakcije između fragmenata. Izmjenu fragmenata unutar aktivnosti obavlja objekt razreda `NavController` inicijaliziran u aktivnosti `EntryActivity`.

```
<?xml version="1.0" encoding="utf-8"?>
<navigation
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/entry_nav_graph"
    app:startDestination="@id/signInFragment">

    <fragment
        android:id="@+id/signInFragment"
        android:name="hr.algebra.tipar.SignInFragment"
```

```
            android:label="fragment_sign_in"
            tools:layout="@layout/fragment_sign_in" >
            <action

android:id="@+id/action_signInFragment_to_signUpFragment"
                app:destination="@id/signUpFragment" />
        </fragment>
        <fragment
            android:id="@+id/signUpFragment"
            android:name="hr.algebra.tipar.SignUpFragment"
            android:label="fragment_sign_up"
            tools:layout="@layout/fragment_sign_up" >
            <action

android:id="@+id/action_signUpFragment_to_signInFragment"
                app:destination="@id/signInFragment"
                app:popUpTo="@id/signInFragment"
                app:popUpToInclusive="true" />
        </fragment>
</navigation>
```

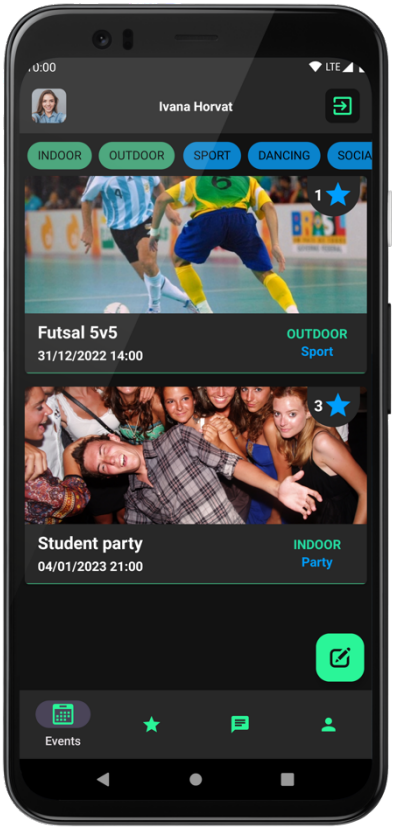
Kôd 4.1 Konfiguracija fragmenata

4.1.2.2 Pregled događaja

Nakon uspješne prijave u sustav, korisniku se prikazuju objave korisnika o događajima kako prikazuje Slika 4.4 Pregled događaja). Objave korisnika prikazane su jedna ispod druge, a svaka sadrži: sliku događaja, naslov, datum i vrijeme, kategoriju i tip događaja te broj korisnika koji su označili objavu sa „Sviđa mi se“. Prilikom klika na objavu, prikazuje se novi zaslon s detaljnijim podacima o događaju.

Navigacija aplikacije nalazi se na samom dnu zaslona i sadrži četiri izbora: „Događaji“ (engl. *Events*), „Omiljeni događaji“ (engl. *Favorite events*), „Čavrljanje“ (engl. *Chat*) i „Korisnički račun“ (engl. *Profile*).

U donjem desnom kutu zaslona predviđenog za prikaz objava, iznad navigacije, nalazi se gumb za stvaranje nove objave, dok se iznad liste objava nalazi filter objava prema tipu i kategoriji. Iznad filtera nalazi se slika korisničkog računa te ime trenutno prijavljenog korisnika uz gumb za odjavljivanje.



Slika 4.4 Pregled događaja

Za prikaz liste objava događaja koristi se *RecyclerView*[27]. Ovu je komponentu potrebno opskrbiti podacima o događajima i definirati kako će svaka objava izgledati. *RecyclerView* ima ulogu učinkovitoga upravljanja prikazivanjem podataka na zaslonu.

Kako bi se još više povećale performanse aplikacije, programsko rješenje „tipar“ koristi *ListAdapter*[28] za prikazivanje podataka unutar *RecyclerViewa*. Problem *RecyclerViewa* je to što kad se lista objava ažurira, *RecyclerView* ponovno prikaže svaku objavu na zaslonu iako je možda došlo do promjene u samo jednoj objavi [14]. *ListAdapter* rješava ovaj problem svojom implementacijom koja služi za uspoređivanje liste događaja. Kako Kôd 4.2 Mehanizam za uspoređivanje) prikazuje, potrebno je definirati radi li se o dva ista događaja te jesu li sva svojstva tih dvaju događaja identična. Ovime svaki put kad se ažurira lista objava, *ListAdapter* svojim algoritmima osigurava učinkovito prikazivanje ažurirane liste.

[27] https://developer.android.com/develop/ui/views/layout/recyclerview
[28] https://developer.android.com/reference/androidx/recyclerview/widget/ListAdapter

```
private class EventDiffCallBack :
DiffUtil.ItemCallback<EventAndCategory>() {
    override fun areItemsTheSame(
        oldItem: EventAndCategory,
        newItem: EventAndCategory
    ): Boolean =
        oldItem.event.idEvent == newItem.event.idEvent

    override fun areContentsTheSame(
        oldItem: EventAndCategory,
        newItem: EventAndCategory
    ): Boolean =
        oldItem == newItem
}
```

Kôd 4.2 Mehanizam za uspoređivanje

Navigacija i podaci o trenutno prijavljenom korisniku uvijek su prisutni na zaslonu, neovisno o tome na kojem modulu aplikacije se korisnik trenutno nalazi. Ovaj zaslon sadržava aktivnost `MainActivity` koja sadrži:

- navigaciju na dnu zaslona
- fragment trenutno odabranog modula u sredini zaslona
- podatke o korisniku i gumb za odjavu na vrhu zaslona.

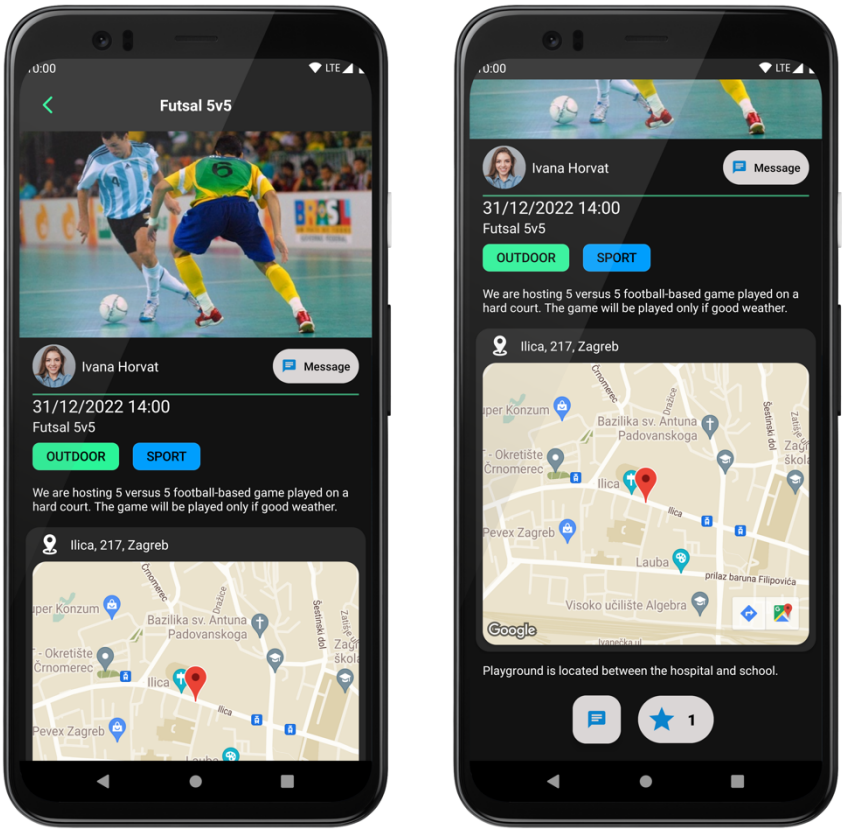
Za navigaciju se koristi implementacija donje navigacije sustava za dizajniranje *Material Design* zvana *BottomNavigationView*.

4.1.2.3 Pregled detalja pojedinog događaja

Ako je korisnik zainteresiran za pojedini događaj, klikom na njega prikazuje mu se novi zaslon s detaljima o događaju. Kao što Slika 4.5 Detaljni prikaz događaja) prikazuje, zaslon s detaljnim informacijama događaja uključuje:

- ime, prezime i sliku korisničkog računa autora događaja
- gumb za „čavrljanje“ s autorom
- sliku događaja
- datum i vrijeme događaja
- naslov događaja
- tip i kategoriju događaja

- opis događaja
- adresu događaja i kartu s označenom lokacijom
- dodatnu napomenu vezanu uz lokaciju događaja
- gumb za grupno „čavrljanje“ svih zainteresiranih za događaj
- gumb za označavanje događaja sa „Sviđa mi se“ s brojem korisnika koji su označili da im se sviđa događaj.



Slika 4.5 Detaljni prikaz događaja

Za prikaz lokacije događaja koristi se *Maps SDK,* stvoren od strane tvrtke Google. Ovo omogućuje integraciju s aplikacijom Google Maps, što znači da korisnici imaju mogućnost brzo i jednostavno doći do navigacije za pojedini događaj.

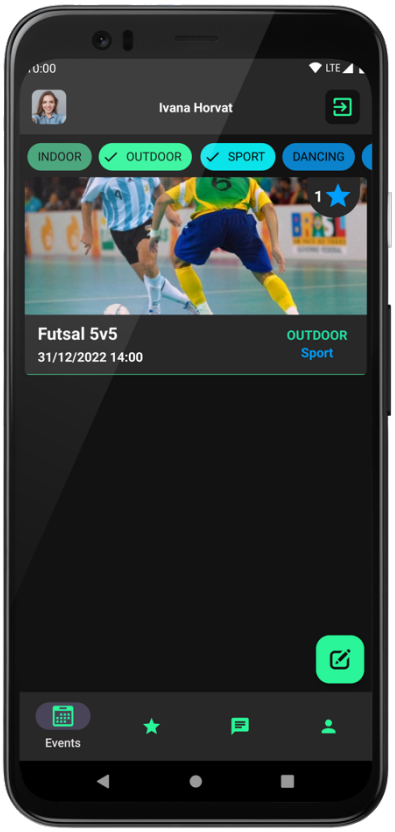
4.1.2.4 Filtriranje objava

Korisnik ima mogućnost jednostavno filtrirati objave prema željenom kriteriju. Objave je moguće filtrirati na temelju:

- kategorije
- tipa

- kategorije i tipa istodobno.

Ovisno o korisnikovom odabiru kriterija filtriranja, lista objava se ažurira (Slika 4.6 Filtriranje objava). Kategorije i tipovi objava prikazani su pomoću komponente *Chip* te su organizirani unutar komponente *ChipGroup*.



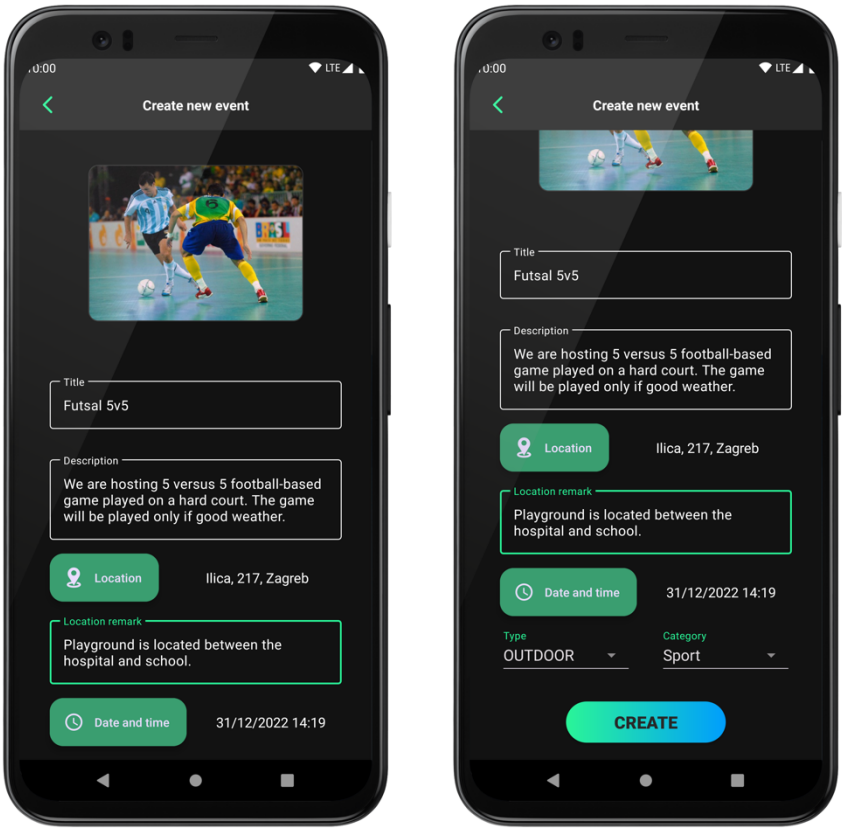
Slika 4.6 Filtriranje objava

4.1.2.5 Stvaranje objave o događaju

Prilikom pregledavanja objava, korisnik ima mogućnost stvaranja objave o događaju klikom na gumb za stvaranje nove objave. Zaslon za stvaranje nove objave prikazuje Slika 4.7 Stvaranje nove objave). Prilikom stvaranja nove objave korisnik:

- odabire sliku događaja
- unosi naslov događaja
- unosi opis događaja
- postavlja lokaciju događaja
- unosi napomenu o lokaciji
- postavlja datum i vrijeme događaja

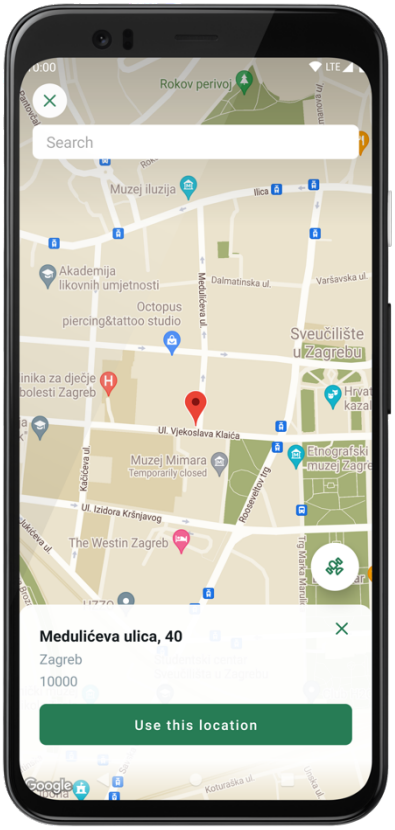
- odabire tip događaja
- odabire kategoriju događaja.



Slika 4.7 Stvaranje nove objave

Klikom na gumb „Lokacija“ za postavljanje lokacije, korisniku se prikazuje novi zaslon koji prikazuje Slika 4.8 Odabir lokacije). Korisnik ima mogućnost upisati adresu u polje predviđeno za unos adrese ili na prikazanoj karti kliknuti i tako postaviti lokaciju označenu crvenim markerom. Kada se korisnik odluči za lokaciju, svoj unos završava klikom na gumb „Koristi ovu lokaciju“ (engl. *Use this location*) te nastavlja stvaranje objave.

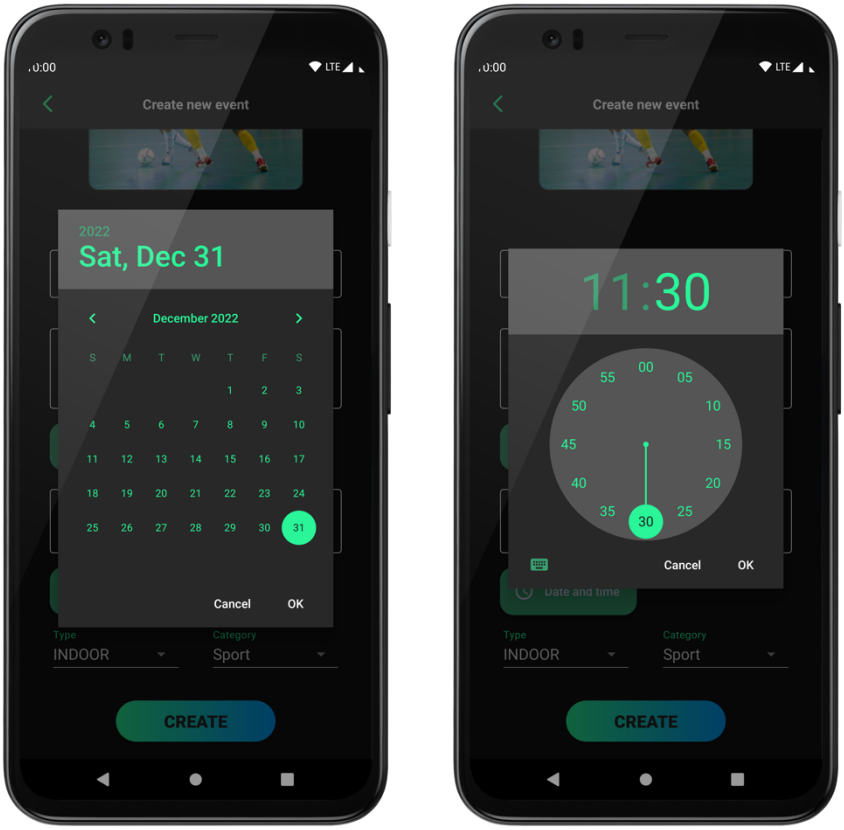
Korisnik odabire tip i kategoriju događaja iz padajućih izbornika. Postoje dva izbora za tip događaja: „Zatvoreni prostor“ (engl. *Indoor*) i „Na otvorenom“ (engl. *Outdoor*). Postoji nekoliko opcija za odabir kategorije događaja. S obzirom na to da se podaci o kategorijama pohranjuju u bazi podataka, na jednom centralnome mjestu, one su otvorene za nadogradnju u budućnosti.



Slika 4.8 Odabir lokacije

Za odabir datuma koristi se komponenta *DatePickerDialog,* dok se za odabir vremena događaja koristi komponenta *TimePickerDialog.* Kao što Slika 4.9 Odabir datuma i vremena) prikazuje, korisnik prvo odabire datum i nakon toga vrijeme događaja. Ova su dva unosa povezana pa korisnik mora uspješno postaviti datum i vrijeme kako bi imao mogućnost stvoriti događaj.

Nakon što su svi potrebni podaci uneseni, korisnik završava stvaranje događaja klikom na gumb „Stvori“ (engl. *Create*). Aplikacija poziva API programskog rješenja „tipar“ te zahtijeva povratnu informaciju (engl. *Callback*) o uspješnosti stvaranja novog događaja. Na temelju povratne informacije, korisniku se prikazuje primjerena poruka.



Slika 4.9 Odabir datuma i vremena

4.1.2.6 Omiljeni događaji

Modul „Omiljeni događaji“ omogućuje korisniku pregledavanje svih događaja koje je označio sa „Sviđa mi se“. Ova funkcionalnost omogućuje laku dostupnost svih događaja za koje je korisnik iskazao interes. Klikom na objavu, koja se sastoji od slike i osnovnih obilježja događaja, korisniku se prikazuje zaslon s detaljnijim prikazom događaja. Slika 4.10 Omiljeni događaji) prikazuje dizajn zaslona za prikaz omiljenih događaja.

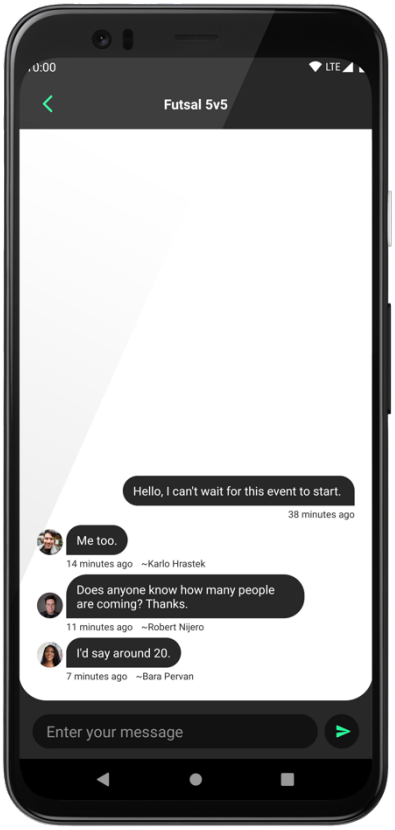
Slika 4.10 Omiljeni događaji

4.1.2.7 Čavrljanje

Korisnici mobilne aplikacije programskog rješenja „tipar“ imaju mogućnost korištenja „čavrljanja“ na dva načina:

- „Grupno čavrljanje“
- „Čavrljanje“ između dva korisnika.

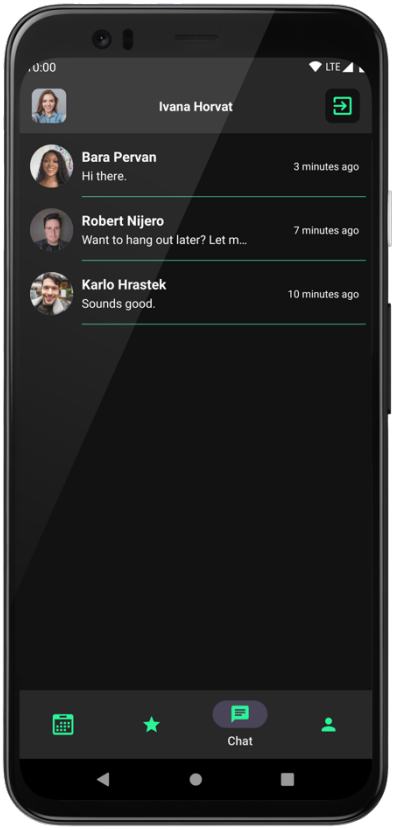
Gumbi koji omogućuju korištenje „čavrljanja“ nalaze se na zaslonu koji prikazuje detalje događaja. „Grupno čavrljanje“ ima ulogu povezati i omogućiti komunikaciju svih korisnika zainteresiranih za određeni događaj. „Čavrljanje“ između dva korisnika služi za komunikaciju između korisnika kojima tema razgovora nije striktno vezana uz određeni događaj. Kao što Slika 4.11 Soba za čavrljanje) prikazuje, svaka poruka „čavrljanja“ sadrži: tekst poruke, vrijeme slanja te ime i sliku korisničkog računa pošiljatelja.



Slika 4.11 Soba za čavrljanje

Slika 4.12 Čavrljanje prikazuje jedan od četiri modula ove mobilne aplikacije. Do ovog zaslona moguće je doći koristeći navigaciju prisutnu na dnu zaslona. Ovaj zaslon sadrži informacije o nedavnim „čavrljanjima“, jednu ispod druge, organizirane u listu. Informacije uključuju: posljednju poruku, vrijeme slanja, ime, prezime i sliku korisničkog računa sudionika. Klikom na prostor, koji prikazuje navedene podatke određenog posljednjeg „čavrljanja“, korisniku se prikazuje novi zaslon: „soba za čavrljanje“ s odabranim sudionikom.

Podaci o „čavrljanju“ pohranjuju se u bazu podataka *Firebase Firestore*. Za prikazivanje poruka koristi se slušač (engl. *listener*). Ovaj slušač radi na način da ga registriramo za praćenje promjena u kolekciji određene sobe za čavrljanje. Nakon što dođe do promjene, slušač se okida i dostavlja aplikaciji informacije do koje je promjene došlo. Na temelju toga, prostor za prikaz poruka se ažurira. Prilikom slanja poruke, osim kolekcije koja pohranjuje podatke o porukama određene „sobe za čavrljanje“, ažurira se i kolekcija koja pohranjuje podatke o posljednjim porukama soba za čavrljanje.

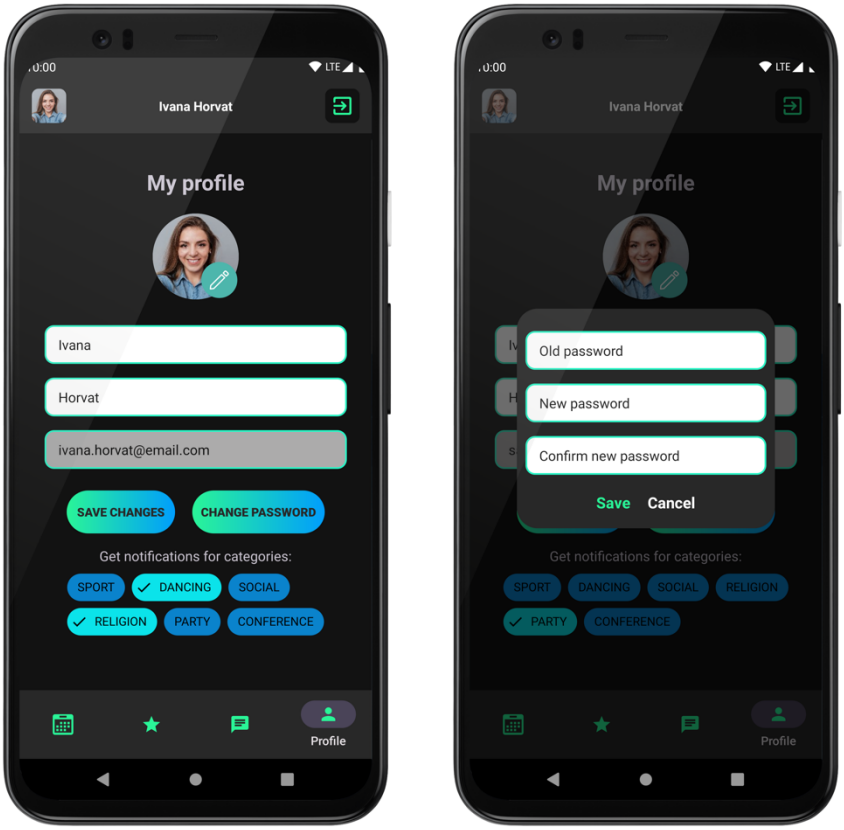


Slika 4.12 Čavrljanje

4.1.2.8 Uređivanje korisničkog računa

Modul „Korisnički račun“ omogućuje korisniku uređivanje slike korisničkog računa, imena i prezimena. Dizajn korisničkog sučelja zaslona za pregled i uređivanje korisničkog računa prikazuje Slika 4.13 Pregled i uređivanje korisničkog računa). Nakon što korisnik unese sve željene izmjene, klikom na gumb „Spremi promjene“ (engl. *Save changes*) potvrđuje svoj odabir.

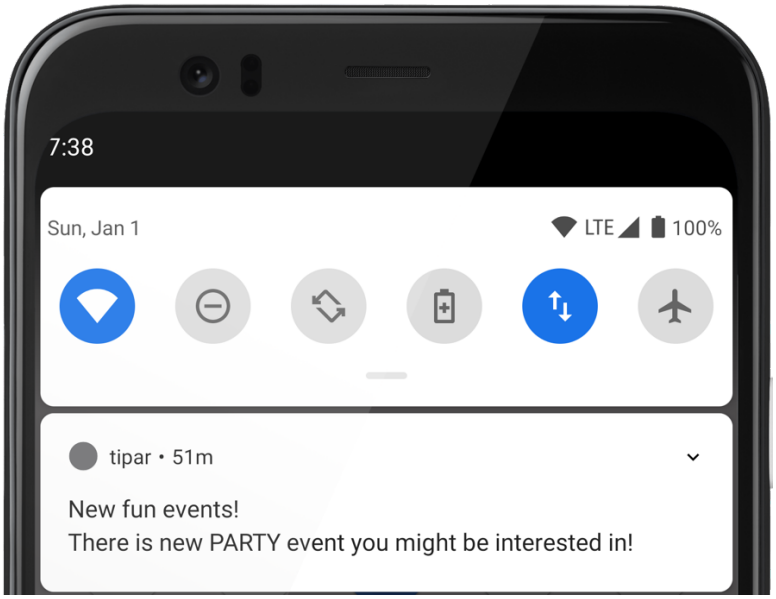
Klikom na gumb „Promijeni lozinku“ korisniku se otvara dijalog i od njega se traži unos lozinke koju želi promijeniti, nove lozinke i ponovljene nove lozinke. Ako je korisnik unio sve točne tražene podatke, promjenu lozinke završava klikom na gumb „Spremi“ (engl. *Save*).



Slika 4.13 Pregled i uređivanje korisničkog računa

4.1.2.9 Obavijesti o događajima

Klijentska aplikacija programskog rješenja „tipar“ omogućuje korisnicima pretplatu na dobivanje obavijesti o novim događajima. Moguće je pretplatiti se na jednu ili više kategorija za koje korisnik želi primati obavijesti. Pretplaćivanje, kao i otkazivanje pretplate, odvija se u modulu „Korisnički račun“. Korisnik se pretplaćuje klikom na gumb s nazivom kategorije, a nakon uspješne pretplate, gumb mijenja svoja vizualna svojstva signalizirajući korisniku da se uspješno pretplatio. Korisnik prima obavijesti u obliku *push* obavijesti koje predstavljaju informativnu poruku izvan korisničkog sučelja mobilne aplikacije. Za razliku od običajnih obavijesti sustava android, koje su stvorene unutar aplikacije na uređaju, *push* obavijesti mogu se shvatiti kao poruke poslane izvan uređaja koje ciljaju određenu aplikaciju. Aplikacija tada prikazuje korisniku obavijest na temelju primljene poruke. U slučaju programskog rješenja „tipar“ pošiljatelj obavijesti je servisni sloj, odnosno server koji cilja mobilnu aplikaciju. Primjer obavijesti prikazuje Slika 4.14 Push obavijest). Slanjem i primanjem obavijesti upravlja servis *Firebase Cloud Messaging*. Za pretplatu se koristi mehanizam kojim se pretplaćuje na određenu temu (engl. *topic*), gdje tema predstavlja naziv kategorije.



Slika 4.14 Push obavijest

Prije korištenja servisa *Firebase Cloud Messaging,* potrebno ga je deklarirati u manifestu mobilne aplikacije, kao što prikazuje Kôd 4.3 Deklaracija servisa za obavijesti).

```
<service
    android:name=".fcm.MessagingService"
    android:exported="false">
    <intent-filter>
        <action
android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

Kôd 4.3 Deklaracija servisa za obavijesti

4.2. Servisni dio

Servisni dio programskog rješenja „tipar“ sastoji se od *web* API-ja. Glavna mu je zadaća povezati bazu podataka s klijentskim slojem. Radi na način da izloži pristupne točke na temelju kojih klijentska aplikacija poziva radnje koje želi izvršiti nad bazom podataka.

4.2.1. Pristupne točke

Glavna komponenta servisnog sloja koji poslužuje podatke je kontroler. Uloga kontrolera je zaprimiti zahtjev i izvršiti traženu radnju nad bazom podataka. Broj kontrolerskih razreda obično odgovara broju tablica nad kojima vršimo radnje. Kontroleri se sastoje od metoda, odnosno pristupnih točaka gdje svaka predstavlja određenu akciju. Tablica 4.1 Pristupne točke) prikazuje kontrolere i pristupne točke od kojih se sastoje, uz kratki opis radnji za koje se koriste. Ovisno o vrsti zahtjeva, podaci se predaju pristupnim točkama u tijelu zahtjeva ili u putanji pristupne točke. Pristupna točka, koja vrši ažuriranje događaja, koristi oba načina predaje podataka, parametar za identificiranje entiteta se predaje u putanji, dok tijelo sadrži podatke ažuriranog događaja.

Tablica 4.1 Pristupne točke

Kontroler	HTTP verb	Pristupna točka	Opis
Like	POST	/api/Like/LikeEvent	Označavanje događaja sa „Sviđa mi se“
	POST	/api/Like/UnlikeEvent	Poništavanje akcije “Sviđa mi se”
Category	GET	/api/Categorty	Dohvaćanje svih kategorija
	GET	/api/Categorty/{id}	Dohvaćanje pojedine kategorije na temelju predanog identifikatora
Events	GET	/api/Events	Dohvaćanje svih događaja
	POST	/api/Events	Stvaranje novog događaja
	GET	/api/Events/UpcomingEvents	Dohvaćanje nadolazećih događaja
	GET	/api/Events/{id}	Dohvaćanje detalja događaja na temelju predanog identifikatora
	GET	/api/Events/Liked/{userUID}	Dohvaćanje događaja označenih sa “Sviđa mi se” određenog korisnika
	PUT	/api/Events/{id}	Ažuriranje događaja
	DELETE	/api/Events/{id}	Brisanje događaja

Kôd 4.4 Primjer pristupne točke) prikazuje metodu pristupne točke koja služi za dohvaćanje kategorije na temelju predanog identifikatora. Atributom iznad metode definira se tip

zahtjeva te, opcionalno, parametri i njihovi tipovi. Navedena metoda koristi *GET* zahtjev. Radnje nad bazom izvršavaju se asinkrono, što omogućuje izvršavanje više radnji u isto vrijeme. Dvije su ključne riječi programskog jezika C#, koje se koriste u asinkronom programiranju, *async* i *await*. Ključnom riječju *async* označava se metoda koja planira koristiti ključnu riječ *await* da asinkrono izvrši određenu radnju. Ključna riječ *await* omogućuje izvršavanje radnje bez zaustavljanja glavne dretve i nastavak izvršavanja programa na glavnoj dretvi nakon što se radnja izvrši.

Također, prilikom vraćanja odgovara klijentu, metoda vraća povratnu informaciju o uspješnosti radnje koja se trebala izvršiti. Povratne poruke govore je li radnja, koja se trebala izvršiti, bila uspješna, neuspješna ili pak pruža neku povratnu informaciju.

```
[HttpGet("{id:int}")]
public async Task<ActionResult<Category>> GetCategory(int id)
{
   try
   {
     var result = await categoryRepository.GetCategory(id);
     if (result == null)
     {
     return NotFound();
     }
     return result;
   }
   catch (Exception)
   {
     return
StatusCode(StatusCodes.Status500InternalServerError,
"Error retrieving data from the database");
   }
}
```

Kôd 4.4 Primjer pristupne točke

4.2.2. Dizajn servisnog sloja

Za rad s bazom podataka korišten je *EF Core*. Stvaranje samog servisnog sloja odvijalo se u obliku nekoliko slijednih procesa. Prvi je korak bio stvaranje razreda koji predstavljaju tablice i veza između njih. Kao što Kôd 4.5 Razred kategorija) prikazuje, potrebno je

definirati svojstva razreda i atributima postaviti dodatne opcije koje su važne za stvaranje baze podataka.

```
public class Category
 {
  [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
  public int IDCategory { get; set; }

  [Required]
  public string Name { get; set; }
 }
```

Kôd 4.5 Razred kategorija

Atributom `[Key]` daje se do znanja da prilikom stvaranja baze, ovaj stupac predstavlja ključ. `[DatabaseGenerated]` atribut govori na koji način baza podataka generira vrijednost za određeno svojstvo. Koristeći atribut `[Required]` eliminira se mogućnost nulte vrijednosti.

Nakon definiranja razreda entiteta, sljedeći je korak stvaranje razreda koji predstavlja kontekst baze podataka. Ovaj razred ima iznimno važnu ulogu servisnog sloja jer se brine o radnjama nad podacima baze podataka. Također, ovaj razred koristi se prilikom stvaranja baze podataka. Sljedeći kod definira svojstva kontekstnog razreda, odnosno koje razrede kontekst obuhvaća pa će se na temelju njih kreirati tablice:

```
public DbSet<Event> Events { get; set; }

public DbSet<Category> Categories { get; set; }

public DbSet<Like> Likes { get; set; }
```

Kako bi aplikacija servisnog sloja znala koji kontekstni razred koristiti s kojom instancom servera baze podataka, u ulaznoj točki aplikacije potreban je sljedeći kod:

```
builder.Services.AddDbContext<ApplicationDbContext>(o =>
o.UseSqlServer(builder.Configuration.GetConnectionString("DBC
onnection")));
```

Potrebno je stvoriti migracije nad bazom podataka. Migracija služi kako bi se na temelju kontekstnog razreda stvorio nacrt baze podataka. Naredba za stvaranje migracije je `Add-Migration`. Nakon stvorene migracije, primjenjuje se nad bazom, koristeći naredbu `Update-Database`. Obe naredbe unose se u *Package Manager Console*.

Prije stvaranja pristupnih točaka, potrebno je stvoriti sučelja te implementacije sučelja koja se bave upravljanjem podacima spremljenih u bazi podataka. Kontroleri će primati ove implementacije sučelja u svojim konstruktorima kako bi obavljali radnje nad bazom podataka. Primjer sučelja za upravljanje tablicom kategorija prikazuje Kôd 4.6 Sučelje za repozitorij kategorija).

```
public interface ICategoryRepository
    {
        Task<IEnumerable<Category>> GetCategories();
        Task<Category> GetCategory(int categoryID);
        Task<Category> AddCategory(Category categoryToAdd);
        Task<Category> UpdateCategory(Category category);
        Task DeleteCategory(int categoryID);
    }
```

Kôd 4.6 Sučelje za repozitorij kategorija

5. Nedostaci i moguća poboljšanja programskog rješenja

Prilikom usporedbe programskog rješenja „tipar“ sa sličnim postojećim rješenjima utvrđeno je kako ovo rješenje ne posjeduje jednu karakteristiku koju posjeduju slična rješenja. Dakle, slična rješenja uz mobilnu android aplikaciju imaju i *web*-aplikaciju, dok se programsko rješenje „tipar“ trenutno sastoji samo od jedne klijentske aplikacije. Za daljnji napredak predlaže se razvoj *web* i mobilne iOS aplikacije. Također, predlaže se mogućnost dodavanja događaja u kalendarsku aplikaciju. Ovom integracijom s kalendarom korisnici bi mogli upravljati podsjetnicima za događaje koji su im važni. Trenutno klijentska aplikacija podržava samo engleski jezik, prijedlog za daljnji razvoj je implementacija višejezičnosti.

Prilikom razvoja primijećeno je da ponekad nisu dovoljna samo dva kriterija filtriranja. Prijedlog je da se korisnicima, osim po kategoriji i tipu, omogući filtriranje po lokaciji. S obzirom na to da programsko rješenje „tipar“ pohranjuje podatke o geolokaciji pojedinog događaja, moguće je poboljšanje implementacija novog modula. Ovaj bi modul prikazivao sve nadolazeće događaje na zemljopisnoj karti te bi korisnicima omogućio pregled događaja čija je lokacija na određenoj Zemljinoj površini.

Slična rješenja, osim s adresom elektroničke pošte i lozinkom, omogućuju i autentifikaciju koristeći korisnički račun društvenih mreža kao što su *Facebook*, *Twitter* i *Google*. Ova bi integracija uvelike pojednostavila autentifikaciju programskog rješenja „tipar“ pa je to jedan od prijedloga za daljnji razvoj.

Zaključak

U današnje vrijeme, postoji velik broj društvenih mreža koje po svojim svrhama i sadržaju sliče jedna drugoj. Osim svoje mogućnosti da povežu ljude, često se koriste kao izvor zabavnog sadržaja te zbog toga korisnici i provode dio slobodnog vremena koristeći ih. Zbog iznimno velikog broja aktivnih korisnika, koji raste iz dana u dan, društvene mreže svojom mogućnošću dijeljena sadržaja stvaraju mjesto koje često korisnici koriste kao izvor informacija.

Programsko rješenje „tipar“ prepoznalo je problem s kojim se susreću ljudi koji žele učinkovito ispuniti svoje slobodno vrijeme pa je usvojilo korisna svojstva aktualnih društvenih mreža. Također, uočilo je i glavne nedostatke kao što je nestrukturiran sadržaj. Ovo je programsko rješenje usmjereno i prilagođeno ljudima koji žele učinkovito ispuniti svoje slobodno vrijeme i steći nova poznanstva sa sebi sličim ljudima. Društvene mreže često su način umrežavanja korisnika u virtualnim zajednicama. Problem generičkih društvenih mreža je što je sadržaj određene teme često raspršen po mnoštvu različitih zajednica. Svojim postojanjem, programsko rješenje „tipar“ zadovoljava ljudsku potrebu za socijaliziranjem, a uz to pruža organiziran sadržaj. Uz dijeljenje i pregledavanje sadržaja, „tipar“ omogućuje korisnicima i komunikaciju. Tri najvažnije karakteristike programskog rješenja „tipar“ zbog kojih se ističe u mnoštvu društvenih mreža su: dobivanje personaliziranih obavijesti, filtriranje objava i dodavanje lokacije događaja. Nakon razvoja programskog rješenja, prepoznati su nedostaci te su ponuđeni prijedlozi za daljnji razvoj.

Otkad nas je zadesila pandemija koronavirusa, koja je rezultirala distanciranjem, ljudi su bili primorani pronaći novi način socijaliziranja. Veliki dio njih upravo se odlučio za upoznavanje drugih korištenjem različitih društvenih mreža. S obzirom na štetu nanesenu mentalnom zdravlju pojedinca zbog epidemioloških mjera, ovo programsko rješenje nameće se kao alat koji će ljudima omogućiti povratak izgubljenog vremena.

Popis kratica

API	*Application Programming Interface*	aplikacijsko programsko sučelje
BI	*Business intelligence*	poslovna inteligencija
EF	*Entity framework*	radni okvir za entitete
ER	*Entity relationship*	odnos između entiteta
HTTP	*Hypertext Transfer Protocol*	protokol za prijenos hiperteksta
IDE	*Integrated development environment*	integrirano razvojno okruženje
LINQ	*Language Integrated Query*	jezično integrirani upit
NDK	*Native development kit*	nativni alati za razvoj softvera
ORM	*Object–relational mapping*	objektno relacijsko mapiranje
REST	*Representational state transfer*	prijenos reprezentativnog stanja
SDK	*Software development kit*	alati za razvoj softvera
SQL	*Structured Query Language*	strukturirani jezik upita
SWOT	*Strengths, Weaknesses,*	snage, slabosti,
	Opportunities, Threats	prilike, prijetnje
T-SQL	*Transact - Structured Query Language*	transakcijski strukturirani jezik upita
XML	*Extensible Markup Language*	proširivi jezik za označavanje

Popis slika

Popis tablica

Popis kôdova

Literatura

[1] Antičević, Vesna. "Učinci pandemija na mentalno zdravlje". *Društvena istraživanja* 30, br. 2 (2021): 423-443. https://doi.org/10.5559/di.30.2.12

[2] Alan Sarsby; SWOT Analysis: A Guide to Swot for Business Studies Students; Spectaris Limited; 2016

[3] Dabbagh, M. and Lee, S.P. (2014) "An approach for integrating the prioritization of functional and nonfunctional requirements", *The Scientific World Journal*, 2014, pp. 1–13. Dostupno na: https://doi.org/10.1155/2014/737626.

[4] Raj, P., Raman, A.C. and Subramanian, H. (2017) Architectural patterns: Uncover essential patterns in the most indispensable realm of enterprise architecture. Birmingham, UK: Packt Publishing.

[5] IBM Cloud Education (2020) *What is three-tier architecture*, *IBM*. Dostupno na: https://www.ibm.com/cloud/learn/three-tier-architecture (Pristupljeno: 15.11.2022).

[6] IBM Cloud Education (2020) *What is an application programming interface (API)?*, *IBM*. Dostupno na: https://www.ibm.com/cloud/learn/api (Pristupljeno: 16.11.2022).

[7] Phillips, B., Marsicano, K. and Stewart, C. (2018) *Android programming the Big Nerd Ranch Guide*. Atlanta: Big Nerd Ranch.

[8] Braun, M. (2022) *Celebrating 5 years of Kotlin on Android*, *Android Developers Blog*. Dostupno na: https://android-developers.googleblog.com/2022/08/celebrating-5-years-of-kotlin-on-android.html (Pristupljeno: 20.11.2022).

[9] Leiva, A. (2016) *Kotlin for Android developers: Learn Kotlin the easy way while developing an Android app*. Victoria, British Columbia: Leanpub.

[10] Landwerth, I. (2021) *Introducing .NET core*, *.NET Blog*. Dostupno na: https://devblogs.microsoft.com/dotnet/introducing-net-core/ (Pristupljeno: 29.12.2022).

[11] Clark, D. (2022) *SQL vs T-SQL: Understanding the differences*, *Dataquest*. Dostupno na: https://www.dataquest.io/blog/sql-vs-t-sql/ (Pristupljeno: 28.12.2022).

[12] Hughes, A. and Stedman, C. (2019) *What is Microsoft SQL server? A definition from whatis.com*, *Data Management*. TechTarget. Dostupno na: https://www.techtarget.com/searchdatamanagement/definition/SQL-Server (Pristupljeno: 28.12.2022).

[13] Kaseb, K. (2021) *Android apps design and User Experience*, *Medium*. Software Development. Dostupno na: https://medium.com/kayvan-kaseb/android-apps-design-and-user-experience-15f2bfc368e3 (Pristupljeno: 22.12.2022).

[14] Anubhav (2021) *Android: ListAdapter, a better implementation for the Recyclerview*, *Medium*. Geek Culture. Dostupno na: https://medium.com/geekculture/android-listadapter-a-better-implementation-for-the-recyclerview-1af1826a7d21 (Pristupljeno: 28.12.2022).