Mobilna Android aplikacija i web aplikacija za organizaciju narudžbi u ugostiteljskim objektima skupine "Barovi"

Skočibušić, Pavo

VELEUČILIŠTE U VIROVITICI

Stručni prijediplomski studij Računarstva

PAVO SKOČIBUŠIĆ

MOBILNA ANDROID APLIKACIJA I WEB APLIKACIJA ZA ORGANIZACIJU NARUDŽBI U UGOSTITELJSKIM OBJEKTIMA SKUPINE „BAROVI“

ZAVRŠNI RAD

VIROVITICA, 2023.

VELEUČILIŠTE U VIROVITICI

Stručni prijediplomski studij Računarstva

MOBILNA ANDROID APLIKACIJA I WEB APLIKACIJA ZA ORGANIZACIJU NARUDŽBI U UGOSTITELJSKIM OBJEKTIMA SKUPINE „BAROVI“

ZAVRŠNI RAD

Predmet: Projektiranje informacijskih sustava

Mentor:

Marko Hajba, mag.math., pred.

Student:

Pavo Skočibušić

VIROVITICA, 2023.

Veleučilište u Virovitici

Stručni prijediplomski studij Računarstva - Smjer Programsko inženjerstvo

OBRAZAC 1b

ZADATAK ZAVRŠNOG RADA

Student/ica: **SKOČIBUŠIĆ PAVO** **JMBAG:** **0307017218**

Imenovani mentor: **Marko Hajba, mag. math., pred.**

Imenovani komentor: **-**

Naslov rada:

Mobilna Android aplikacija i web aplikacija za organizaciju narudžbi u ugostiteljskim objektima skupine „Barovi“

Puni tekst zadatka završnog rada:

Izraditi mobilnu Android aplikaciju pomoću programskog jezika Java te web aplikaciju pomoću Reacta koja služi za organizaciju narudžbi ugostiteljskog objekta iz skupine „Barovi“. Bazu podataka je potrebno napuniti testnim podacima, a potrebno je koristiti Firebase bazu podataka.

Korisnici aplikacije mogu biti korisnici (konobari zaposlenici), vlasnici objekata i administratori. Aplikacija treba imati različite funkcionalnosti, ovisno o tipu korisnika koji ju koristi:

- Administratori brinu o održavanju funkcionalnosti aplikacije, primaju prijedloge i rješavaju probleme koje mogu prijaviti ostali korisnici.
- Vlasnik objekta može prijaviti ugostiteljski objekt. Nadalje, unosi dostupne uređaje u objektu. Ima uvid u statistiku objekta.
- Korisnici (konobari) ugostiteljskog objekta mogu postavljati narudžbe, izvršavati ih te ažurirati ponudu objekta pomoću mobilne aplikacije. Web aplikacija omogućava prikaz svih narudžbi u objektu s detaljima narudžbe, ažuriranje njihovog statusa te pretraživanje narudžbi.

Osim programskog rješenja, u pisanom dijelu završnog rada opišite ukratko korištene tehnologije te detaljno opišite arhitekturu sustava i odabrane procese i/ili funkcije unutar same aplikacije. Prilikom opisivanja, osim neformalnih koristite i neke od formalnih metoda koje poznajete.

Datum uručenja zadatka studentu/ici: 31.07.2023.
Rok za predaju gotovog rada: 08.09.2023.

Mentor:

Marko Hajba, mag. math., pred.

M. Hajba

Dostaviti:
1. Studentu/ici
2. Povjerenstvu za završni rad - tajniku

MOBILNA ANDROID APLIKACIJA I WEB APLIKACIJA ZA ORGANIZACIJU NARUDŽBI U UGOSTITELJSKIM OBJEKTIMA SKUPINE „BAROVI"

Sažetak

Cilj ovog završnog rada bio je razviti mobilnu Android aplikaciju i web aplikaciju za potrebe ugostiteljskih objekata u kategoriji „Barovi". Mobilna aplikacija je namijenjena konobarima kako bi olakšala proces kreiranja i upravljanja narudžbama te organizacijom menija pića objekta. S druge strane, aplikacija omogućuje vlasnicima ugostiteljskih objekata cjelovit uvid u relevantne informacije kao i ažuriranje informacija objekta i telefonskih brojeva mobitela pomoću kojih se zaposlenici prijavljuju. Putem web aplikacije, konobari imaju mogućnost pregledavanja kreiranih narudžbi. Imaju opciju odbijanja ili postavljanja narudžbe u stanje spremnosti, pri čemu će oba slučaja rezultirati obavještavanjem konobara koji je inicijalno napravio narudžbu. Opisani par aplikacija omogućava brzu komunikaciju i koordinaciju među osobljem, što ispunjava glavnu svrhu mobilne i web aplikacije. U nastavku teksta opisane su tehnologije koje su korištene za razvoj aplikacija. Zatim će se objasniti povezanost aplikacija te pojedinačne funkcionalnosti prema različitim korisničkim ulogama. Također će biti objašnjeno programsko rješenje završnog rada, funkcionalnosti mobilne Android aplikacije koja je podijeljena na dva dijela, jedan dio koji predstavlja stranu konobara, te drugi dio koji predstavlja stranu vlasnika ugostiteljskog objekta. Nakon toga slijedi opis funkcionalnosti web aplikacije, koja omogućava registraciju ugostiteljskog objekta u sustav aplikacija te organizaciju narudžbi.

Ključne riječi: *Android, Firebase, konobari, sustav naručivanja, React*

MOBILE ANDROID APPLICATION AND WEB APPLICATION FOR ORGANIZING ORDERS IN CATERING ESTABLISHMENTS IN THE "BARS" CATEGORY

Abstract

The goal of this project was to develop a mobile Android application and a web application for the needs of catering establishments in the " Bars" category. The mobile application is intended for waiters to facilitate the process of creating and managing orders and organizing the facility's beverage menu. On the other hand, the application enables the owners of catering establishments to have a comprehensive view of the relevant information as well as updating facility information and mobile phone numbers that employees use to log in. Through the web application, waiters can view the created orders. They have the option of rejecting the order or setting the order in a ready state, both cases will result in notifying the waiter who initially created the order. This kind of work results in quick communication and coordination among the staff, which fulfills the main purpose of the mobile and web application. The following text will describe the technologies that were used for the development of the applications. Then, the connection of applications and individual functionalities according to different user roles will be explained. It will also explain the programming solution of the project, the functionality of the mobile Android application, which is divided into two parts, one part representing the side of the waiter, and the other part representing the side of the owner of the catering establishment. This is followed by a description of the functionality of the web application, which enables the object registration in the application system and the orders organization.

Keywords: *Android, Firebase, waiters, order system, React*

Sadržaj

1. Uvod

U ugostiteljskim objektima, konobari su suočeni s izazovima koji se povećavaju s razmjernošću veličine objekta, obimom menija i brojem gostiju. Potrebna je efikasna komunikacija među kolegama kako bi osigurali brzu i točnu isporuku narudžbi. Takvo radno okruženje konobarima donosi opterećenost kao i mogućnost nenamjernih grešaka. Karakteristično za konobarsko zanimanje jest učestalost promjene zaposlenika. To dodatno naglašava potrebu za preciznom evidencijom tko ima pravo prijave i u kojem objektu.

Svrha ovog projekta je smanjenje opterećenosti zaposlenika te time pogrešaka u radu. Sve informacije o pićima trebaju biti dostupne i ažurne. Aplikacije omogućavaju stalnu automatiziranu komunikaciju u procesu pravljenja, pripreme i dostave narudžbe. S druge strane vlasnici ugostiteljskih objekata trebaju imati na raspolaganju uvid u informacije objekta. Određuju telefonske brojeve uređaja za rad poput tableta ili mobitela s kojima se zaposlenici prijavljuju. Time nemaju potrebu stalnog ažuriranja izmjene zaposlenika, jer se prijava vrši isključivo preko uređaja koji su vezani za određeni objekt.

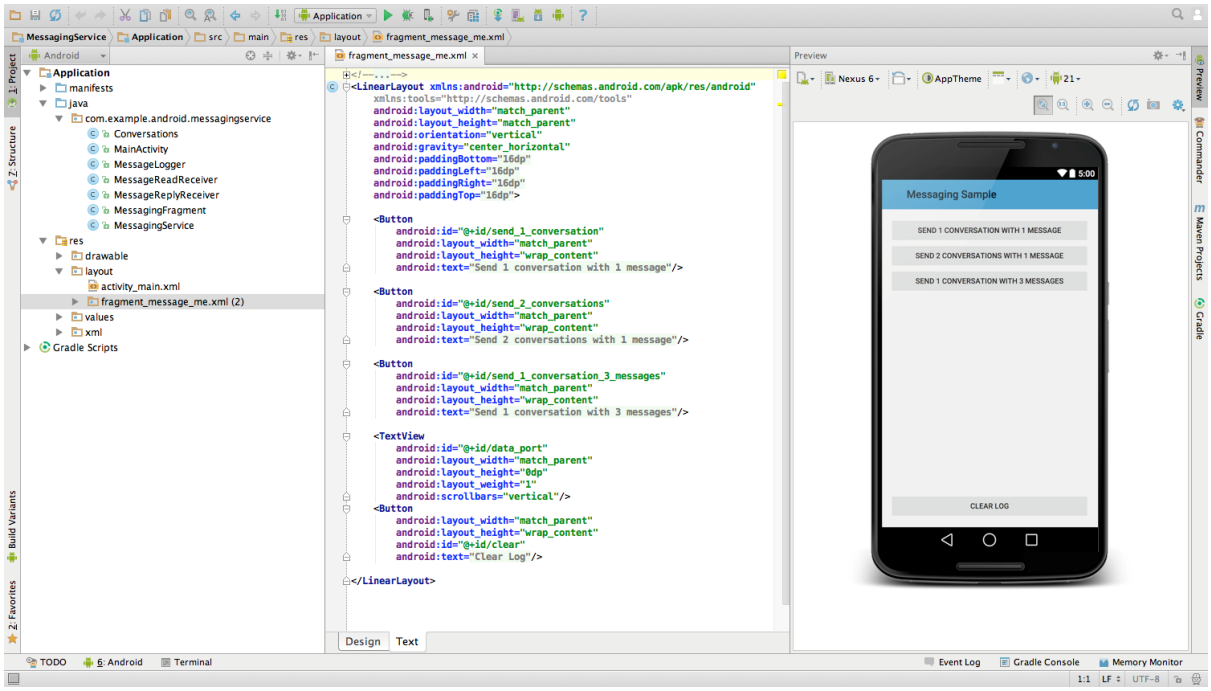
Ovaj rad obuhvaća funkcionalnosti mobilne Android i web aplikacije koje su povezane te tvore jedinstvenu cjelinu. Rad se sastoji od tri osnovna dijela. U prvom dijelu su opisane tehnologije koje su korištene pri izradi ovih aplikacija. Drugi dio pruža detaljniji uvid u arhitekturu aplikacija te preciznije razlaže koji dijelovi aplikacija su namijenjeni za koje uloge korisnika. Treći, završni dio razlaže funkcionalnosti koje su dostupne putem aplikacija.

2. Tehnologije i alati korišteni pri izradi

2.1. Android Studio

Android Studio je Googleovo službeno razvojno okruženje koje je namijenjeno za razvoj i izradu Android mobilnih aplikacija. Zasnovan je na IntelliJ IDEA platformi. Besplatan je za korištenje i dostupan na većini operacijskih sustava, uključujući, Windows, macOS i Linux. Slika 1 prikazuje izgled rada unutar Android Studija, odnosno korisničko sučelje ovog razvojnog okruženja. Android Studio nudi razne alate kojima se olakšava razvoj mobilnih aplikacija kao što su:

- Razvojni alat koji koristi fleksibilan Gradle[1]
- Emulator za pokretanje i *debuggiranje* aplikacija
- Inteligentni uređivač koda
- Alati za analiziranje brzinu izvedbe
- Opsežni alati i okviri za testiranje [1].



Slika 1. Primjer izrade aplikacije u Android Studiju [13]

[1] Gradle – alat za automatizaciju izrade za višejezični razvoj softvera

2.2. Java programski jezik

Programski jezik Java nastao je 1995. godine. Prvenstveno se koristio za *web* aplikacije, no danas je riječ o programskom jeziku opće uporabe, temeljenom na objektno orijentiranom programiranju. Java je posebno popularna u poslovnim aplikacijama zahvaljujući svojoj jednostavnosti, stabilnosti i kompatibilnosti s prethodnim verzijama. Otvorenost programskog jezika omogućava razvojnu zajednicu da pridonose razvitku. Bitna karakteristika Jave je princip WORA (engl. *write once, run anywhere*), što znači da se napisani Java kod kompajlira jednom te se može izvršiti na različitim platformama bez potrebe za dodatnim prevođenjem koda. Java aplikacije se najčešće kompajliraju u *bytecode*[2] što čini rezultat sastavljanja Java programa. Zatim se izvršava na Java virtualnom stroju (JVM), neovisno o računalnoj arhitekturi [2].

Korištenje Jave za razvoj Android aplikacija pruža programerima pristup velikom obilju alata koji značajno olakšavaju proces kreiranja aplikacije. Za razvoj Android aplikacija koristi se pomoć Android API-ja[3] koji omogućava programerima pristup funkcionalnostima uređaja poput kamere.

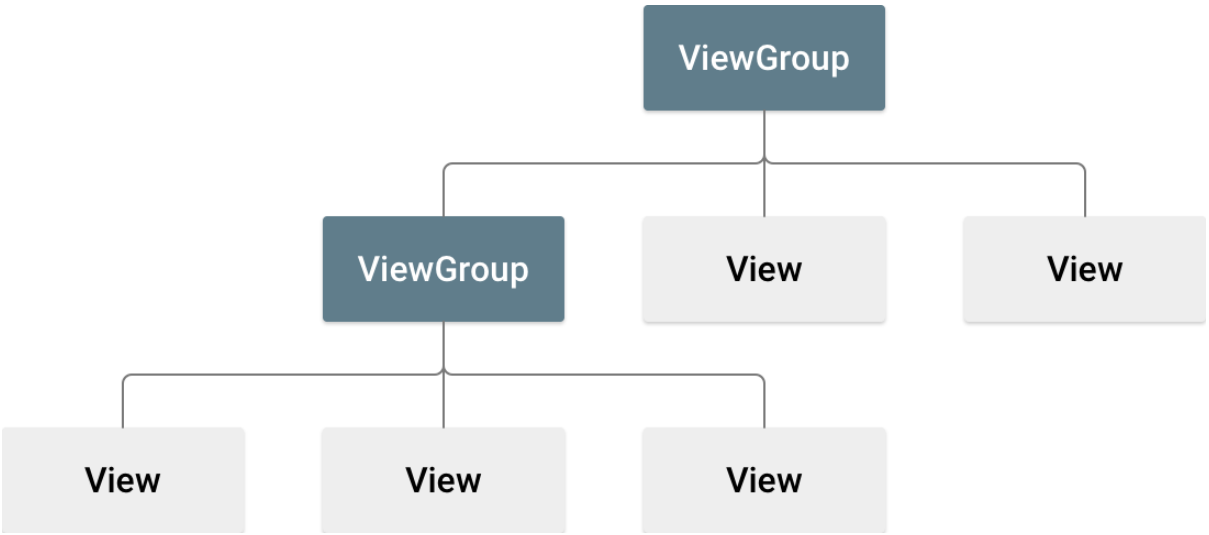
2.3. XML

XML (engl. *Extensible Markup Language*) je jezik koji definira pravila strukturiranja XML dokumenta u formatu koji je čitljiv ljudima i računalnima. Karakteristika jezika je uokvirenost odgovarajućeg sadržaja odgovarajućim oznakama koje ga opisuju. Sličan je jeziku HTML (engl. *HyperText Markup Language*), glavna razlika je što se u HTML-u koriste prederfinirane oznake, a XML jezik nudi mogućnost samostalnog definiranja i dodavanja novih oznaka koje se oblikuju prema potrebama. Izuzetno je moćan jezik za pohranu podataka koji omogućava jednostavno čuvanje, pretraživanje i dijeljenje informacija. Ključna prednost je u standardiziranom XML formatu što znači da XML datoteke lako mogu biti prenosive preko različitih sustava i platformi [3]. Da bi se dokument smatrao valjanim XML dokumentom, mora zadovoljiti sintaktička pravila. Jedno od pravila je sadržavanje isključivo jednog elementa koji se još naziva korijenskim elementom. Unutar njega se spremaju svi ostali elementi.

[2] Bytecode – kod koji se pretvara u binarni strojni kod
[3] API (engl. *Application Programming Interface*) – softverski posrednik koji omogućuje strukturirani prijenos podataka između aplikacija

U Android aplikacijama XML jezik se koristi za kreiranje korisničkih sučelja. Svi elementi korisničkog sučelja izgrađeni su pomoću hijerarhije objekata *View* i *ViewGroup*. Slika 2 prikazuje primjer hijerarhije izgrađene pomoću navedenih objekata. *View* se još naziva *widget* te predstavlja objekte od kojih će se sastojati korisničko sučelje Android aplikacije. *ViewGroup* se još naziva i *layout*, služi kao spremište za *widgete* [4]. Postoje različiti *widgeti*, ovisno o ulozi mogu predstavljati gumb ili tekstualni okvir za prikaz i unos teksta. Svaki *View* ima vlastiti naziv, najpoznatiji su:

- TextView – element korisničkog sučelja koji samo prikazuje tekst
- EditText – element korisničkog sučelja koji služi za unos teksta
- Button – element korisničkog sučelja koji predstavlja gumb, služi da pritiskom pokreće određenu radnju
- ImageView – element korisničkog sučelja koji služi za prikaz resursa slika.



Slika 2. Ilustracija hijerarhije korisničkog sučelja [4]

2.4. React

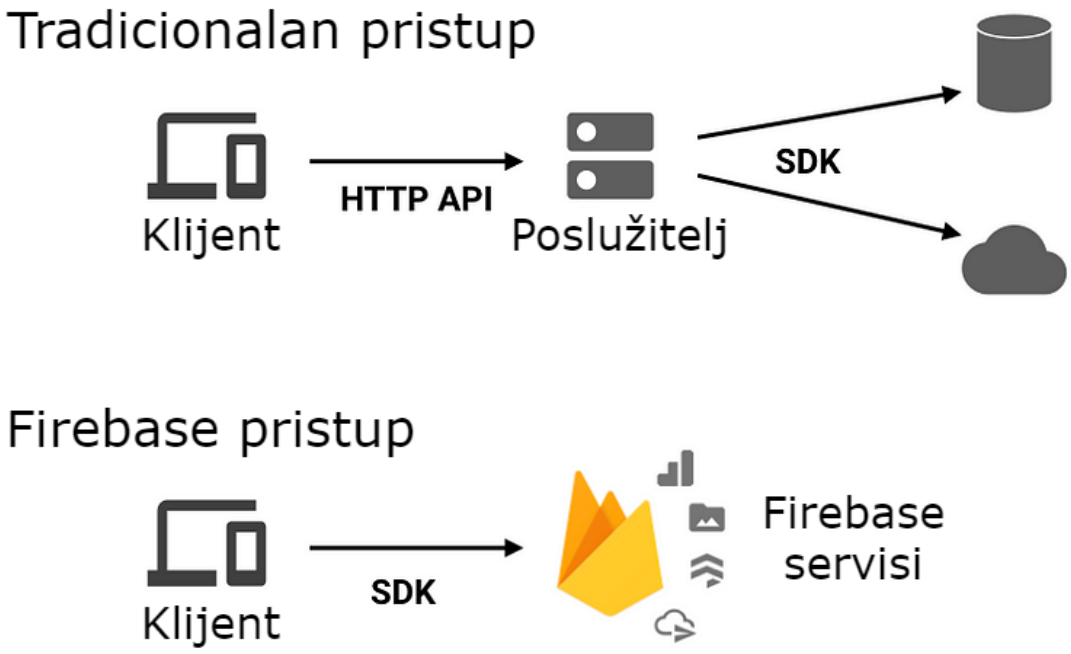
React je besplatna biblioteka otvorenog koda za programski jezik JavaScript koja omogućava razvoj korisničkih sučelja SPA (engl. *single-page application*). To je tip web aplikacije koja u interakciji s korisnikom dinamički mijenja dijelove stranice. Za razliku od tradicionalnih web aplikacija gdje se kod svakog izvršavanja akcije od strane klijenta obavlja niz DOM[4] promjena. Tradicionalno manipuliranje DOM-a putem JavaScripta je spor proces.

[4] DOM (engl. *Document Object Model*) – hijerarhijski prikaz grafičkog sučelja na web-u.

React takve procese odrađuje puno efikasnije zahvaljujući virtualnom DOM-u koji smanjuje opterećenost mreže pri svakoj akciji tako da se ažuriranje vrši samo na neophodnim DOM čvorovima. Uz pomoć Reacta izrađuju se i mobilne aplikacije (React Native). Prava snaga biblioteke je u njezinim komponentama koje programeri stvaraju kako bi prikazali dijelove korisničkog sučelja. To svojstvo čini programska rješenja skalabilnima jer se jedna komponenta može koristiti više puta u aplikaciji. React funkcionalne komponente su JavaScript funkcije koje mogu primati parametre, izvršavati radnje te imati povratnu vrijednost JSX kod. JSX je proširenje programskog jezika JavaScript (JavaScript XML). Omogućava pisanje oznaka nalik na HTML unutar JavaScript datoteke [5]. Takvim načinom pisanja koda dobivamo preglednost, ali internet preglednici nisu u stanju interpretirati JSX kod pa ga je potrebno pretvoriti u standardni JavaScript kod uz pomoć alata koji se naziva Babel [6].

2.5. Firebase

Firebase je Googleova BaaS (engl. *Backend-as-a-Service*) platforma koja pruža niz alata za razvoj aplikacija [7]. Servisi BaaS platforme smješteni su u oblaku te za njihovo održavanje nije odgovoran programer već pružatelj servisa. Prednost takvog koncepta je jednostavnost izrade aplikacije jer se otklanja potreba upravljanja bazama podataka te pisanja pozadinskog koda za server aplikacije. Klijentska strana za dohvaćanje podataka iz baze ne koristi programska sučelja, tzv. API-je koji bi preuzeli podatke s poslužitelja, obradili ih te poslali klijentu koji je poslao upit za njih. Firebase za takve radnje ima drugačiji pristup, koristi alate za razvijanje softvera ili kraće SDK (engl. *Software Development Kit*) koji pružaju izravno komuniciranje aplikacije i Firebase servisa. Razlika između tradicionalnog pristupa i Firebase pristupa u procesu komunikacije klijenta i poslužitelja vidljiva je na slici 3 koja je preuređena prema izvoru [14].



Slika 3. Prikaz tradicionalne komunikacije i komunikacije korištenjem Firebasea [14]

Podaci se pohranjuju u NoSQL bazu podataka u JSON (engl. *JavaScript Object Notation*) formatu. JSON je jednostavni format dizajniran da zauzima što manje podatkovnog prostora kako bi razmjena podataka bila što brža. Ljudima ga je lako čitati i pisati, a računala ga lako generiraju i analiziraju [8]. Firebase nudi dvije vrste baza podataka, *Realtime Database* i *Cloud Firestore*. *Realtime Database* sprema podatke kao JSON stablo te pri dodavanju novog podatke stvara se novi čvor u JSON strukturi s pripadajućim ključem kojeg može korisnik stvoriti ili će biti automatski generiran. *Cloud Firestore* je novija verzija, pruža fleksibilne hijerarhijske strukture podataka. Podaci se sastoje od dokumenata i zbirki.

U projektu je korištena baza *Realtime Database* koja sprema podatke u JSON strukturu. Dobrim smještajem podataka u relevantne čvorove stvara se pregledna struktura iz koje se podaci mogu brzo izvlačiti bez potrebe za kompliciranim upitima. Baza također pruža iznimno koristan alat – mogućnost postavljanja slušatelja na čvorove podataka. Slušatelji kontinuirano prate promjene podataka koji se nalaze unutar odabranog čvora. Ova funkcionalnost omogućava automatsko praćenje i reagiranje na određene promjene u stvarnom vremenu. Ostali korišteni Firebase servisi za izradu ovog rada su:

- Authentication – servis sigurne registracije i prijave korisnika u aplikaciju. Autentifikaciju nudi putem elektronične pošte, telefonskog broja kao i drugih pružatelja identiteta, poput Google, Twitter, Facebook ili GitHub računa.
- Cloud Storage – servis za pohranu datoteka generiranih od strane korisnika, datoteke mogu biti i fotografije i videozapisi [9].
- Performance Monitoring – servis koja pruža uvid u brzinu rada određenih stavki aplikacije. Pregledavanja performansi i analiziranje pomaže da se u stvarnom vremenu uvide moguća poboljšanja [10]. Performanse je moguće pratiti ovisno o verziji aplikacije, državi iz koje se koristi aplikacija, API verziji Android sustava, modelu uređaja te drugim parametrima.
- Crashlytics – servis izvjestitelj o rušenjima aplikacije u stvarnom vremenu. Pomaže pri određivanju prioriteta za rješavanje problema stabilnosti aplikacije [11]. Servis iznosi naziv i opis pogreške te pruža precizne informacije o datoteci u kojoj je nastala greška pa čak i broj linije koda te datoteke. Dodatne informacije su model mobitela, verzija Android operacijskog sustava, stanja u kojem je bio uređaj kada je nastala greška te vrijeme nastanka greške.

2.5.1. *Realtime Database* SDK implementacija u Android aplikaciju

Implementacija Firebase *Realtime Database* SDK-a u Android aplikaciju uključuje sljedeće korake:

1. Stvaranje Firebase projekta – prvo je potrebno stvoriti projekt s Firebase konzolom
2. Dodavanje konfiguracije – nakon što je projekt stvoren, potrebno je dodati konfiguracijski JSON u projekt
3. Dodavanje SDK-a – SDK se uključuje dodavanjem odgovarajuće zavisnosti u Gradle datoteku modula projekta na razini aplikacije – isječak programskog koda 1.

Isječak programskog koda 1. Firebase preporučana implementacija [15]

```
dependencies {
    // Import the BoM for the Firebase platform
    implementation(platform("com.google.firebase:firebase-bom:32.2.2"))

    // Add the dependency for the Realtime Database library
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation("com.google.firebase:firebase-database")
}
```

2.5.2. *Realtime Database* SDK implementacija u web aplikaciju

Implementacija Firebase *Realtime Database* SDK-a u web aplikaciju uključuje sljedeće korake:

1. Stvaranje Firebase projekta – prvo je potrebno stvoriti projekt s Firebase konzolom
2. Instalirati Node.js – poslužiteljsko okruženje otvorenog koda koji omogućava izvršavanje JavaScript jezika na poslužitelju
3. Instalirati npm (engl. *Node Package Manager*) – upravitelj paketa za JavaScript programski jezik pomoću kojeg se vrši instalacija paketa te njegova organizacija u web aplikaciju
4. Instalacija Firebase paketa – u terminal je potrebno unijeti naredbu: „npm install firebase“ koja pokreće npm za instaliranje Firebase paketa
5. Dodavanje SDK-a – inicijalizacija baze podataka u skriptu koja postaje konfiguracijskom skriptom – isječak programskog koda 2.

Isječak programskog koda 2. Firebase preporučeni isječak koda [16]

```
// Replace the following with your app's Firebase project configuration
// See: https://firebase.google.com/docs/web/learn-more#config-object
const firebaseConfig = {
  // ...
  // The value of `databaseURL` depends on the location of the database
  databaseURL: "https://DATABASE_NAME.firebaseio.com",
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);

// Initialize Realtime Database and get a reference to the service
const database = getDatabase(app);
```

3. Arhitektura aplikacija

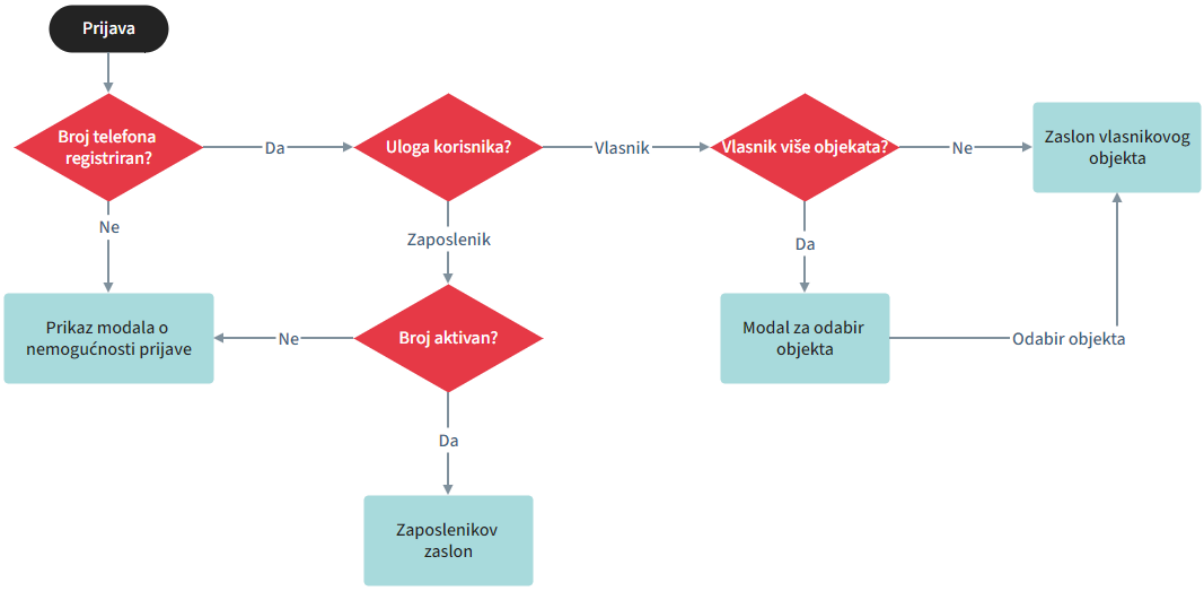
U nastavku rada, bit će opisani dijagrami toka i arhitektura mobilne Android i web aplikacije. Cilj ovog odlomka je pružiti jasnije razumijevanje u povezanost aplikacija te funkcionalnosti obje aplikacije prema različitim korisničkim ulogama.

3.1. Dijagram toka

Dijagram toka je grafički prikaz programa koji pojednostavljuje i vizualno ilustrira tijek aplikacije, čineći time razumijevanje problema ili zadatka programa lakšim. Dijagram toka se može koristiti neovisno o programskom jeziku. Komponente dijagrama su univerzalno definirane, a često imaju oblik geometrijskog lika. Na primjer, elipsa se koristi za označavanje početka, kraja ili prekida programa, dok romb označava uvjetno grananje. Elementi su međusobno povezani linijama koje simboliziraju prijelaze iz jedne komponente u drugu.

3.1.1. Dijagram toka mobilne Android aplikacije

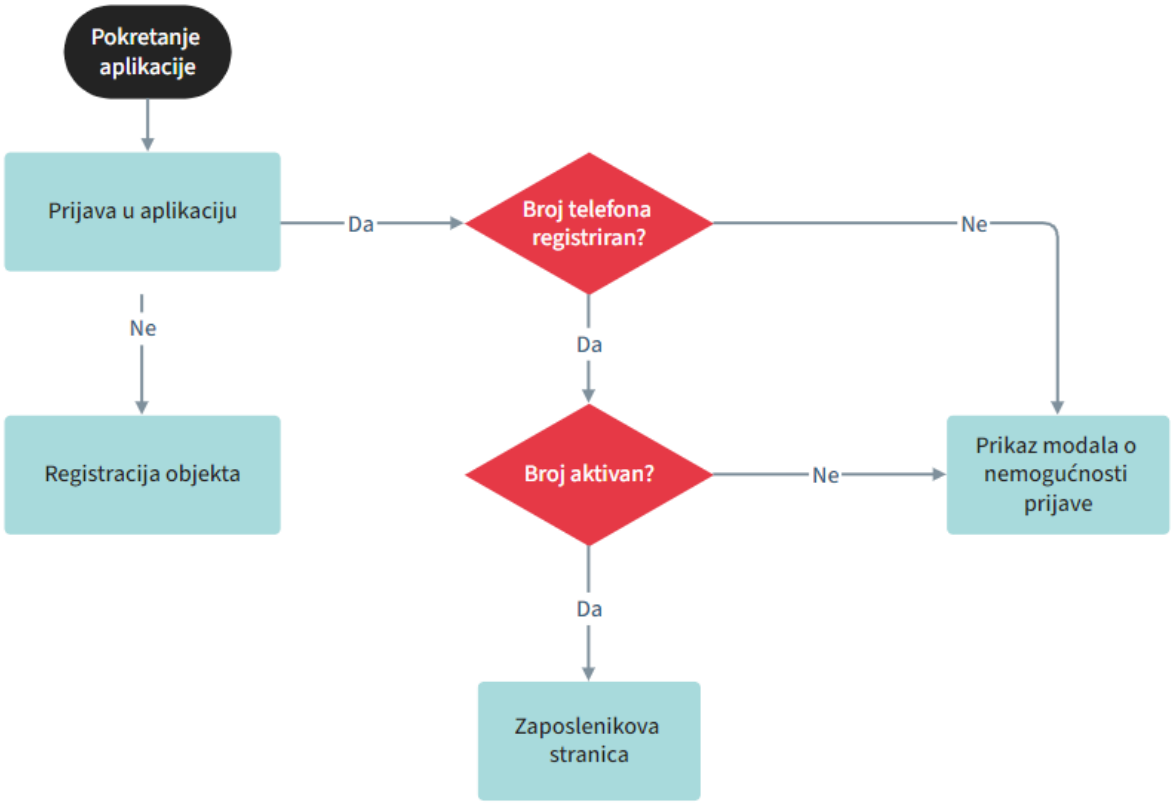
Mobilna aplikacija je namijenjena zaposlenicima, konobarima koji će prikupljati narudžbe i vršiti ažuriranja menija objekta u kojem rade. Vlasnicima je također namijenjena aplikacija koji mogu biti prijavljeni kao vlasnici jednog ili više objekata. Stoga je od iznimne važnosti osigurati temeljitu i dobro osmišljenu funkcionalnost prijave u aplikaciju. Na slici 4 prikazan je dijagram toka prijave u mobilnu Android aplikaciju.



Slika 4. Dijagram toka mobilne aplikacije

3.1.2. Dijagram toka web aplikacije

Web aplikacija ima jednostavniju logiku. Namijenjena je svima koji žele registrirati ugostiteljski objekt u sustav aplikacija. Prijavljuju se zaposlenici koji imaju ulogu pripreme narudžbe. Na slici 5 prikazan je dijagram toka koji ilustrira moguće kretnje korisnika kroz aplikaciju.



Slika 5. Dijagram toka web aplikacije

3.2. Arhitektura aplikacija

Obje aplikacije, i mobilna i web imaju komponente kojima mogu pristupiti svi korisnici. Tek nakon prijave, ovisno o ulozi korisnika prikazuju se pojedinačni dijelovi vezani za ulogu korisnika.

3.2.1. Arhitektura web aplikacije

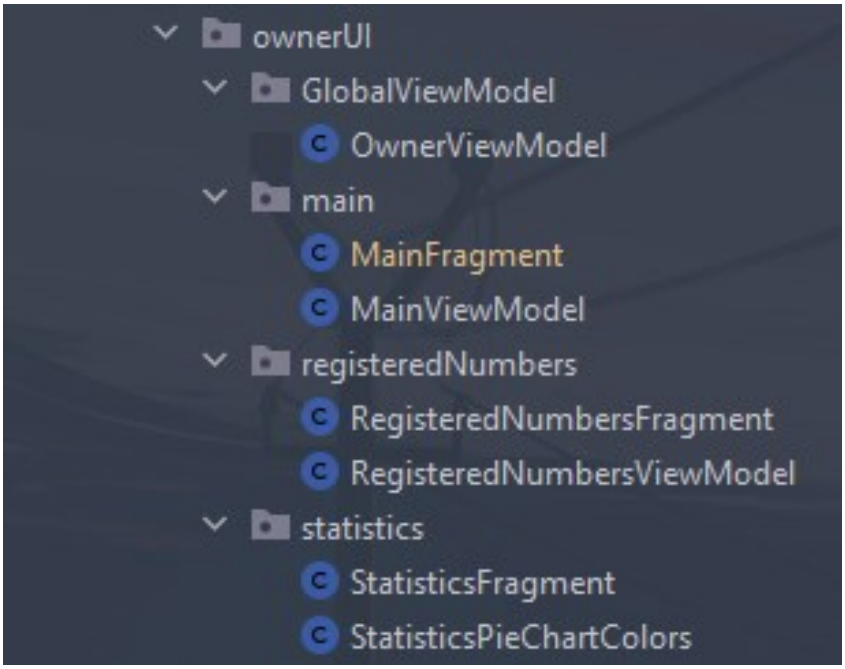
Arhitektura web aplikacije je jednostavna, jer je strukturirana s jednom osnovnom korisničkom ulogom kojoj se pristupa uspješnom prijavom, čime je olakšan dizajn i izvedba. S obzirom na to da je glavna usmjerenost zaposlenika, koji koristi web aplikaciju, priprema narudžbi, aplikacija bi mu trebala služiti kao jasan prikaz narudžbi i njenih detalja. Time je

ograničen opseg interakcija s podacima kako bi se održala jednostavna logika usmjerena na osnovne funkcionalnosti.

3.2.2. Arhitektura mobilne Android aplikacije

Arhitektura je oblikovana s dvije definirane prijavljene korisničke uloge. Svaka uloga ima svoju skupinu funkcionalnosti koje su organizirane u podskupine. Svaka podskupina funkcionalnosti nalazi se na pojedinačnom fragmentu[5]. Aktivnost je ključna komponenta Android aplikacije koja brine o stvaranje korisničkog sučelja [12]. Jedna aktivnost će sadržavati fragmente, a svaki fragment sadržava korisničko sučelje uz pomoć kojeg se izvršavaju podskupine funkcionalnosti. Fragmenti su prikazani pomoću iste aktivnosti te je potrebno osigurati komunikacije fragment – aktivnost i fragment – fragment. U takvom principu rada problem je skladištenje podataka tijekom izmjene fragmenata koji se uništavaju skupa sa svojim podacima. Jedan primjer rješenja je pri svakom uništenju fragmenta spremiti podatke u aktivnost, no to zahtjeva puno resursa zbog česte potrebe za komunikacijom fragment – aktivnost. Drugo rješenje, koje je implementirano u mobilnu aplikaciju ovog rada je mogućnost vezanja komponente koja se koristi za pohranu i upravljanje podacima koji će biti sačuvani nakon uništenja fragmenta. Takva komponenta u Android okruženju naziva se *ViewModel*. Ako fragment ima podatke koji su potrebni ostalim fragmentima ili aktivnosti sadržavat će svoju *ViewModel* klasu. Za pristup čitanju i ažuriranju podataka određene *ViewModel* klase prvobitno ju je potrebno implementirati u vlastitu klasu te zatim pozvati određeni podatak kroz implementiranu klasu. Podaci koji će biti potrebni svim fragmentima i aktivnosti nalazit će se u globalnoj klasi *ViewModela*. Slikom 6 prikazana je struktura korisničkog sučelja prijavljenog vlasnika ugostiteljskog objekta, aktivnost sadrži tri fragmenta, dva fragmenta imaju klasu za spremanje podataka te postoji globalna *ViewModel* klasa kojoj svi fragmenti i aktivnost pristupaju.

[5] Fragment – predstavlja dio korisničkog sučelja koji ima vlastiti životni ciklus i ovisan je drugom fragmentu ili aktivnosti



Slika 6. Prikaz strukture klasa korisničkog sučelja vlasnika

4. Programsko rješenje

Ovo poglavlje detaljno opisuje funkcionalnosti aplikacija. Prvo će biti objašnjen *backend* kojeg čini Firebase radi boljeg razumijevanja radnih aspekata aplikacija. Nakon toga će se analizirati mobilna aplikacija, pružajući opis prijave te mogućnosti koje imaju prijavljeni korisnici. Posljednji dio posvećen je web aplikaciji, registraciji ugostiteljskog objekta, prijavi korisnika te korisničkom sučelju prijavljenog zaposlenika.

4.1. Baza podataka

Realtime Database je odabrano tehnološko rješenje jer pruža mogućnost postavljanja slušatelja na čvorove podataka. To pojednostavljuje proces praćenja i upravljanja podacima koji će se automatski ažurirati i prikazati korisnicima, a promjene su stalne zbog mogućnosti ažuriranje menija te pravljenja i organizacije narudžbi. Važno je napomenuti da se slušatelji na bazi moraju ručno zaustaviti, jer platforma ne pruža automatski mehanizam za njihovo gašenje. Ukoliko slušatelji nisu deaktivirani, oni će i dalje ostati aktivni i pratiti promjene. To može rezultirati višestrukim slušateljima na istom čvoru podataka, što uzrokuje usporenje dohvaćanja podataka iz baze. Zbog toga je ključno upravljati slušateljima kako bi se osiguralo optimalno funkcioniranje aplikacije i brza interakcija s podacima.

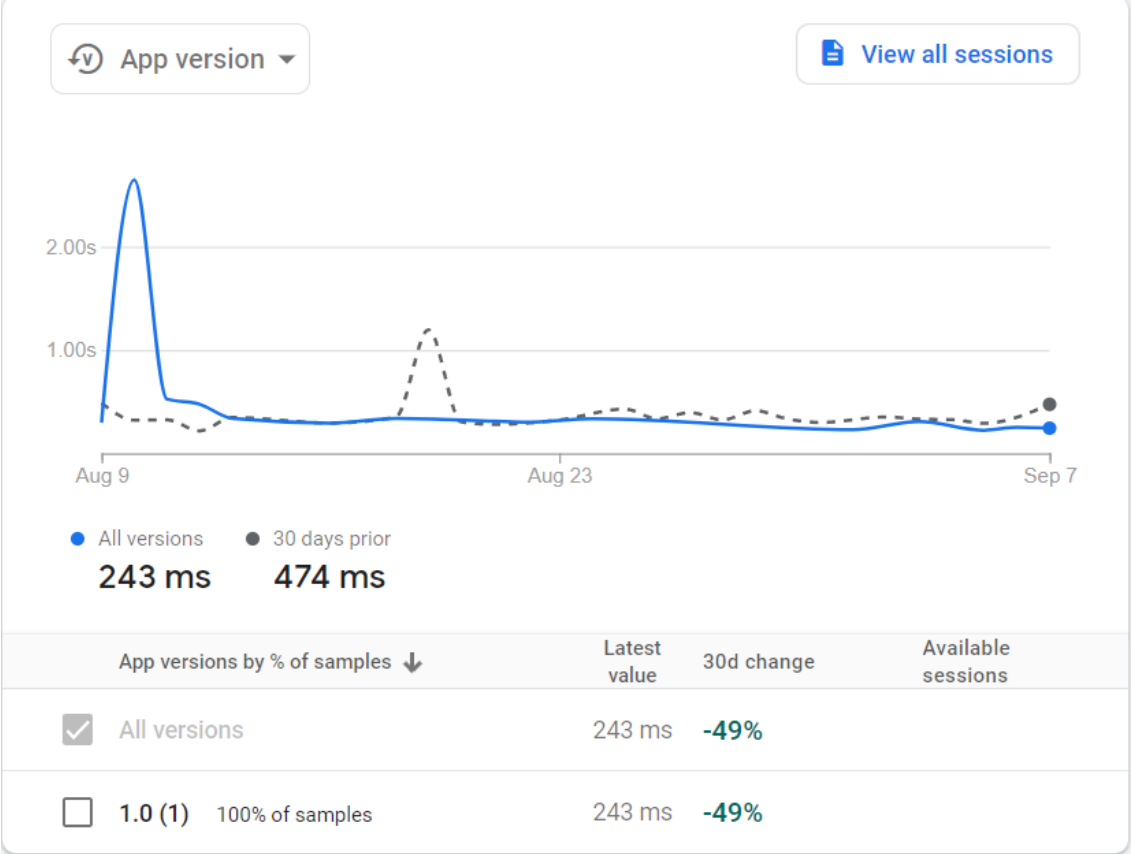
4.1.1. Firebase autentifikacija

Kako bi u aplikaciji mogli koristiti Firebase autentifikaciju, potrebna je implementacija servisa u aplikacije. Za Android aplikaciju potrebno je dodati još jednu ovisnost koju web aplikacija ima instaliranjem Firebase paketa preko terminala. Nakon toga se u Firebase konzoli mogu pratiti i kontrolirati svi korisnici. Prijava u aplikacije se vrši isključivo putem telefonskog broja. Uloge korisnika nisu pohranjene direktno unutar sustava autentifikacije. Umjesto toga, nakon uspješne autentifikacije korisnika, aplikacija će iz baze dohvatiti ulogu korisnika pomoću povezanog identifikatora.

4.1.2. Firebase performanse

Za praćenje performansi aplikacija, Firebase Performance Monitoring servis bio je ključan alat koji je omogućavao nadzor na određenim aspektima. Posebnu pozornost stavljena je na praćenje brzine pokretanja aplikacije te brzina aplikacija kada se sadržaj često izmjenjiva. Primjer je česta navigacija u mobilnoj aplikaciji zbog koje se uništava fragment koji se napušta

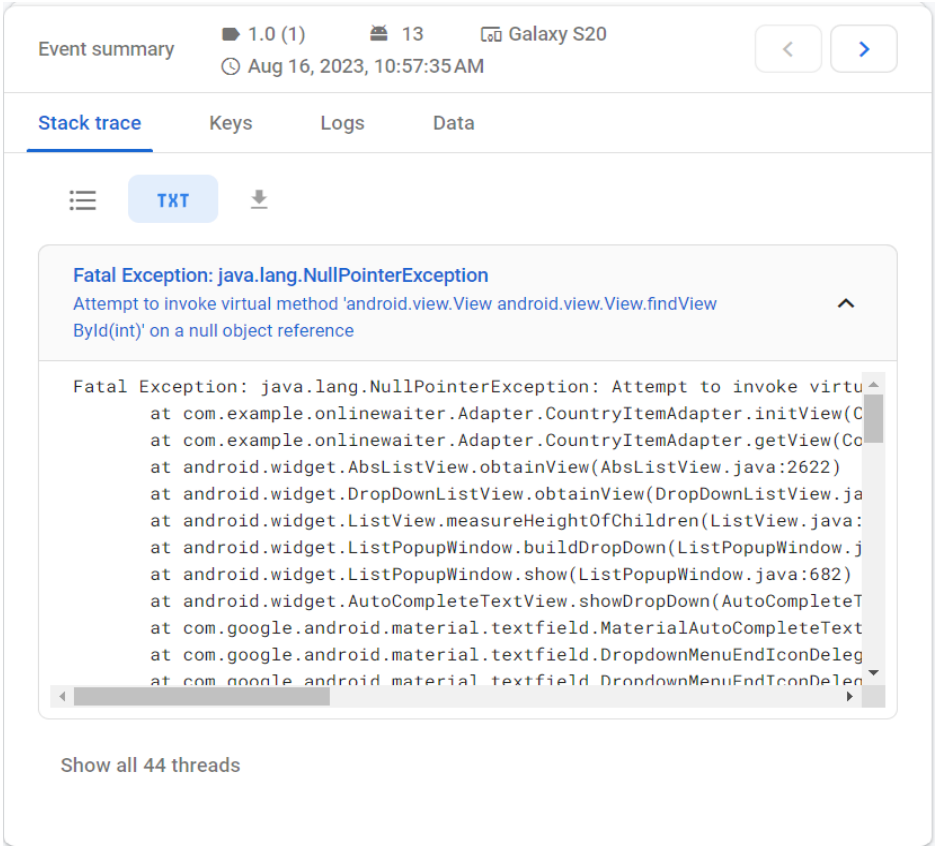
i stvara odredišni fragment. Na slici 7 prikazano je korisničko sučelje Firebase Performance usluge, gdje su prikazani detalji vezani uz aspekt pokretanja Android aplikacije.



Slika 7. Detalji brzine izvođenja pri pokretanju mobilne aplikacije

4.1.3. Firebase Crashlytics

Android aplikacija omogućava prijavu opisa greške administratorima, no nije sigurno da će korisnici razumjeti tehničke detalje pogreške. Alat je testiran u simuliranom okruženju, bez pravih korisnika, stoga će njegova učinkovitost biti potvrđena tek kada bude u uporabi od strane stvarnih korisnika aplikacije. Testiranje alata je pokazalo uspješne rezultate i spremnost za stvarne korisnike. Korisničko sučelje koje servis pruža administratorima sastoji se od liste grešaka u određenom vremenskom razdoblju. Odabirom pojedine greške prikazuju se detalji vezani za odabranu grešku. Na slici 8 prikazan je izgled korisničkog sučelja koji sadrži detalje odabrane greške. Vidljivi su detalji uređaja na kojem je nastala pogreška, opis pogreške temeljito je objašnjen tekstualno.



Slika 8. Prikaz pogreške u simuliranom okruženju

4.2. Mobilna Android aplikacija

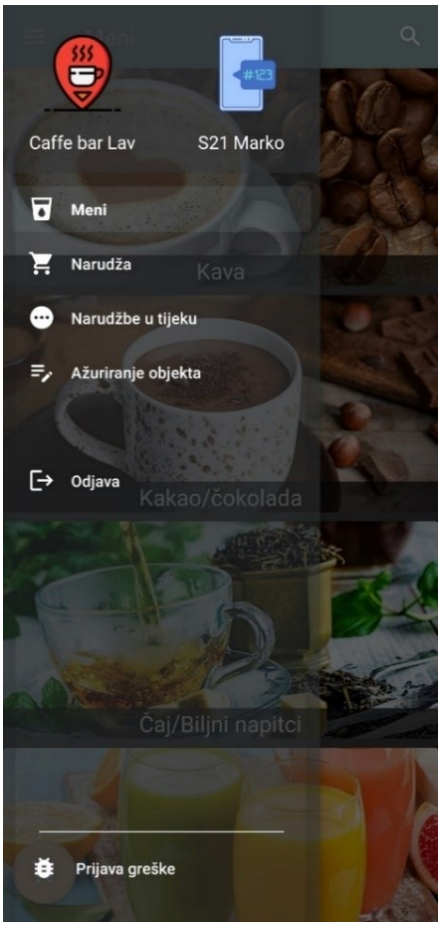
4.2.1. Prijava u aplikaciju

Kao što je već napomenuto, u aplikaciju se može prijaviti zaposlenik ili vlasnik ugostiteljskog objekta skupine „Barovi“. Princip prijave je putem telefonskog broja na koji se prima jednokratna lozinka koja sadržava šesteroznamenkasti broj koji je potrebno potvrditi. Nužan uvjet za slanje jednokratne lozinke na uređaj je prethodna registracija tog uređaja u Firebase autentifikaciji. Vlasnik registrira brojeve za određeni objekt, dok se vlasnikov broj registrira pri registraciji ugostiteljskog objekta, o tome će se više reći u opisu web aplikacije. Mobilna aplikacija zahtjeva od korisnika potvrdu za pristup telefonskom broju uređaja kako bi se olakšao proces prijave tako što će aplikacija unositi broj mobitela. Telefonski broj može biti prikriven, ovisno o operateru telefonskih usluga što znači da se broj neće moći pročitati. U takvim situacijama aplikacija kako bi osigurala dosljednost i korisničko iskustvo pamtit će uneseni telefonski broj kada je uspješno odrađena prijava korisnika. Još jedan način olakšavanje prijave je hvatanje primljene poruke koja sadrži potrebnu lozinku. Korisnika se pita za dopuštenje da se lozinka unese pomoću aplikacije te time završi proces provjere

autentifikacije. Ta radnja je omogućena osluškivanjem i reagiranjem na dolazne poruke pomoću Android komponente *Broadcast Receiver*. Ona omogućava aplikaciji da obradi dolazne poruke prema definiranim kriterijima kao što je prepoznavanje šesteroznamenkaste lozinke. Kontrola prijave korisnika je prikazana slikom 4.

4.2.2. Korisničko sučelje prijavljenog zaposlenika

Uspješnom prijavom zaposlenika otvara se aktivnost sa četiri fragmenta, svaki pruža svoj skup funkcionalnosti. Kretanje kroz fragmente izvršava se izbornikom za navigaciju – slika 9. Aktivnost postavlja slušatelje na određenim čvorovima podataka unutar baze.



Slika 9. Izgled početnog zaslona i navigacijskog izbornika

Početni zaslon prikazuje ponudu pića prema kategorijama koje sadrži ugostiteljski objekt. Svaka kategorija sadrži jedno ili više pića koja su prikazana u obliku kartica. Kartice sadrže relevantne informacije pića poput naziva, opisa, fotografije, cijene, dostupne količine, i količine u trenutnoj narudžbi. Korisnicima su omogućene akcije dodavanja i uklanjanja pića iz narudžbe. Svaka promjena u meniju koju obave ostali zaposlenici odmah se osvježava na

korisničkom sučelju. U cilju ubrzavanja procesa prikupljanja narudžbi, implementirana je funkcionalnost pretraživanja pića prema nazivu. Ova opcija omogućava konobarima brži pristup željenim pićima tako da jednostavno unesu naziv u pretraživač i pronađu odgovarajuće piće.

Fragment narudžbe predočava pića koja su dodana u narudžbu. Moguće im je izmijeniti količinu. Osim toga prikazane su informacije o ukupnoj količini pića te cijeni narudžbe. Brisanje narudžbe moguće je pomoću jednog klika. Za potvrdu pravljenja narudžbe potrebno je odabrati stol za koji je narudžba namijenjena. Ulaskom u fragment trenutne narudžbe sve cijene i ostale informacije pića ostat će nepromijenjene. Promjene koje drugi korisnici vrše neće automatski utjecati na prikaz kao što je to u slučaju početnog fragmenta. Ako se bilo kakve promjene dogode na pićima koja su dodana u narudžbu, one će biti ažurirane tek kada se korisnik ponovno vrati u taj fragment. Ova logika je osmišljena s namjerom osiguravanja konzistentnosti cijena i sprječavanja situacija u kojima bi gostima objekta bila predočena različita cijena od one koji im je prvobitno iznesena. Konobar će biti obaviješten ako drugi zaposlenici naprave promjene poput brisanja pića ili ukoliko piće više nije dostupno u zadovoljavajućoj količini, a nalazi se u narudžbi koja još nije trenutno napravljena. Obavijest je prikazana u modalu koji se automatski otvara. U ovisnosti o situaciji, ako je piće obrisano ili njegova količina postane nedostupna (količina je nula), tada će se pića ukloniti iz narudžbe. S druge strane, ukoliko promjena smanji količinu pića ispod zadovoljavajuće granice, tada će piće zadržati preostalu količinu. Konobar će primiti obavijest bez obzira na to gdje se trenutno nalazi unutar aplikacije, bilo u fragmentu narudžbi ili bilo kojem drugom dijelu. Ovo predstavlja jedan od primjera funkcionalnosti aplikacije koja osigurava visoko kvalitetno radno iskustvo. Ova mogućnost postiže se putem slušatelja postavljenih na bazi podataka pomoću *Realtime Database* servisa, što omogućuje trenutačno i pouzdano obavještavanje o svim bitnim promjenama u stvarnom vremenu.

Sljedeći fragment obrađuje funkcionalnosti o narudžbama u tijeku koje mogu biti u jednom od četiri stanja.

1. Narudžba u priprema
2. Narudžba spremna – ukazuje na to da je narudžba spremna za isporuku, pri čemu konobar koji je inicijalno kreirao narudžbu prima obavijest na prijavljenom uređaju da je narudžba pripremljena i spremna za isporuku na određeni stol

3. Narudžba odbijena – odbijanje se vrši od strane konobara koji ju priprema, konobar koji je napravio narudžbu dobiva obavijest o odbijanju narudžbe, a nakon što pročita razlog, narudžba se uklanja iz sustava
4. Zahtjev za brisanje narudžbe – zadnje moguće stanje označava situaciju u kojoj konobar koji preuzima narudžbu od gostiju šalje zahtjev za brisanjem. Ovo stanje je uvedeno kako bi se omogućile ispravke u slučaju mogućih grešaka pri kreiranju narudžbe ili na zahtjev samih gostiju. Konobar koji priprema narudžbu ima ovlasti odlučiti hoće li prihvatiti brisanje ili odbiti zahtjev.

U svakom trenutku je moguć pregled pića sadržana u narudžbi Dostupno je pretraživati narudžbe prema broju stola.

U zadnjem fragmentu zaposlenik ima mogućnost kreiranja novog pića. Potrebno je odabrati kategoriju kojoj će pripadati, unijeti naziv, opis, cijenu i količinu, a opcionalno je dodavanje fotografije. Osim dodavanja novog pića moguće je i ažuriranja i brisanja pića iz menija objekta. Posljednja dostupna akcija je ažuriranje broja stolova u objektu na kojima je moguća ispostava narudžbi. Modal za ažuriranje broja stolova vidljiv je na slici 10.

Slika 10. Izgled modala za ažuriranje broja stolova objekta

Navigacijski izbornik ima još dvije dodatne radnje koje ne otvaraju zasebne fragmente. Prva radnja je odjava iz aplikacije, druga je otvaranje modala u koji je moguće unijeti opis pogreške koja se šalje administratorima aplikacije.

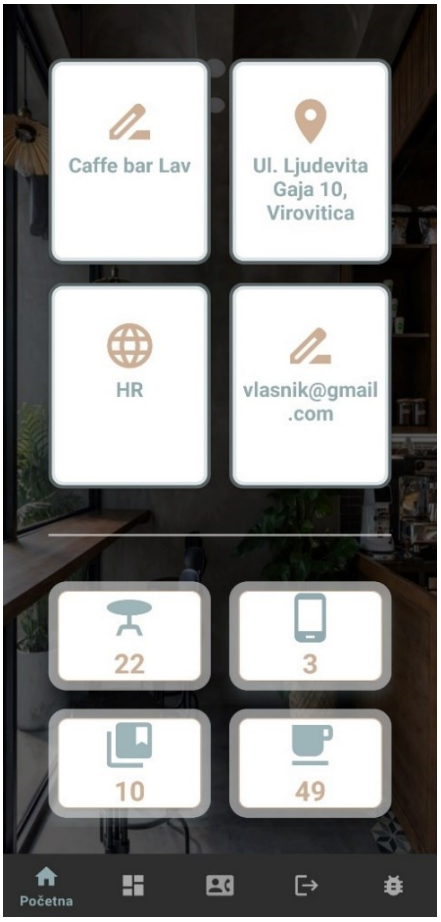
4.2.3. Korisničko sučelje prijavljenog vlasnika

Nakon uspješne prijave vlasnika, slijedi sličan proces kao kod zaposlenikovog sučelja, Aktivnost sadrži tri fragmenta, navigacijska traka se nalazi na donjem dijelu zaslona, postavljen je jedan slušatelj na određeni čvor baze podataka. Isječak programskog koda 3 prikazuje XML kod umetanja fragmenata unutar navigacijske trake.

Isječak programskog koda 3. Vlasnikova mobilna navigacija s fragmentima

```
 1. <?xml version="1.0" encoding="utf-8"?>
 2. <navigation xmlns:android="http://schemas.android.com/apk/res/android"
 3.     xmlns:app="http://schemas.android.com/apk/res-auto"
 4.     xmlns:tools="http://schemas.android.com/tools"
 5.     android:id="@+id/mobile_navigation_owner"
 6.     app:startDestination="@+id/nav_owner_main">
 7.
 8.     <fragment
 9.         android:id="@+id/nav_owner_main"
10.         android:name="com.example.onlinewaiter.ownerUI.main.MainFragment"
11.         android:label="@string/owner_nav_main_title"
12.         tools:layout="@layout/fragment_main" />
13.
14.     <fragment
15.         android:id="@+id/nav_owner_statistics"
16.         android:name="com.example.onlinewaiter.ownerUI.statistics.StatisticsFragment"
17.         android:label="@string/owner_nav_statistics_title"
18.         tools:layout="@layout/fragment_statistics" />
19.
20.     <fragment
21.         android:id="@+id/nav_owner_registered_numbers"
22.         android:name="com.example.onlinewaiter.ownerUI.registeredNumbers.RegisteredNumbersFr
          agment“
23.         android:label="@string/owner_nav_registered_numbers_title"
24.         tools:layout="@layout/fragment_registered_numbers" />
25.
26. </navigation>
```

Početni zaslon prikazuje elemente u dva rešetkasta rasporeda (engl. *GridLayout*). Prvi skup elemenata uključuje naziv objekta, njegovu lokaciju, standard države te email kontakt vlasnika. Standard države definira način na koji će se prikazivati različite vrijednosti kao što su valuta, decimalni razdjelnik, pozivni broj, oblik datuma te druge vrijednosti. Na primjer, prema hrvatskim standardima, cijena treba biti prikazana s dvije decimale gdje će decimalni dio biti odvojen zarezom. Vrijednosti sadržane u elementima prvog rešetkastog rasporeda je moguće uređivati klikom na pojedini element. Drugi skup informacija prikazuje relevantne podatke o objektu, uključujući broj stolova, registrirane telefonske brojeve, broj kategorija pića te ukupan broj pića. Početni zaslon prikazan je slikom 11. Vlasnik ima mogućnost mijenjati broj registriranih brojeva u drugom fragmentu dok zaposlenici imaju ovlasti za mijenjanje ostalih podataka. Promjene će biti odmah prikazane zbog slušatelja na bazi.



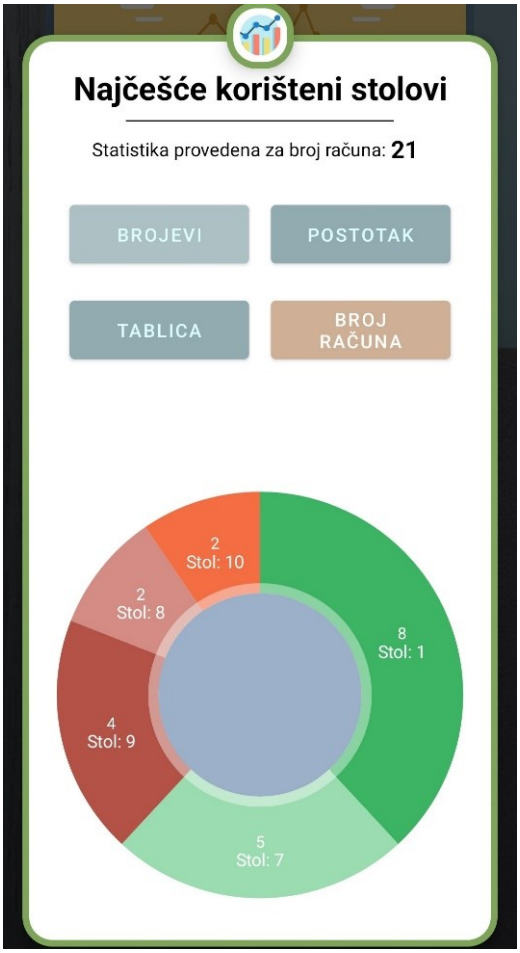
Slika 11. Izgled početnog zaslona prijavljenog vlasnika

Sljedeći fragment nazvan „Statistika“ pruža pregled relevantnih analiza. Postoje tri različite statistike. Prva analizira broj kreiranih računa prema telefonskim brojevima uređaja, pružajući uvid u koliko je svaki uređaj kreirao narudžbi. Druga analiza prikazuje najčešće naručivana pića. Treća statistika prikazuje najčešće korištene stolove, odnosno najviše naručivanih narudžbi za stolove. Isječak programskog koda 4 prikazuje prikupljanje podataka iz baze ovisno o odabranoj statistici za prikaz. Svaka statistika predstavljena je putem modalnog prozora. Pri otvaranju modala statistike se prezentiraju pomoću kružnog grafikona. Moguće je prikazati broj entiteta prema kriteriju statistike ili taj broj zamijeniti postotkom. Radi bolje čitljivosti, grafikon će prikazati najviše deset entiteta te interaktivno je rotirajući. Za dublju analizu, korisnici mogu kliknuti na gumb „tablica“ koji će zamijeniti grafikon tablicom koja pruža detaljniji prikaz podataka i ne ograničava se na samo 10 entiteta, već su svi prikazani. Na raspolaganju je i posljednji gumb koji omogućuje skrivanje i prikazivanje elementa u koji se može unijeti broj računa na kojima će se provesti odabrana statistika. Izgled modala statistike najčešće korištenih stolova prikazan je slikom 12.

Isječak programskog koda 4. Prikupljanje podataka iz baze ovisno o odabranoj statistici

```
queryCafeBills.addListenerForSingleValueEvent(new ValueEventListener() {
  @Override
  public void onDataChange(@NonNull DataSnapshot cafeBillsSnapshot) {
      if(!cafeBillsSnapshot.exists()) {
          return;
      }
      tvStatisticsSize.setText(
              String.valueOf(cafeBillsSnapshot.getChildrenCount()) );
      for(DataSnapshot cafeBillSnapshot : cafeBillsSnapshot.getChildren()) {
          if(!cafeBillSnapshot.exists()) {
              return;
          }
          CafeBill cafeBill = cafeBillSnapshot.getValue(CafeBill.class);
          switch (queryType) {
              case QUERY_TYPE_EMPLOYEES: {
                  if(pieChartStatistic == null || pieChartStatistic.isEmpty()) {
                      pieChartStatistic.put(cafeBill.getCafeBillDelivererEmployee(), 1);
                  }
                  else {
                      if(pieChartStatistic.containsKey(cafeBill.getCafeBillDelivererEmployee
                                                                                  ())) {
                          pieChartStatistic.put(cafeBill.getCafeBillDelivererEmployee(),
                                                pieChartStatistic.get(
                                  cafeBill.getCafeBillDelivererEmployee()) + 1);
                      }
                      else {
                          pieChartStatistic.put(cafeBill.getCafeBillDelivererEmployee(), 1);
                      }
                  }
                  break;
              }
              case QUERY_TYPE_DRINKS: {
                  for(String cafeBillDrinkKey : cafeBill.getCafeBillDrinks().keySet()) {
                      CafeBillDrink cafeBillDrink =
                                    cafeBill.getCafeBillDrinks().get(cafeBillDrinkKey);
                      if(pieChartStatistic == null || pieChartStatistic.isEmpty()) {
                          pieChartStatistic.put(cafeBillDrink.getDrinkName(),
                                                cafeBillDrink.getDrinkAmount());
                      }
                      else {
                          if(pieChartStatistic.containsKey(cafeBillDrink.getDrinkName())) {
                              pieChartStatistic.put(cafeBillDrink.getDrinkName(),
                                                    pieChartStatistic.get(
                                      cafeBillDrink.getDrinkName()) +
                                                    cafeBillDrink.getDrinkAmount());
                          }
                          else {
                              pieChartStatistic.put(cafeBillDrink.getDrinkName(),
                                                    cafeBillDrink.getDrinkAmount());
                          }
                      }
                  }
                  break;
              }
              case QUERY_TYPE_TABLES: {
                  if(pieChartStatistic == null || pieChartStatistic.isEmpty()) {
                      pieChartStatistic.put(getResources().getString(R.string.statistics_tab
                                                                     les_table_number) +
                              AppConstValue.characterConstValue.CHARACTER_SPACING +
                                   String.valueOf(cafeBill.getCafeBillTableNumber()), 1);
                  }
                  else {
                      if(pieChartStatistic.containsKey(getResources().getString(R.string.sta
                                                       tistics_tables_table_number) +
                              AppConstValue.characterConstValue.CHARACTER_SPACING +
                                   String.valueOf(cafeBill.getCafeBillTableNumber()))) {
                          pieChartStatistic.put(getResources().getString(R.string.statistics
```

```
                                                                _tables_table_number) +
                                        AppConstValue.characterConstValue.CHARACTER_SPACING +
                        String.valueOf(cafeBill.getCafeBillTableNumber()), pieChartStatistic.get(
                                        getResources().getString(R.string.statistics_tables_table_
                                                                number) +
                                                AppConstValue.characterConstValue.CHARACTER_SPACIN
                                    G + String.valueOf(cafeBill.getCafeBillTableNumber())) + 1);
                            }
                            else {
                                pieChartStatistic.put(getResources().getString(R.string.statistics
                                                                _tables_table_number) +
                                        AppConstValue.characterConstValue.CHARACTER_SPACING +
                                         String.valueOf(cafeBill.getCafeBillTableNumber()), 1);
                            }
                        }
                        break;
                    }
                }
            }
            if(showChart) {
                makePieChart(false);
            }
            else {
                populateEmployeesTable();
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            ServerAlertDialog serverAlertDialog = new ServerAlertDialog(getActivity());
            serverAlertDialog.makeAlertDialog();
            String currentDateTime = simpleDateFormat.format(new Date());
            appError = new AppError(
                    mainViewModel.getOwnerCafeId().getValue(),
                    mainViewModel.getOwnerPhoneNumber().getValue(),
                    AppErrorMessage.Title.RETRIEVING_FIREBASE_DATA_FAILED_OWNER,
                    error.getMessage().toString(),
                    currentDateTime,
                    AppConstValue.errorSender.APP
            );
            appError.sendError(appError);
        }
    });
```



Slika 12. Modal statistike najčešće korištenih stolova objekta

Zadnji fragment prikazuje registrirane telefonske brojeve putem svoga sučelja. Korisnicima je omogućeno dodavanje novih telefonskih brojeva unosom broja i memorijske riječi koja će predstavljati taj broj kada bude potreban njegov prikaz. U element koji zahtjeva telefonski broj, aplikacija samostalno unosi pozivni broj ovisno o standardu države. Postojeće brojeve moguće je uređivati, isključiti, uključiti i odjaviti. Isključivanjem broja onemogućuje se daljnja prijava u aplikaciju tim brojem, ali broj se ne briše. U slučaju isključivanja ili odjave telefonskog broja dok je zaposlenik prijavljen, on će odmah biti odjavljen te će mu se okolnost objasniti putem modalnog prozora. Vlasnik objekta može promijeniti vlastiti telefonski broj. Za izmjenu broja, prvo je potrebno unijeti jednokratnu lozinku poslanu na njegovu email adresu. Uspješnim unosom lozinke, vlasnik može izmijeniti broj. Potrebno je napomenuti da se telefonski broj vlasnika može promijeniti samo jednom po prijavi, a za dodatne promjene potrebna je nova prijava s novim brojem.

Odjava iz aplikacije moguća je pomoću navigacijske trake, kao i slanje opisa pogreške administratorima aplikacije.

4.3. Web aplikacija

4.3.1. Registracija ugostiteljskog objekta

Korisnicima web aplikacije omogućeno je registrirati ugostiteljski objekt u sustav aplikacija. Registracija je podijeljena u tri koraka. Prvi korak odnosi se na unos osnovnih podataka vlasnika, obavezno je unijeti email i telefonski broj. U drugom koraku, korisnici popunjavaju informacije objekta, naziv i lokaciju te lokalizaciju prema kojoj će biti prikazani valuta, decimalni razdjelnik, pozivni broj, oblik datuma te druge vrijednosti. Završni korak je opcionalan, a omogućava unos menija pića putem Excel dokumenta – slika 13. Moguće je preuzeti predložak s primjerima ispravno ispunjenih podataka te ga prilagoditi željenim podacima. Sadržaj preuzetog dokumenta bit će ispunjen informacijama prilagođenim odabranom jeziku u zaglavlju aplikacije. Dokumenti su pohranjeni u Firebase servisu Cloud Storage.

Slika 13. Izgled završnog koraka registracije objekta

Kretanje između koraka je moguće, svi uneseni podaci ostat će spremljeni. Ovo je omogućeno korištenjem prilagođene komponente „*useMultiStepForm*“ kojoj se prosljeđuje lista React komponenti. Komponenta „*useMultiStepForm*“ vraća informacije o trenutnom koraku i omogućuje radnje koje ovise o određenom koraku na kojem se nalazimo – isječak

programskog koda 5. Svakoj komponenti prosljeđujemo objekt koji će biti ispunjen podacima relevantnim za tu komponentu.

Isječak programskog koda 5. Kod komponente koja omogućava kretanje kroz formu registracije

```
const useMultiStepForm = (steps) => {
  const [currentStepIndex, setCurrentStepIndex] = useState(0);

  function next() {
    setCurrentStepIndex((i) => {
      if (i > steps.length - 1) return i;
      return i + 1;
    });
  }

  function back() {
    setCurrentStepIndex((i) => {
      if (i <= 0) return i;
      return i - 1;
    });
  }

  function goTo(index) {
    setCurrentStepIndex(index);
  }

  return {
    currentStepIndex,
    step: steps[currentStepIndex],
    steps,
    isFirstStep: currentStepIndex === 0,
    isLastStep: currentStepIndex === steps.length - 1,
    goTo,
    next,
    back,
  };
};

export default useMultiStepForm;
```

4.3.2. Prijava u aplikaciju

Prijava u web aplikaciju odvija se na isti način kao i u Android aplikaciji, putem unosa telefonskog broja i jednokratne lozinke. Ako je drugi zaposlenik prijavljen u web aplikaciju putem tog broja, korisnik će biti obaviješten o toj situaciji. Ova mjera je uspostavljena kako bi se spriječilo da više zaposlenika koristi isti broj za prijavu, čime se osigurava da operacije nad narudžbama budu dostupne samo jednom zaposleniku.

4.3.3. Korisničko sučelje prijavljenog korisnika

Korisničko sučelje prijavljenog zaposlenika prikazuje tablicu narudžbi u tijeku. Aplikacija automatski dodaje nove narudžbe na kraj tablice putem slušatelja na čvoru podataka narudžbi u tijeku. Isječak programskog koda 6 prikazuje postavljanje slušatelja te dohvaćanje podataka

iz baze. Ažuriranje tablice izvršava se pretraživanjem narudžbi prema broju stola ili promjenom vrijednosti lokalizacije u bazi podataka.

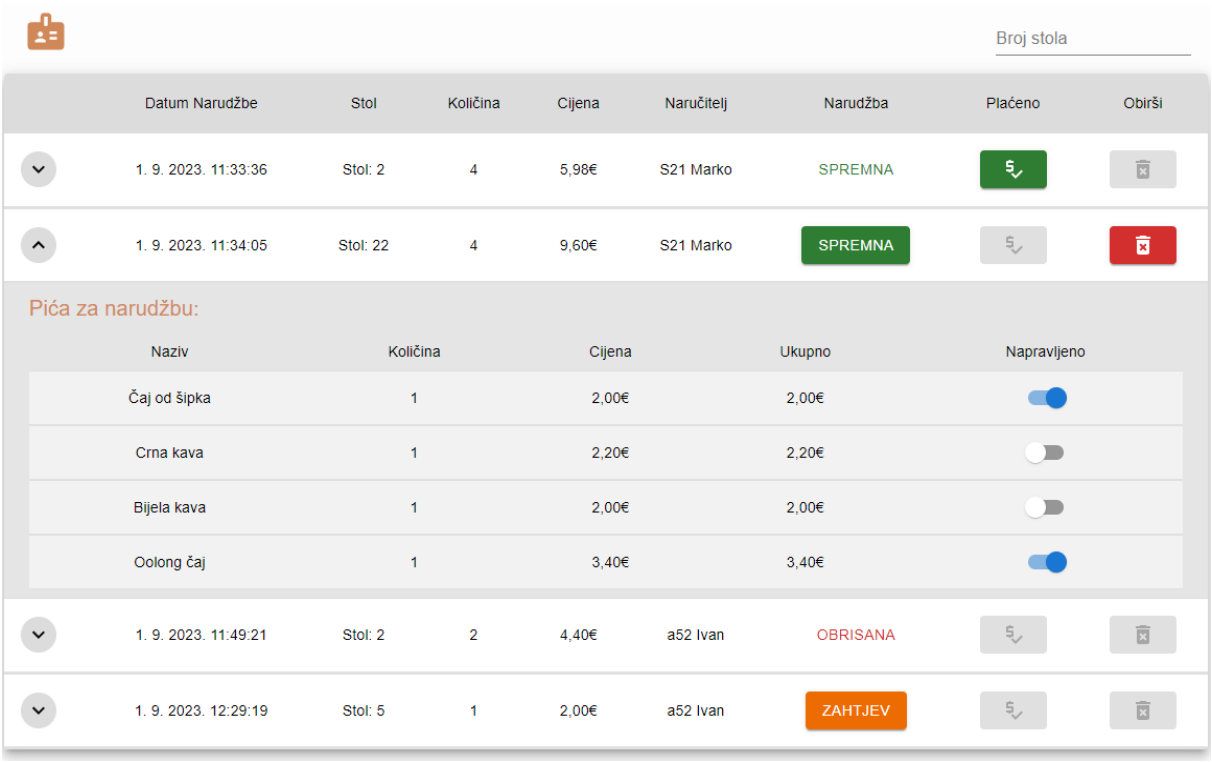
Isječak programskog koda 6. Postavljanje slušatelja na čvor podataka te njihovo dohvaćanje

```
useEffect(() => {
  async function fetchCurrentOrders() {
    try {
      const currentOrdersRef = ref(
        database,
        cafeCurrentOrdersPath(state.loggedPhone.cafeId)
      );

      const unsubscribe = onValue(currentOrdersRef, (snapshot) => {
        const currentOrders = [];
        snapshot.forEach((childSnapshot) => {
          const orderId = childSnapshot.key;
          const order = childSnapshot.val();
          if (
            ordersQuery &&
            parseInt(order.currentOrderTableNumber) ===
              Number(orderTableQuery)
          ) {
            const data = createOrderData(
              orderId,
              order.currentOrderDatetime,
              order.currentOrderTableNumber,
              order.currentOrderProductAmount,
              formatNumberLocalization(
                order.currentOrderTotalPrice,
                localizationData.decimalSeperator
              ),
              order.currentOrderMakerMemory,
              order.currentOrderDrinks,
              order.currentOrderStatus,
              order.currentOrderMakerEmployee
            );
            currentOrders.push(data);
          } else if (!ordersQuery) {
            const data = createOrderData(
              orderId,
              order.currentOrderDatetime,
              order.currentOrderTableNumber,
              order.currentOrderProductAmount,
              formatNumberLocalization(
                order.currentOrderTotalPrice,
                localizationData.decimalSeperator
              ),
              order.currentOrderMakerMemory,
              order.currentOrderDrinks,
              order.currentOrderStatus,
              order.currentOrderMakerEmployee
            );
            currentOrders.push(data);
          }
        });
        setRows(currentOrders);
      });

      return () => {
        unsubscribe();
      };
    } catch (error) {
      console.error(error);
    }
  }
  fetchCurrentOrders();
}, [orderTableQuery, localizationData]);
```

Svaku narudžbu moguće je proširiti kako bi se prikazala njena pića. Zaposlenik treba preuzeti željenu narudžbu kako bi mu se omogućilo izvršavanje akcija nad tom narudžbom. Moguće akcije su:

- Brisanje narudžbe – otvara se modal u kojem je potrebno unijeti razlog brisanja narudžbe. Konobar koji je inicijalno napravio tu narudžbu bit će obaviješten da je narudžba obrisana.
- Postavljanje narudžbe u stanje spremnosti – ova radnja također pokreće slanje obavijesti konobaru, u ovom slučaju da je narudžba za određeni stol spremna za isporuku.
- Naplaćivanje narudžbe – otvara se modal koji služi kao brzi kalkulator, moguće je unijeti primljenu vrijednost te vidjeti potreban iznos za povrat. Ovom radnjom narudžba se uklanja iz tablice.

Kao što je već spomenuto, konobar koji pravi narudžbe može poslati zahtjev da se narudžba ukloni. Odbijanje ili prihvaćanje tog zahtjeva mogu odraditi svi zaposlenici koji koriste web aplikaciju bez potrebe za prethodnim prihvaćanje te narudžbe. Ova funkcionalnost je osmišljena kako bi konobar što prije dobio odgovor na svoj zahtjev. Izgled tablice narudžbi u svim mogućim stanjima prikazano je slikom 14.



Slika 14. Tablica narudžbi

5. Zaključak

Za izradu aplikacija završnog rada korištene su brojne tehnologije, pri čemu se mogu istaknuti Java, XML, React biblioteka Javascripta te Firebase.

Svrha izrađenih aplikacija je pojednostaviti rad konobara u zahtjevnim radnim uvjetima, posebno u okruženjima gdje je potrebno brzo i efikasno upravljati velikim brojem narudžbi. Vlasnicima ugostiteljskih objekata omogućena je analiza narudžbi koja daje uvid u preferencije gostiju, kao što su najtraženija pića te lokacije objekta. Statistička analiza narudžbi može potencijalno pomoći u organizaciji nabavki, kao i uređenju prostora. Sve navedeno trebalo bi pomoći podizanju zadovoljstva gostiju ugostiteljskih objektata.

Izrada programskog rješenja znatno je pridonijela dubljem razumijevanju poslovne logike povezane s ovim specifičnim područjem. Naročito u dijelu kreiranja robusnog rješenja problema razvoja velikog sustava, koje zahtjeva logičko i kritičko razmišljanje o svakom segmentu aplikacija. Kroz zadatak završnog rada proširio sam znanje u području svih koraka kreiranja korisnikovog sučelja kao i tehnologija koje sam koristio u izradi. Tu mi je uvelike pomogao i predmet „Projektiranje informacijskih sustava“.

Sljedeći korak bio bi razvoj mobilne aplikacije za iOS operacijski sustav. Potrebno je pratiti potrebe korisnika kao i Firebase servise kako bi se rad aplikacija sustavno nadograđivao i dodatno poboljšao.

Popis literature

[1] Android. Pristupljeno: 16. kolovoza 2023. [Online]. Dostupno na: https://developer.android.com/studio/intro

[2] Sierra Kathy, Bert Bates: Head First Java 2nd Edition, 2005.

[3] Mozilla. Pristupljeno: 17. kolovoza 2023. [Online]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction

[4] Android. Pristupljeno: 16. kolovoza 2023. [Online]. Dostupno na: https://developer.android.com/develop/ui/views/layout/declaring-layout

[5] React. Pristupljeno: 18. kolovoza 2023. [Online]. Dostupno na: https://react.dev/learn/writing-markup-with-jsx

[6] Babel. Pristupljeno: 18. kolovoza 2023. [Online]. Dostupno na: https://babeljs.io/docs/

[7] Educative. Pristupljeno: 19. kolovoza 2023. [Online]. Dostupno na: https://www.educative.io/answers/what-is-firebase

[8] JSON. Pristupljeno 19. kolovoza. 2023. [Online]. Dostupno na: https://www.json.org/json-en.html

[9] Firebase. Pristupljeno 19. kolovoza 2023. [Online]. Dostupno na: https://firebase.google.com/docs/storage

[10] Firebase. Pristupljeno 19. kolovoza 2023. [Online]. Dostupno na: https://firebase.google.com/docs/perf-mon

[11] Firebase. Pristupljeno 19. kolovoza. 2023. [Online]. Dostupno na: https://firebase.google.com/docs/crashlytics

[12] Android. Pristupljeno 19. kolovoza 2023. [Online]. Dostupno na: https://developer.android.com/guide/components/activities/intro-activities

[13] Android. Pristupljeno: 16. kolovoza 2023. [Online]. Dostupno na: https://android-developers.googleblog.com/2014/12/new-code-samples-for-lollipop.html

[14] Doug Stevenson, Medium (2018). Pristupljeno: 18. kolovoza 2023. [Online]. Dostupno na: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

[15] Firebase. Pristupljeno: 19. kolovoza. 2023. [Online]. Dostupno na: https://firebase.google.com/docs/database/android/start

[16] Firebase. Pristupljeno: 19. kolovoza. 2023. [Online]. Dostupno na: https://firebase.google.com/docs/database/web/start

Popis slika

Popis programskih kodova

Veleučilište u Virovitici

OBRAZAC 5

IZJAVA O AUTORSTVU

Ja, Petar Ivačilušić

izjavljujem da sam autor/ica završnog/diplomskog rada pod nazivom

Mobilna Android aplikacija i web aplikacija za organizaciju rasporeda u ugostiteljskim objektima skupine „Borovi"

Svojim vlastoručnim potpisom jamčim sljedeće:
- da je predani završni/diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija,
- da su radovi i mišljenja drugih autora/ica, koje sam u svom radu koristio/la, jasno navedeni i označeni u tekstu te u popisu literature,
- da sam u radu poštivao/la pravila znanstvenog i akademskog rada.

Potpis studenta/ice

Petar Ivačilušić

Veleučilište u Virovitici

OBRAZAC 6

ODOBRENJE ZA OBJAVLJIVANJE ZAVRŠNOG/DIPLOMSKOG RADA U DIGITALNOM REPOZITORIJU

Ja Pero Ykočilušić

dajem odobrenje za objavljivanje mog autorskog završnog/diplomskog rada u javno dostupnom digitalnom repozitoriju Veleučilišta u Virovitici sadržanom u Dabar (Digitalni akademski arhivi i repozitoriji) te u javnoj internetskoj bazi završnih radova Nacionalne i sveučilišne knjižnice bez vremenskog ograničenja i novčane nadoknade, a u skladu s odredbama članka 58. stavka 5., odnosno članka 59. stavka 4. Zakona o visokom obrazovanju i znanstvenoj djelatnosti (NN 119/22).
Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog završnog/diplomskog rada. Ovom izjavom, kao autor navedenog rada dajem odobrenje i da se moj rad, bez naknade, trajno javno objavi i besplatno učini dostupnim na sljedeći način:

a) Rad u otvorenom pristupu
b) Rad dostupan nakon: _____ (upisati datum nakon kojeg želite da rad bude dostupan)
c) Pristup svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Pristup korisnicima matične ustanove
e) Rad nije dostupan (*u slučaju potrebe dodatnog ograničavanja pristupa Vašem završnom/diplomskom radu, podnosi se pisani obrazloženi zahtjev*).

Potpis studenta/ice

Pero Ykočilušić

U Virovitici, 8. 9. 2023.