Aplikacija za rezervaciju termina u studentskim referadama

Matak, Ivan

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

IVAN MATAK

APLIKACIJA ZA REZERVACIJU TERMINA U STUDENTSKIM REFERADAMA

Diplomski rad

Pula, lipanj 2020.

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

IVAN MATAK

APLIKACIJA ZA REZERVACIJU TERMINA U STUDENTSKIM REFERADAMA

Diplomski rad

JMBAG: 0303061164, redoviti student

Studijski smjer: Diplomski sveučilišni studij Informatika

Kolegij: Mobilne aplikacije

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijsko-komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Siniša Sovilj

Komentor: dr. sc. Nikola Tanković

Pula, lipanj 2020.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Ivan Matak, kandidat za magistra informatike ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima, te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

Ivan Matak

U Puli, lipanj, 2020. godine

IZJAVA

o korištenju autorskog djela

Ja, Ivan Matak dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom „Aplikacija za rezervaciju termina u studentskim referadama“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli,_____(datum)

Potpis

Ivan Matak

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Pula, 11. ožujka 2020.

DIPLOMSKI ZADATAK

Pristupnik:	**Matak Ivan (0303061164)**
Studij:	Sveučilišni diplomski studij informatike
Naslov (hrv.):	**Aplikacija za rezervaciju termina u studentskim referadama**
Naslov (eng.):	Application for appointment reservation at student offices
Opis zadatka:	Zadatak je izraditi web i mobilnu aplikaciju za rezervaciju termina u studentskim referadama koji olakšava i automatizira proces odlaska u studentsku referadu na jednostavan i moderan način. Mobilna aplikacija namijenjena studentima se temelji na Java programskom jeziku te Android operacijskom sustavu. Web aplikacija namijenjena osoblju referade se temelji na JavaScript programskom jeziku te Vue.js okruženju. Za pohranu i upravljanje podacima koristi se baza podataka Google Firebase. Nakon registracije i prijave, studenti preko mobilne aplikacije pregledavaju slobodne termine te rezerviraju termine prema datumima, razlogu i fakultetu. Studenti mogu svoje rezervacije uređivati, ažurirati i izbrisati. Moguće je i vidjeti status rezervacija te pregledati prošle i nadolazeće rezervacije. Osoblje referade pregledava rezervacije te ih ovisno o dostupnosti odobravaju ili otkazuju. Rezervacije se mogu sortirati prema datumu, razlogu i ostalim parametrima. Istražiti i provesti usporedbu Vue.js-a s ostalim okruženjima za izradu web aplikacija.

Zadatak uručen pristupniku: 11. ožujka 2020.
Rok za predaju rada: 11. veljače 2021.

Mentor:	Komentor:
Siniša Sovilj	
_____	_____
doc.dr.sc. Siniša Sovilj	dr.sc. Nikola Tanković

Sadržaj

1. Uvod 1
2. Usporedba okruženja za izradu web aplikacija 3
2.1. *Frontend* okruženja za izradu web aplikacija 3
2.1.1. React 3
2.1.2. Vue.js 4
2.1.3. Angular 5
2.2. *Backend* okruženja za izradu web aplikacija 5
2.2.1. Node.js 5
2.2.2. Django 6
2.2.3. Laravel 7
2.3. Zaključak usporedbi i odabir okruženja 7
3. Postojeće stanje, motivacija i mogućnosti aplikacije 9
3.1. Postojeće stanje i motivacija 9
3.2. Logika i mogućnosti aplikacije 12
4. Korisnički scenariji i prototip sučelja 16
4.1. Korisnički scenariji 16
4.2. Prototip sučelja 20
5. Implementacija web aplikacije 25
5.1. Osnovni oblik aplikacije i usmjeravanje na stranice 25
5.2. *Vuex state management* i baza podataka Firebase 26
5.3. Registracija i prijava 28
5.4. Nadolazeće i prošle rezervacije 29
5.5. Promjena podataka za prijavu 30
6. Implementacija mobilne aplikacije 32
6.1. Registracija i prijava 32
6.2. Moje rezervacije, prošle rezervacije i glavni izbornik 34
6.3. Izrada nove rezervacije 36
6.4. Uređivanje rezervacije i opcije 37
7. Korisničke upute 40
7.1. Korisničke upute web aplikacije 40
7.1.1. Registracija i prijava 41
7.1.2 Nadolazeće rezervacije i prošle rezervacije 42
7.1.3. Promjena podataka za prijavu i odjava 43
7.2. Korisničke upute mobilne aplikacije 45
7.2.1. Registracija, prijava, izbornik 45
7.2.2. Glavni izbornik, opcije i moje rezervacije 46
7.2.3. Rezervacija i uređivanje rezervacije 47
8. Zaključak 49

9. Literatura.....50
10. Popis slika.....51
11. Sažetak.....52
12. Abstract.....52

1. Uvod

Globalizacija je trend koji se ubrzao pojavom Interneta. Komunikacija i usluge koje su nekad zahtijevale mnogo vremena i fizičkih materijala su stvar prošlosti. E-mail poruke dolaze u bilo koji kraj svijeta u sekundi na jednostavan način. Trenutni je trend automatizacija i digitalizacija raznih procesa koji su se obavljali fizički. Internet u svojim začecima nije bio dinamički dizajniran već su informacije bile statičke. Stranice su se sastojale od hard kodiranog hiperteksta što znači da su se sve modifikacije na stranicama trebale odvijati od strane autora. Zatim se 1993. godine pojavio *Common Gateway Interface* (CGI) standard koji je omogućio prikazivanje eksternih aplikacija preko web poslužitelja što je omogućavalo dinamičku interakciju između korisnika i aplikacije (Common Gateway Inteface, 1997). Kasnije su se razvila takozvana „full stack“ okruženja za razvoj web aplikacija. Ta okruženja pokušavaju imati što više mogućnosti za korisnika kao što su upravljanje bazom podataka, usmjeravanje stranica, komponente za izradu stranica, registracija i prijava korisnika na poslužitelja.

Bitno je napomenuti kako su *frontend* i *backend* danas bazična struktura većine web i mobilnih aplikacija. *Frontend* se sastoji od korisničkog sučelja, a *backend* se sastoji od upravljanja komunikacijom sa poslužiteljem i bazom podataka. U zadnjih 10 godina uz web aplikacije pojavljuju se i mobilne aplikacije. Pametni telefoni su proširili tržište aplikacija te su otvorili pregršt mogućnosti za razvoj preko raznih ugrađenih senzora unutar pametnih telefona. Aplikacije su danas toliko proširene da njihov broj s vremenom eksponencijalno raste. Naravno, postoje razne kategorije aplikacija sa različitim funkcionalnostima. Trendovi se mijenjaju, no mogućnosti pametnih telefona se povećavaju konstantno što dovodi i do inovacija kod razvoja aplikacija.

Web okruženja za razvoj danas ima mnogo, svaki razvojni programer može birati okruženje prema programskim jezicima koje koriste ili prema nekim drugim preferencijama kao što su dostupni paketi i proširenja za razvoj web aplikacija. Moderne, ali i jednostavne baze podataka isto tako danas dostupnije su no ikad. Dvije najveće mobilne platforme za razvoj aplikacija danas su iOS i Android. Android aplikacije se razvijaju u službenom programu za razvoj „Android Studio“, a koristi Java i Kotlin programske jezike prilikom razvoja. S druge strane, iOS koristi službeni program „Xcode“ sa programskim jezikom Swift. Te dvije platforme danas zauzimaju ogromni dio tržišta mobilnih aplikacija te su vrlo raširene.

Rad se sastoji od osam glavnih poglavlja. Uvodno poglavlje prikazuje kratki povijesni razvoj i utjecaj web i mobilnih aplikacija na društvo. Drugo poglavlje obuhvaća detaljnu usporedbu mogućnosti, prednosti i mana okruženja za izradu web aplikacija. Provodi se analiza i objašnjenje čimbenika koji dolaze u obzir prilikom odabira okruženja za razvoj web aplikacija ovisno o situaciji i potrebama razvojnog programera. Treće poglavlje prikazuje trenutno stanje studentskih referada te SWOT analizu trenutnog stanja iz koje proizlazi i motivacija za izradu web aplikacije i mobilne aplikacije kako bi se ubrzao, modernizirao i automatizirao proces odlaska u studentsku referadu. Isto tako, u trećem poglavlju se razrađuju logika i funkcionalnosti aplikacije pomoću use case dijagrama. Četvrto poglavlje obuhvaća korisničke scenarije koji su prikazani sekvencijskim dijagramima kako bi se što preciznije prikazao proces odvijanja glavnih mogućnosti za web aplikaciju i mobilnu aplikaciju. U četvrtom poglavlju se isto tako prikazuje prototip korisničkog sučelja koji služi kao skica za izradu aplikacija. Peto i šesto poglavlje se fokusiraju na implementaciju funkcionalnosti i izradu web aplikacije i mobilne aplikacije. Sedmo poglavlje sadrži detaljne korisničke upute za korištenje aplikacija te zaključno poglavlje objašnjava kako aplikacije mijenjaju i pojednostavljuju način života.

2. Usporedba okruženja za izradu web aplikacija

Odabir okruženja za izradu web aplikacija nije lak izbor. Radi njihove mnogobrojnosti, evaluacija prednosti i mana pojedinog okruženja može biti težak zadatak. Odabirom pravilnog okruženja može se znatno ubrzati proces izrade kvalitetne aplikacije. Preporuka je u obzir uzeti okruženje koje koristi jezik s kojim je razvojni programer upoznat. U tom slučaju proces učenja je mnogo brži, no i same funkcionalnosti imaju veću ulogu pošto utječu na kvalitetu aplikacije. Popularnost okruženja je isto faktor koji utječe na ažuriranja samog okruženja kao i na dostupnost rješenja problema na raznim forumima i ostalim izvorima.

2.1. *Frontend* okruženja za izradu web aplikacija

2.1.1. React

React po čistoj definiciji nije okruženje, već JavaScript biblioteka koja je razvijena od strane multinacionalne kompanije Facebook. React je nastao 2013. godine te je postao popularan u 2014. godini. Trenutno je najpopularnija biblioteka koja koristi JavaScript programski jezik. Vrlo je koristan kod velikih projekta koje se baziraju na razmjeni podataka, iako nema baš sve mogućnosti koje bi trebale koristiti razvojnim programerima. Isto tako, *ECMAScript* standard od 2015. godine donosi nove mogućnosti JavaScript jeziku na godišnjoj bazi što može biti malo previše za nove korisnike zbog novih sintaksi koje se konstantno mijenjaju. Također, biblioteka se konstantno ažurira te se dodaju novi i unaprjeđuju postojeći alati. Funkcionira na način da se postavlja kod koji izgleda kao HTML unutar JavaScript datoteka (Banks i Porcello, 2017). Naglasak se postavlja na funkcionalno programiranje, a ne na objektno orijentirano programiranje kao kod nekih ostalih jezika. Arhitektura biblioteke se bazira na komponentama, a osnove se lako nauče. Najpoznatije aplikacije koje koriste React su Facebook i Instagram.

Prednosti React biblioteke uključuju laku čitljivost koda, vrlo laka SEO interakcija, ponovna iskoristivost komponenti, pregršt dodatnih biblioteka i paketa te VDOM koja predstavlja koncept programiranja koji omogućuje deklarativnu API React biblioteku za prilagodbu korisničkog sučelja. Glavna mana je učenje funkcionalnosti

biblioteke, pogotovo jer se razne funkcionalnosti često ažuriraju, no osnove nisu toliko zahtjevne. Dokumentacija oko kompleksnijih radnji nije toliko razumljiva.

2.1.2. Vue.js

Vue.js je jedan od novijih okruženja koji raste po popularnosti zbog svoje jednostavne strukture i lakoće korištenja. To je vrlo progresivno okruženje što znači da, ukoliko već postoji određeni projekt, Vue.js se može primijeniti na samo jedan dio projekta bez problema sa kompatibilnosti. Iako je to vrlo fleksibilno i jednostavno okruženje, problem je što Vue.js nije podržan od stana velikih multinacionalnih kompanija kao što su Google i Facebook pa nije toliko poznat i korišten kao React i Angular. Ne zahtijeva dodatne biblioteke te je jednostavan za naučiti i koristiti. Za razliku od svoje konkurencije, Vue.js je isto tako zauzima manje prostora te je poznat po svojoj detaljnoj dokumentaciji za korištenje.

Način na koji se Vue.js koristi i integrira predstavlja kombinaciju HTML i JavaScript jezika. Pri tome HTML i Vue.js komponente služe za izradu korisničkog sučelja, dok JavaScript datoteke obično sadrže definicije funkcija koje Vue.js datoteke pozivaju i koriste. Vue.js koristi i komponente koje predstavljaju dijelove web aplikacije koje sadrže vlastite podatke te se mogu ponovno koristiti u drugim dijelovima aplikacije (Filipova, 2016).

Prednosti Vue.js okruženja su jednostavnost korištenja, detaljna dokumentacija, te fleksibilnost. Mane su što mu komponente nisu dovoljno stabilne i razrađene pa korisnici obično ovise o dodatnim bibliotekama kako bi imali sve dostupne komponente za rad. Iako se fleksibilnost uvijek spominje kao vrlina, ista može biti i mana u kontekstu kompleksnosti i neovisnosti okruženja za razvoj. Također zbog toga što nije među najpopularnijim okruženjima, manji je broj razvojnih programera koji koriste to okruženje što automatski označava da će se teže rješavati problemi preko foruma i sličnih izvora.

2.1.3. Angular

Angular je nastao kako bi se pojednostavio razvoj web i mobilnih aplikacija, a razvila ga je multinacionalna kompanija Google 2010. godine. Koristi se za *frontend* razvijanje aplikacija te se bazira na TypeScript programskom jeziku koji je nadskup JavaScripta. Jedan je od starijih okruženja u usporedbi s konkurencijom, no velika ažuriranja u zadnjih 8 godina mu pomažu držati korak s vremenom. Iako je u početku bilo problema sa kompatibilnosti sa starijim verzijama, Google je pravovremenim ažuriranjima ispravio greške. To je specijalizirano okruženje za *single-page* aplikacije, odnosno aplikacije na kojima se podaci dinamički izmjenjuju na jednoj bazičnoj stranici. Ne zahtijeva dodatne biblioteke za razvoj te sadrži razne funkcionalnosti kao što su povezivanje podataka i usmjeravanje na stranice preko komponenti. Zbog svoje kompleksnosti, okruženje je namijenjeno i prilagođeno za veće projekte tako da svaki zaposlenik može raditi na svom dijelu koda bez utjecaja na tuđi kod.

Prednosti Angular okruženja su duga podrška od strane kompanije Google te pregršt komponenti za korištenje uz objektno orijentirano programiranje. Uz te prednosti, TypeScript je jezik koji je vrlo koristan nakon što se savlada te zahtijeva manje kodiranja uz dobre performanse. Mana Angular okruženja je zapravo njegova veličina u usporedbi s konkurencijom te kompleksnost učenja za rad u tom okruženju. Također, zbog čestih ažuriranja koja su često pozitivna, ista mogu zapravo problematična radi kompatibilnosti sa starijim verzijama što forsira konstantno ažuriranje koda (Freeman, 2018).

2.2. *Backend* okruženja za izradu web aplikacija

2.2.1. Node.js

Node.js je okruženje koje pomaže pri izvršavanju JavaScript programskog koda na poslužiteljskoj strani. To je okruženje otvorenog koda koje je vrlo popularno u svojoj kategoriji. Podržava preko milijun paketa otvorenog koda koji se besplatno mogu koristiti. Prije pojave Node.js okruženja, uporaba JavaScript jezika na poslužiteljskoj strani je bila rijetkost. Inputi i outputi koji rade bez blokada su prije zahtijevali posebne biblioteke, no Node.js ih podržava što u kombinaciji sa JavaScript jezikom čini moćnu kombinaciju. Omogućuje da se čitanje i pisanje datoteka odvija asinkrono u istom

procesu pošto u klasičnom pristupu sa blokadama treba čekati da jedan proces završi kako bi drugi mogao započeti (Young i Harter, 2017).

Prednost Node.js sučelja je uporaba jezika koji se može koristiti i za *frontend* i za *backend* svrhe te je vrlo lagan za naučiti i savladati. Nudi vrlo dobre performanse te zbog svoje popularnosti ima mnogo dostupnih i besplatnih materijala te aktivnu skupinu razvojnih programera. Zbog svoje fleksibilnosti i slobode pri razvijanju aplikacija, nudi inovativno rješenje za *backend* razvoj sa performansama. Iako ima mnogo prednosti, Node.js ima i svoje nedostatke. API tog okruženja nije stabilan te ponekad zna patiti od problema vezanih za kompatibilnost sa starijim verzijama okruženja. Isto tako, okruženje je ovisno o paketima kako bi se izvele neke osnovne i često korištene funkcionalnosti. Iako nudi dobre performanse radi mogućnosti inputa i outputa bez blokade, to isto tako znači da se programiranje treba provoditi prema asinkronom modelu kojeg neki programeri smatraju teškim za naučiti.

2.2.2. Django

Django je *backend* okruženje bazirano na Python programskom jeziku. Vrlo je popularan te ga koriste poznate kompanije kao što su Google, YouTube, Mozilla, National Geographic te Pinterest. Jedan je od starijih okruženja te je nastao 2003. godine. Sadrži pregršt dokumentacije koja je organizirana prema referencama, primjerima i temama (Django documentation, 2019). Zbog svoje skalabilnosti i prilagodljivosti, predstavlja jedan od najboljih izbora među okruženjima. Skalabilnost omogućuje prilagođavanje mogućnosti okruženja prema potrebama programera. Okruženje je građeno za veće i kompleksnije aplikacije te nije idealan za manje projekte koji mogu rezultirati aplikacijom koja sadrži mnogo nepotrebnih ili suvišnih mogućnosti.

Uz sve prednosti bitno je napomenuti kako je to okruženje otvorenog koda koje je vrlo prilagodljivo za SEO radi pronalaska aplikacije u tražilici te sadrži i sigurnosne mogućnosti. Django je vrlo kompatibilan sa aplikacijama koje sa baziraju na pregledu i interakciju sa sadržajem kao što su vijesti, video zapisi, društvene mreže i slično. U kombinaciji sa moćnim Python jezikom, Django je jedan od najkorištenijih i okruženja sa jednostavnom sintaksom. Mana okruženja je u nekim slučajevima brzina te

određivanje stanica za usmjeravanje zna biti problematično za početnike. Isto tako, pomalo monotoni pristup pri razvoju ne dozvoljava mnogo različitih načina razvoja.

2.2.3. Laravel

Laravel je okruženje bazirano na PHP programskom jeziku koji je jedan od najpopularnijih jezika za razvoj web stranica. Uz jednostavnu sintaksu, implementacija pojedinih mogućnosti kao što su autorizacija i autentikacija je jednostavna. Omogućuje brzu i jednostavnu autentikaciju više korisnika istovremeno uz pomoć tokena. Fokus se postavlja na pojednostavljenje procesa razvoja za razvojne programere početnike, ali i za iskusnije korisnike (Laravel installation, 2020). Okruženje je idealno prilikom izrade e-mail servisa pošto sadrži drivere kao što su SMTP i Mailgun. Brzina i *cache* menadžment je bitan radi ukupnih performansi aplikacija, a Laravel podržava napredno upravljanje *cache* memorijom. Upravljanje greškama i sigurnost su isto značajke koje su podržane na napredan i efektivan način.

Uglavnom, uz ugrađenu API potporu, Laravel sadrži već ugrađene pakete koji predstavljaju njegovu najveću prednost te je dobar za razvojne programere početnike. Mane Laravel okruženja leže u tome što nije preporučljiv za veće projekte pošto nema performanse u usporedbi sa konkurencijom kao što je Django. Isto tako, Laravel zna biti problematično okruženje ukoliko razvojni programer planira napraviti migraciju postojeće aplikacije na isti. Jednostavnost je dobra, no za tu jednostavnost se moraju žrtvovati neke funkcionalnosti koje nadodaju na kompleksnosti.

2.3. Zaključak usporedbi i odabir okruženja

Čim se nudi izbor između različitih okruženja, većina razvojnih programera profesionalno donosi oduku na temelju mnogo čimbenika. Izbor kao takav se može bazirati na čistoj statistici i brojkama, no i ne mora, ovisno o situaciji. Ukoliko je izbor profesionalne prirode, brojke imaju veću ulogu pri odabiru. Često se u obzir uzimaju performanse i mogućnosti kao glavni faktori. Za profesionalne potrebe je isto tako bitno uzeti u obzir potrebe kompanije te postojeća okruženja s kojih se potencijalno može migrirati. Naravno, proces migracije mora biti što direktniji i bezbolniji za kompaniju

kako se ne bi stvorili gubitci ili slučajna šteta. Sigurnost je isto tako jako bitni čimbenik, pogotovo ako se aplikacija koristi povjerljivim podacima. Programski jezik je isto čimbenik koji može biti presudan. Ukoliko su ostale aplikacije rađene u specifičnom programskom jeziku, faktor odabira može biti sučelje koje se koristi istim programskim jezikom radi lakše interakcije i kompatibilnosti između aplikacija.

Sa osobne razine ili usavršavanja, odabir pravog okruženja može biti razlog za učenje. Popularnost je u tom slučaju isto jako dobar čimbenik zbog količine dostupnih materijala i dokumentacije za učenje. Za razvojne programere početnike jednostavnost je jedan od najbitnijih čimbenika pošto tek ulaze u svijet razvoja aplikacija, dok za one koji samo žele proširiti svoje horizonte i imati kompetitivne vještine na današnjem tržištu rada, odabir popularnog sučelja je najbolji izbor. Ponuda se rapidno proširuje i trendovi se mijenjaju konstantno, u zadnje vrijeme nastaje i sve veći broj sučelja za razvoj mobilnih aplikacija, od kojih je primjer Flutter multinacionalne kompanije Google.

Dakle, odgovor na koje je sučelje najbolje zapravo ne postoji. Statistikom i čistim brojkama se može doći do zaključka koje je sučelje najbolje na temelju svih ranije navedenih čimbenika. No, bitna je stavka kako se sve svodi na trenutnu situaciju i osobne preferencije što znači da brojevi i statistika mogu samo poslužiti kao vodilja za odabir, no nisu krucijalni dio izbora najboljeg sučelja. Za pravu kombinaciju *frontend* i *backend* okruženja potrebno je balansirati statistiku i preferencije razvojnog programera.

Na temelju svoje jednostavnosti, lakog učenja i fleksibilnosti, odabir sučelja za izradu web aplikacije u ovom slučaju je Vue.js. Segment koji je presudio u odabiru jest detaljna dokumentacija, fleksibilnost ali i osobna preferencija JavaScript programskog jezika koji je vrlo praktičan kod izrade web aplikacija.

3. Postojeće stanje, motivacija i mogućnosti aplikacije

3.1. Postojeće stanje i motivacija

Postojeće stanje u studentskim referadama se bazira na čekanju u redu. To se čekanje dijeli na dolazak i klasično čekanje u redu ili na uzimanje papirića sa brojevima za dolazak na red. Taj proces ponekad zna oduzeti puno vremena studentima koji u nekim situacijama mogu čekati i više od jednog sata. Prilikom većih događaja kao što su upisi ili kada dođe do neželjenih događaja poput pada sustava, čekanje u redu zna biti i veće od 2 sata. Često takvi slučajevi izazivaju nelagodu i frustraciju među studentima, ali i zaposlenicima studentskih referada te se javlja potreba za prilagodbom, automatizacijom i modernizacijom tog procesa.

Slika 1. Red ispred referade

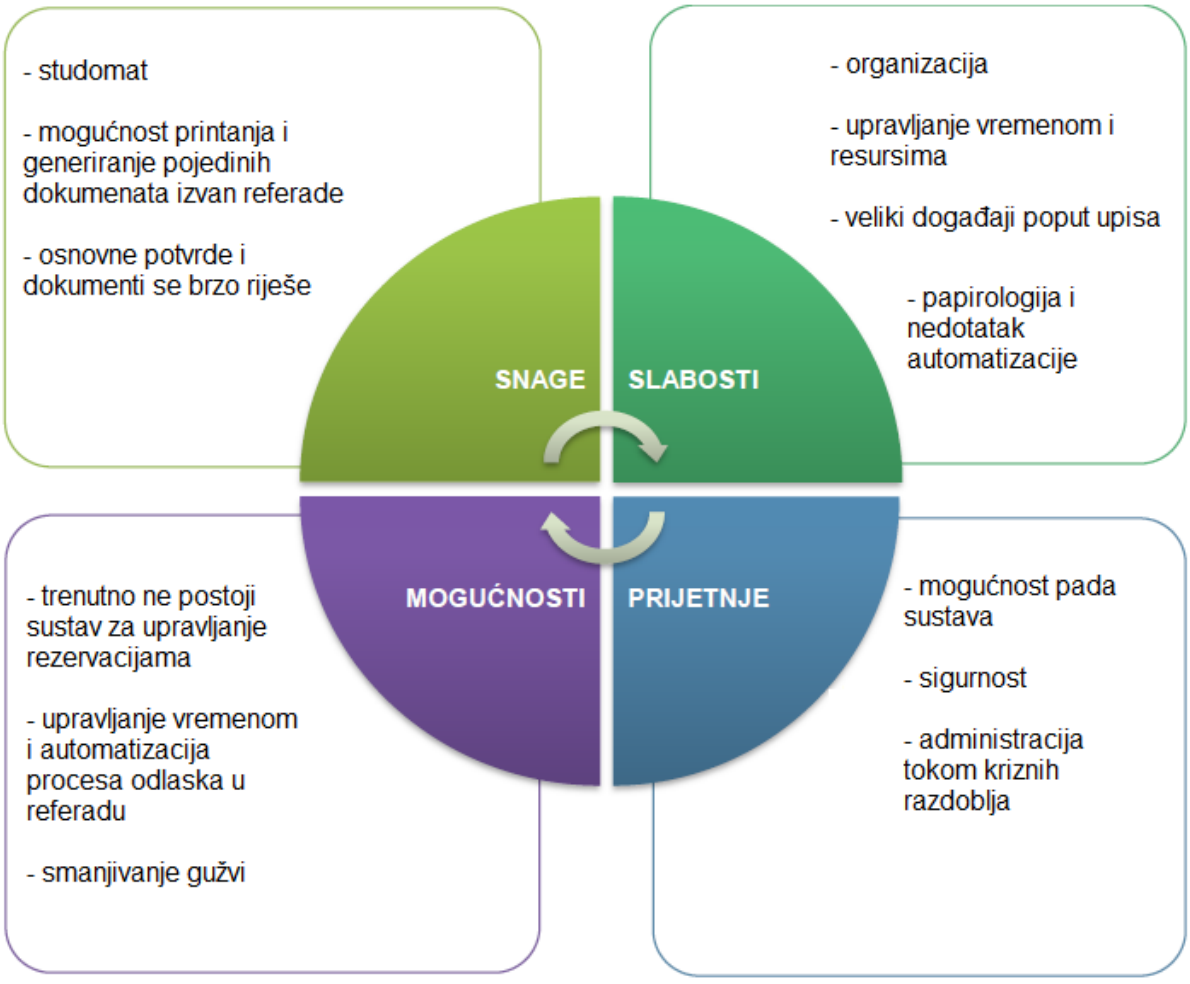
Izvor: http://mojfaks.com/akademska-cetvrt/propustaju-predavanja-zbog-reda-zadarski-studenti-frustrirani-dugim-cekanjima-pred-vratima-referade

Upravo proces odlaska u referadu studenti doživljavaju kao jedan od gorih iskustava tokom studiranja (Šoda, 2019). Nerijetko studenti propuštaju predavanja radi

čekanja u redu. Dodatni problemi se stvaraju kada se zaposlenici susretnu sa neočekivanim situacijama koje produžuju proces čekanja.

Na temelju trenutnog stanja u studentskim referadama nastaje motivacija kako bi se taj proces prije svega automatizirao i ubrzao. Problem čekanja često izaziva negodovanje i frustraciju kod studenata. Aplikacije su općenito u današnje vrijeme vrlo raširene i količinski njihov broj iz dana u dan raste. Ideja je koristiti aplikacije kao alate koji će poboljšati odlazak u referadu na Sveučilištu Jurja Dobrile u Puli.

Slika 2. SWOT analiza trenutnog stanja



Izvor: autor

Na temelju SWOT analize moguće je prikazati trenutno stanje studentske referade na temelju kategorija kao što su snage, slabosti, mogućnosti i prijetnje. Snaga

svih studentskih referada je Studomat sustav koji olakšava procese kao što su prijava i odjava ispita, pregled ocjena, generiranje digitalnog dokumenta o statusu studenta i slično. Specifično vezano za referadu u Puli, snaga je isto tako što se takve potvrde mogu isprintati pomoću specijaliziranih uređaja izvan prostorija referade. No, isti ti dokumenti zahtijevaju potpis zaposlenika referade što postavlja generirane dokumente sa sustava Studomat kao bolju opciju. Snaga je isto tako što se neke osnovne potrebe studenata mogu riješiti u vrlo kratkom vremenom roku. Slabosti referade su veliki i neočekivani događaji poput pada Studomat sustava ili upisa na studij. Često kompliciraniji zahtjevi studenata trebaju više vremena za rješavanje. Posljedično tome produžuje se red čekanja kao negativni aspekt referade. Iz tih slabosti proizlaze mogućnosti koje mogu poboljšati trenutno stanje. Trenutno ne postoji sustav na tržištu koji bi trebao automatizirati i modernizirati proces odlaska u referadu. Glavna mogućnost i poboljšanje je ušteda i bolje upravljanje vremenom kako bi se vrijeme čekanja smanjilo za studente i za zaposlenike. Za takav sustav postoje i prijetnje koje se mogu poistovjetiti sa većinom ostalih sustava, a to su sigurnost, mogućnost pada sustava te način administracije u kriznim razdobljima.

Osnovna ideja je izraditi web aplikaciju i mobilnu aplikaciju. Mobilna aplikacija je namijenjena studentima, pošto su pametni telefoni danas najbrži način pristupa informacijama, aplikacijama i ostalim uslugama. Web aplikacija je namijenjena zaposlenicima referade pošto njihov posao uključuje rad na računalima te je to najpraktičniji pristup za obje strane. Cijeli koncept se bazira na tome da studenti rezerviraju termin, a zaposlenici odobre ili otkažu tu rezervaciju. Na taj način se izbjegavaju nepotrebne gužve ispred studentske referade jer studenti sa svojim rezerviranim terminom znaju točno kada trebaju doći. S druge strane, prednost imaju i zaposlenici pošto mogu rasporediti svoje radno vrijeme efektivnije i unaprijed znaju razlog posjeta.

Mobilna aplikacija se izrađuje u programu Android Studio u programskom jeziku Java. Web aplikacija se izrađuje u Vue.js okruženju se paketom Vuetify. Kao baza podataka koristi se Google Firebase. Takva vrsta baze podataka predstavlja jednostavno i praktično rješenje za spremanje podataka besplatno. Web aplikacija i mobilna aplikacija nemaju direktnu međusobnu interakciju, već obje aplikacije imaju zajedničku interakciju sa bazom podataka na način da dohvaćaju, spremaju, ažuriraju i brišu podatke.

3.2. Logika i mogućnosti aplikacije

Prije razrade funkcionalnosti aplikacija, potrebno je razraditi logiku iste. Dostupni termini za izradu rezervacija su prilagođeni trenutnom radnom vremenu referade, isto tako logika uzima u obzir da postoji jedan zaposlenik za svaku sastavnicu fakulteta pošto se raspored zaposlenika prema sastavnicama konstantno mijenja.

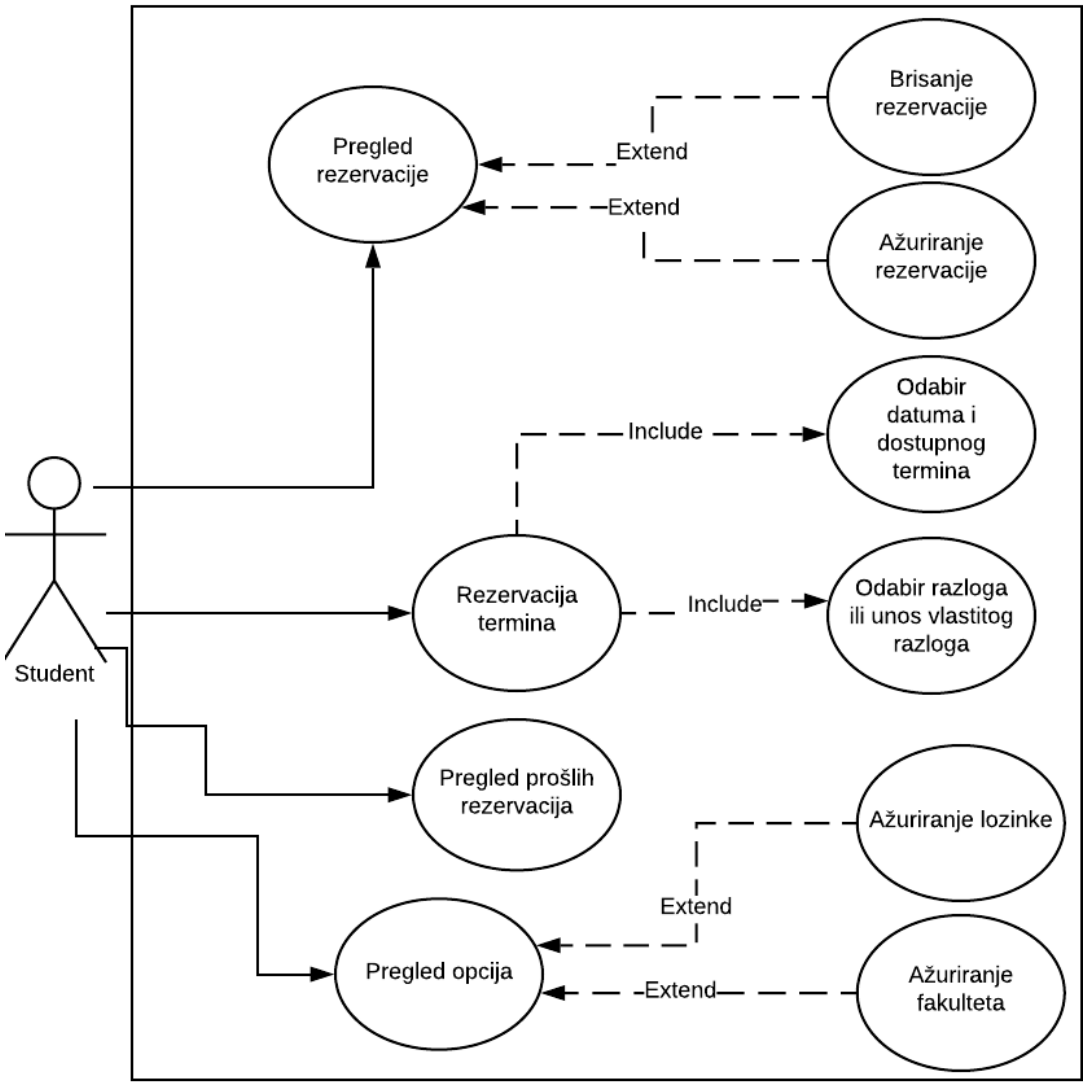
Aplikacija za studente omogućuje studentima registraciju i prijavu pomoću e-mail adrese. Prilikom rezervacije, student odabire dostupni termin. Vikendom nije moguće odabrati termin pošto je referada zatvorena. Logika isto tako funkcionira na način da je moguće rezervirati termin na isti datum u isto vrijeme uz uvjet da studenti nisu sa iste sastavnice fakulteta. Primjerice, rezervacija za 4.10.2020. u 13:00 sati je rezervirana od strane studenta A koji je student FIPU sastavnice. Student B koji je isto tako dio FIPU sastavnice može napraviti rezervaciju na isti datum, ali ne na isti termin pošto je taj termin rezerviran od strane studenta A. S druge strane, student C koji je dio FET sastavnice može rezervirati isti datum i termin kao student A pošto su njihove sastavnice različite. Tom logikom se postiže sustav da postoji jedan zaposlenik za svaku sastavnicu fakulteta. Isto tako, postoje restrikcije kod rezervacija. Dozvoljena je samo jedna rezervacija dnevno kako bi se spriječila zlouporaba sustava.

Nakon odabira željenog datuma, studentu se nude dostupni termini. Zauzeti termini se ne povlače iz baze podataka te se prikazuju samo dostupni termini na temelju odabranog datuma. Ta činjenica se veže na ranije objašnjenu logiku sastavnica, studenti će vidjeti samo dostupne i zauzete termine svoje sastavnice. Nakon odabira termina student odabire razlog posjeta referadi uz mogućnost dodavanja vlastitog razloga ukoliko se razlog ne nalazi u jednom od ponuđenih. Student zatim sprema rezervaciju na bazu podataka klikom na gumb.

Nakon spremljene rezervacije, student svoje vlastite rezervacije može pregledati. Rezervacije se sortiraju prema datumu na način da se rezervacija koja je najbliža realizaciji nalazi na početku. Odabirom rezervacije student istu može urediti ukoliko se predomislio oko nekih detalja kao što su datum, termin i razlog. Ukoliko student želi otkazati rezervaciju, može to učiniti odabirom gumba za brisanje te se rezervacija briše sa baze podataka. Isto tako, student može pregledati svoje prošle rezervacije sortirano koje služe kao arhiva što znači da se ne mogu uređivati ili brisati. Student isto tako ima dostupne opcije pomoću kojih može ažurirati svoju lozinku te

promijeniti svoju sastavnicu fakulteta ukoliko se prebace na neki drugi smjer studija. Aplikacija funkcionira na načina da sačuva trenutna sesija korisnika što znači da, ukoliko se aplikacija zatvori i ugasi, ponovnim ulaskom u aplikaciju sesija se samo nastavlja te se student automatski prijavi u aplikaciju.

Slika 3. Use case dijagram - student



Izvor: autor

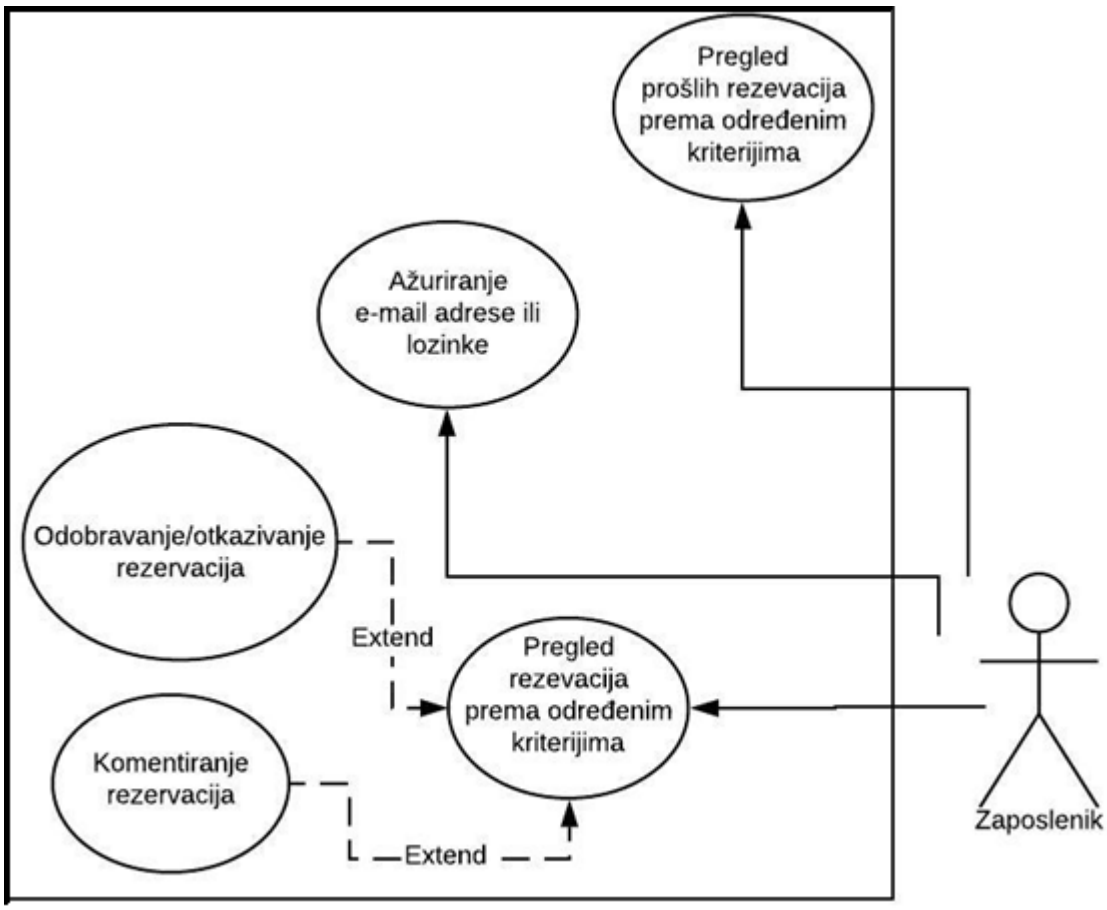
Zaposlenici koriste web aplikaciju radi praktičnijeg i efektivnijeg korištenja. Zaposlenici već sad koriste računala pa je web aplikacija jako praktična u tom slučaju. Glavna mogućnost zaposlenika jest pregled nadolazećih rezervacija sortiranih prema vremenu. Logika iz aplikacije za zaposlenike funkcionira na način da svi zaposlenici

mogu vidjeti popis svih nadolazećih rezervacija bez obzira za koju sastavnicu fakulteta su zaduženi. Razlog tome je vrlo funkcionalni filter koji će omogućiti zaposlenicima filtriranje rezervacije prema svojim preferencijama kao što su fakultet, e-mail studenta, razlog, datum i ostale informacije vezane za rezervacije. Zaposlenici isto tako imaju mogućnost odobravanja i otkazivanja rezervacija. Razlog zašto svi zaposlenici vide sve rezervacije je u tome što se u izvanrednim situacijama može omogućiti zaposlenicima da međusobno preuzmu studente ovisno o situaciji ukoliko primjerice jedan zaposlenik hitno mora napustiti referadu, a razlog posjeta je lako rješiv. Isto tako, upravljanje vremenom je efektivnije pošto se neki slučajevi mogu riješiti bez odlaska u referadu što doprinosi brzini i protoku studenata na dnevnoj bazi.

Prilikom odobrenja ili otkazivanja rezervacije, e-mail adresa zaposlenika koji je napravio tu radnju se sprema na bazu podataka te postaje vidljiv i studentu na mobilnoj aplikaciji prilikom pregleda rezervacije. Na taj način student može vidjeti e-mail adresu zaposlenika koji je njihovu rezervaciju odobrio ili otkazu u slučaju da ih žele kontaktirati preko e-mail servisa. Uz te mogućnosti, zaposlenici mogu komentirati svaku rezervaciju. Taj komentar služi kako bi student mogao vidjeti ukoliko zaposlenik ima nekakvu napomenu vezano uz odobrenu ili odbijenu rezervaciju što omogućuje direktniju komunikaciju bez upotrebe e-mail adresa. Uz navedene mogućnosti, zaposlenici isto tako mogu pregledavati i filtrirati prošle rezervacije koje služe kao arhiva što znači da se njihov status i komentar ne mogu ažurirati. Pomoću te arhive moguće je pregledati prošle slučajeve koji mogu pomoći pri rješavanju trenutnih problema. Zaposlenici još mogu i ažurirati svoje e-mail adrese i lozinke.

Odobravanjem, otkazivanjem ili komentiranjem na rezervaciju, podaci se ažuriraju u stvarnom vremenu, što znači da se ne provodi nikakvo osvježavanje stranice te su ažurirani podaci odmah vidljivi na korisničkom sučelju. Web aplikacija će isto tako sadržavati zaštitu za provjeru autentičnosti. To znači da, ukoliko korisnik nije prijavljen na aplikaciju, ne može pristupiti njenim sadržajima koji su namijenjeni zaposlenicima upisom linka u adresnu traku web preglednika. Na taj način se omogućuje da sustav kao cjelina funkcionira na vrlo praktičan način sa primarnim ciljem smanjenja redova čekanja, ali i bolju organizaciju vremena i ubrzavanje procesa odlaska u referadu pomoću bržeg odgovora zaposlenika preko komentara.

Slika 4. Use case dijagram – zaposlenik referade



Izvor: autor

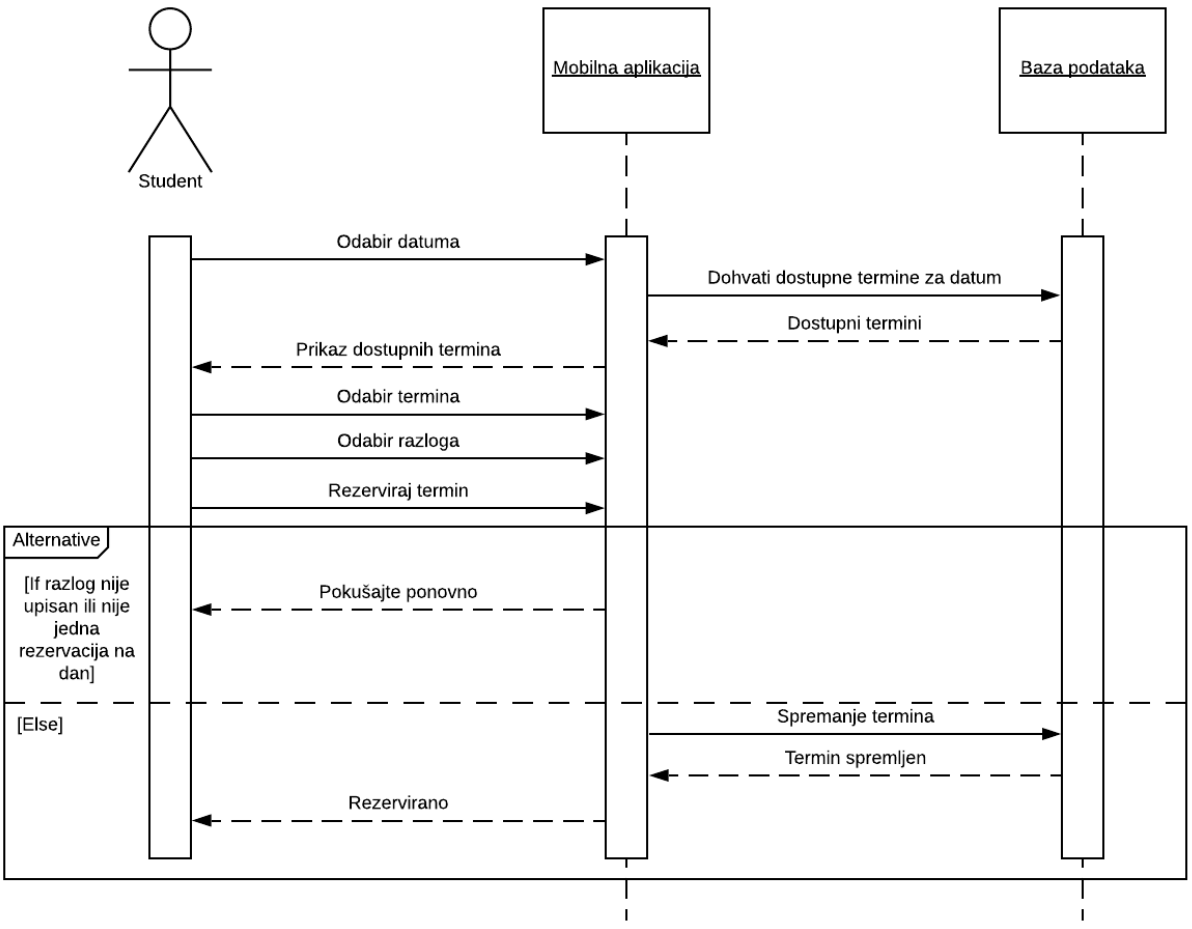
4. Korisnički scenariji i prototip sučelja

4.1. Korisnički scenariji

Korisnički scenariji se najlakše prikazuju sekvencijskim dijagramima. To su dijagrami koji na jednostavan način prikazuju proces obavljanja određenih radnji korak po korak, odnosno po određenoj sekvenci. Dijagram se sastoji od aktera i objekata. Akteri su našem slučaju studenti i zaposlenici referade, a objekti su mobilna aplikacija, web aplikacija te baza podataka.

Prilikom rezervacije, student odabire datum preko sučelja aplikacije, te se prilikom te radnje dohvaćaju dostupni termini iz baze podataka. Nakon odabranog datuma, termina i razloga provodi se provjera je li to prva rezervacija na taj datum.

Slika 5. Sekvencijski dijagram – student – rezervacija

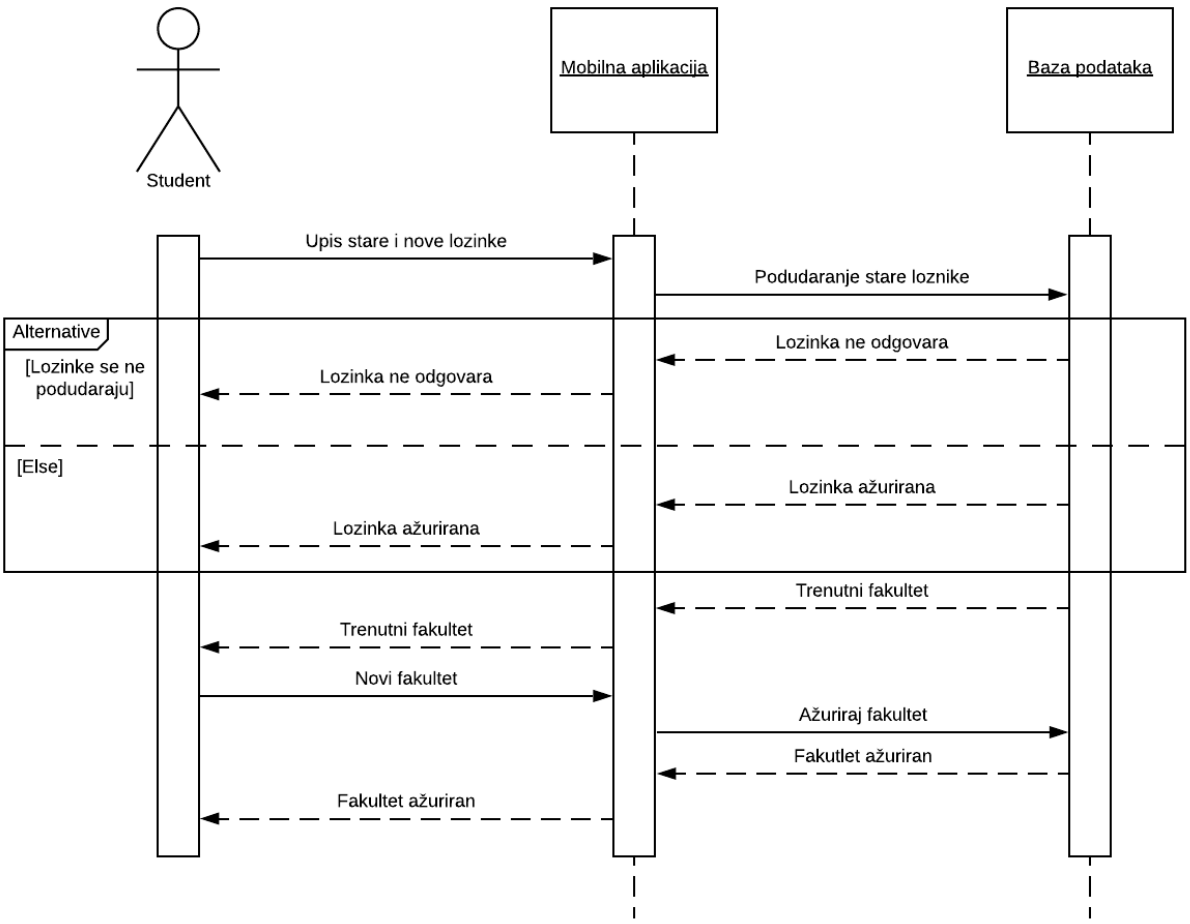


Izvor: autor

Uz taj uvjet, provjerava se i je li vlastiti razlog popunjen, ukoliko je odabran. Dohvaćanje dostupnih termina se bazira na ranije objašnjenom logici sastavnica fakulteta. Ukoliko neki od tih uvjeta nisu zadovoljeni, aplikacija ispisuje grešku. Ako su svi uvjeti zadovoljeni, termin se sprema na bazu podataka. Kod procesa uređivanja rezervacija proces je vrlo sličan, jedina krucijalna razlika je što se kao prvi korak dohvaćaju trenutni podaci o rezervaciji kako bi se mogli urediti. Proces pregleda rezervacija je jednostavan te se samo dohvaćaju sortirane rezervacije.

Kod opcija, student bira između ažuriranja lozinke ili fakulteta. Proces ažuriranja lozinke se sastoji od unosa trenutne (stare) i nove lozinke. Ukoliko lozinke ne odgovaraju, aplikacija prikazuje grešku. Ukoliko lozinke odgovaraju, ažuriranje lozinke se provodi na bazi podataka. Promjena fakulteta jednostavnija te se samo odabirom fakulteta isti i ažurira na bazi podataka bez potencijalnih greški.

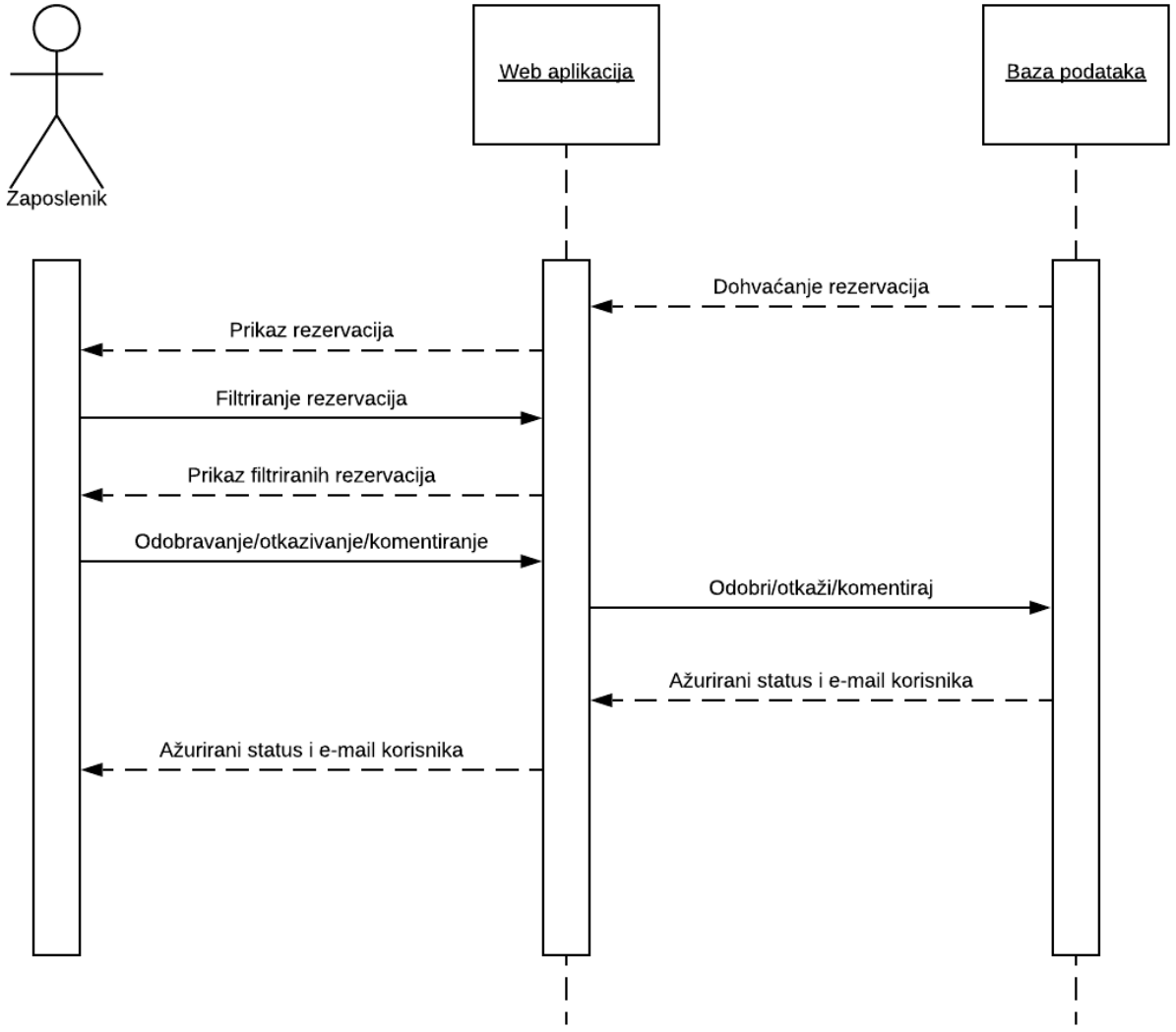
Slika 6. Sekvencijski dijagram – student – opcije



Izvor: autor

Kod zaposlenika glavna mogućnost je pregled rezervacija, filtriranje rezervacija te odobravanje/otkazivanje/komentiranje. Prilikom pregleda rezervacija, prvo se dohvaćaju i sortiraju sve rezervacije iz baze podataka te se iste prikazuju zaposleniku preko korisničkog sučelja web aplikacije. Zaposlenik zatim može rezervacije pretraživati i filtrirati po želji, a taj se proces obavlja isključivo preko web aplikacije i njenih ugrađenih funkcija koje upravljanju već dohvaćenim podacima iz baze podataka. Za razliku od pretraživanja i filtriranja, radnje kao što su odobravanje, otkazivanje ili komentiranje rezervacije zahtijeva spremanje ažuriranih podataka na bazu podataka. Sukladno tome, ažurirani podaci se spremaju na bazu podataka, no radi brzine prikaza podaci se isto tako ažuriraju na način da korisnici ne primijete proces spremanja na bazu. Na taj način se i ažurira i e-mail zaposlenika koji prikazuje zaposlenika koji je zadnji napravio promjenu na rezervaciji.

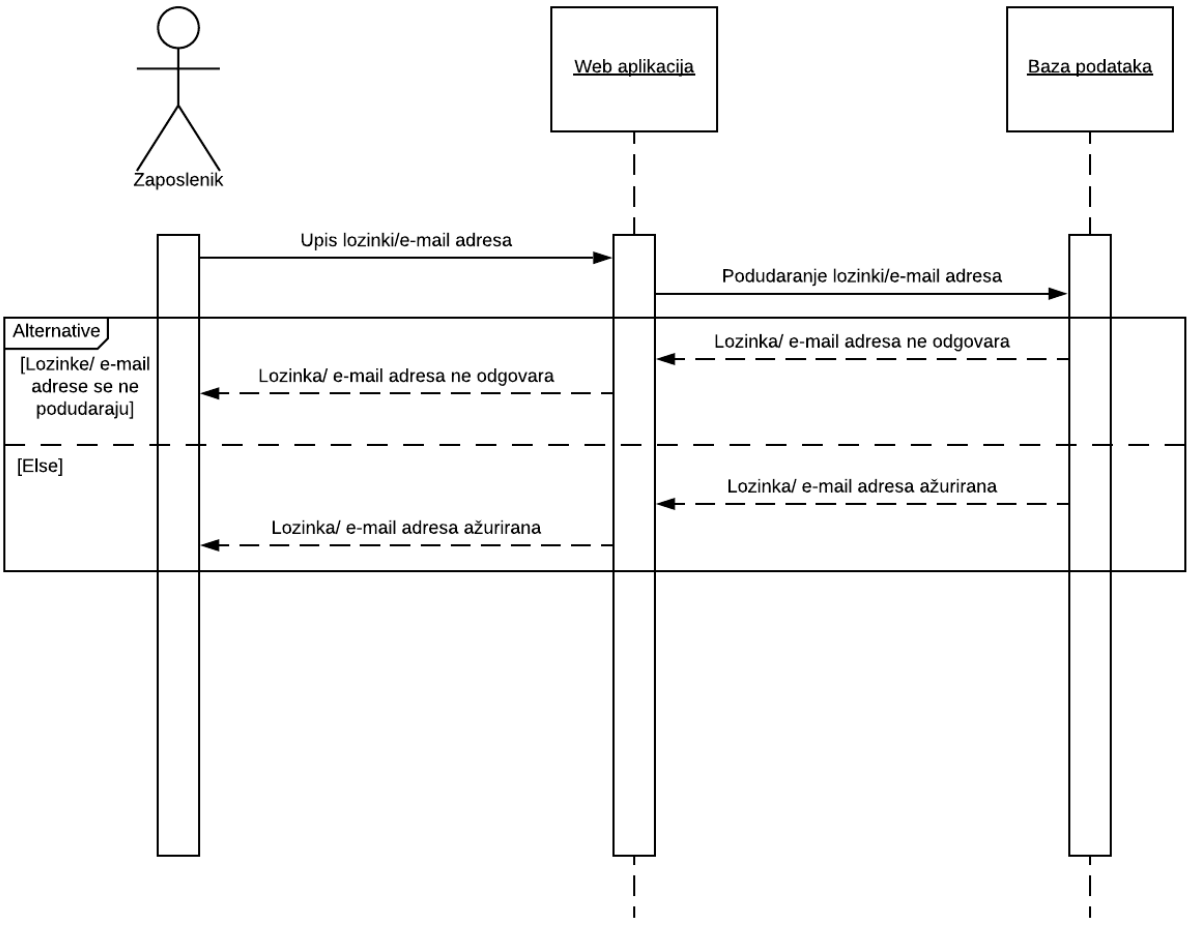
Slika 7. Sekvencijski dijagram – zaposlenik – rezervacije



Izvor: autor

Kao i studenti, zaposlenici isto tako imaju svoje opcije za ažuriranje e-mail adrese i lozinke. Proces se sastoji od unosa trenutne lozinke i e-mail adrese te nove e-mail adrese i lozinke, ovisno po želji. Naravno, ukoliko se lozinke ne podudaraju, zaposleniku se prikazuje greška. U protivnom, lozinka i e-mail adresa se uspješno ažuriraju preko baze podataka. Važno je napomenuti kako je cilj što manje koristiti i pozivati bazu podataka pošto radnje preko baze podataka znaju usporiti proces na korisničkom sučelju, a cilj je napraviti brzu i responzivnu aplikaciju. Dohvat podataka prilikom pokretanja aplikacije je jedna od tehnika koja upravlja podacima bez pozivanja baze podataka. U našem slučaju baza podataka se poziva samo kod ažuriranja podataka, a to znači da se korištenjem raznih tehnika podaci mogu ažurirati odmah, a pravo ažuriranje na bazi podataka se izvodi u pozadini. Riječ je ponekad o milisekundama, no pošto je cilj brza i efektivna aplikacija, takve tehnike su dobrodošle za poboljšavanje korisničkog iskustva s aplikacijom.

Slika 8. Sekvencijski dijagram – zaposlenik – opcije



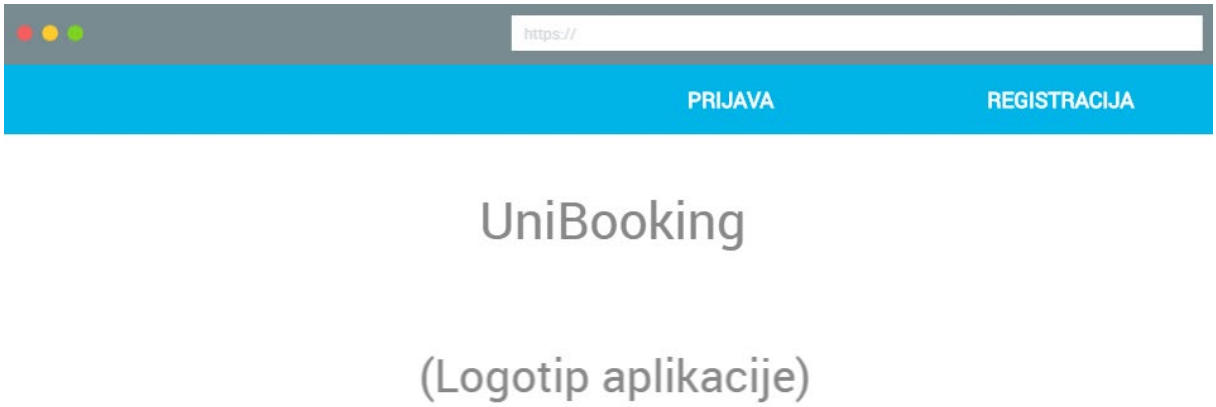
Izvor: autor

4.2. Prototip sučelja

Prototip sučelja služi za prezentaciju izgleda korisničkog sučelja prije same implementacije. Izrada prototipa se provodi pomoću raznih programa, a prototip može biti statički ili dinamički. Statički prikaz su samo slike, a dinamički prikaz omogućuje određenu razinu interakcije kao što je usmjeravanje na stranice klikom na gumbe. U poslovnim situacijama prototip služi kako bi se kupcu prikazalo što očekivati od krajnjeg proizvoda te se uz njegove povratne informacije prototip lako prilagođava. Bitno je što više informacija izvući iz kupca prije same implementacije kako ne bi došlo do neželjenih ili naglih promjena prilikom implementacije, no to je ponekad neizbježno.

Početni zaslon web aplikacije se sastoji od alatne trake te gumbi za prijavu i registraciju korisnika, a glavni sadržaj stranice se sastoji od imena i logotipa aplikacije UniBooking. Stranica za prijavu se sastoji od polja za unos e-mail adrese i lozinke te gumba za prijavu. Registracija slijedi slični koncept, jednina je razlika u tome što korisnika mora unijeti lozinku dva puta prilikom registracije. Nakon registracije, korisniku se nudi više mogućnosti unutar alatne trake kao što su pregled nadolazećih rezervacija, prošlih rezervacija, opcije te odjava. Prilikom pregleda nadolazećih rezervacija, zaslon prikazuje prostor za pretraživanje u kojeg korisnik upisuje pojmove koje pretražuje. Tu funkciju nadopunjuje filter koji omogućuje odabir kategorije pretraživanja. Filtriranje rezultata se provodi na temelju raznih čimbenika rezervacija kao što su fakultet, razlog, status, e-mail studenta i ostali čimbenici. Ispod prostora za pretraživanje i filtera nalazi se popis rezervacija koje su sortirane prema datumu uzlazno. Svaka rezervacija ima svoj odvojeni prostor (karticu) te sadrži sastavnicu fakulteta studenta, e-mail adresu studenta, datum, vrijeme i razlog. Zatim slijede gumbi koje zaposlenik referade koristi, a to su gumbi za odobrenje, otkazivanje i komentiranje. Odabir tih gumbi utječe na donji dio kartice što znači da se podaci ažuriraju ovisno o tome koji korisnik je odabrao gumb. Status se mijenja ovisno o odabiru gumba, ali informacija o tome tko je uredio status ili komentar se postavlja na e-mail adresu korisnika koji je radio promjene na rezervaciju pomoću tih gumbi. Komentiranjem se otvara skočni prozor u kojem se piše komentar. Ukoliko već postoji komentar, otvaranje skočnog prozora sadrži postojeći komentar. Prošle rezervacije imaju sve iste mogućnosti kao i nadolazeće rezervacije, no nemaju gumbe pošto služe kao arhiva za pregled i filtriranje. Opcije se sastoje od prostora za upis trenutnih podataka i novih za prijavu koji se ažuriraju klikom na gumb.

Slika 9. Web aplikacija – početni zaslon

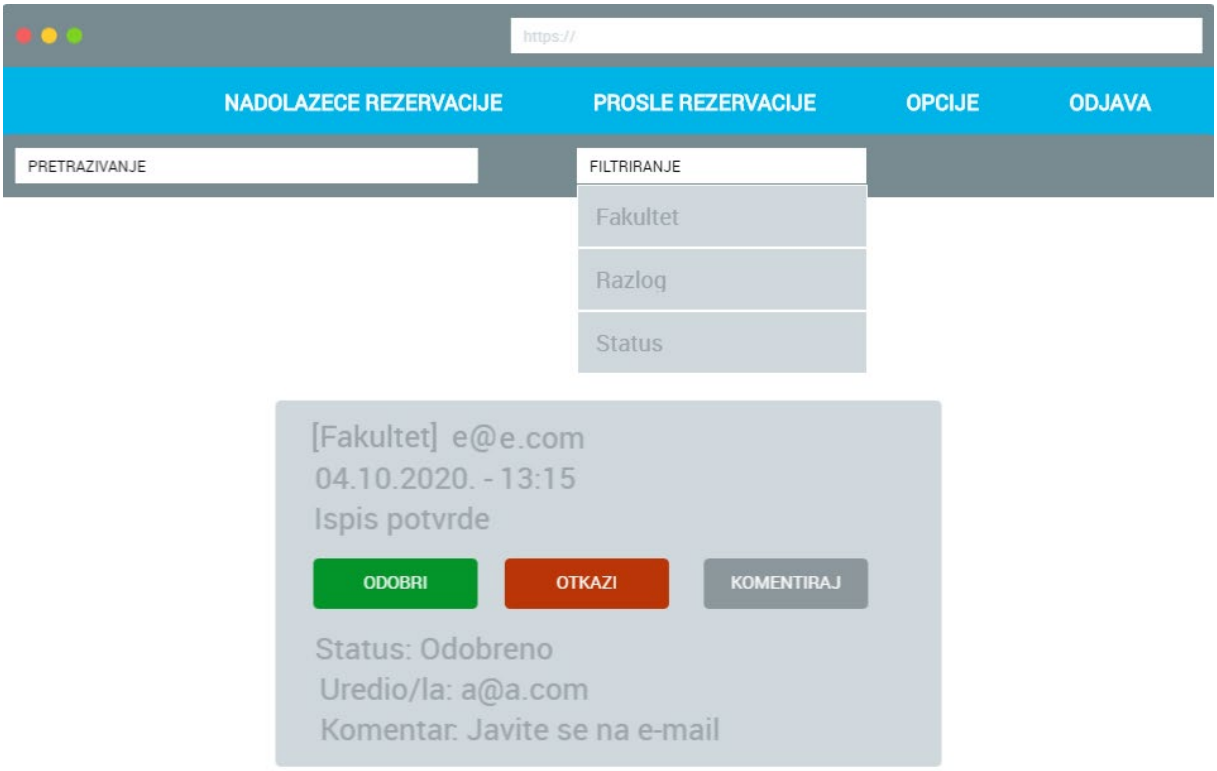


Izvor: autor

Slika 10. Web aplikacija – prijava

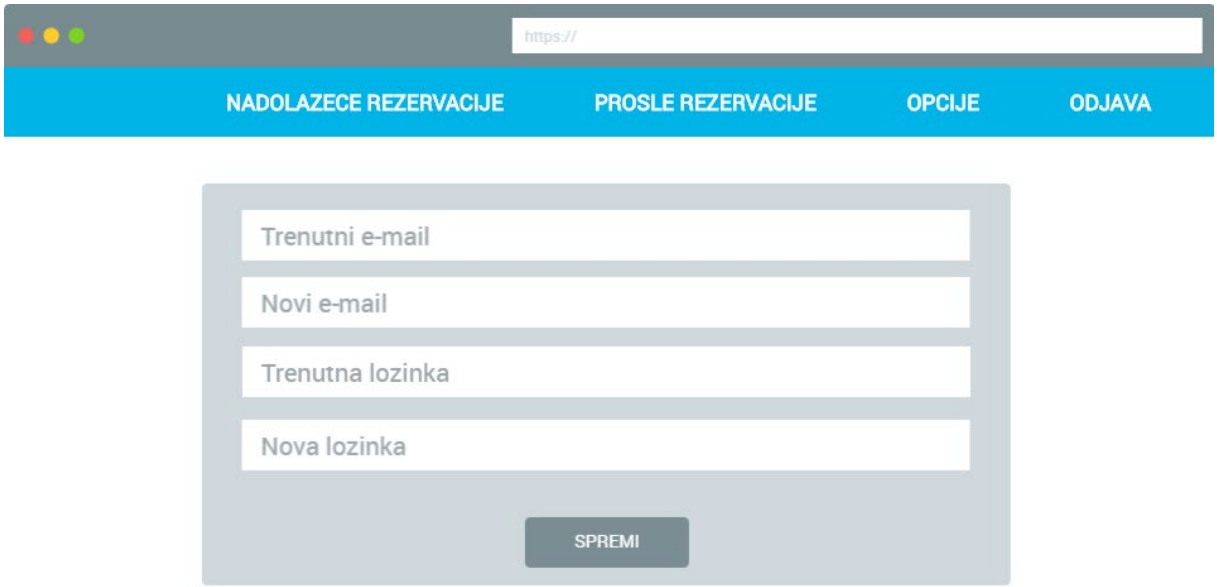
Izvor: autor

Slika 11. Web aplikacija – nadolazeće rezervacije



Izvor: autor

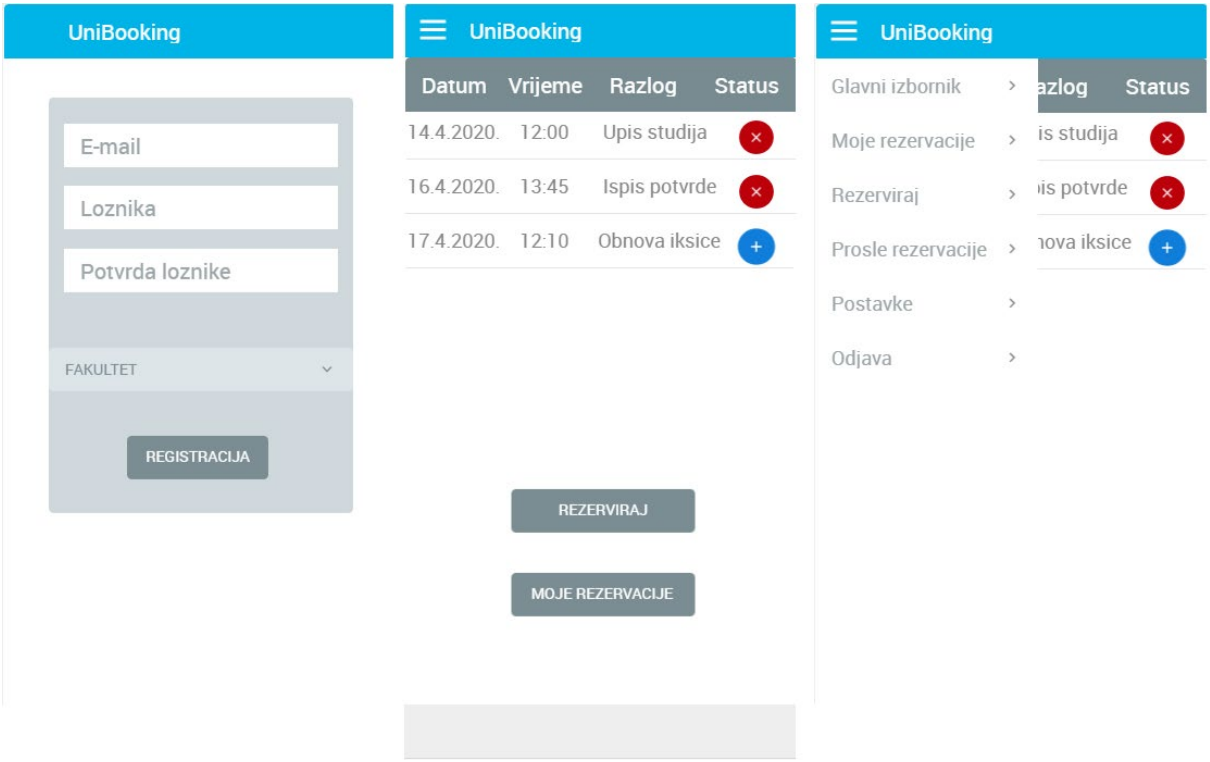
Slika 12. Web aplikacija – opcije



Izvor: autor

Mobilna aplikacija prilikom prvog pokretanja prikazuje početni zaslon sa logotipom aplikacije te dva gumba za registraciju ili prijavu. Zaslon za registraciju se sastoji od unosa e-mail adrese, lozinke koju je potrebno i potvrditi te odabira fakulteta. Na dnu se nalazi gumb za registraciju ukoliko su svi uneseni podaci korektni. Zaslon za prijavu ima sličan izgled, no bez odabira fakulteta te se lozinka unosi samo jednom. Ukoliko je korisnik već bio prijavljen, početni zaslon prikazuje popis top 3 nadolazeće rezervacije koje su sortirane uzlazno. Te rezervacije prikazuju datum, vrijeme, razlog te status koji je prikazan slikom, slika može prikazati otkazanu rezervaciju, odobrenu rezervaciju ili rezervaciju na čekanju. Isto tako, početni zaslon sadrži gumbe za rezervaciju i pregled vlastitih rezervacija. Popis rezervacija sadrži sortiranu listu rezervaciju prema istom formatu kao i početni zaslon, no ne sadrži gumbe već se na glavnom području zaslona su samo rezervacije. Kroz cijelu aplikaciju, osim kod prijave i registracije, za navigaciju se koristi izbornik sa strane koji se poziva odabirom simbola za izbornik u gornjem lijevom kutu.

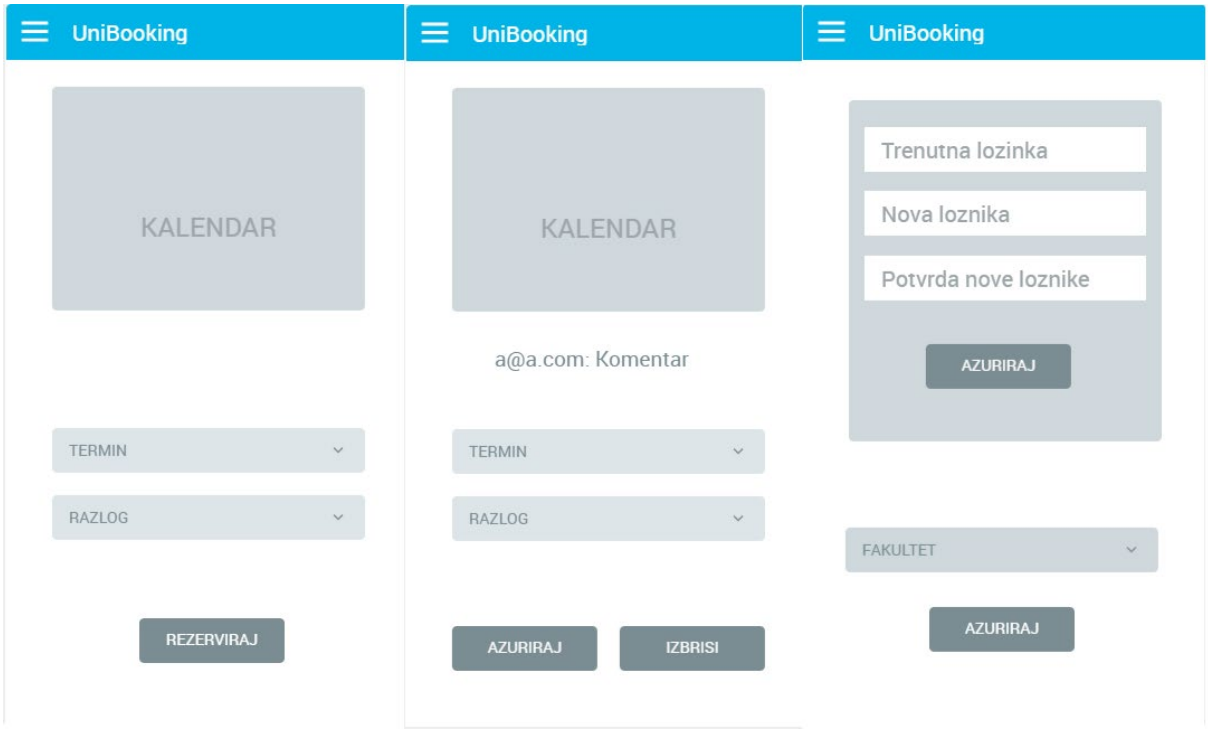
Slika 13. Mobilna aplikacija – registracija, početni zaslon i izbornik



Izvor: autor

Zaslon za rezervaciju se sastoji od kalendara, odabira termina i razloga. Na dnu zaslona se nalazi gumb za spremanje rezervacije. Na početnom zaslonu te na zaslonu s popisom rezervacija se odabirom rezervacije dolazi na zaslon za uređivanje rezervacije. Zaslon sadrži podatke odabrane rezervacije te uz to sadrži i e-mail adresu i komentar osobe koja je rezervaciju odobrila ili otkazala. Ukoliko zaposlenik nije komentirao odabranu rezervaciju, komentar i e-mail adresa se ne prikazuju. Na kraju, postoje gumbi za spremanje ažuriranih podataka ili brisanje rezervacije. Opcije isto tako sadrže prostor za unos trenutne i nove lozinke sa odgovarajućim gumbom za spremanje promjena. Promjena se još može izvesti i za fakultet odabirom fakulteta te pritiskom na gumb za ažuriranje.

Slika 14. Mobilna aplikacija – rezervacija, uređivanje rezervacije i opcije



Izvor: autor

5. Implementacija web aplikacije

Implementacija UniBooking web aplikacije uzima u obzir poznavanje osnova JavaScript i HTML jezika i njihovih funkcionalnosti. Ovo poglavlje služi za objašnjenje i opis implementacije ključnih dijelova web aplikacije koji su krucijalni za njenu funkcionalnost. UniBooking web aplikacija koristi Vue.js okruženje, Vuetify paket komponenti te bazu podataka Firebase. Koncept Vue.js aplikacije se bazira na tome da vue datoteke pozivaju funkcije koje su definirane u JavaScript datotekama, a Vuetify paket komponenti se koristi za slaganje i prikazivanje komponenti unutar vue datoteka. Mogućnosti aplikacije su registracija, prijava, pregled nadolazećih rezervacija, pregled prošlih rezervacija, te promjena podataka za prijavu.

5.1. Osnovni oblik aplikacije i usmjeravanje na stranice

Upute za instalaciju Vue.js okruženja se nalaze na njihovoj službenoj stranici (Vue.js Installation, 2020). Nakon instalacije okruženja, potrebno je instalirati i paket komponenti Vuetify, a upute za instalaciju se nalaze na njihovoj službenoj stranici (Vuetify Quick start, 2020). Nakon instalacije potrebnih komponenti, započinje se rad u generiranoj App.vue datoteci. Ta datoteka služi za generiranje alatne trake *single-page* aplikacije. *Property* `<template>` služi za označavanje granica koda za gradnju izgleda stranice, dok `<script>` služi za upravljanje podacima i pozivanje funkcija.

Za izradu izgleda stranice koriste se Vuetify komponente. Komponente se sastoje od lista, gumbi, ikonica, alatne trake, kalendara, formi i mnogo drugih komponenti. Izrada izgleda aplikacije bazira se na HTML jeziku i strukturi. Pomoću `<v-toolbar>` komponente za izrađuje alatna traka, a sadržaj trake se povlači iz polja. Pošto polje predstavlja podatke, to je indikator da se to polje mora definirati unutar `<script>`. Cilj je postaviti uvjet da se sadržaj alatne trake mijenja s obzirom na to je li korisnik prijavljen ili nije. Radi toga, izrađuju se dva polja i s time da se postavlja uvjet je li korisnik prijavljen. Funkcija će biti definirana nakon implementacije i inicijalizacije baze podataka, no zasada je dovoljno napisati ime funkcije koja je u ovom slučaju `userIsAuthenticated`. Kako bi se popis opcija alatne trake prikazao, potrebno je unutar komponente ispisati podatke pomoću petlje, pošto se radi o polju podataka. Unutar komponente je moguće provoditi uvjete i petlje kao što su `v-if` i `v-for`. Također, potrebno je povezati to polje pomoću `:key` koji služi kao unikatni identifikator.

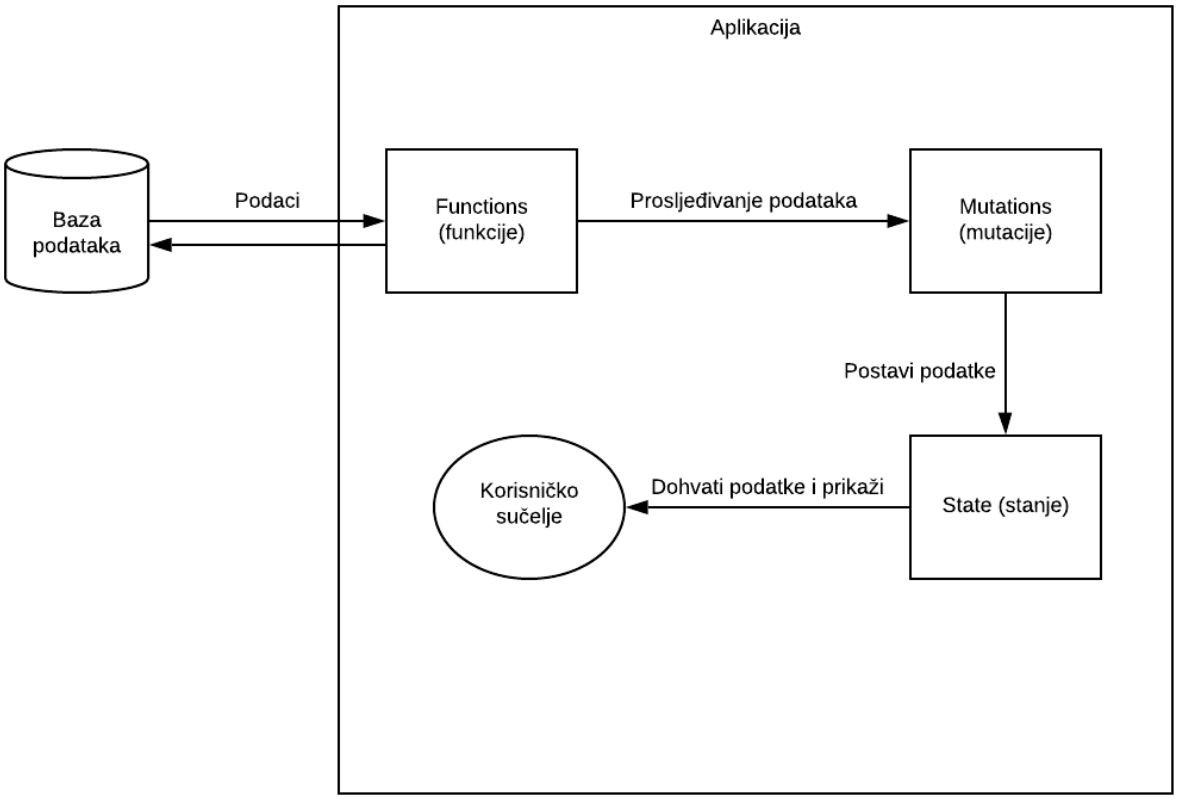
Prvi korak pri usmjeravanju stranica je izraditi vue datoteke za sve stranice na koje želimo povezati i usmjeriti. Unutar App.vue datoteke je zato potrebno definirati `<router-view>` unutar `<main>` komponente za usmjeravanje. Unutar mape „router“ postoji indeks.js datoteka u kojoj se provodi usmjeravanje, odnosno *routing* stranica. Mapa i datoteka su generirane ukoliko se pri instalaciji specificira da se ta mogućnost želi dodati u aplikaciju. Početna stranica će imati adresu „/“ pošto je to prva stranica koja se prikazuje korisniku pri otvaranju web aplikacije. Za ostale stranice se adrese mogu odrediti po želji. Za svaku stranicu koja se definira i usmjeri, potrebno je napraviti uvoz, odnosno import iste na vrhu datoteke uz točnu adresu gdje se odgovarajuća datoteka za tu stranicu nalazi. Povezane stranice je isto tako potrebno navesti i unutar App.vue datoteke u ranije definirana polja. Potrebno je dodati link *property* za svaku stranicu koja je navedena u polju te unutar jednostrukih navodnika navesti adresu na koju stranica vodi prema ranije definiranom usmjerivaču. Kako bi se preko korisničkog sučelja moglo usmjeriti na druge stranice, potrebno je unutar for petlje dodati `router :to=item.link` za povezivanje stranica iz polja. Uz povezivanje stranica, potrebno je postaviti zaštitu za provjeru autentičnosti koja će spriječiti korisnike da pristupe adresama stranica koje su namijenjene samo prijavljenim korisnicima. Unutar *routing* mape izrađuje se nova JavaScript datoteka te će se ta funkcionalnost implementirati nakon povezivanja sa bazom podataka jer se preko baze podataka dobiva informacija o tome je li korisnik prijavljen.

5.2. *Vuex state management* i baza podataka Firebase

Za mogućnost korištenja funkcija te njihovo urednije spremanje i pozivanje koristi se Vuex *state management*. Upute za instalaciju se nalaze na njihovoj službenoj stranici (Vuex Installation, 2020). Nakon instalacije potrebno je definirati mapu „store“ koja će sadržavati JavaScript datoteke sa funkcijama koje se mogu kategorizirati na korisnika i rezervaciju. Store u sebi sadrži stanje, mutacije, *gettere* i akcije. Stanje se koristi za povezivanje, punjenje i dohvaćanje podataka sa popisa. Pomoću njega se dodjeljuju vrijednosti podacima iz polja tako da se ti podaci mogu pozvati iz vue datoteka. Mutacije služe za izmjenu tih podataka iz stanja dok akcije služe za izvršavanje upita na bazu podataka. Dakle, u slučaju nadolazećih rezervacija, akcija dohvaća podatke koje prosljeđuje u mutaciju. Mutacija zatim puni te podatke u stanje

te su ti podaci dostupni za dohvaćanje iz vue datoteka. Za dohvat tih datoteka se koriste *getteri*.

Slika 15. Protok podataka iz baze podataka do korisničkog sučelja



Izvor: autor

Firebase je baza podataka čiji je vlasnik multinacionalna kompanija Google. Predstavlja efektivno i besplatno rješenje za korištenje baze podataka bez potrebe plaćanja hosting servisa. Upotreba baze podataka Firebase zahtijeva Google račun te je proces integracije u aplikaciju jednostavan. Pomoću web preglednika određuje se način prijave korisnika na bazu podataka, što je ovom slučaju e-mail adresa i lozinka. Prilikom integracije baze u web aplikaciju, potrebno je kopirati generirani kod koji je nastao odabirom na Firebase izborniku preko web preglednika. Taj kod služi za inicijalizaciju baze podataka sa aplikacijom te se isti postavlja unutar Main.js datoteke koja sadrži sve glavne funkcije prilikom pokretanja web aplikacije.

Nakon postavljanja baze podataka, poželjno je odmah i kreirati funkciju koja prati je li korisnik prijavljen. Funkcije vezane za autentikaciju koje prate promjenu korisnika se pozivaju pomoću `firebase.auth().onAuthStateChanged`. Ukoliko je

korisnik prijavljen, poziva se `autoSignIn` funkcija kojoj se prosljeđuju podaci korisnika. Definicija te funkcije se nalazi unutar „store“ mape te dodjeljuje ID korisniku. Alatna traka isto tako ima ažuriranje s obzirom na bazu podataka. Jednostavno se poziva *getter* koji provjerava je li korisnik prijavljen te vraća *true* ili *false* što je nadovezivanje na funkciju koja prati promjene stanja korisnika.

5.3. Registracija i prijava

Registracija i prijava su po svojoj strukturi vrlo slični procesi u web aplikaciji. Prilikom registracije koristi se `<form>` koja se puni sadržajem za popunjavanje. Svaki `<text-field>` ima svoje ime, ID, model, tip unosa i indikator je li to polje obavezno. Kod registracije je još bitno provjeriti i odgovaraju dvije lozinke iz različitih `<text-field>` komponenti. Provjera se provodi jednostavnom usporedbom koja vraća *true* ili *false* bez potrebe pozivanja funkcija iz drugih datoteka. Ukoliko su podaci krivo upisani, ispisuje se poruka greške na način da se dohvati i ispiše poruka preko baze podataka Firebase. Klikom na gumb, ukoliko su svi podaci pravilno uneseni, prosljeđuju se podaci potrebni za registraciju, a to su e-mail adresa i lozinka. Poziva se ranije navedeni *store* te `dispatch` uz naziv funkcije koja se poziva sa argumentima koji se šalju u zagradama. Unutar funkcije, moguće je postaviti i pozvati animirani rotirajući simbol koji služi za čekanje dok se funkcija izvršava. Simbol je moguće implementirati unutar gumba, ali i preko cijelog korisničkog sučelja. Iako su u većini slučajeva u pitanju samo milisekunde, radi korisničkog doživljaja rotirajući simbol „loading“ daje korisniku obavijest kako se proces izvršava. Unutar funkcije, korisnik se registrira pomoću ugrađenih Firebase funkcija u kategoriji `auth()`. Funkcija koristi proslijeđene podatke iz korisničkog sučelja te se nakon izvršenja funkcije izrađuje konstanta koja se prosljeđuje do stanja preko mutacije. Na taj način se vrlo brzo pomoću tog stanja dohvaća trenutno prijavljeni korisnik aplikacije.

Proces prijave prati slični postupak, no ima nekoliko manjih razlika. Forma za popunjavanje podataka sadrži samo polja za unos e-mail adrese i lozinke uz isti format kao i registracija. Razlika je što se tijekom prijave koristi druga Firebase funkcija sa podacima koje je korisnik upisao u formu. Nakon izvršavanja, ukoliko prijava nije uspješna, dohvaća se poruka u grešci i prikazuje korisniku. Ostatak funkcije na isti način prosljeđuje podatke u stanje za brzo dohvaćanje.

5.4. Nadolazeće i prošle rezervacije

Prilikom izrade stranice za popis svih nadolazećih rezervacija koristi se lista pošto se radi o nizu podataka koji se ponavljaju. Nakon pozicioniranja i definiranja dimenzija sadržaja na stranici, potrebno je koristiti `<v-card class="info">` komponentu koja predstavlja karticu, odnosno zasebni prostor za svaku pojedinačnu rezervaciju. Svaka kartica će imati isti oblik, odnosno prikazivat će se niz podataka u istom obliku. Podaci se dijele na pisani dio i dio za akcije od strane korisnika. Korisnik može imati interakciju sa gumbima svake kartice. Podaci sadrže sve podatke o rezervaciji sa baze podataka dok gumbi utječu na status svake rezervacije.

Pravi korak je dohvaćanje rezervacija iz baze podataka. Prilikom samog pokretanja aplikacije poziva se funkcija koja učitava sve rezervacije iz main.js datoteke. To znači da se iz vue datoteke koriste samo *getteri* za dohvaćanje već pripremljenih podataka iz stanja. Dohvaćanje podataka se unutar funkcije izvršava pomoću petlje pošto se prolazi kroz niz podataka. Dohvaćanje podataka se provodi korištenjem Firebase funkcije koja pomoću reference dohvaća vrijednosti iz definirane grane podataka.

Baza podataka Firebase je strukturirana u obliku stabla što znači da se provodi grananje podataka. U ovom slučaju, grana „Rezervacije" se sastoji od niza ID podataka koji u sebi sadrže podatke o rezervacijama. Svaki ID predstavlja granu u kojoj se nalaze podaci za pojedinu rezervaciju. Dakle, odabirom grane za dohvaćanje podataka, dohvaćaju se svi podaci unutar te grane. Kako bi se dohvaćene podatke spremilo, potrebno je napraviti polje. Za prolazak kroz dohvaćene podatke koristi se petlja, a prije prolaska kroz petlju potrebno je definirati konstantu koja služi da dohvaćanje podataka iz svake grane. Ulaskom u petlju potrebno je dohvatiti samo nadolazeće rezervacije. Svaka rezervacija sadrži i*nteger* tip podataka nazvan „timeStamp" koji sadrži broj milisekundi od 1970. godine. Na taj način se omogućuje jednostavna usporedba podataka sa trenutnim vremenom. Trenutno vrijeme za dohvaća u formatu datuma te se pretvara *integer* kako bi se mogao usporediti. Ukoliko je datum rezervacije iz baze podataka veći od trenutnog datuma, to je indikator kako se rezervacija nalazi u budućnosti te se ista sprema u ranije deklarirano polje. Pošto se radi o petlji, taj se proces ponavlja dokle god ima dodanih grana u bazi podataka. Pomoću funkcija referenci baze podataka moguće je pozicionirati na bilo koji dio baze

podataka ukoliko se zna ime ili ID grane kao referenca. Nakon izvršavanja petlje podaci se prosljeđuju kroz mutaciju koja postavlja podatke unutar stanja. Podaci se zatim iz stanja jednostavno dohvaćaju iz vue datoteka i prikazuju u obliku kartica koristeći for petlju za prikaz.

Prošle rezervacije prate slični proces dohvaćanja i prikazivanja podataka uz nekoliko manjih razlika. Funkcija je gotovo ista, no razlika je u usporedbi za spremanje rezervacija u polje. Razlika je samo u tome što se odabir vrši prema suprotnom uvjetu. Ukoliko je datum rezervacije manji od trenutnog datuma isti se sprema u polje što je to indikator da je rezervacija u prošlosti. Naravno, prošle rezervacije koriste vlastitu funkciju, mutaciju i stanje. Razlika je još u korisničkom sučelju ta da se unutar kartica ne prikazuju gumbi za interakciju te je to jedina razlika u prikazu tih podataka.

Nadolazećim i prošlim rezervacijama zajedničke su mogućnosti pretraživanja i filtriranja. Te su mogućnosti zapravo međusobno povezane pošto filter utječe na koji tip podataka se pretražuje. Na korisničkom sučelju se prikazuje `<input>` komponenta u obliku za pretraživanje, odnosno `v-model="search"`. Upravo taj model koristi filter kako bi prikazao željeni podatak korisniku. Filter je zapravo obična `<select>` komponenta sa ponuđenim opcijama za odabir. No unutar te komponente se promjenom odabira poziva funkcija koja prilagođava pretraživanje ovisno o odabranoj vrsti podataka iz filtera. Naravno petlja za prikazivanje kartica se bazira na filtriranom sadržaju, a ne na svim rezervacijama.

5.5. Promjena podataka za prijavu

Promjena podataka za prijavu ima sličnu strukturu na korisničkom sučelju kao i registracija ili prijava. Sastoji se od forme koju je potrebno popuniti sa trenutnom e-mail adresom i lozinkom te novom e-mail adresom i lozinkom. Gumb ostaje sakriven dok se cijela forma ne popuni, a to je moguće postavljanjem uvjeta koji vraća *true* ukoliko je cijela forma popunjena. Na taj način se sprječava korisnika da pošalje nepotpunu formu.

Uneseni podaci se spremaju u objekt preko kojeg je moguće pristupiti svim njegovim sadržajima. Funkcija koja mijenja podatke o korisniku prvo vrši prijavu korisnika radi provjere unesenih podataka za trenutnu prijavu. Ukoliko su podaci točni,

koriste se, dohvaća se aktivni korisnik te se ažuriraju e-mail adresa i lozinka pomoću odgovarajućih Firebase funkcija. Nakon uspješno obavljene radnje promjene podataka, korisniku se osim prestanka učitavanja prikazuje i skočna poruka unutar web preglednika koja ga informira o uspješno ažuriranim podacima. Ukoliko trenutni podaci korisnika nisu bili točno uneseni, ispisuje se poruka greške koja je vidljiva korisniku na sučelju.

6. Implementacija mobilne aplikacije

Mobilna aplikacija uzima u obzir znanje Java ili Kotlin programskog jezika za razvoj. Koncept razvoja Android aplikacije se zasniva na izradi korisničkog sučelja unutar XML datoteka te povezivanje i dodavanje funkcionalnosti komponenta unutar java datoteka pošto u ovom slučaju aplikacija je izrađena u Java programskom jeziku. Implementacija prati na koji način se postavljanju i izvršavaju glavne funkcionalnosti aplikacije. Prvi korak uzima u obzir izradu nove aplikacije u Android Studio programu te postavljanje baze podataka Firebase. Proces postavljanja baze je pojednostavljen unutar Android Studio programa te se cijeli proces bazira na nekoliko klikova koji su prikazani uz interaktivne korake. Naravno, za postavljanje i inicijalizaciju baze podataka potrebno na sličan način kao i u web aplikaciji postaviti tražene podatke kao što su jedinstveni ID baze u odgovarajuću datoteku. Taj proces je pojednostavljen te se baza inicijalizira kopiranjem generirane JSON datoteke sa službene Firebase stranice.

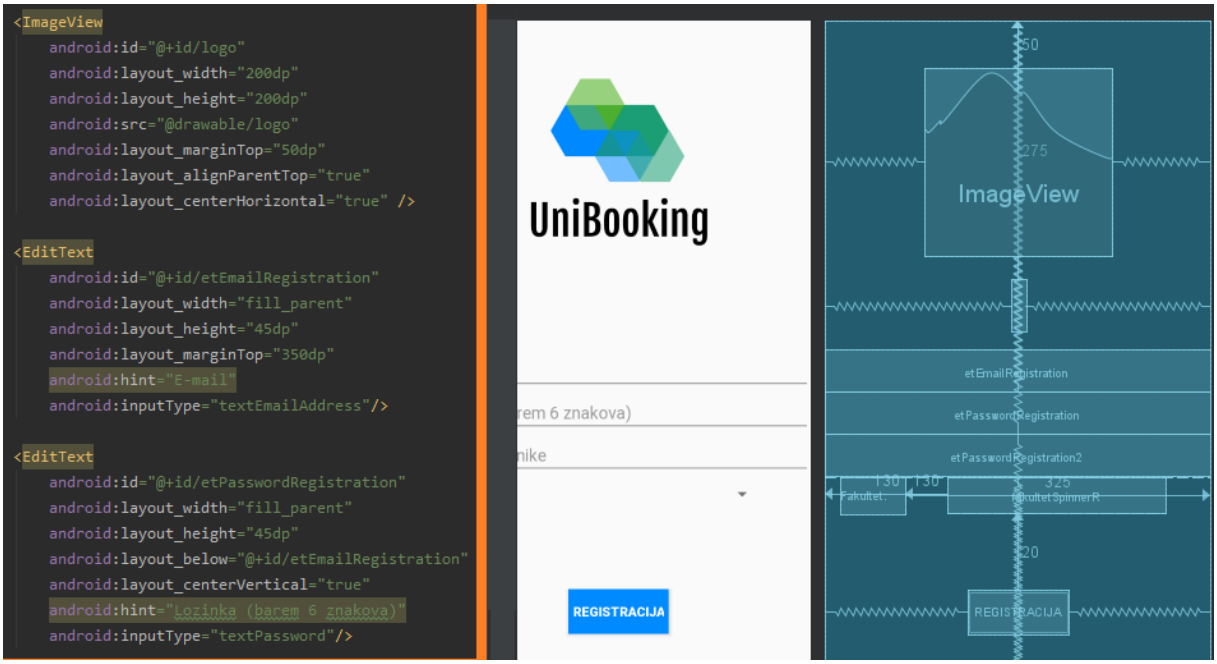
6.1. Registracija i prijava

Nakon izrade novog projekta i postavljanje baze podataka, prvi korak kod izrade aplikacije za studentsku referadu su funkcionalnosti registracije i prijave. Za izradu korisničkog sučelja koristi se XML datoteka. Komponentama iz XML datoteke se dodaju funkcionalnosti preko java datoteke. Java datoteka sadrži sve funkcije i procese upravljanja podacima dok XML datoteke isključivo služe za prikazivanje korisničkog sučelja i interakciju sa korisnikom.

Nakon kreiranja nove XML datoteke, započinje gradnja korisničkog sučelja koja se sastoji od slaganja komponenti. Svaka komponenta ima svoj ID koji se kasnije koristi za povezivanje sa Java datotekom. Svakoj komponenti moguće odrediti razne specifikacije kao što su širina, dužina, margine i centriranje. Kod registracije postavlja se *ImageView* koji sadrži i izvor slike koja se prikazuje, odnosnu njezinu adresu. *EditText* se koristi za lozinke što znači da se može postaviti i smjernica korisniku unutar prostora komponente te može postaviti način unosa tako da se prilagodi prikaz teksta ukoliko se nalazi o e-mail adresi ili lozinki. *Spinner* za odabir fakulteta sadrži samo svoju definiciju i izgled, a njegove mogućnosti se postavljaju i pune pomoću Java

datoteke. Gumb za potvrdu registracije se nalazi ispod navedenih komponenti te sadrži boju pozadine i boju i veličinu slova. Slično kao u web aplikaciji, mobilna aplikacija isto tako može koristiti *progress bar*, odnosno simbol za učitavanje kojim se isto tako upravlja pomoću Java datoteke.

Slika 16. XML datoteka – slaganje i definiranje komponenti



Izvor: autor

Nakon uređivanja korisničkog sučelja, Prvi korak unutar Java datoteke je definirati i povezati komponente iz XML datoteke. Komponente se povezuju pomoću `findViewById` funkcije te se unutar zagrada poziva komponenta prema svom ID imenu uz R.id prije imena. Za korištenje baze podataka i njezinih mogućnosti, potrebno je napraviti referencu na željenu poziciju. Isto tako, za upravljanje korisnicima moguće je pozvati instancu `FirebaseAuth`.

Za postavljanje opcija u *spinner* potrebno je napraviti listu tipa *String* te dodati sve željene opcije te dodijeliti ih pomoću adaptera. Također je potrebno dodijeliti listener koji prati kada je odabrana opcija za *spinner*. *Listener* je funkcija koja se poziva ukoliko je zadovoljen određeni uvjet što je u ovom slučaju svaki put kada se odabere nova opcija. Funkcija sprema odabir korisnika unutar instance klase student. Klasa student se sastoji od svih podataka koji se spremaju na bazu podataka, a to su ID, e-mail adresa i lozinka. Gumb isto tako ima svoj *listener* koji se poziva klikom od strane

korisnika. Prvi korak nakon klika korisnika potrebno je napraviti provjeru jesu li sva polja popunjena. Ukoliko nisu, prikazuje se poruka greške korisniku. Zatim se postavlja *progress bar* koji je indikator korisniku da je proces registracije započeo. Pomoću Firebase funkcije se obavlja registracija te se spremaju podaci na bazu koji su prikupljeni iz instance klase student. Nakon obavljenog procesa uklanja se *progress bar* te se poziva nova aktivnost koja preusmjerava korisnika na novu stranicu, što je ovom slučaju glavni izbornik aplikacije.

Proces prijave prati vrlo sličan proces kao i registracija. XML datoteka je jednostavnija te za izradu jer ne sadrži *spinner* već sadrži samo prostore za unos e-mail adrese, lozinke i gumba za prijavu. Na isti se način povezuju komponente i dodjeljuju se funkcije koje se pozivaju ovisno u uvjetima. Te su funkcije zapravo već ugrađene u sklopu Java programskog jezika i Android studio programa te ih nije potrebno programirati samostalno. Početni zaslon aplikacije je isto vrlo jednostavan za implementaciju te sadrži samo dva gumba koji usmjeravaju na registraciju ili prijavu. Aplikacije isto tako prije pokretanja unutar aktivnosti „Home“ provjerava je li korisnik prijavljen u aplikaciju. Ukoliko je, korisnik se odmah prebacuje na glavni izbornik bez potrebe za ponovnom prijavom u aplikaciju. To se postiže sa pozivanjem instanci ugrađenih Firebase funkcija vezanih za trenutno prijavljenog korisnika, a funkcija vraća *true* ili *false* što je povratna informacija o statusu korisnika.

6.2. Moje rezervacije, prošle rezervacije i glavni izbornik

Moje rezervacije predstavljaju popis nadolazećih rezervacija prijavljenog studenta. Ideja se temelji na dohvaćanju rezervacija studenta iz baze podataka i njihovo prikazivanje na korisničkom sučelju. Podatke je poželjno formatirati unutar liste i omogućiti da se klikom na njih prebacuje na novu stranicu, odnosno aktivnost. Za formatiranje adaptera liste potrebno je napraviti posebnu Java datoteku te definirati popis podataka za prikazivanje unutar liste. Taj popis je potrebno povezati sa XML datotekom kako bi prikaz liste bio omogućen. Povezivanje se provodi na sličan način kao i prilikom registracije i prijave, no u ovom slučaju koristi se integrirani `LayoutInflater`. U toj se datoteci isto dodjeljuju slike ovisno o statusu rezervacije pošto je status rezervacije na bazi podataka spremljen u tekstualnom obliku.

Prije prikazivanja sadržaja na sučelju, potrebno je dohvatiti podatke iz baze podataka. Ponovno se koristi referenca na željenoj poziciji koja je u ovom slučaju „Rezervacije“ u kojoj se nalazi popis rezervacija na temelju njihovog ID podatka. Unutar reference se postavlja *listener* te se izrađuje *snapshot* svih podataka. Kroz *snapshot* se prolazi petljom te se tako slijedno dohvaćaju podaci. Prije prikazivanja u listi mora se napraviti provjera je li rezervacija zaista napravljena od strane korisnika na temelju e-mail adrese koja se automatski sprema prilikom rezervacije. Nakon provjere potrebno je podatke spremiti u instancu klase „Rezervacija“. To je klasa koja sadrži sve podatke koje sadrže rezervacije na bazi podataka. Klasa ima svoje *getter* i *setter* funkcije kao i konstruktor što omogućuje lako upravljanje podacima.

Potrebno je podatke dodijeliti datotekama tipa *String* te napraviti provjeru vezanu za datum rezervacije, pošto je cilj prikazati samo buduće rezervacije. Još je potrebno postaviti mogućnost da se nakon klika korisnik prebaci na novu aktivnost, odnosno stranicu. To se postiže na način da se postavi *listener* na listu koji se aktivira kada korisnik klikne na određenu rezervaciju. Iz adaptera se dohvaća odabrana rezervacija te se stvara novi *intent*, odnosno namjera koja pokreće novu aktivnost. U novu aktivnost moguće je proslijediti odabrane podatke od strane korisnika pomoću `putExtra` mogućnosti u kojoj se navodi identifikator proslijeđenih podataka te sadržaj koji je u ovom slučaju spremljen u instancu klase „Rezervacija“. Podaci se koriste za dohvaćanje rezervacije koja se uređuje na novoj stranici. Preko namjera se na brz i jednostavan način prebacuju aktivnosti i prosljeđuju se podaci.

Prošle rezervacije su vrlo slične popisu rezervacije korisnika. Glavna je razlika u tome što se prilikom sortiranja koristi obrnuti pristup pa se uzimaju u obzir rezervacije čiji je datum manji od trenutnog. Razlika je i u tome što je status tih rezervacija automatski postavljen na „prošlo“ te je simbol statusa prikazan kao istek vremena. Osim statusa, razlika je u tome što nije moguće odabrati, urediti i izbrisati te rezervacije što znači da je njihova implementacija čak i jednostavnija od nadolazećih rezervacija pošto sadrže jednostavnije mogućnosti pri samom prikazu.

Glavni izbornik je konstrukciji vrlo sličan popisu rezervacija korisnika, no sadrži samo top 3 nadolazeće rezervacije uz dva gumba za preusmjeravanje na rezerviranje i moje rezervacije. Gumbi sadrže ranije definirani *listener* koji se poziva prilikom klika na gumb. Klikom na gumb ponovno se koriste namjere koje preusmjeravaju korisnika

na novu aktivnost, odnosno stranicu. Popis sa top 3 rezervacije prati istu strukturu tablice kao i rezervacije korisnika, no prilikom dohvaćanja rezervacija potrebno ih je na pravilan način sortirati. Logika sortiranja uzima u obzir da se kroz petlju provjerava je li datum rezervacije u budućnosti, odnosno je li vrijeme rezervacije veće od trenutnog vremena. Ukoliko je to istina, provodi se usporedba način da prilikom prve provjere datum postavlja kao prvi za prikaz. Slijedeći datum u budućnosti se nakon toga uspoređuje sa prvim datumom radi pozicioniranja, te ukoliko je novi datum manji od trenutno pozicioniranog datuma, on preuzima prvu poziciju dok se datum koji je bio prvi pomiče na drugu poziciju. Isti proces se ponavlja i za treću poziciju te se pomoću petlje rotiraju svi datumi budućih rezervacija kako bi se dobile top 3 nadolazeće rezervacije.

6.3. Izrada nove rezervacije

Jedna od glavnih funkcionalnosti mobilne aplikacije jest mogućnost izrade rezervacije. Sam izgled stranice se uvijek bazira na slaganju komponenti te se sastoji od kalendara, dva *spinnera* da odabir termina i razloga te gumbom za rezerviranje. Uz isti pristup povezivanja komponenti sa Java datotekom te instancama baze podataka, primijenjena su pravila preko kojih proces rezervacije funkcionira. Prvi korak je dohvaćanje fakulteta trenutno prijavljenog korisnika kako bi se prikazali dostupni termini njegove sastavnice. Taj se proces provodi postavljanjem reference na bazu podataka te petljom kojom se dohvaća trenutni korisnik kako bi se pristupilo podatku o njegovom fakultetu. Taj se podatak sprema u instancu klase „Rezervacija“ radi spremanja na bazu podataka prilikom rezervacije te u varijablu koja se koristi za dohvaćanje dostupnih termina. Prije dohvaćanja termina, potrebno je odrediti koji je dan u tjednu odabran u kalendaru. Komponenta kalendara dozvoljava izradu instance pomoću koje se može dohvatiti dan u tjednu. Pomoću dana u tjednu koji je prikazan u obliku broja, poziva se funkcija koja postavlja termine u *spinner* ovisno o tome koji je dan u tjednu. Ista funkcija sakriva gumb i ostale opcije te prikazuje tekst „Zatvoreno“ ukoliko korisnik odabere subotu ili nedjelju.

Slijedeća funkcija postavljanjem reference i petlje dohvaća zauzete termine. Prva usporedba provjerava je li odabrani datum s kalendara jednak datumu rezervacije te je li fakultet rezervacije jednak fakultetu prijavljenog korisnika. Sve rezervacije koje

odgovaraju navedenim uvjetima spremaju se u listu. Pošto sada postoji popis zauzetih termina, potrebno je prikazati dostupne termine, odnosno suprotne podatke od postojećih. To se postiže na način da se naprave liste koje se sadrže sve termine, zauzete termine i jedna prazna lista. Prazna lista se puni kao razlika između svih termina za odabrani datum i termina koji su zauzeti.

Potrebno je napraviti listu razloga te ih povezati sa adapterom. Razlog i termin isto tako imaju svoje `setOnItemSelectedListener` koji spremaju odabranu opciju u objekt klase „Rezervacija“. Kalendar je isto tako potrebno optimizirati na način da se postave minimalni i maksimalni datum. Minimalni datum rezervacije je sutrašnji, dok najkasniji datum rezervacije je 14 dana unaprijed. Kalendar isto ima svoj vlastiti *listener* koji se poziva prilikom odabira datuma. Sprema se trenutni odabir u objekt klase „Rezervacija“ te se poziva funkcija koja ažurira dostupne termine za trenutno odabrani datum.

Klikom na gumb „Spremi“ poziva se odgovarajući *listener* za klik. Inicijalizira se *Boolean* koji ima ulogu zastavice ukoliko korisnik već ima rezervaciju na trenutno odabrani datum. Postavljaju se podaci na ranije navedeni objekt klase te se odabrani datum i termin konvertiraju numerički format radi usporedbe u bazi podataka. Referencom na bazu podataka i petljom provodi se provjera ima li korisnik rezervaciju na odabrani datum te se koristi zastavicu kao indikator. Ukoliko korisnik ima rezervaciju, ispisuje se greška. Greška se isto tako ispisuje ukoliko je odabrani razlog „Vlastiti razlog:“ a prostor za upisivanje razloga nije popunjen. Ukoliko su svi navedeni uvjeti zadovoljeni, popunjena instanca klase „Rezervacija“ se sprema na bazu podataka te se korisniku objavljuje poruka kako je termin uspješno rezerviran. Nova namjera tada usmjerava korisnika na glavni izbornik gdje je rezervacija odmah i vidljiva ukoliko se nalazi u top 3 nadolazeće rezervacije.

6.4. Uređivanje rezervacije i opcije

Opcija za uređivanje rezervacija je vrlo povezana unutar aplikacije. Namjere sa glavnog izbornika i popisa rezervacija vode upravo na tu aktivnost prosljeđujući joj podatke. Ti podaci služe kako bi se korisniku na sučelju prikazale specifičnosti trenutno odabrane rezervacije kao što su datum, termin, razlog te e-mail zaposlenika i komentar

ukoliko je rezervacije odobrena ili otkazana. Logika ove aktivnosti je vrlo slična izradi rezervacije po funkcijama za ažuriranje dostupnih termina. Prilikom uređivanja rezervacije bitno je dohvatiti i prikazati proslijeđene podatke iz drugih aktivnosti. Sve se odabrane rezervacije za uređivanje odabiru iz tablice koja sadrži podatke o istoj. Za dohvaćanje podataka potrebno je znati dodijeljeno ime koje podaci nose iz jedne aktivnosti u drugu, a to je u ovom slučaju ime „Uredi“. Razlog zašto se koriste imena je kako bi se mogli razlikovati podaci koji se prosljeđuju pošto se može prosljeđivati više podataka u različitim situacijama. Naravno podaci se prilikom dohvaćanja spremanju u novi objekt klase „Rezervacija“ pošto ti objekti po strukturi podataka imaju istu klasu. Dohvaćeni se podaci zatim spremaju u varijable. Zatim se ti podaci koriste za postavljanje datuma i *spinnera* tako da njihove informacije odgovaraju informacijama odabrane rezervacije. E-mail adresa i komentar obično nisu vidljivi ukoliko ne postoje tako da se prikazuju samo ako je status rezervacije promijenjen.

Funkcije prilikom ažuriranja rezervacije su iste, no ukoliko se rezervacija koja je odobrena ili otkazana ažurira, status se ponovno postavlja „Na čekanju“ radi reevaluacije iste. Brisanjem rezervacije korisnik se usmjerava na glavni izbornik uz poruku kako je rezervacija uspješno obrisana što je odmah i vidljivo na popisu rezervacija. Proces brisanja se isto provodi preko petlje gdje se uspoređuje ID rezervacije te ukoliko ID odgovara, rezervacija se briše pomoću ugrađene Firebase funkcije.

Opcije omogućuju korisniku da ažurira svoju lozinku ili fakultet. Prvo se dohvaća reference na bazu podataka na pozicije gdje se nalaze podaci o studentima te se izrađuje objekt klase „Student“. Postavlja se *spinner* sa dostupnim sastavnicama fakulteta te pomoću petlje na bazi podataka dohvaća trenutni fakultet korisnika kako bi se spinner postavio na fakultet koji trenutno pohađa. Za *spinner* postoji i *listener* koji sprema podatke o odabranom fakultetu. Pritiskom na gumb fakultet studenta se ažurira preko reference na bazi podataka te se korisnik preusmjerava na glavni izbornik.

Ažuriranje lozinke sadrži polja za popunjavanje koje je potrebno popuniti pravilnim podacima prema uputama. Klikom na gumb za ažuriranje lozinke prvo se pomoću petlje dohvaća korisnikova trenutna lozinka sa baze podataka radi usporedbe. Prvo se provjerava jesu li zadana polja popunjena te odgovara li trenutna lozinka unesenoj lozinki od strane korisnika. Ukoliko neki od navedenih uvjeta nisu

zadovoljeni, korisniku se ispisuje greška. Zadovoljeni uvjeti omogućuju ažuriranje lozinke korisnika temeljem ugrađene Firebase funkcije te se korisnik preusmjerava na glavni izbornik s obavijesti o uspješno ažuriranoj lozinki.

7. Korisničke upute

Korisničke upute služe kako bi se detaljno objasnile funkcionalnosti aplikacije na korisničkoj razini. Korisnik ne treba znati kakve se funkcije pozivaju za izvršavanje određenih radnji niti proces implementacije istih. Bitno je korisniku objasniti krajnju aplikaciju te kako bi trebala njegova interakcija sa korisničkim sučeljem trebala biti. Upute služe za potpuno nove korisnike, ali i za postojeće korisnike ukoliko im neki dio aplikacije nije jasan. U praksi, korisničke upute je bitno ažurirati s obzirom na ažuriranje funkcionalnosti aplikacije.

7.1. Korisničke upute web aplikacije

Prilikom otvaranja aplikacije, prikazuje se početna stranica koja se sastoji od logotipa aplikacije u sredini, te alatne trake na vrhu stranice. Alatna traka na lijevoj strani sadrži gumb za otvaranje izbornika sa strane, no on je namijenjen ukoliko se aplikacija koristi na web pregledniku mobilnog uređaja. Sa desne strane na alatnoj traci se isto tako nalaze opcije za prijavu postojećeg korisnika ili registraciju novog korisnika.

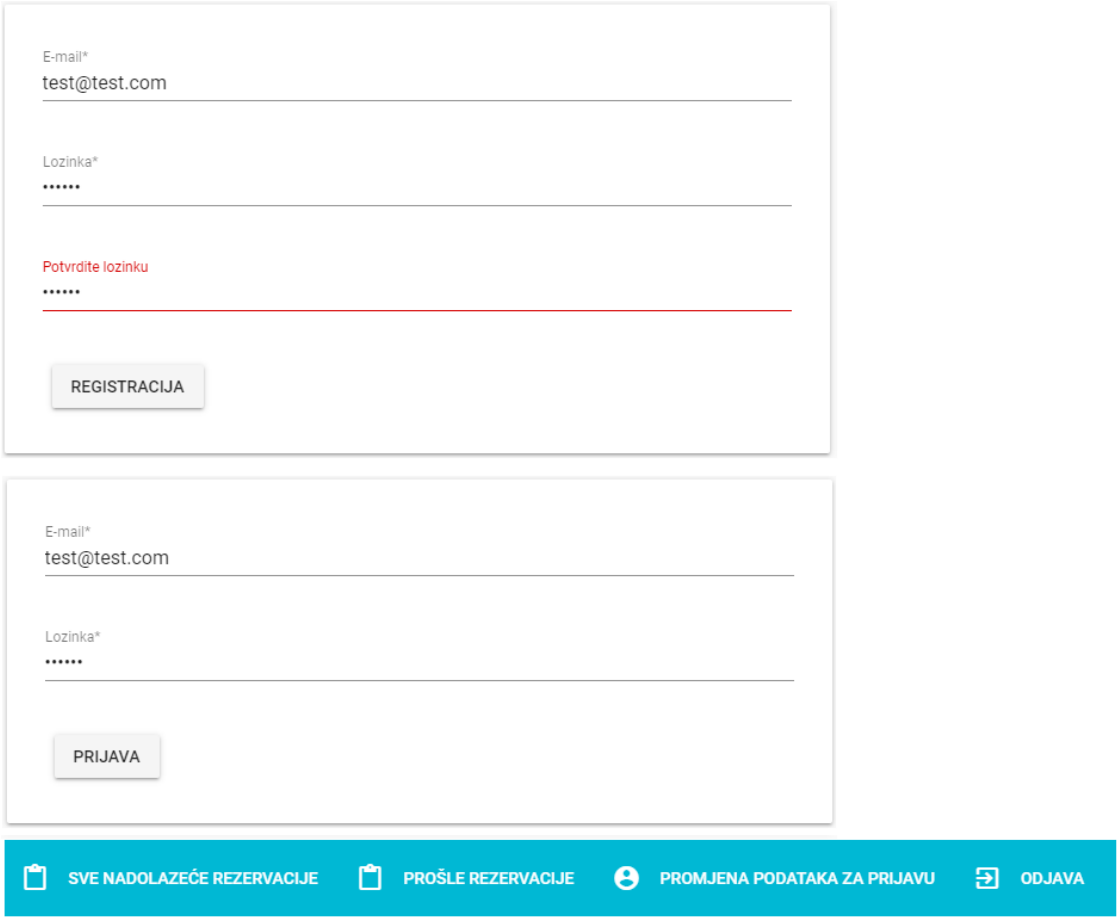
Slika 17. Web aplikacija – početni zaslon prilikom otvaranja aplikacije

Izvor: autor

7.1.1. Registracija i prijava

Odabirom registracije korisniku se prikazuje forma za popunjavanje svojih podataka. E-mail adresa se postavlja jednom, a lozinka dva puta. Ukoliko aplikacija javlja grešku prilikom registracije, postoji nekoliko razloga. Ukoliko se pokuša registrirati sa e-mail adresom već postojećeg korisnika, to nije dozvoljeno. Lozinke pri registraciji moraju biti jednake te moraju sadržavati barem 6 znakova što je jedini uvjet pri odabiru lozinke. E-mail adresa mora biti u e-mail formatu. Poruke grešaka su obično crvene boje i lako su uočljive. Klikom na gumb za registraciju, potrebno je pričekati nekoliko trenutaka te se korisnika usmjerava na početnu stranicu sa više mogućnosti na alatnoj traci. Ukoliko korisnik već ima račun na aplikaciji, umjesto registracije odabire se prijava. Kod prijave je potrebno upisati e-mail adresu i lozinku, a ukoliko se pojavi greška, e-mail adresa ili lozinka nisu točne. Uspješnom prijavom korisniku se prikazuju dodatne opcije na alatnoj traci. Isto tako, moguće je osvježiti stranicu bez da se izgubi trenutna sesija tako da repetitivna prijava na aplikaciju nije potrebna.

Slika 18. Web aplikacija – registracija, prijava, i mogućnosti alatne trake



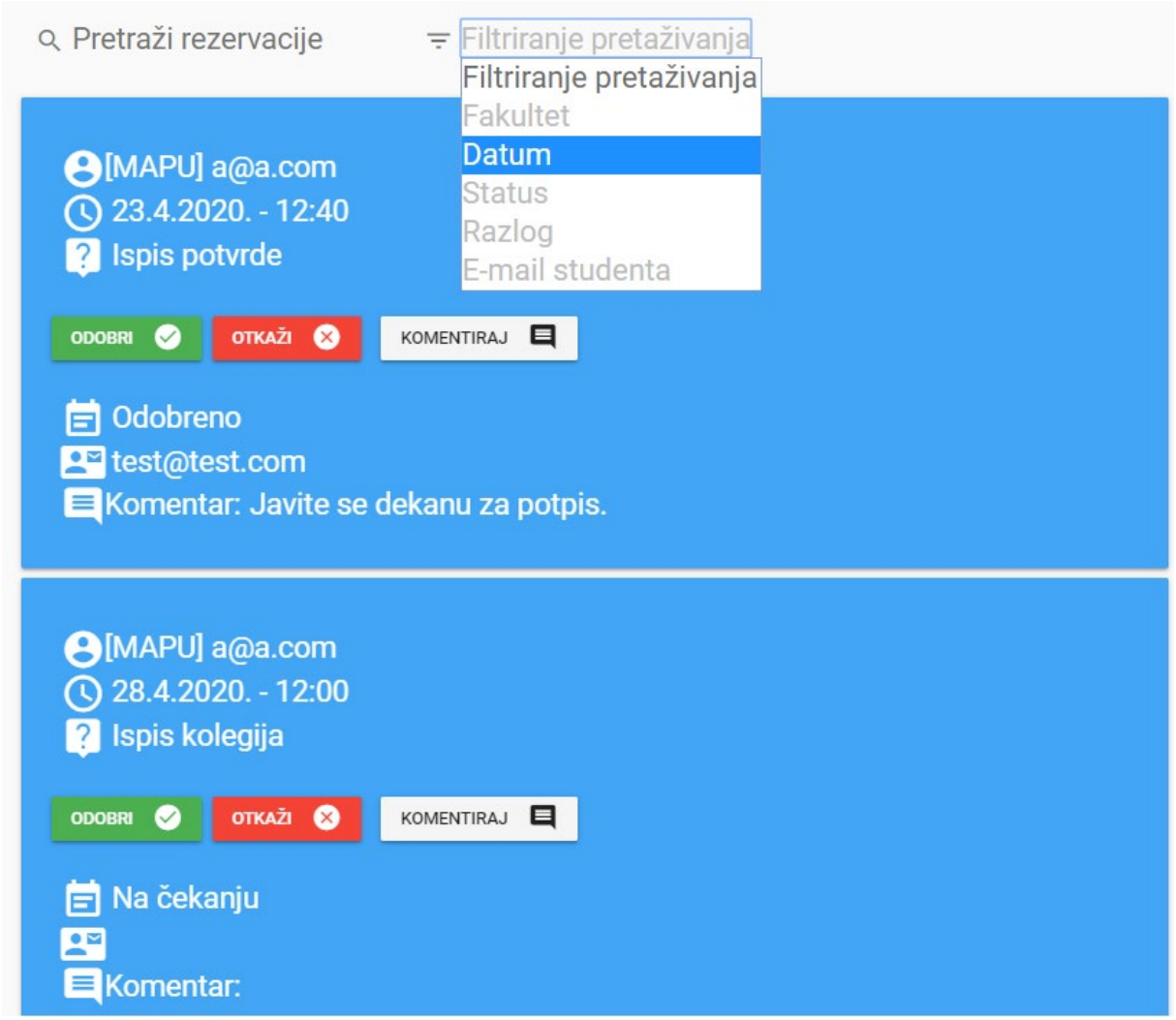
Izvor: autor

7.1.2 Nadolazeće rezervacije i prošle rezervacije

Odabirom svih nadolazećih rezervacija, korisniku se prikazuje popis rezervacija sortiranih prema vremenu. Iznad popisa rezervacija nalazi se prostor koji služi za upisivanje pojmova za pretraživanje, pored prostora za pretraživanje nalazi se filter. Filter omogućuje korisniku da definira što želi filtrirati, odnosno koja vrsta podatka se koristi za pretraživanje. Moguće je pretraživati prema fakultetu, datumu, statusu, razlogu i e-mail adresi studenta. Rezervacije se sastoje od podataka koji služe sa čitanje i od gumbi koji služe za kontroliranje rezervacija. Redoslijedno, u vitičastim zagradama se prikazuje sastavnica fakulteta, zatim e-mail adresa studenta koja može poslužiti za kontakt, datum i vrijeme rezervacije te razlog. Na temelju tih informacija slijede gumbi za odobravanje, otkazivanje ili komentiranje rezervacije. Prostor ispod gumbi je direktno podložan promjenama korisnika web aplikacije, odnosno zaposlenika studentske referade. Redoslijedno, ispod gumbi prikazuju se status, e-mail adresa zaposlenika koji je obradio rezervaciju te komentar. Klikom na odobravanje ili otkazivanje rezervacije, status se mijenja na „Odobreno“ ili „Otkazano“ te se e-mail zaposlenika postavlja na e-mail adresu trenutno prijavljenog korisnika.

Nova rezervacija ima status „Na čekanju“ te nema e-mail adresu zaposlenika prikazanu ispod gumbi što znači da nitko nije odobrio, otkazao ili komentirao rezervaciju. Bilo kakva interakcija sa gumbima rezervacije bilježi e-mail adresu trenutno prijavljenog korisnika, a tu e-mail adresu vidi student na svojoj rezervaciji na mobilnoj aplikaciji. Na taj način student zna koji zaposlenik je preuzeo njegovu rezervaciju te može kontaktirati zaposlenika preko e-mail adrese ukoliko je to potrebno. Komentari kao takvi nisu obvezni, već služe ukoliko zaposlenik želi komentirati rezervaciju kako bi se dalo dodatno objašnjenje vezano za rezervaciju. Na taj način se ubrzava kontakt između zaposlenika referade i studenta pošto student može vidjeti komentar i status vezan za svoju rezervaciju. Studenti mogu napraviti samo jednu rezervaciju dnevno te se rezervacije istih sastavnica fakulteta ne mogu poklapati. Isto tako, korisnik može promijeniti status rezervacije ili komentara rezervacije koja je već obrađena, što znači da se bilježi e-mail adresa korisnika koji je posljednji napravio promjene. Ta funkcionalnost služi ukoliko zaposlenik želi preuzeti studenta čiju je rezervaciju odobrio drugi zaposlenik. Komentiranje rezervacije otvara skočni prozor sa prostorom za upisivanje komentara koji se može spremiti ili zatvoriti bez spremanja promjena klikom na previđene gumbe „Spremi“ ili „Odustani“.

Slika 19. Web aplikacija – nadolazeće rezervacije i filter



Izvor: autor

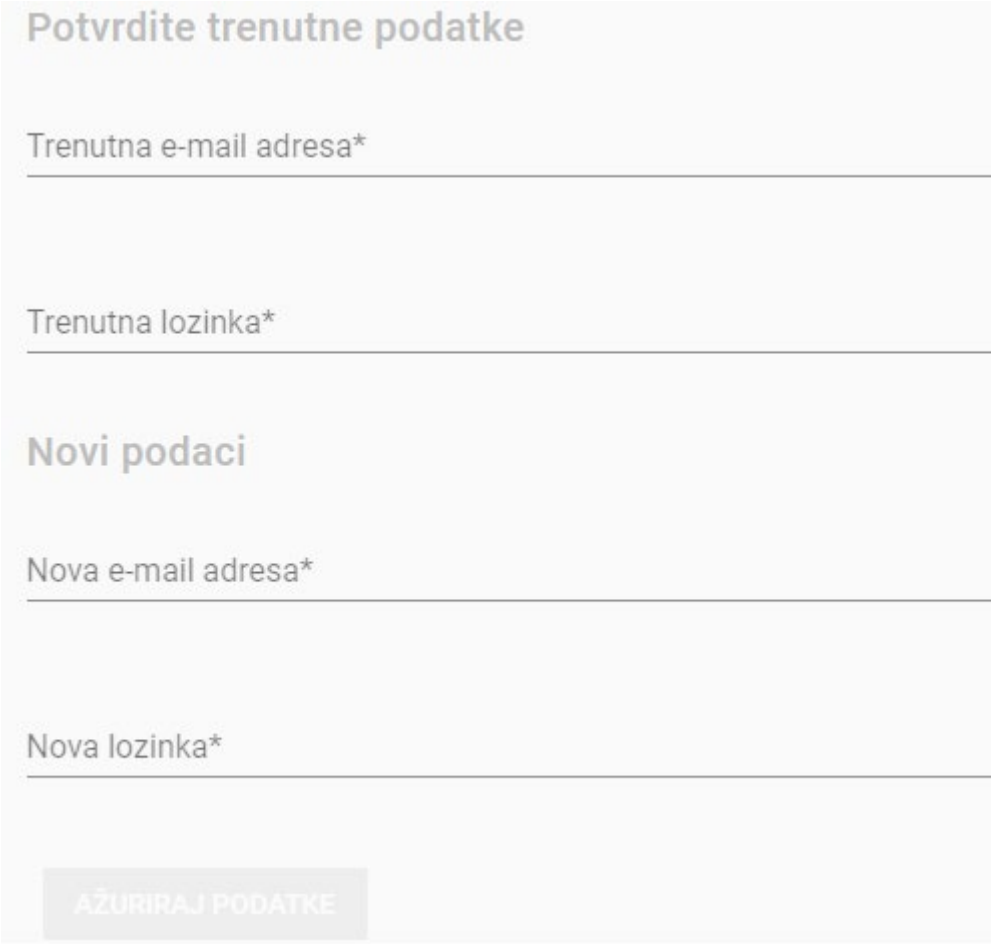
Klikom na prošle rezervacije preko alatne trake, korisniku se prikazuju sve rezervacije koje su vremenski gledano prošle. Prikazane su u istom obliku kao i nadolazeće rezervacije sa prostorom za pretraživanje i filterom. Jedina je razlika što korisnik ne može imati interakciju sa prošlim rezervacijama pošto su za njih uklonjeni gumbi. Prošle rezervacije služe isključivo kao arhiva za pregled.

7.1.3. Promjena podataka za prijavu i odjava

Klikom na promjenu podataka za prijavu korisniku se prikazuje forma koja služi za ažuriranje podataka za prijavu, a sastoji se od unosa trenutnih i novih podataka za

prijavu. Prilikom ažuriranja podataka za prijavu, nije potrebno promijeniti i e-mail adresu i lozinku, već ukoliko se samo želi promijeniti samo e-mail adresa, kod unosa novih podataka je potrebno napisati istu lozinku. Ista logika se primjenjuje ukoliko korisnik želi ažurirati samo lozinku. Greške se mogu pojaviti ukoliko trenutni podaci za prijavu nisu točni te ukoliko nova lozinka ima manje od 6 znakova. Naravno, pravilo formata e-mail adrese isto vrijedi te može izbaciti grešku ukoliko nije zadovoljeno. Gumb za spremanje podataka će postati dostupan kada su svi podaci u formi uneseni. Nakon završetka rada, korisnik klikom gumba za odjavu se odjavljuje sa aplikacije te mu se ponovno prikazuju opcije za registraciju i prijavu na alatnoj traci. Pošto je sesija korisnika zaključena odjavom, nije moguće klikom na gumb „Back“ u web pregledniku ponovno se prijaviti, već je potrebno ponovno unijeti podatke za prijavu.

Slika 20. Web aplikacija – promjena podataka za prijavu



Izvor: autor

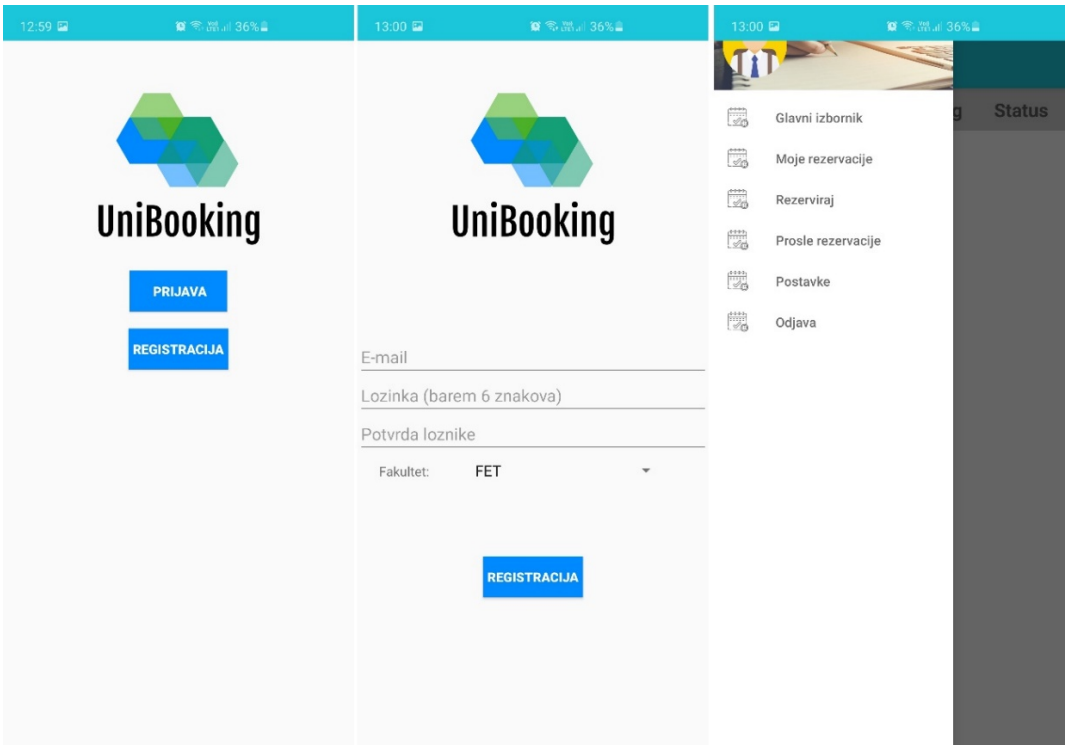
7.2. Korisničke upute mobilne aplikacije

Mobilna aplikacija je namijenjena studentima kako bi izrađivali, uređivali, pregledavali i brisali svoje rezervacije. Aplikacija isto tako sadrži opcije za ažuriranje lozinke i fakulteta. Potrebno je razumjeti nekoliko pravila koje aplikacija prati te je bazirana na tome da studenti obave posjet referadi na što brži način uz minimalno vrijeme čekanja.

7.2.1. Registracija, prijava, izbornik

Prilikom prvog pokretanja aplikacije, korisniku se prikazuje logotip aplikacije uz dva gumba za registraciju i prijavu. Novi korisnici odabiru registraciju te im se prikazuje forma koju moraju popuniti. Ukoliko se pri registraciji pojavljuje greška, podaci nisu pravilno uneseni ili korisnik sa upisanom e-mail adresom već postoji. Istom logikom funkcionira i prijava, upisom e-mail adrese i lozinke. Ukoliko je sve pravilno uneseno, aplikacija prebacuje korisnika na glavni izbornik. Pritiskom na gumb u gornjem lijevom kutu aplikacije, prikazuje se izbornik sa strane. Aplikacija pamti sesiju što znači da ako se aplikacija zatvori i nakon dužeg vremena otvori, korisnik će ostati prijavljen.

Slika 21. Mobilna aplikacija – početni zaslon, registracija, izbornik



Izvor: autor

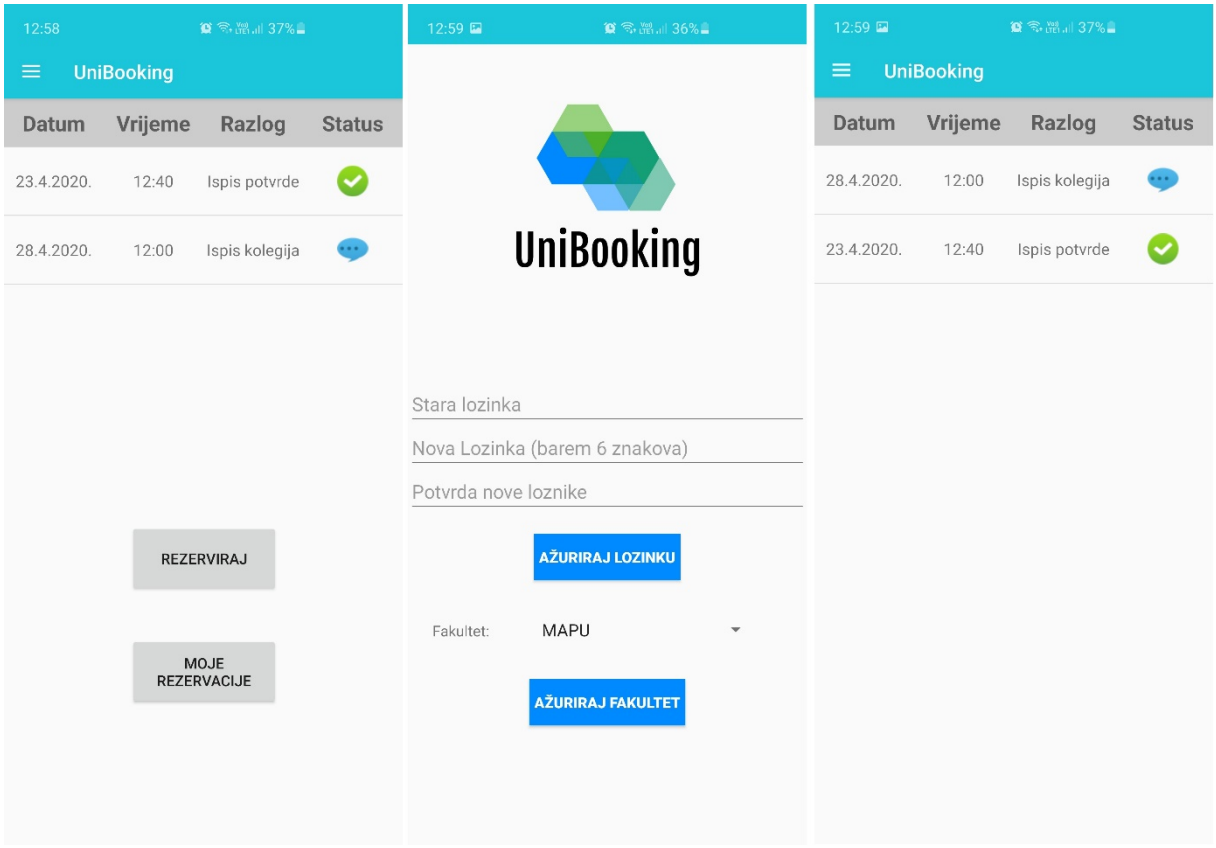
7.2.2. Glavni izbornik, opcije i moje rezervacije

Nakon registracije ili prijave, za navigaciju se koristi ranije spomenuti izbornik koji se nalazi u gornjem lijevom kutu aplikacije. Glavni izbornik sadrži top 3 nadolazeće rezervacije koje su sortirane prema vremenu. Prikazuje se datum, vrijeme, razlog posjeta te status. Status se prikazuje simbolima koje je lako razumjeti, pa tako zeleni simbol predstavlja odobrenu rezervaciju, crveni simbol status predstavlja otkazanu rezervaciju, a plavi simbol predstavlja rezervaciju na čekanju. Korisnici mobilne aplikacije ne mogu mijenjati status, to je funkcionalnost koju koriste zaposlenici referade preko svoje web aplikacije. Status služi kao prikaz povratne informacije zaposlenika referade na rezervaciju. Uz kratki popis, glavni izbornik sadrži i dva gumba koji služe kao prečaci na dvije najčešće korištene funkcionalnosti, a to su rezerviranje te popis vlastitih rezervacija korisnika. Cilj glavnog izbornika je pružati i prikazati najbitnije informacije i najkorištenije funkcionalnosti korisniku. Rezervacije sa popisa se mogu odabrati po želji ukoliko ih se želi pregledati, urediti ili izbrisati.

Odabirom opcije „Moje rezervacije“ iz izbornika prikazuje se popis svih nadolazećih rezervacija sortiranih prema vremenu, ali bez gumbi na korisničkom sučelju. Isto kao i sa glavnog izbornika, moguće je odabrati bilo koju rezervaciju sa popisa. Naravno, rezervacija svaka se nova rezervacija odmah pohranjuje te je dostupna čim se prikaže obavijest kako je nova rezervacija spremna. Isto pravilo vrijedi i za uređivanje i brisanje rezervacija što znači da bi svaka promjena od strane korisnika bila odmah vidljiva.

Prošle rezervacije prikazuju sve prošle rezervacije isto tako sortirane prema vremenu. Status tih rezervacija je simbol koji označava da su prošle te se ne mogu odabrati za uređivanje. Opcije omogućuju korisnicima promjenu lozinke za prijavu unosom traženih podataka. Ukoliko podaci nisu pravilno uneseni, aplikacije će ispisati grešku. Stara lozinka mora odgovarati lozinki spremljenoj na bazi podataka te nova lozinka mora imati najmanje 6 znakova. Uz promjenu lozinke, korisnik može promijeniti i svoj fakultet. Razlog tome je što studenti mogu prebacivati studij pa je bitno da su njihove rezervacije usmjerene prema odgovarajućem zaposleniku studentske referade s obzirom na sastavnicu fakulteta studenta. Fakultet je bitan zato što odabirom svoje sastavnice student može imati pregled svih dostupnih termina vezanih za svoju sastavnicu.

Slika 22. Mobilna aplikacija – glavni izbornik, opcije i moje rezervacije



Izvor: autor

7.2.3. Rezervacija i uređivanje rezervacije

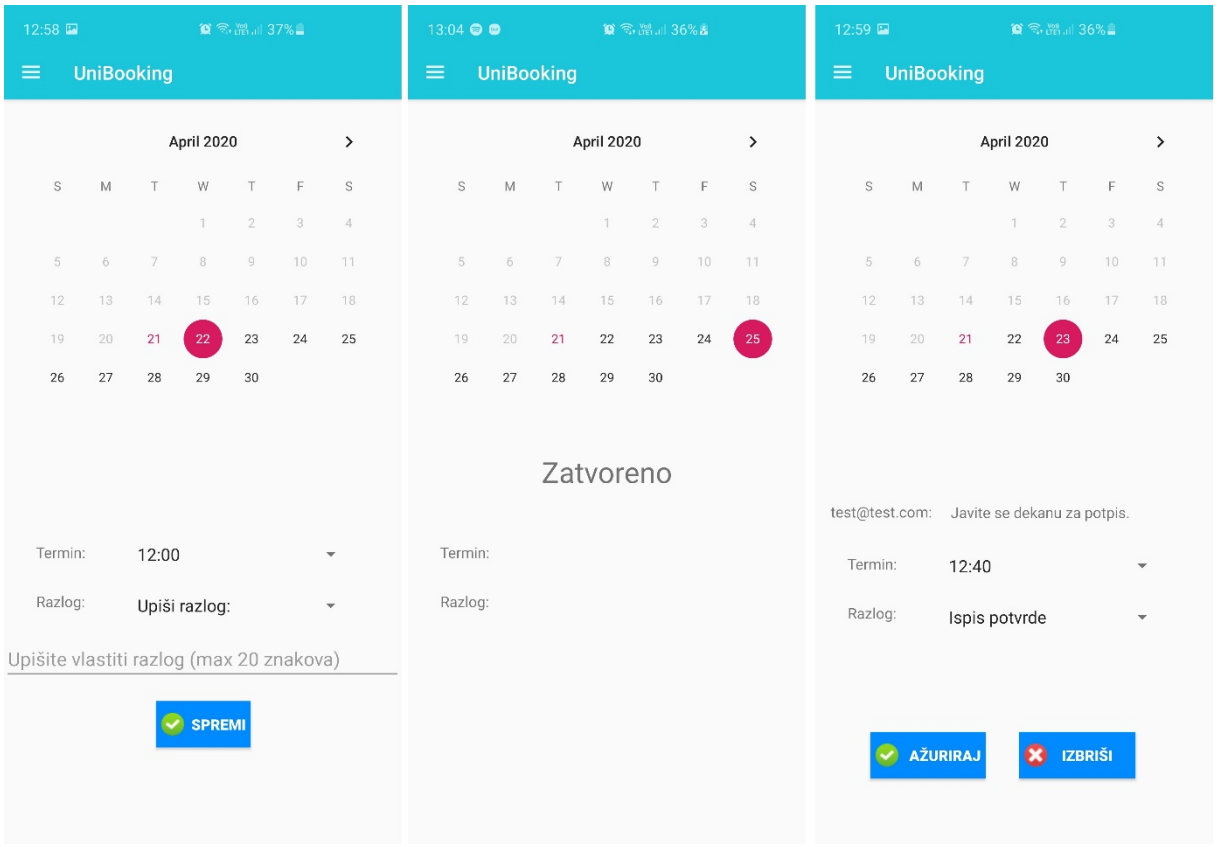
Nakon odabira opcije za rezerviranje, korisniku se prikazuje kalendar sa dostupnim datumima za rezervaciju. Korisnik može rezervirati termin do dva tjedna unaprijed uz određene restrikcije. Rezervacije se mogu samo rezervirati prema dostupnim terminima te rezervacije nije moguće napraviti subotom i nedjeljom pošto je studentska referada zatvorena. U tom slučaju korisniku se ispisuje obavijest kako je referada zatvorena. Dostupni termini funkcioniraju po logici da postoji jedan zaposlenik za jednu sastavnicu fakulteta. Korisniku se prikazuju dostupni termini njegove sastavnice, odnosno fakulteta koji je odabrao pri registraciji te se taj fakultet može promijeniti u opcijama. Isto tako, pravilo je da se ne može izraditi jedna rezervacija dnevno. Ukoliko se pokuša izraditi više od jedne rezervacije dnevno, aplikacija će ispisati grešku.

Nakon odabranog dostupnog termina, korisnik odabire razlog posjeta sa popisa najčešćih razloga. Ukoliko student ima neki drugi razlog koji želi navesti, isto može

učiniti odabirom vlastitog razloga. Polje za vlastiti razlog u tom slučaju mora biti popunjeno, a ukoliko nije popunjeno ispisuje se greška. Ukoliko su svi uvjeti zadovoljeni, rezervacija će biti uspješno spremljena te će korisnik biti preusmjeren na glavni izbornik.

Odabirom rezervacije sa popisa ili glavnog izbornika, studentu se prikazuju informacije odabrane rezervacije koje se mogu ažurirati. Uz prikazane podatke koji se mogu ažurirati prema istim pravilima kao i kod obične rezervacije, prikazuju se i povratne informacije zaposlenika referade ukoliko je rezervacija odobrena ili otkazana. Prikazuje se e-mail adresa zaposlenika studentske referade koji je promijenio status rezervacije te komentar ukoliko postoji. Ako komentar ne postoji, prikazuje se samo e-mail adresa zaposlenika. Ukoliko student ažurira rezervaciju koja je odobrena ili otkazana, njezin status postaje „Na čekanju“ pošto zaposlenik mora ponovno pogledati rezervaciju jer su njeni podaci izmijenjeni. Uz ažuriranje, korisnik može i izbrisati rezervaciju. Odabirom na gumb za brisanje, rezervacija se briše bez obzira na svoj status.

Slika 23. Mobilna aplikacija – rezervacija, zatvorena referada, uređivanje rezervacije



Izvor: autor

8. Zaključak

Aplikacije su postale veliki dio svakodnevnog privatnog i profesionalnog života. Od jednostavnih aplikacija za čitanje i slanje poruka pa sve do aplikacija koje koriste argumentiranu realnost, njihov broj, ali i mogućnosti eksponencijalnu rastu čak i na dnevnoj razini. Okruženja za razvoj web aplikacija danas su vrlo raširena te je njihova ponuda mogućnosti vrlo bogata. Predstavljaju moderan način izrada web aplikacija uz konstantna ažuriranja i nadogradnje. Mogućnosti okruženja se konstantno proširuju te ih je moguće i nadopuniti raznim paketima. Idealno okruženje za razvoj web aplikacija ne postoji, već se izbor bazira na mnogo čimbenika koji utječu na konačnu odluku. Izbor kao takav zapravo najviše ovisi o trenutnoj situaciji te preferencijama pojedinca ili kompanije sa aspekta kompatibilnosti. Svako okruženje ima programski jezik kojem je namijenjeno za razvoj te se dijele na *frontend* i *backend* okruženja.

Frontend okruženja se odnose na razvoj korisničkog sučelja, njihovih funkcionalnosti i interakciju sa krajnjim korisnicima. Ta okruženja koriste programske jezike za web razvoj kao što su HTML, CSS i JavaScript. *Backend* okruženja odnose se na komunikaciju sa poslužiteljem što uključuje učitavanje, slanje, izmjenu i brisanje podataka te nisu namijenjena za prikaz krajnjim korisnicima. Okruženja koriste jezike kao što su Python i Ruby te su zadužene za komunikaciju sa poslužiteljem i sa sigurnosnog aspekta.

Vue.js predstavlja dobru opciju ukoliko je cilj korištenje jednostavnog i fleksibilnog okruženja čije se mogućnosti mogu proširiti paketima. Baza podataka Firebase je besplatna opcija koja omogućuje spremanje i upravljanje podacima preko web aplikacija i mobilnih aplikacija koje ostvaruju međusobnu interakciju. Mobilne aplikacije za operacijski sustav Android se programiraju u Java ili Kotlin programskom jeziku u programu Android Studio koji redovito dobiva ažuriranja na svoje mogućnosti. Pomoću navedenih okruženja i programa moguće je napraviti interaktivne i vizualno lijepe web i mobilne aplikacije za široku upotrebu.

Izvorni kod za obje aplikacije je postavljen na GitHub repozitorije te je dostupan na slijedećim poveznicama: https://github.com/imatak/UniBooking_Mobile za mobilnu aplikaciju te https://github.com/imatak/UniBooking_Web za web aplikaciju.

9. Literatura

KNJIGE

1) Banks, A. i Porcello, E. (2017) Learning React. Sebastopol: O'Reilly Media, Inc.
2) Filipova, O. (2016) Learning Vue.js 2. Birmingham: Packt Publishing Ltd.
3) Freeman, A. (2018) Pro Angular 6, 3rd Edition. London: Apress Media LLC
4) Young, A. i Harter, M. (2017) Node.js in practice. Shelter Island: Manning Publications Co.

INTERNET IZVORI

1) Common Gateway Interface (1997). URL: https://web.archive.org/web/20090409213905/http://hoohoo.ncsa.uiuc.edu/cgi/intro.html [10.4.2020.]
2) Django documentation (2019). URL: https://docs.djangoproject.com/en/3.0/ [12.4.2020.]
3) Laravel installation (2020). URL: https://laravel.com/docs/7.x [12.4.2020.]
4) Šoda, A. (2019) Najgori trenuci u životu svakog studenta. URL: https://www.srednja.hr/faks/najgori-trenuci-zivotu-svakog-studenta-posjet-referadi [14.4.2020.]

10. Popis slika

Slika 1. Red ispred referade 9
Slika 2. SWOT analiza trenutnog stanja 10
Slika 3. Use case dijagram - student 13
Slika 4. Use case dijagram – zaposlenik referade 15
Slika 5. Sekvencijski dijagram – student – rezervacija 16
Slika 6. Sekvencijski dijagram – student – opcije 17
Slika 7. Sekvencijski dijagram – zaposlenik – rezervacije 18
Slika 8. Sekvencijski dijagram – zaposlenik – opcije 19
Slika 9. Web aplikacija – početni zaslon 21
Slika 10. Web aplikacija – prijava 21
Slika 11. Web aplikacija – nadolazeće rezervacije 22
Slika 12. Web aplikacija – opcije 22
Slika 13. Mobilna aplikacija – registracija, početni zaslon i izbornik 23
Slika 14. Mobilna aplikacija – rezervacija, uređivanje rezervacije i opcije 24
Slika 15. Protok podataka iz baze podataka do korisničkog sučelja 27
Slika 16. XML datoteka – slaganje i definiranje komponenti 33
Slika 17. Web aplikacija – početni zaslon prilikom otvaranja aplikacije 40
Slika 18. Web aplikacija – registracija, prijava, i mogućnosti alatne trake 41
Slika 19. Web aplikacija – nadolazeće rezervacije i filter 43
Slika 20. Web aplikacija – promjena podataka za prijavu 44
Slika 21. Mobilna aplikacija – početni zaslon, registracija, izbornik 45
Slika 22. Mobilna aplikacija – glavni izbornik, opcije i moje rezervacije 47
Slika 23. Mobilna aplikacija – rezervacija, zatvorena referada, uređivanje rezervacije 48

11. Sažetak

Aplikacije su postale neizbježni dio svakodnevnog života. Zato je bitno upoznati se sa okruženjima za razvoj web aplikacija koje se dijele na *frontend* i *backend*. Teorijski dio rada temelji se na opisu nekoliko okruženja te kako odabrati idealno okruženje temeljem potreba i osobnih preferencija. Praktični dio rada obuhvaća sveukupni proces izrade web i mobilne aplikacije za rezervaciju u studentskim referadama. Cilj aplikacije je ubrzati i automatizirati proces posjeta referadi na moderan način preko web aplikacije za zaposlenike i mobilne aplikacije za studente. Praktični dio se sastoji od izrade prototipa korisničkog sučelja, opisivanja funkcionalnosti i njihov dijagramski prikaz. Slijedi implementacija važnih funkcionalnosti aplikacija te korisničke upute.

Ključne riječi: Web aplikacija, Mobilna aplikacija, Okruženje, Implementacija, Android Studio, Vue.js, Firebase

12. Abstract

Applications have become an unavoidable part of our daily lives. That is why there is significant importance when it comes to learning about web application development interfaces which can be divided into frontend and backend frameworks. The theoretical section is based on describing several frameworks to demonstrate the decision-making process on which framework to choose based on needs and personal preferences. The practical section includes the process of making web and mobile applications for reservations at student offices. The main goal is to speed up and automate the process of visiting the student offices by using a web application for employees and a mobile application for students. The practical section also includes making a UI prototype and functionality description with diagrams. The following chapters focus on the implementation of important functionalities of both applications and the user manual.

Keywords: Web application, Mobile application, Framework, Implementation, Android Studio, Vue.js, Firebase