MOBILNA APLIKACIJA ZA UPRAVLJANJE STUDENTSKIM OGRANKOM

Pavković, Nikola

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

MOBILNA APLIKACIJA ZA UPRAVLJANJE STUDENTSKIM OGRANKOM

Završni rad

Nikola Pavković

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 05.09.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Nikola Pavković
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4112, 28.07.2017.
OIB studenta:	41043811283
Mentor:	Izv. prof. dr. sc. Josip Balen
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Mobilna aplikacija za upravljanje studentskim ogrankom
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	05.09.2021.
Datum potvrde ocjene Odbora:	08.09.2021.

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

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 16.09.2021.

Ime i prezime studenta:	Nikola Pavković
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4112, 28.07.2017.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Mobilna aplikacija za upravljanje studentskim ogrankom**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Josip Balen

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:

Sadržaj

1. UVOD

Velik broj udruga, tvrtki i organizacija u današnje vrijeme još uvijek ovisi isključivo o ljudskom radu u području organizacije događaja, komunikacije sa strankama i upravljanja ljudskim resursima. Mnoge od navedenih aktivnosti često su repetitivne i oduzimaju dragocjeno vrijeme koje bi se u suprotnom moglo utrošiti u poboljšanje rada i stvaranje novih idejnih rješenja unutar udruge. Iznimno brz napredak tehnologije u posljednjem desetljeću rezultirao je mnogim programskim alatima koji, ako su korišteni zajedno unutar smislene cjeline, omogućuju automatizaciju brojnih poslova poput organizacije događaja, registracije novih korisnika, predstavljanje rada udruge korisnicima i obavještavanje korisnika o novim aktivnostima unutar udruge.

Programsko rješenje koje se ističe kao dobar alat za automatizaciju spomenutih aktivnosti upravo su mobilne aplikacije. Značajke poput integracije geografske lokacije, spremanja podataka u vanjsku bazu podataka, jednostavno obavještavanje korisnika o promjenama unutar aplikacije i brojne biblioteke koje razvojna sučelja mobilnih aplikacija nude, mogu se iskoristiti u svrhu izrade aplikacije koja će korisniku omogućiti brz, učinkovit i vjerodostojan dohvat potrebnih informacija i dijeljenje istih. Unatoč tomu što već postoji velik broj aplikacija koje nude određenu razinu automatizacije navedenih poslova, vrlo je teško pronaći aplikaciju koja postiže ravnotežu između dizajna, intuitivnog korisničkog sučelja [1] i funkcionalnosti. Postavlja se pitanje kako ponuditi korisniku iskustvo koje ga neće prezasititi informacijama i mogućnostima te će mu predstaviti samo neophodne funkcionalnosti zbog kojih će ostati motiviran koristiti aplikaciju. Odgovor na ovo pitanje jest analiza potreba svakog pojedinačne udruge te izrada personalizirane aplikacije. Nažalost, izrada i dizajn aplikacije koja je prilagođena za potrebe pojedinačne udruge skup je i dugotrajan proces zbog čega na tržištu postoje aplikacije koje nude određene opće mogućnosti potrebne većini udruga i organizacija. Takve aplikacije često su neovisne o mobilnoj platformi i operativnom sustavu te su izrađene u višeplatformskim (*engl. multi-platform*) [2] razvojnim sučeljima poput Fluttera [3] i Xamarina [4] čime su ograničene na funkcionalnosti koje su zajedničke mobilnim platformama za koje se izrađuju.

Zbog prethodno navedenih razloga, aplikacija za organizaciju studentskog ogranka IEEE-a, koja je ujedno i tema ovog rada, bit će izrađena nativno [5], točnije, isključivo za Android mobilnu platformu te će, koristeći geografski prikaz lokacije, Google Firebase [6] baze podataka i druge tehnologije i biblioteke koje Android platforma nudi, predstaviti automatizirano rješenje glavnih

organizacijskih problema ogranka. Analizom rada osječkog studentskog ogranka utvrđeno je kako su ti problemi upravo obavještavanje članova ogranka o novim aktivnostima i njihova izrada te dugo trajanje registracije velikog broja korisnika u sustav IEEE-a.

1.1. Zadatak završnog rada

Cilj ovog rada jest predstaviti osnovne elemente Android mobilne platforme, Android Studio IDE-a (*engl. IDE – Integrated development environment*) i programskog jezika Kotlin [7] te prikazati rad i svrhu korištenih biblioteka, alata i aplikacijskih programskih sučelja. Cilj je također predstaviti već postojeća programska rješenja za organizaciju različitih dobrovoljnih ili poslovnih društava. Navedene teme i zadaci rada bit će obrađeni na primjeru izrade prilagođene aplikacije za organizaciju IEEE studentskog ogranka Osijek, točnije, ogranka globalne organizacije inženjera i studenata s tehničkih područja s ciljem promicanja tehnologije i inovacije za boljitak čovječanstva. Aplikacija vodstvu studentskog ogranka nudi mogućnost izrade događaja i prikaza istih ostalim članovima ogranka, upoznavanje članova s prednostima i radom ogranka te pojednostavljenje procesa registracije.

2. AUTOMATIZACIJA RADA UDRUGE

2.1. Analiza potreba pojedinačne udruge

Analiza potreba neke udruge jest postupak koji zahtijeva veoma duboko i detaljno razumijevanje rada udruge što naposlijetku omogućuje implementaciju određenih potrebnih specifičnosti u programsko rješenje. Općenito, analiza potreba udruge višeslojan je proces koji osim izrade tehničkog rješenja zahtijeva i socijalnu analizu rada udruge.

Prvi korak prilikom automatizacije elemenata rada udruge jest analizirati specifične potrebe koje ta udruga ima. To predstavlja dugotrajan proces nadgledanja rada i bilježenja poslova koji su repetitivni ili oduzimaju veliku količinu vremena te, kao takvi, predstavljaju kandidate za potencijalnu automatizaciju. Već vrlo rano u procesu analize rada pojavljuje se pitanje važnosti ljudskih odnosa i socijalnih interakcija unutar udruge i koliko je ustvari ljudska interakcija prilikom obavljanja nekih poslova esencijalna za osjećaj zajedništva i dobrodošlice te motivaciju članova.

Na primjeru rada IEEE studentskog ogranka Osijek primijećeno je kako veliki broj studenata nakon uspješno obavljenog upoznavanja s radom ogranka i velikom količinom interesa za učlanjenje, odustaje od učlanjivanja zbog složenog postupka prijave. Prilikom gotovo svakog novog učlanjivanja jedan od članova vodstva primoran je provesti novog potencijalnog člana kroz postupak registracije koji u prosjeku traje 10-15 minuta. Ovakav način rada održiv je kada je u pitanju manja udruga, no jednostavno nije prilagodljiv na veći priljev članova prouzročen rastom udruge i zahtijeva automatizaciju. Također je utvrđeno kako je vrlo teško pronaći prikladan medij za informiranje postojećih članova o novim aktivnostima unutar ogranka. Kao logično rješenje navedenog problema ističu se socijalne mreže poput Facebooka i Instagrama na kojima bi se objavljivale najave aktivnosti, no detaljnom analizom zaključeno je kako većina gostiju prikupljenih ovim putem nisu aktivni članovi ogranka već vanjski posjetitelji koji su u velikom broju zainteresirani za učlanjenje. Unatoč tomu što spomenuti način rada rezultira priljevom novih članova, također stvara distancu između ogranka i već registriranih članova jer im najave aktivnosti nisu ciljano predstavljene nekim službenim medijem ili članovi ne koriste navedene socijalne mreže i nemaju pristup najavama. Na taj način aktivni članovi gube interes i ne razvijaju svoju karijeru unutar samog ogranka zbog čega ogranak gubi ljudske resurse potrebne za organizaciju i razvoj te samim time postaje manje aktivan i nedinamičan. Cilj je postići ravnotežu između priljeva novih članova te edukacije i razvoja već registriranih aktivnih članova.

Osim administrativnih problema registracije i pronalaska medija za komunikaciju, ističe se i složen proces upoznavanja članova i zainteresiranih vanjskih posjetitelja s radom ogranka koji također oduzima veliku količinu vremena i izravno ovisi o kvalitetnim ljudskim resursima.

Sljedeći korak analize rada jest odabrati koje od stavki poslovanja i rada uistinu ima smisla automatizirati. Kako su registracija i obavještavanje članova o novim aktivnostima iznimno repetitivni i osobna komunikacija s članovima prilikom obavljanja tih poslova nije od velike važnosti već, štoviše, može negativno utjecati na reputaciju ogranka i motiviranost potencijalnih i aktivnih članova, ove je poslove potrebno automatizirati. S druge strane upoznavanje zainteresiranih studenata s radom ogranka delikatan je posao s mnoštvom detalja i temelji se na kvalitetnoj i vjerodostojnoj komunikaciji koju je iznimno teško postići nekim tehničkim rješenjem. Unatoč tomu što takav oblik komunikacije zahtijeva veliku količinu organizacije, pripreme i vještina, nema potrebe za automatizacijom jer je u ovom slučaju štetna za rad ogranka.

Posljednji korak analize potreba udruge jest pronaći adekvatno tehničko rješenje koje je prikladno za automatizaciju rada udruge. Naravno, ovaj proces ovisi o samom načinu organizacije neke udruge, o količini i frekvenciji aktivnosti, načinu registracije članova, statusu i usmjerenju članova i sličnim karakteristikama. Za potrebe IEEE studentskog ogranka Osijek zaključeno je kako je najbolje tehničko rješenje mobilna aplikacija za Android platformu jer omogućuje rješenje svih prepoznatih problema unutar jednostavnog i intuitivnog korisničkog sučelja.

Analogno primjeru IEEE studentskog ogranka sličan se postupak analize može primijeniti i na druge udruge uz prilagodbu odabira tehničkog rješenja koje je prikladno za automatizaciju poslova. Najoptimalniji ishod ovakve analize može biti pronalazak već gotovog tehničkog rješenja što čini korak analize neophodnim jer može dovesti do uštede resursa potrebnih za izradu prilagođene aplikacije ili web stranice.

Izrada prilagođenog rješenja, osim automatizacije poslova udruge, ima i mnogobrojne druge koristi. U slučaju da postoji određen broj udruga koje imaju slične zahtjeve, moguće je uz manje prilagodbe dizajna i funkcionalnosti primijeniti već gotovo tehničko rješenje na te udruge što se u krajnosti može monetizirati. Osim monetizacije, izrada tehničkog rješenja može se iskoristiti u svrhu organizacije posebne aktivnosti kao u slučaju IEEE studentskog ogranka Osijek. S obzirom da se radi o udruzi studenata tehničkog opredjeljenja, izrada aplikacije osječkog ogranka zamišljena je kao

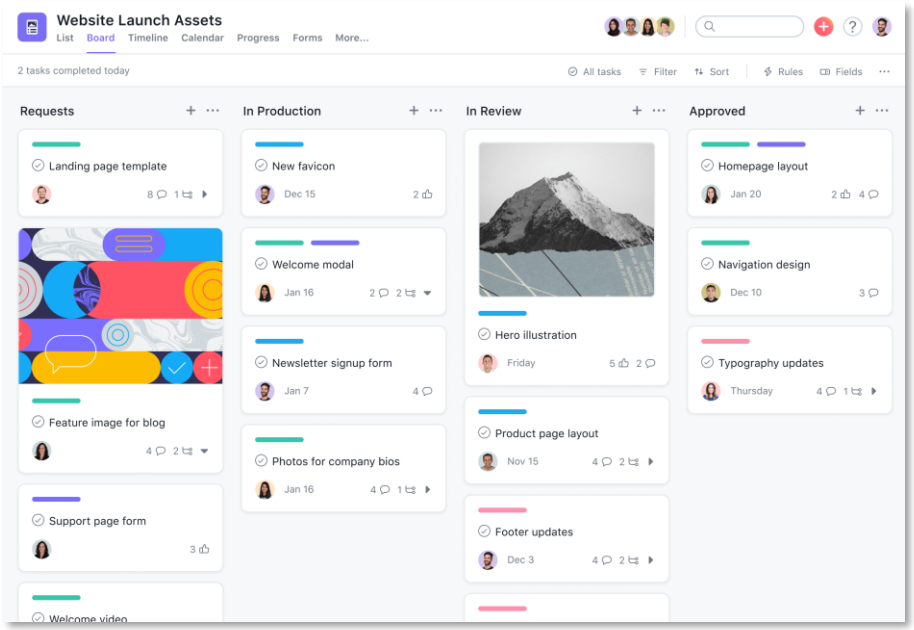
višegodišnji projekt na čijem će razvoju sudjelovati studenti članovi ogranka. To omogućuje članovima karijerni napredak i razvoj socijalnih i tehničkih vještina.

Razvoj ovakve aplikacije objedinjuje znanja iz mnogih disciplina zbog čega je moguće raspodijeliti posao razvoja studentima i članovima ovisno o njihovim afinitetima. Dizajn, izrada korisničkog sučelja i poboljšanje korisničkog iskustva, programiranje pozadinskog rada aplikacije, rad s bazama podataka i marketing samo su neke od disciplina koje je potrebno obuhvatiti izradom aplikacije.

2.2. Postojeća rješenja za organizaciju udruga

Unatoč činjenici da je izrada prilagođene aplikacije za svaku konkretnu udrugu savršeno rješenje za automatizaciju specifičnih poslova koje ta udruga obavlja, ono sa sobom nosi i velik trošak vremena, resursa i potrebu za kvalificiranim razvojnim programerima i dizajnerima. Upravo zbog toga na tržištu već postoje određena tehnička rješenja koja automatiziraju neke opće prepoznate poslove. Manji je broj ovakvih aplikacija i web stranica čak i modularan te omogućuje prilagodbu sučelja i funkcionalnosti specifičnim potrebama.

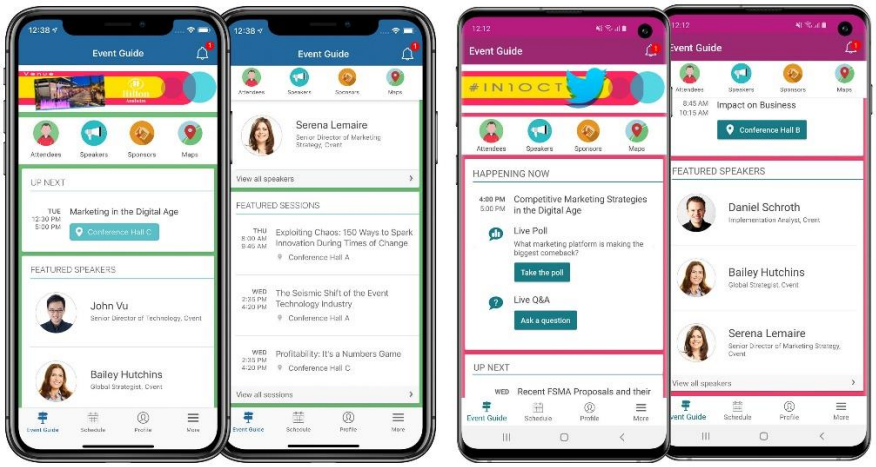
Na tržištu se posebice ističu višeplatformske aplikacije poput Asane i Monday.com-a. Navedene aplikacije nude rješenja za organizaciju unutrašnjeg rada tvrtke ili udruge te raspodjelu poslova zaposlenicima i članovima. Također ovakav tip aplikacija nudi funkcionalnosti poput jednostavne međusobne komunikacije zaposlenika i članova, dijeljenje sadržaja, notifikacije i brojne druge mogućnosti. Nažalost, velik broj mogućnosti često djeluje negativno na korisničko iskustvo jer u većini slučajeva sve funkcionalnosti ovakvih aplikacija nisu potrebne svakoj tvrtki ili udruzi. Često na web stranicama navedenih aplikacija možemo pronaći popis klijenata koji su nerijetko velike tvrtke s mnoštvom zaposlenika i naprednim potrebama. Takve napredne i mnogobrojne funkcionalnosti često nisu od koristi manjim udrugama i ponekad negativno utječu na organizaciju zbog dugotrajne i kompleksne krivulje učenja i razabiranja korisnih alata od svih onih koji su ponuđeni. Osim toga, spomenute aplikacije ili alati često su orijentirani isključivo prema zaposlenicima i članovima te ne nude mogućnost prikaza aktivnosti vanjskim korisnicima u svrhu marketinga ili upoznavanja s radom tvrtke ili udru. Na slici 2.1 na sljedećoj stranici prikazano je sučelje aplikacije Asana te je lako zaključiti kako ono, unatoč jednostavnosti i dobrom dizajnu, nije idealno rješenje za potrebe IEEE studentskog ogranka Osijek jer ne nudi mogućnost oglašavanja aktivnosti ili registracije u sustav IEEE-a.



Slika 2.1 Korisničko sučelje aplikacije Asana

Prednost ovakvih alata jest njihova neovisnost o platformi ili operacijskom sustavu. Najčešće se radi o web aplikacijama čiji je prikaz prilagođen i mobilnim uređajima zbog čega nije potrebno izrađivati posebne aplikacije za web, Android i IOS platforme.

Osim aplikacija koje služe za unutarnju organizaciju udruga, na tržištu se pojavljuje sve više rješenja sa svrhom prikaza događaja i aktivnosti na određenoj lokaciji. Tu se posebice ističu aplikacije poput CrowdCompassa i Whovea. Takve aplikacije nude jednostavno korisničko sučelje kao što prikazano na slici 2.2, brzu registraciju i lako objavljivanje događaja, dok posjetiteljima nude mogućnost prijave na događaj i informiranje o događaju. Također, navedene aplikacije prikazuju i lokaciju događaja na karti pomoću Google Maps API-a (*engl. API – application programming interface*) [8] integriranog u sučelje. Sudeći prema potrebama IEEE studentskog ogranka Osijek, aplikacije za organizaciju događaja djeluju kao izvrsno rješenje za automatizaciju poslova ogranka, no s obzirom da ne nude opciju registracije na vanjski servis poput IEEE-a [9] što je ujedno i najveći problem ogranka, takva rješenja nisu prikladna.



Slika 2.2 Sučelje aplikacije CrowdCompass

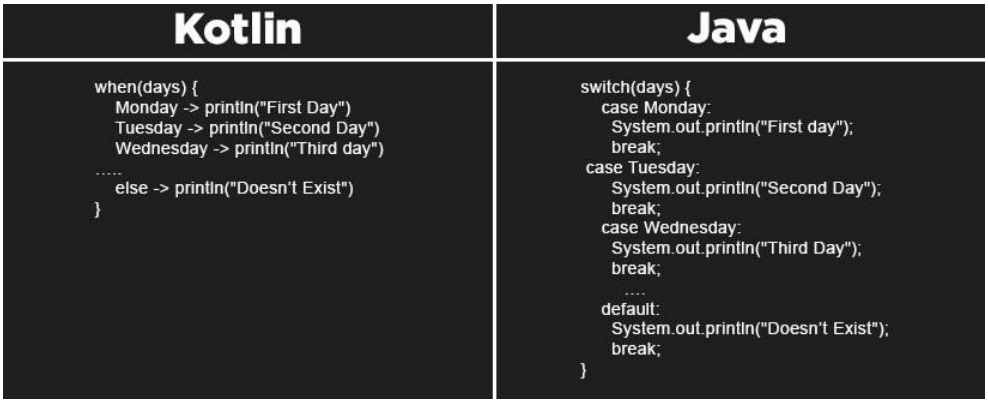
Analiza dostupnih alata na tržištu ponovno sugerira izradu vlastite aplikacije prilagođene potrebama IEEE studentskog ogranka Osijek, ali kao što je već spomenuto, analiza je neophodan korak koji može iznjedriti potpuno drugi zaključak ovisno o vrsti udruge.

2.3. Odabir programskih alata za razvoj aplikacije

Aplikacija namijenjena za organizaciju jedne udruge sadrži brojne funkcionalnosti poput ažuriranja podataka u stvarnom vremenu, geografske lokacije, autentifikacije korisnika i mnoge druge. To su iznimno kompleksni programski elementi koji predstavljaju rezultat istraživačkog i inženjerskog rada proteklog stoljeća zbog čega je nerealno očekivati razvoj svih pojedinačnih elemenata aplikacije ispočetka. Upravo iz tog razloga, Android platforma nudi mnoštvo već gotovih alata, aplikacijskih programskih sučelja i biblioteka koji čine izradu aplikacije puno jednostavnijom.

Prvenstveno je potrebno odabrati programski jezik za izradu aplikacije. Android platforma nudi dva dominantna jezika za razvoj aplikacija – Kotlin i Java [10]. Java je objektno-orijentirani programski jezik korišten kao primarni jezik na Android platformi od njenog nastanka. Razvojem Android platforme i rastom broja funkcionalnosti, zahtjevi razvojnih programera za jednostavnijim, bržim i optimiziranijim jezikom postajali su sve češći. Kao odgovor na zahtjeve programera, dugogodišnji Googleov partner JetBrains najavljuje razvoj programskog jezika Kotlin 2011. godine. Cilj razvoja Kotlina bio je stvoriti objektno orijentirani jezik koji će biti jednostavniji za pisanje, donijeti nove značajke i ujedno biti jednako brz prilikom prevođenja kao Java. Danas je Kotlin

primarni jezik za razvoj Android aplikacija i koristi ga preko 60% razvojnih programera na Android platformi. Najnovija stabilna inačica Kotlina (Kotlin 1.3) je višeplatformski, objektno orijentirani jezik koji je u potpunosti interoperabilan [11] s Javom. To znači da isti programski projekt može sadržavati oba programska jezika u željenoj mjeri. Ova značajka čini Kotlin idealnim za razvoj aplikacija na Androidu jer omogućuje postepeni prijelaz [12] razvojnih programera s Jave na Kotlin. Osim interoperabilnosti s Javom, Kotlin omogućuje automatsko prepoznavanje tipova podataka te jednostavno definiranje promjenjivih (*engl. mutable*) i nepromjenjivih (*engl. imutable*) vrijednosti pomoću ključnih riječi *var* za promjenjive i *val* za nepromjenjive. Kotlin također uvodi pojam „Nullable“ i „NonNull“ tipova podataka kako bi se smanjio broj iznimki u kodu, jednostavne lambda izraze koji smanjuju količinu koda i nova rješenja za pisanje asinkronog koda. Na slici 2.3 prikazano je koliko Kotlin uistinu smanjuje količinu koda i povećava čitljivost i brzinu pisanja.



Slika 2.3 Usporedba naredbi grananja u Kotlin i Java programskim jezicima

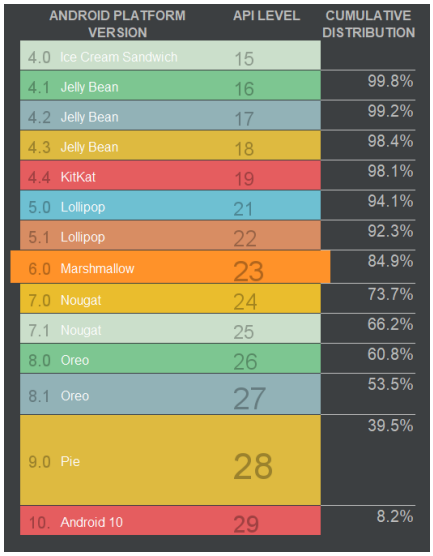
Osim programskog jezika bitno je odabrati i prikladno programsko radno okruženje. Za potrebe izrade ovog rada odabrana najnovija stabilna verzija Android Studio IDE-a 4.2.2. Android Studio službeni je alat za razvoj Android mobilnih aplikacija koji u potpunosti podržava Kotlin programski jezik. Ovo programsko okruženje nudi mogućnost dizajna i izrade korisničkog sučelja aplikacije pisanjem XML [13] datoteka ili koristeći grafičko korisničko sučelje te nudi mnoštvo prečaca i funkcionalnosti za brži i kvalitetniji razvoj aplikacija što uključuje i jednostavnu integraciju novih biblioteka i aplikacijskih programskih sučelja pomoću Android Studio SDK Managera [14]. Intuitivna struktura projekta unutar Android Studio IDE-a i mogućnost emulacije rada aplikacije na virtualnom uređaju i instaliranja aplikacije u razvoju na fizički uređaj samo su neke od mnogih prednosti ovog razvojnog sučelja.

Aplikacija za organizaciju IEEE studentskog ogranka Osijek zahtijeva i skladištenje podataka o događajima koji će se prikazivati korisnicima u mrežnoj bazi podataka. Za potrebe ove aplikacije korišten je Google Firebase servis koji predstavlja cjelokupno rješenje za skladištenje podataka na internetu bilo kojeg tipa te je integriran u Android Studio IDE.

3. IZRADA APLIKACIJE ZA ORGANIZACIJU IEEE STUDENTSKOG OGRANKA

3.1. Početno postavljanje razvojnog okruženja i pripadajućih komponenti

Android Studio IDE, kao što je navedeno u prethodnom poglavlju, bit će glavni alat za izradu aplikacije IEEE studentskog ogranka. Bitno je odabrati odgovarajuću inačicu ovog razvojnog okruženja. S obzirom na činjenicu da se ne preporučuje korištenje „beta“ inačica ovog razvojnog okruženja, koristit će se stabilna verzija 4.2.2. Zbog čestog ažuriranja ovog razvojnog sučelja i njegovih komponenti, mnoge značajke koje su prije bile dostupne i preporučene za korištenje, vremenom su zamijenjene novijim zbog čega se oprezno treba odabrati minimalna (najstarija) verzija Android operacijskog sustava koju će aplikacija podržavati. Za potrebe razvoja ove aplikacije odabrana je verzija Androida 6.0 Marshmallow, API razina 23 [15], koja podržava sve potrebne tehnologije koje će aplikacija koristiti poput geolokacije, RecyclerViewa [16] i ViewPager2 sučelja te rad s vanjskim aplikacijskim programskim sučeljima poput Glidea [17] i Firebase baza podataka. Problem brzog napretka tehnologije i zastarijevanja prethodnih inačica Android operacijskog sustava opće je prepoznat i dokumentiran. Google kao nositelj Android platforme upozorava na problem odabira ciljane inačice već prilikom izrade prvog projekta u Android Studio IDE-u i na zaslonu prikazuje statistiku korištenja određenih verzija Android operativnog sustava kao što je prikazano slikom 3.1 na sljedećoj stranici.



Slika 3.1 Statistika pokrivenosti tržišta određenom verzijom Android operacijskog sustava

Trenutno je najnovija verzije Android operativnog sustava Android 10 ili API razine 29. S obzirom na malu pokrivenost tržišta ovom verzijom, većina razvojnih programera odlučuje se razvijati programska rješenja koja su kompatibilna s prethodnim verzijama Androida. Iznimno je bitno proučiti dokumentaciju svih korištenih biblioteka i aplikacijskih programskih sučelja prilikom odabira minimalne API razine jer često dijelovi ili cjelokupno sučelje neće raditi na određenim nižim API razinama. Kako su svi elementi ovog programskog rješenja podržani API razinom 6.0 koja pokriva 84.9% mobilnih uređaja, nema potrebe za odabirom niže razine. Zbog Googleove inicijative da se većina razvojnih programera na Android platformi migrira na korištenje Kotlina zbog prednosti navedenih u prethodnom poglavlju, za potrebe ove aplikacije koristit će se programski jezik Kotlin. Rad aplikacije koja će biti razvijena za potrebe ogranka uvelike ovisi o korištenju baza podataka, kako Kotlin nudi jednostavno rukovanje asinkronim pozivima baze podataka, omogućuje jednostavnije dohvaćanje XML dokumenata i rukovanje njima unutar Kotlin klasa te je prilagođen za rad s novim inačicama prikaza (*engl. View*) izvrstan je izbor za potrebe razvoja moderne i intuitivne aplikacije.

Porast broja proizvođača mobilnih uređaja s Android operativnim sustavom koji ciljaju na niskobudžetni i srednjebudžetni spektar tržišta velik je problem Android platforme jer često takvi proizvođači prekidaju podršku uređaju nakon jedne do dvije godine i samim time onemogućuju ažuriranje operacijskog sustava na noviju inačicu. Takva politika proizvođača onemogućuje velikom broju korisnika prelazak na noviji sustav unatoč tomu što hardver uređaja podržava ažuriranje. Prilagodba korisničkog iskustva na nekoj inačici Androida iznimno je skup proces zbog kojeg se proizvođači mobilnih uređaja često opredjeljuju na korištenje softvera koji su izradili za prethodne generacije uređaja. Takav stav proizvođača koči razvoj aplikacija u najnovijim tehnologijama podržanim u višim API razinama, no s druge strane privlači sve više korisnika na Android platformu zbog niskih cijena uređaja.

3.2. Tehnička koncepcija aplikacije

Mobilna aplikacija izrađena za potrebe IEEE studentskog ogranka Osijek, naziva „.Act„ osmišljena je na način da svaki element aplikacije ima svoju jednoznačnu svrhu. Stoga se aplikacija može podijeliti u šest glavnih cjelina: autentifikacija, upoznavanje korisnika s radom aplikacije, prikaz svih aktivnosti ogranka, detaljni prikaz pojedinačne aktivnosti, prijenos novih događaja i registracija u sustav IEEE-a. Google i drugi razvijači nude mnoštvo alata koji čine izradu pojedinačnih elemenata

aplikacije jednostavnijom, od kojih će u nastavku biti objašnjeni Firebase baze podataka, ViewPager2, Glide API, Google Maps API i klase vezane uz animaciju pojedinih elemenata. Za izradu koncepta korisničkog sučelja i dizajna grafičkih elemenata odabran je Adobe Photoshop koji predstavlja industrijski standard za izradu grafičkih elemenata statičke veličine.

3.3. Firebase paket za izradu baza podataka

Firebase predstavlja paket različitih usluga koji razvojnim programerima olakšava izradu baza podataka, skladištenje velikih datoteka u oblaku (*engl. Cloud*), izradu servisa za komunikaciju velikog broja korisnika te, odnedavno, korištenje Googleovih servisa za strojno učenje. Google se opredijelio za korištenje NoSQL [18] tehnologije prilikom razvoja Firebasea jer olakšava snalaženje i rad s velikim bazama podataka u usporedbi s drugim rješenjima koristeći sučelje slično organizaciji datoteka na osobnom računalu s operativnim sustavom Linux. Za potrebe izrade ove aplikacije, bit će korištene Firebase Authentication, Firebase Firestore i Firebase Storage značajke navedene platforme.

Firebase Authentication jest usluga koja omogućuje jednostavnu registraciju i autentifikaciju korisnika mobilne ili web aplikacije. Za korištenje ove usluge, potrebno je u projekt dodati implementaciju Firebase ovisnosti (*engl. Dependency*) i posjedovati Google račun. Firebase Firestore predstavlja NoSQL bazu podataka za skladištenje manjih datoteka i njihov dohvat u realnom vremenu. Googleovo rješenje za skladištenje velikih datoteka je Firebase Storage koji svakoj uskladištenoj datoteci pridodaje jednoznačnu oznaku pomoću koje se pristupa datoteci unutar programskog koda. Količina prostora na Firebase platformi ovisi o količini prostora na Google računu te Google korisnicima besplatno osigurava 15 Gb prostora. Implementacije ovisnosti navedenih usluga u „Gradle“ datoteci prikazane su programskim kodom 3.1 gdje je moguće primijetiti sufiks „-ktx“ koji označava implementaciju specifične ovisnosti razvijene za paket proširenja standardne Kotlin biblioteke pod nazivom KotlinX. Osim implementacije ovisnosti potrebno je i osigurati da aplikacija ima pristup internetu te, u slučaju podizanja datoteka na bazu podataka iz aplikacije, pristup lokalnoj pohrani mobilnog uređaja. Spomenuta se dopuštenja mogu jednostavno osigurati umetanjem programskog koda u „Manifest.xml“ datoteku prikazanim programskim kodom 3.2.

```
implementation 'com.google.firebase:firebase-auth:21.0.1'
implementation 'com.google.firebase:firebase-firestore-ktx:23.0.1'
implementation 'com.google.firebase:firebase-storage-ktx:20.0.0'
```

Programski kod 3.1 Implementacija ovisnosti Firebase servisa

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

Programski kod 3.2 Implementacija dopuštenja potrebnih Firebaseu

Firebase bazi podataka pristupa se pomoću grafičkog korisničkog sučelja Firebase Console kojemu je moguće pristupiti nakon registracije aplikacije u Firebase sustav što je moguće učiniti iz Android Studio IDE-a. Pomoću Firebase Console sučelja moguće je iz uloge razvojnog programera brisati i dodavati podatke te upravljati kojim podacima korisnik aplikacije može pristupiti. Upravljanje pristupom podacima vrši se izmjenom pristupnih pravila pomoću Firebase opisnog jezika. Iz sigurnosnih razloga ograničavanje pristupa podacima neophodan je korak prilikom postavljanja Firebase servisa te će ova aplikacija omogućiti pristup podacima isključivo registriranim i prijavljenim korisnicima. Izgled sigurnosnih pravila „.Act“ aplikacije prikazan je slikom 3.2 na kojoj se može vidjeti dopuštenje čitanja i pisanja u bazu podataka samo u slučaju kada je korisnik prijavljen. Također je moguće i pristupiti prethodnim inačicama pravila u svrhu analize promjena istih.

Slika 3.2 Prikaz sigurnosnih pravila .Act aplikacije

Korištenje značajki Firebase usluga nakon osnovnog postavljanja iznimno je jednostavno jer Firebase pruža velik broj metoda za manipulaciju bazom podataka pomoću programskog jezika Kotlin. Programskim kodom 3.3 prikazana je metoda koja omogućuje podizanje jedne fotografije nasumično generiranog naziva na Firebase Storage bazu podataka. Fotografija se funkciji predaje pomoći Uri identifikatora te se pomoću reference na bazu podataka i metode putFile() podiže u određeni direktorij. Potrebno je obratiti pozornost na činjenicu da su pozivi prema bazi podataka asinkroni jer ovise o brzini internetske veze i ne mogu se izvršavati trenutno. Zbog toga je potrebno svakom pozivu metode za rad s bazom pridodati i „listener" metodu. U slučaju ove aplikacije koristit ćemo metode addOnCompleteListener() i addOnFailureListener(). Metoda addOnCompleteListener() definira logiku koja će se izvršiti nakon što je podizanje datoteke na bazu uspješno odrađeno, dok addOnFailureListener() definira što će se dogoditi u slučaju neuspješnog podizanja podataka na bazu.

```
private fun savePhoto(uri: Uri) {
    randomTitleImage=getRandomString(10)
    storageReference?.child("eventTitleImages/${randomTitleImage}.jpg")
        ?.putFile(uri)
        ?.addOnCompleteListener { task ->
            if (task.isComplete && task.isSuccessful) {
                Log.e("Upload successful:", "Upload successful")



            } else {
                Log.e("Upload unsuccessful:", "Upload unsuccessful")
            }
        }
        ?.addOnFailureListener {
            Log.e("Upload unsuccessful:", it.message.toString())
        }
}
```

Programski kod 3.3 Podizanje podataka na Firebase Storage bazu podataka

Na sličan način vrši se i spuštanje datoteka s baze podataka te autentifikacija korisnika zbog asinkrone naravi svih navedenih procesa. Autentifikacija korisnika i podizanje podataka na druge Firebase servise poput Firestorea također je iznimno jednostavna. Programskim kodom 3.4 prikazan je kod za registraciju i autentifikaciju korisnika u aplikaciji. Zbog asinkrone naravi rada s bazom podataka bitno je voditi računa o činjenici da glavna nit (*engl. thread*) programa čeka odaziv baze i kratko zaustavlja ostatak programa. Kotlin nudi jednostavno rješenje za ovaj problem te je pomoću Kotlin Coroutines značajke moguće lako delegirati posao komunikacije s bazom podataka na IO nit i time osloboditi glavnu nit. Nakon toga je vrlo jednostavno registrirati novog korisnika koristeći metodu createUserWithEmailAndPassword() koja kao argumente prima niz znakova koji

predstavljaju adresu elektroničke pošte i lozinku korisnika. S obzirom na činjenicu da korisnik sam upisuje navedene podatke, postoji velika mogućnost unosa nepravilno oblikovanih podataka zbog čega će Firebase Authentication sustav ponekad izbaciti iznimku s podacima o vrsti greške te je potrebno proces autentifikacije zatvoriti u try-catch [19] programski blok.

```
private fun registerUser(){
    val email=etEmail.text.toString()
    val password=etPassword.text.toString()
    if(email.isNotEmpty()&&password.isNotEmpty()){
        CoroutineScope(Dispatchers.IO).launch {
            try {
                auth.createUserWithEmailAndPassword(email, password).await()
                withContext(Dispatchers.Main){
                    checkLoggedInState()
                }
            }catch (e: Exception){
                withContext(Dispatchers.Main){
                    Toast.makeText(this@RegistrationActivity, e.message,
                      Toast.LENGTH_LONG).show()
                }
            }
        }
    }
}
```

Programski kod 3.4 Funkcija za registraciju korisnika pomoću Firebase Authentication servisa

Rad s Firebase Firestore uslugom također je vrlo jednostavan i omogućuje brz dohvat podataka s baze podataka u aplikaciju te podizanje istih. Programskim kodom 3.5 predstavljeno je podizanje jednog objekta tipa „Event“ na Firestore bazu podataka. Objekt se na bazu podataka podiže pomoću metode add() i reference na bazu „db“. Također je potrebno istaknuti u koji direktorij baze podataka treba spremiti podatke. Ovo je učinjeno predavanjem argumenta „events“ metodi collection() zbog čega će se u konačnici predani objekt spremiti u direktorij naziva „events“

```
db.collection("events")
        .add(event)
        .addOnSuccessListener(OnSuccessListener<DocumentReference> { documentReference ->
Log.d("Event upload", "ID: " + documentReference.id) })
        .addOnFailureListener(OnFailureListener {
                    e -> Log.w("Event upload", "Error adding event", e) })
```

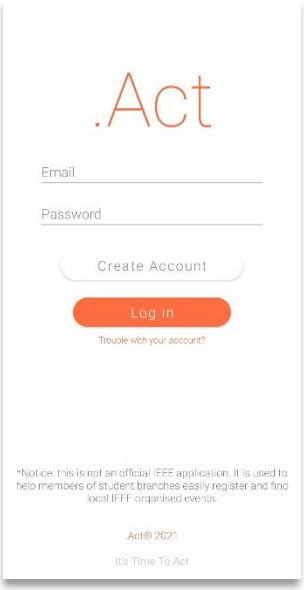
Programski kod 3.5 Podizanje objekta na Firestore bazu podataka

3.4. Korisničko sučelje aplikacije

Kao što je prethodno navedeno, aplikacija za organizaciju IEEE studentskog ogranka sastoji se od šest osnovnih cjelina: registracija i autentifikacija, upoznavanje korisnika s radom aplikacije,

prikaz svih aktivnosti ogranka, detaljni prikaz pojedinačne aktivnosti, prijenos novih događaja i registracija korisnika u sustav IEEE-a

Nakon prikaza uvodne animacije (*engl. splash-screen*) korisnik je usmjeren na početni zaslon aplikacije na kojemu je objedinjena registracija i prijava u aplikaciju upisom adrese elektroničke pošte i lozinke. Korisnik nakon upisa podataka za autentifikaciju može izabrati opciju izrade novog računa ili prijave kao što je prikazano slikom 3.3.



Slika 3.3 Početni zaslon aplikacije

Dodirom na tipke „Create Account“ i „Log In“ pozivaju se odgovarajuće Firebase Authentication metode za autentifikaciju objašnjene u prethodnom poglavlju. Također je moguće primjetiti poluproziran tekst „It's Time To Act“ u podnožju zaslona na slici 3.3. Ovaj tekst je u stvarnosti animiran na način da se uzastopno pojavljuje i nestaje koristeći klasu AnimationUtils. Pomoću navedene klase učitana je XML datoteka animacije nakon čega se pomoću metode startAnimation() animira TextView [20] „tvWelcomeAnimated“. Prikaz koda za animaciju jednog tekstualnog objekta dan je programskim kodom 3.6.

```
val animation = AnimationUtils.loadAnimation(this, R.anim.fade_in)
 tvWelcomeAnimated.startAnimation(animation)
```

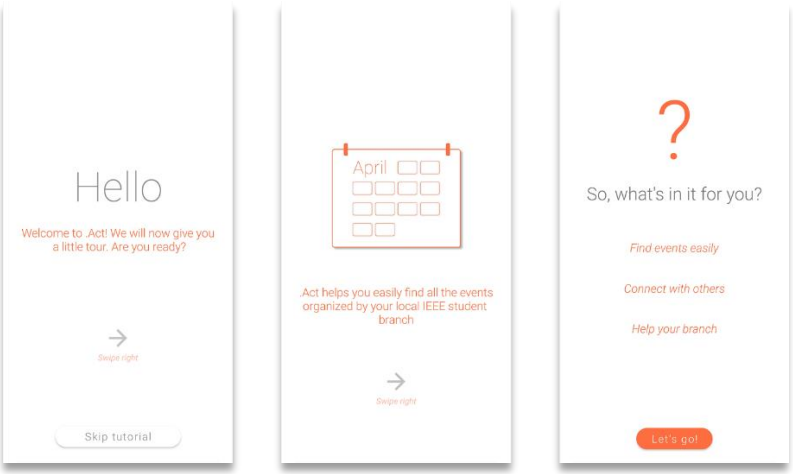
Programski kod 3.6 Programski kod za animaciju objekata

XML datoteka animacije dana je programskim kodom 3.7 u kojemu je moguće vidjeti kako je efekt animacije postignut promjenom postotka prozirnosti objekta svake sekunde. Animacija se kontinuirano pokreće na način da se ista animacija svaki drugi put izvršava unazad čime je postignut efekt pulsiranja.

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
        <alpha
            android:duration="1000"
            android:fromAlpha="0.1"
            android:toAlpha="1.0"
            android:repeatCount="infinite"
            android:repeatMode="reverse"
            />
</set>
```

Programski kod 3.7 XML datoteka pulsirajuće animacije

Nakon uspješne autentifikacije ili registracije, aplikacija korisnika preusmjerava na zaslon za upoznavanje sa svrhom i radom aplikacije. Navedeni zaslon prikazuje korisniku značajke aplikacije i pozitivne strane učlanjenja u IEEE organizaciju. Prelaskom prsta preko zaslona mobilnog uređaja, korisnik prelazi na sljedeću stranicu s novim informacijama poput onih prikazanih slikom 3.3. Na svakoj stranici nalaze se animirani objekti čija je animacija također postignuta koristeći klasu AnimationUtils.



Slika 3.3 Uvodni zasloni aplikacije

Ovakav je prikaz omogućen korištenjem ViewPager2 biblioteke, najnovijom inačicom hvaljene ViewPager biblioteke. Navedena biblioteka je proširenje osnovne RecyclerView biblioteke te horizontalno prikazuje listu različitih objekata. U slučaju .Act aplikacije radi se o listi fragmenata [21] kako bi se osiguralo samostalno funkcioniranje svakog pojedinačnog zaslona. S obzirom na činjenicu da će ovu aplikaciju u budućnosti razvijati i drugi razvojni programeri, odabir liste fragmenata je dobar izbor zbog proširivosti funkcionalnosti. Kako bi ViewPager2 ispravno funkcionirao, potrebno je definirati adapter klasu koja će voditi računa o pravilnom prikazu svakog fragmenta iz predane liste fragmenata s obzirom na stanje njegovog životnog ciklusa (*engl. lifecycle*) [22] te prepisati određene metode nužne za rad ove biblioteke. Programski kod 3.8 prikazuje ViewPager2 adapter klasu prilagođenu za rad s fragmentima unutar .Act aplikacije.

```
class WelcomeSliderViewPagerAdapter (
    list: ArrayList<Fragment>,
    fm: FragmentManager,
    lifecycle: Lifecycle
    ) : FragmentStateAdapter(fm, lifecycle) {

    private val fragmentList: ArrayList<Fragment> =list


    override fun getItemCount(): Int {
        return fragmentList.size
    }

    override fun createFragment(position: Int): Fragment {
        return fragmentList[position]
    }
}
```

Programski kod 3.8 Adapter klasa za prikaz i rad ViewPager2 biblioteke

Nakon završetka upoznavanja s radom aplikacije, korisnika se usmjerava na glavni zaslon. Na glavnom zaslonu kartično su prikazani svi trenutno aktivni događaji unutar nekog ogranka. Kartice se prikazuju na način da se najnoviji događaji nalaze na vrhu liste kartica, dok se stariji događaji, analogno, prikazuju na dnu zaslona. Kartični prikaz omogućen je korištenjem CardView i RecyclerView biblioteka, gdje su svi detalji pojedinačnog događaja pohranjeni unutar CardView kontejnera, a svaki događaj predstavlja element RecyclerViewa. Korištenjem RecyclerView biblioteke u ovu svrhu postignuta je veća brzina učitavanja kartica jer RecyclerView ograničava broj trenutno prikazanih kartica na zaslonu, što je iznimno korisno u slučaju velikog broja aktivnih događaja unutar ogranka. Programskim kodom 3.9 prikazan je dio klase EventRecyclerViewAdapter

unutar kojeg se iz baze podataka dohvaćaju podaci o događaju i prikazuju unutar CardView kontejnera.

```
override fun onBindViewHolder(holder: EventViewHolder, position: Int) {

    val currItem = events[position]
    var storageReference=FirebaseStorage.getInstance().reference


    holder.title.text=currItem.title
    holder.briefDescription.text = currItem.briefDescription
    holder.date.text = currItem.date
    GlideApp.with(holder.itemView).load(storageReference?.child("eventTitleImages/${currI-
tem.titleImage}.jpg"))
        .placeholder(R.mipmap.no_title_image).override(holder.eventPicture.ivCardEventPic-
ture.width).into(holder.eventPicture.ivCardEventPicture
```

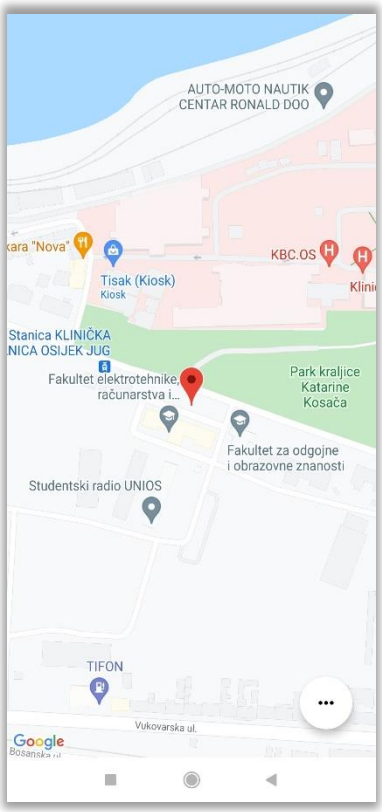
Programski kod 3.9 Prikaz umetanja podataka s baze podataka u CardView kontejner

Osim kartičnog prikaza događaja, na glavnom su zaslonu prikazane i ikone na čiji se dodir korisnik može odjaviti iz aplikacije ili osvježiti prikaz događaja. Osvježeni prikaz događaja postignut je ponovnim pokretanjem glavnog zaslona što rezultira novim pozivom funkcija za dohvaćanje podataka s baze podataka. Na slici 3.4 prikazan je izgled korisničkog sučelja glavnog zaslona. Na slici je moguće uočiti i plutajući gumb (*eng. Floating action button*) koji na dodir otkriva četiri gumba za navigaciju ostalim značajkama aplikacije.

Slika 3.4 Prikaz glavnog zaslona s potpuno otvorenim izbornikom za navigaciju

Dodirom na pojedinačnu karticu otvara se poseban fragment s prikazom detalja o događaju. Na ovom zaslonu korisnik može pročitati detalje o događaju i prijaviti se na događaj pomoću obrasca za online prijavu. Obrasci za prijavu na događaj definiraju se pomoću linka za online obrazac koji se postavlja pri izradi događaja. Navedeno rješenje odabrano je zbog fleksibilnosti u slučaju da određeni studentski ogranak koristi prilagođeni način prijave za događaj te jednostavnosti izvoženja podataka iz online organizacijskih alata. Također, korisnik pritiskom na tipku „Show on map“, može pogledati lokaciju događaja na karti. Izgled fragmenta za prikaz detalja događaja prikazan je na slici 3.5, dok je izgled fragmenta za prikaz lokacije događaja predstavljen slikom 3.6.

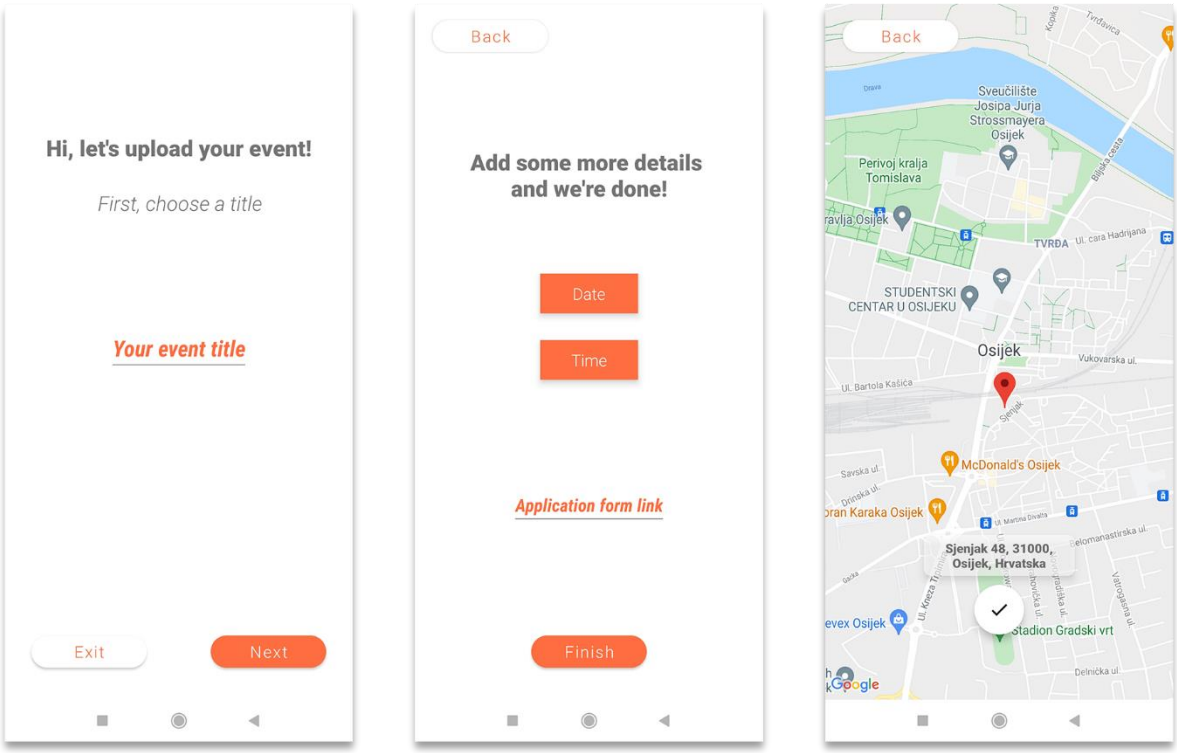
Slika 3.5 Prikaz fragmenta s detaljima o pojedinačnog događaja



Slika 3.6 Prikaz lokacije događaja na karti

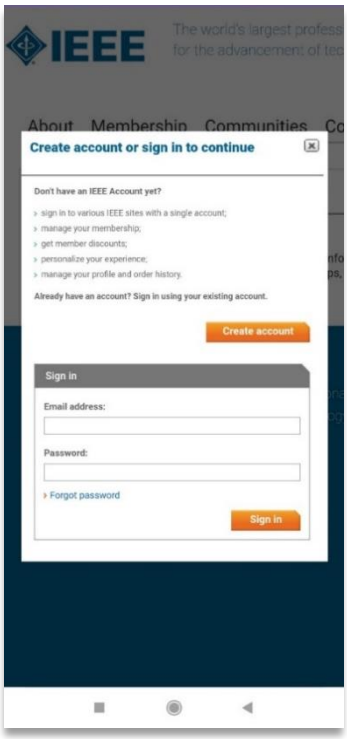
Dodirom na prvi plutajući gumb prikazan na slici 3.4 , gumb za odjavu korisnika, korisnika se odjavljuje i vraća na zaslon za prijavu. Dodirom na drugi plutajući gumb, točnije gumb za prijenos vlastitog događaja, otvara se nova aktivnost koja je nositelj ViewPager2 objekta s listom fragmenata koji vode korisnika kroz proces prijenosa događaja. Kako bi se događaj uspješno prenio, korisnik mora unijeti sve potrebne podatke o događaju uključujući naslovnu sliku i geografsku lokaciju

događaja. Navedeni se podaci zatim prenose na bazu podataka iz koje će glavna aktivnost popunjavati kartični prikaz događaja. Unos svakog podatka potrebnog za izradu događaja smješten je u pojedinačni fragment. Na ovaj način postignuto je intuitivno korisničko iskustvo i jednostavno snalaženje u aplikaciji prilikom unošenja podataka o događaju. Prikaz nekolicine fragmenata za unos podataka o događaju predstavljen je slikom 3.7.



Slika 3.7 Prikaz fragmenata za unos podataka o događaju

Osim prijenosa događaja moguća je i registracija u sustav IEEE-a pritiskom na treći plutajući gumb. Nakon dodira gumba otvara se zaslon na kojemu je prikazana web aplikacija za prijavu na IEEE servis. Takav prikaz postignut je korištenjem WebView kontejnera u kojemu se otvara IEEE web stranica s navedenom web aplikacijom. Ovakvo je rješenje odabrano zbog mogućnosti promjene načina prijave u sustav u budućnosti i manjka službenog aplikacijskog programskog sučelja za prijavu. Važno je istaknuti da se tokom izrade ove aplikacije način za prijavu na IEEE servis promijenio, no zbog korištenja WebView kontejnera nije bilo potrebno raditi izmjene u aplikaciji što dokazuje agilnost i prilagodljivost ovakvog rješenja. Na ovaj se način korisnicima omogućuje brz pristup IEEE servisu i ubrzan proces prijave. Zaslon za prijavu na servis prikazan je slikom 3.8 na kojoj je moguće vidjeti obrazac za prijavu.



Slika 3.8 Prikaz obrasca za prijavu i registraciju u sustav IEEE-a

3.5. Google Maps programsko aplikacijsko sučelje

Kao što je prikazano u prethodnom poglavlju, „.Act“ aplikacija omogućuje prikaz i postavljanje lokacije pojedinog događaja na karti. Za postizanje ovih funkcija korišten je Google Maps API, uvjerljivo najraširenij i najodržavaniji servis za geografsko pozicioniranje na tržištu. Korištenje ovog servisa je besplatno i, s obzirom da Google održava Android platformu, nudi brojne funkcije specifične za prikaz lokacije i karti na pametnim telefonima što uvelike olakšava razvoj aplikacija koje koriste takve funkcionalnosti. U „.Act“ aplikaciji korištene su funkcije za prikaz lokacije pomoću geografske dužine i širine te funkcije za prikaz adrese i drugih detalja o odabranoj lokaciji. Osim navedenih funkcija, Google Maps API nudi i funkcije za animaciju položaja kamere u odnosu na kartu te promjenu razine uvećanja karte ovisno o namjeni.

Za postavljanje Google Maps aplikacijskog programskog sučelja potrebno je generirati jedinstveni API ključ i postaviti ga u manifest datoteku nakon čega je moguće koristiti navedene

metode. Prikaz spajanja na aplikacijsko programsko sučelje pomoću jedinstvenog ključa predstavljen je programskim kodom 3.10.

```
<resources>
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AI-
zaSyCyd9iZaiVmP-hhsoezY_iNQlAZGQAHF4w</string>
</resources>
```

Programski kod 3.10 Prikaz spajanja na Google Maps API pomoću jedinstvenog „translatable“ ključa

Nakon spajanja na Google Maps API izrađen je Google Maps Fragment koji nudi osnovnu programsku strukturu za rukovanje kartama. Prilikom izrade događaja korisnik odabire lokaciju događaja povlačenjem oznake za lokaciju na željeno mjesto na karti nakon čega se geografska širina i dužina odabrane lokacije podižu na bazu podataka. Nakon izrade događaja, iz baze podataka se dohvaćaju informacije o geografskoj širini i dužini te se lokacija prikazuje u posebnoj aktivnosti. Kod za prikazivanje lokacije na karti predstavljen je programskim kodom 3.11 u kojemu je moguće vidjeti callback metodu koja nakon učitavanja karte postavlja oznaku na koordinate željene lokacije. Callback metoda je vrsta metode koja se predaje kao argument drugoj metodi i poziva se kasnije. U ovom slučaju callback metoda poziva se nakon što je karta učitana zbog vremena potrebnog za učitavanje karte. U metodi je definiran i položaj kamere u odnosu na kartu korištenjem metode moveCamera() te je uvećanje postavljeno na razinu grada. Dodana je i oznaka „Event location“ i animacija uvećanja kamere.

```
private val callback = OnMapReadyCallback { googleMap ->
    val bundle = this.arguments
    val latitude=bundle!!.get("latitude").toString().toDouble()
    val longitude=bundle!!.get("longitude").toString().toDouble()
    val location = LatLng(latitude, longitude)
    googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(location, 10f))

    googleMap.addMarker(MarkerOptions().position(location).title("Event location"))

    googleMap.animateCamera(CameraUpdateFactory.zoomTo(17f), 1000, null)
}
```

Programski kod 3.11 Isječak programskog koda za prikaz lokacije na karti

Na sličan način omogućen je i odabir lokacije prilikom izrade događaja, no dodan je atribut draggable koji omogućuje pomicanje oznake lokacije. Nakon toga prepisana je metoda onMarkerDragEnd() koja

definira što će se dogoditi prilikom ispuštanja oznake na željenu lokaciju. Prilikom ispuštanja oznake lokacije, geografska dužina i širina podižu se na bazu podataka za daljnje korištenje.

3.6. Glide programsko aplikacijsko sučelje

Kako bi se događaji cjelovito prikazivali korisniku potrebno je omogućiti prijenos naslovne slike događaja u svrhu vizualne orijentacije korisnika u listi događaja. Prijenos i prikaz slika složeni su procesi koji se sastoje od kompresije, obrezivanja i obrade fotografija zbog čega je izrada vlastitog prilagođenog programskog rješenja iznimno složen i vremenski zahtjevan pothvat. Upravo zbog toga razvojem Android platforme pojavljuju se brojne biblioteke koje služe za jednostavan i intuitivan rad sa slikama unutar Androida.

Najpopularnija biblioteka u vrijeme pisanja ovog rada jest Glide. Glide je biblioteka otvorenog koda koja pruža rješenje za brzo dohvaćanje i dekodiranje slika te njihovu obradu. Jedan od primarnih ciljeva ove biblioteke također je brzo i učinkovito učitavanje slika unutar neke liste, što ga čini pogodnim za korištenjem s bibliotekama baziranim na RecyclerView biblioteci. Zbog ove značajke Glide je izvrstan odabir za razvoj „.Act“ aplikacije s obzirom na velik broj ViewPager2 i RecyclerView objekata.

Glide biblioteka je iznimno jednostavna za korištenje. Nakon što se u Gradle dadoteku postavi implementacija biblioteke, razvojnom se programeru nude brojne funkcije za jednostavan rad s fotografijama. Programskim kodom 3.12 prikazan je isječak koda za učitavanje fotografije s baze podataka u ImageView kontejner, postavljanje fotografije na točno određenu rezoluciju te prikazivanje zamjenske fotografije prije učitavanja željene fotografije (*eng. placeholder image*).

```
GlideApp.with(holder.itemView).load(storageReference?.child("eventTitleImages/${currItem.tit-
leImage}.jpg"))
    .placeholder(R.mipmap.no_title_image).override(holder.eventPicture.ivCardEventPic-
ture.width).into(holder.eventPicture.ivCardEventPicture)
```

Programski kod 3.12 Isječak programskog koda za učitavanje i obradu slike unutar jednog elementa RecyclerViewa

Korištenje Glide biblioteke iznimno ubrzava tok razvoja aplikacije i čini korištenje kompleksnijih metoda za obradu i rad s fotografijama, koje su ugrađene u Android platformu, gotovo nepotrebnim osim u slučajevima kada je potrebno preciznije manipulirati prijenosom i obradom fotografije.

4. ZAKLJUČAK

Ovim završnim radom opisan je složen proces prepoznavanja specifičnih potreba određene udruge te koraci odabira poslova udruge koje je potrebno automatizirati. Objašnjene su i osnovne značajke korisničkog sučelja aplikacije koje su potrebne kako bi se održala motiviranost korisnika za korištenje iste. Predstavljena su i postojeća rješenja za organizaciju i automatizaciju rada udruge te njihove prednosti i mane. Na primjeru izrade aplikacije u svrhu prikaza i prijenosa organiziranih aktivnosti IEEE studentskog ogranka Osijek, „.Act“, predstavljeno je programsko rješenje u obliku aplikacije bazirane na Android mobilnoj platformi i programskom jeziku Kotlin. Detaljno su objašnjene i najbitnije biblioteke za obradu i rad s fotografijama, manipulaciju bazama podataka i autentifikaciju korisnika te rad s kartama i geografskim pozicioniranjem. Osim navedenih biblioteka prikazan je i objašnjen rad biblioteka za prikaz i rad korisničkog sučelja, intuitivan prikaz grafičkih elemenata i animaciju istih. Također je objašnjen i argumentiran odabir navedenih biblioteka te rukovanje asinkronim aspektima rada aplikacije. U radu se, također, ulazi u problematiku odabira prikladne API razine Android operacijskog sustava te statistiku zasićenosti tržišta određenom verzijom Androida. Prikazan je i tok korištenja aplikacije iz perspektive korisnika te sve značajke aplikacije dostupne korisniku.

Aplikacija, čiji je razvoj objašnjen i prikazan ovim radom, namijenjena je kao projekt osječkog studentskog ogranka IEEE-a čiji će članovi u budućnosti raditi na razvoju i održavanju ove aplikacije u praktične i edukativne svrhe. U budućnosti je cilj povećati razinu sigurnosti aplikacije te ponuditi više opcija prijave, poboljšati korisničko iskustvo i brzinu učitavanja pojedinih elemenata aplikacije. Osim toga cilj je i dodati značajke koje će udovoljiti budućim potrebama ogranka nastalim zbog samog širenja i razvoja istog.

LITERATURA

[1] S. Schoger, Adam Wathan, »Starting from scratch,« u *Refactoring UI*, 2018., pp. 7-28.

[2] M. Luetić, "Cross-Platform vs. Native Mobile Development" [Online]. Dostupno na web lokaciji: https://decode.agency/native-vs-cross-platform-mobile-apps/, 15. ožujak 2021.

[3] Google Inc., "About Flutter" [Online]. Dostupno na web lokaciji: https://github.com/flutter/flutter, 2021.

[4] Javapoint.com, "What is Xamarin" [Online]. Dostupno na web lokaciji: https://www.javatpoint.com/what-is-xamarin, 2021.

[5] R. Budiu, "Mobile: Native Apps, Web Apps, and Hybrid Apps" [Online].Dostupno na web lokaciji: https://www.nngroup.com/articles/mobile-native-apps/#:~:text=Native%20apps%20are%20installed%20through,of%20contacts%2C%20and%20so%20on., 2013.

[6] Google Inc., "Get Started With Google Firebase" [Online]. Dostupno na web lokaciji: https://firebase.google.com/docs, 2021.

[7] JetBrains, "What is Kotlin" [Online]. Dostupno na web lokaciji: https://kotlinlang.org/docs/faq.html, 28.5.2021.

[8] Google Inc., "Maps SDK for Android overview".Dostupno na web lokaciji: https://developers.google.com/maps/documentation/android-sdk/overview, 2021.

[9] IEEE, "Learn about IEEE" [Online]. Dostupno na web lokaciji: https://www.ieee.org/about/index.html, 2021.

[10] Oracle, "Oracle Announces Java 16" [Online]. Dostupno na web lokaciji: https://www.oracle.com/news/announcement/oracle-announces-java-16-031621.html, Austin-Texas, 2021.

[11] E. Shapiro, »Kotlin/Java Interoperability,« u *Kotlin Apprentice 3rd Edition - Beginning Programing with Kotlin*, Razeware LLC., 2021., pp. 329-350.

[12] Nat Pryce, Duncan McGregor, »Refactoring to Kotlin,« u *Java to Kotlin - A Refactoring Guidebook*, 202.1, pp. 11-14.

[13] Google inc., "Layouts" [Online]. Dostupno na web lokaciji: https://developer.android.com/guide/topics/ui/declaring-layout, 2021.

[14] Google Inc., "Installing android sdk tools" [Online]. Dostupno na web lokaciji: https://guides.codepath.com/android/installing-android-sdk-tools, 30.12.2020.

[15] AppBrain, "Top Android OS versions" [Online]. Dostupno na lokaciji: https://www.appbrain.com/stats/top-android-sdk-versions, lipanj 2021.

[16] Google Inc., "Create dynamic lists with RecyclerView" [Online].Dostupno na web lokaciji: https://developer.android.com/guide/topics/ui/layout/recyclerview, 2021.

[17] Bumptech, "An image loading and caching library for Android focused on smooth scrolling" [Online]. Dostupno na web lokaciji: https://github.com/bumptech/glide, 30. siječanj 2021.

[18] IBM, "NoSQL Databases" [Online]. Dostupno na web lokaciji: https://www.ibm.com/cloud/learn/nosql-databases, 6. kolovoz 2019.

[19] JetBrains, "Exceptions" Online]. Dostupno na web lokaciji: https://kotlinlang.org/docs/exceptions.html, 28.5.2021.

[20] Google Inc., "TextView" [Online].Dostupno na web lokaciji: https://developer.android.com/reference/android/widget/TextView, 2021.

[21] Google Inc., "Fragments" [Online]. Dostupno na web lokaciji: developer.android.com/guide/fragments, lipanj 2021.

[22] Google Inc., "Handling Lifecycles with Lifecycle-Aware Components" [Online]. Dostupno na web lokaciji: https://developer.android.com/topic/libraries/architecture/lifecycle, lipanj 2021.

SAŽETAK

Završni rad prikazuje proces prepoznavanja poslova udruge koje bi trebalo automatizirati te na primjeru izrade aplikacije za prikaz aktivnih događanja unutar studentskog ogranka IEEE-a, objašnjava korake izrade programskog rješenja na Android platformi. Obrađuje se postavljanje razvojnog okruženja, odabir programskog jezika i usporedba popularnih jezika za razvoj na Android platformi. Također se pažnja posvećuje objašnjenju značajki ugodnog i intuitivnog korisničkog iskustva te korisnih biblioteka za razvoj aplikacija ovog tipa poput RecyclerView, CardView i ViewPager2 biblioteka. Detaljno su predstavljeni i način prikazivanja i obrade fotografija pomoću Glide biblioteke, korištenje Google Maps aplikacijskog programskog sučelja i rukovanje asinkronim pozivima metoda Firebase baza podataka. Navedena su i uspoređena već postojeća slična tehnička rješenja i argumentiran je odabir izrade personalizirane aplikacije za organizaciju ogranka. U završnom su radu navedeni i koraci budućeg razvoja aplikacije i poboljšanja njenih značajki.

Ključne riječi: Android, Firebase, organizacija udruge, korisničko iskustvo

MOBILE APPLICATION FOR STUDENT BRANCH MANGEMENT ABSTRACT

The thesis explains the process of recognizing automatable activities inside an organization and exemplifies this process with a step-by-step description of an event management application development for the IEEE student branch Osijek. The thesis also shows the necessary steps to develop an application on the Android platform. It also contains an explanation of common design characteristics shared by applications with a pleasant and intuitive user experience and frequently used libraries such as RecyclerView, CardView and ViewPager2. It also expounds image displaying and processing using the Glide library, usage of Google Maps API and handling of asynchronous method calls to the Firebase database suite. Other existing technical solutions are also specified and mutually compared alongside giving a line of reasoning for developing a personalized application for the student branch. The thesis also lists the goals of future development of the application and the steps needed to expand the functionality of the application.

Keywords: Android, Firebase, association organization, user experience

ŽIVOTOPIS

Nikola Pavković rođen je 28. studenog 1998. u Osijeku. Upisuje Prirodoslovno-matematičku gimnaziju u Osijeku 2013. godine. 2017. upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija na Sveučilištu Josipa Juraja Strossmayera u Osijeku, smjer računarstvo. Uz izvršavanje studentskih obaveza, postaje predsjednik IEEE studentskog ogranka Osijek u sklopu kojega izvršava administrativne i organizacijske poslove. Tijekom studiranja obavlja posao predavača brojnih predmeta iz područja računarstva u privatnoj tvrtki.

Nikola Pavković