Android aplikacija za digitalizaciju narudžbi i jelovnika u ugostiteljskim objektima

Biondić, Enio

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

ANDROID APLIKACIJA ZA DIGITALIZACIJU NARUDŽBI I JELOVNIKA U UGOSTITELJSKIM OBJEKTIMA

Diplomski rad

Enio Biondić

Osijek, 2021.

SADRŽAJ

1. UVOD

Prateći procese digitalizacije koji se događaju u svakodnevnom životu može se primijetiti kako su određene sfere života ostale nepromijenjene godinama, a poneke i desetljećima. Iako nam tehnologija i znanje omogućuju unapređenje, na pojedinim područjima izostaju promjene. Posebno je specifičan način na koji se u ugostiteljskim obrtima naručuje hrana.

Pitamo li naše roditelje, bake ili djedove kako su oni naručivali hranu u restoranima za vrijeme njihove mladosti primijetit ćemo da se taj postupak gotovo ni u čemu ne razlikuje od današnjice. Vodeći se ovom problematikom cilj završnog rada je pokazati mogućnosti digitalizacije ugostiteljstva. Stoga je izrađena platforma *Menuely* kojoj je svrha digitalizacija jelovnika i narudžbi u ugostiteljskim obrtima.

Ovaj će rad biti podijeljen u četiri poglavlja. Prvo će poglavlje opisati razloge realizacije i mogućnosti primjene, a drugo će dati uvid u primijenjene tehnologije za razvoj aplikacije *Menuely*. Iduće poglavlje opisat će proces realizacije *Android* aplikacije dok će posljednje dati uvid u izgled i način funkcioniranja iste.

1.1. Zadatak diplomskog rada

Zadatak ovog diplomskog rada je izraditi aplikaciju za *Android* mobilne uređaje koja će kroz jednostavno sučelje korisnicima omogućiti korištenje digitalne verzije jelovnika i vršiti narudžbe. Aplikacija treba imati zasebna sučelja ovisno o tome je li korisnik u aplikaciji prijavljen kao ugostiteljski obrt ili kao gost ugostiteljskog obrta.

Kada je korisnik prijavljen kao ugostiteljski obrt pružaju se mogućnosti kreiranja i uređivanja jelovnika, uređivanja profila restorana te zapošljavanja konobara. Ako je korisnik u aplikaciji prijavljen kao gost ugostiteljskog obrta pružaju se mogućnosti pretraživanja restorana po imenu i adresi, skeniranja *QR* koda radi otvaranja digitalnog jelovnika, pregledavanja jelovnika ugostiteljskih obrta, stvaranja narudžbi, pregleda povijesti naručivanja, uređivanja vlastitog profila i zapošljavanja u ugostiteljskom obrtu.

2. RAZLOZI REALIZACIJE I MOGUĆNOSTI PRIMJENE APLIKACIJE

Glavni razlog realizacije platforme *Menuely* je izrada digitalnog rješenja za pojedine procese u ugostiteljstvu čije je unapređenje zanemarivano godinama. Kao potpuno digitalno rješenje ova platforma pruža brojne pogodnosti u pogledu iskoristivosti vremena i osoblja te podiže opću kvalitetu usluge. Svoju primjenu, osim u restoranima, platforma može pronaći i u kafićima i klubovima.

Zbog funkcionalnosti koje su dostupne korisnicima kroz aplikaciju, ugostiteljski obrti nemaju potrebu trošiti financijska sredstva za tiskanje i održavanje jelovnika. Jednom kreiran jelovnik dostupan je doživotno putem aplikacije te ga je moguće uređivati neograničen broj puta. Vrijeme čekanja naručenog proizvoda znatno se smanjuje jer uslužno osoblje ne mora obilaziti stolove radi prikupljanja narudžbi.

Ovo su samo neke od prednosti koje aplikacija pruža te je daljnjim razvojem i usavršavanjem moguće ostvariti i druge prednosti.

2.1. Pregled sličnih rješenja

Aplikacija *Menuely* izrađena je po uzoru na već postojeće aplikacije kao što su [1] *Glovo* i [2] *Wolt*. Glavna je prednost u usporedbi s navedenim aplikacijama ta što je korisnik u mogućnosti izraditi narudžbu unutar ugostiteljskog objekta.

Aplikacija *Glovo* namijenjena je za dostavu gotovo bilo kojih proizvoda unutar grada te na jednostavan način povezuje korisnike i dostavljače.

Wolt je aplikacija također za dostavu proizvoda sličnih principa kao i *Glovo* ,ali glavna je razlika što je priroda proizvoda koji se mogu naručiti preko *Wolta* ograničena na prehrambene proizvode iz ugostiteljskih objekata.

3. PRIMIJENJENE TEHNOLOGIJE RAZVOJA

Za razvoj *Android* aplikacije od samog početka pa sve do finalnog proizvoda korišteni su razni alati i tehnologije. Ovo će poglavlje opisati svaki od korištenih alata te razlog njihove uporabe. Proces izrade prilagođen je propisanim načelima iz službene *Android* dokumentacije te su korištene najbolje raspoložive metode razvoja.

3.1. *Kotlin* programski jezik

Kotlin [3] je programski jezik novije generacije razvijen od tvrtke *JetBrains*. Javno je dostupan pod *Apache 2* licencom, a njegov je razvoj pod vodstvom *Kotlin Foundation*. Glavne su odlike interoperabilnost, sigurnost, jednostavnost i dobra integracija s drugim razvojnim alatima.

Zbog svoje sintakse i interoperabilnosti postao je višeplatformni jezik, što znači da se u njemu mogu razvijati aplikacije kako za mobilne uređaje tako i za *web* i poslužiteljske aplikacije. Godine 2017. na *Google I/O* sajmu proglašen je podržanim jezikom za razvoj *Android* mobilnih aplikacija, a dvije godine nakon postaje i službenim *Android* programskim jezikom. *Kotlin* je objektno orijentiran [4] programski jezik, a uz to sa sobom nosi brojne beneficije koje najviše pogoduju razvojnim programerima.

Glavna prednost *Kotlina* u usporedbi s *Javom* [5], donedavnim službenim jezikom *Androida*, jest rukovanje s *null pointer* iznimkama. Ima ugrađene zaštite koje prepoznaju kada se u kodu pozivaju objekti ili varijable koje nisu inicijalizirane i unaprijed upozoravaju programera da je to mjesto eventualnog rušenja aplikacije. Programer je dužan taj dio koda zaštiti dodatnim blokovima.

Slika 3.1. *Logo programskog jezika Kotlin*

3.2. *Android Studio*

Android Studio [6] službeno je razvojno okruženje od *Googlea* namijenjeno prvenstveno za izradu *Android* aplikacija za mobilne uređaje, tablete, pametne satove i televizore. Zasnovan je na *JetBrainsovom InteliJ* razvojnom okruženju, a dostupan je za *Windows*, *macOs* i *Linux* operativne sisteme.

Unutar *Android Studio* okruženja nalaze se brojni alati koji razvoj aplikacija čine lakšim. Podržani jezici za pisanje aplikacija su *Java* i *Kotlin*. Izrada mobilnog sučelja putem ovog razvojnog okruženja popraćena je automatski generiranim pretpregledom kako bi razvojnim programerima bilo lakše praćenje promjena. Podržava pokretanje aplikacija na virtualnim uređajima čije specifikacije je moguće samostalno uređivati kako bi testirali kompatibilnost aplikacije za različite uređaje. Unutar razvojnog okruženja nalazi se *debuger,* njegova svrha je pomoć kod uklanjanja logičkih grešaka koje su nastale tijekom razvoja. Omogućuje nam da se izvođenje mobilne aplikacije može zaustaviti na bilo kojoj liniji koda i u svakom trenutku pročitati sadržaj varijabli i objekata.

Slika 3.1. *Ikona Android Studia*

3.3. *Postman*

Postman je kolaboracijska platforma razvojnih programera poslužiteljskih i klijentskih aplikacija. Svrha mu je ubrzati testiranje *API-a* [7] (engl. *Application Programming Interface - aplikacijsko programsko sučelje*) i olakšati usklađivanje klijentske i poslužiteljske strane. Zbog mogućnosti poput kreiranja radnih grupa, dokumentacije i timova *Postman* je najpopularnija platforma svoje vrste. Putem jednostavnog sučelja mogu se kreirati i poslati sve vrste *HTTP* (engl. *Hypertext Transfer Protocol*) [8] zahtjeva na određeni poslužitelj te u stvarnom vremenu dobiti odgovor.

Slika 3.3. *Ikona Postmana*

3.4. *Jetpack Compose*

Jetpack Compose [9] je *Androidov* skup alata razvijen od strane *Google-a* za stvaranje modernog i nativnog sučelja uz manje koda, ali drugačiji tok učenja (engl. *learning curve*). Dio je poznatog *Jetpack* paketa kojeg je prema službenoj *Android* dokumentaciji najbolje koristiti za razvoj mobilnih aplikacija.

Usporedo s dosadašnjim načinom stvaranja korisničkog sučelja pokreće novu eru u pogledu toka podataka. Donedavni konvencionalni pristup razvoja aplikacija nije imao jasno definiran tok podataka, odnosno, izostankom arhitekture bilo je moguće napraviti grananja tokova podatka i tako dodatno zakomplicirati razvoj aplikacije. *Jetpack Compose* donosi revolucionarne promjene tako što je tok podataka od izvora do prikaza na sučelju jednosmjeran te pomaže da se prezentacijski sloj aplikacije odvoji od sloja za obradu podataka.

Glavna gradivna komponenta sučelja su *Composable* funkcije. Ovaj tip funkcija kroz parametar prima trenutnu kompoziciju podataka nekog modela i na temelju tih podataka izvršava stvaranje jednog dijela sučelja. Kako bi prikazani podaci na sučelju bili ažurni potrebno je prilikom promjene kompozicije podataka u modelu iznova pozvati funkcije. Taj proces se naziva rekompozicija. C*omposable* funkcije imaju mogućnost rukovanja stanjima (engl. *state*) varijabli te zbog toga rekompozicija postaje automatizirani proces. Drugim riječima, dobro napisane *Composable* funkcije pratit će svaku promjenu kompozicije podataka i brinut se o njihovom pravovremenom prikazu.

Slika 3.4. *Logo Jetpack Composea*

3.5. *Retrofit* i *OkHttp*

Retrofit [10] i *OkHttp* [11] su dvije najpoznatije i najkorištenije biblioteke kada je riječ o mrežnom sloju *Android* aplikacija. Njihova uloga je oblikovanje, slanje i primanje paketa podataka putem interneta.

Kroz anotacije koje su dostupne u *Retrofit* biblioteci, moguće je kreirati sve vrste *HTTP* zahtjeva.

```
@GET("group/{id}/users")

Call<List<User>> groupList(@Path("id") int groupId);
```

Slika 3.5. *Primjer GET HTTP zahtjeva*

Na slici 3.5. vidljiv je primjer funkcije koja je popraćena *GET* anotacijom. Prilikom poziva ove funkcije kreirat će se *HTTP GET* zahtjev s proslijeđenim parametrom za varijablu *„id“* koji će biti poslan na poslužiteljev *endpoint „group/{id}/users“*. Na ovaj način pišu se sve funkcije za komunikaciju aplikacije s poslužiteljem, jedina je razlika *endpoint* na koji se šalju.

OkHttp je biblioteka koja služi za rukovanje podacima koji se šalju odnosno primaju s udaljenog poslužitelja. Najčešće se integrira u mrežni sloj aplikacije zajedno s *Retrofitom*, a svrha biblioteke je pretvaranje *Java/Kotlin* objekata u *JSON* (engl. *JavaScript Object Notation*) [12] objekte i obratno. Ova je konverzija nužna jer se komunikacija putem interneta s udaljenim poslužiteljem najčešće vrši putem *JSON* objekata, dok se unutar *Android* aplikacije rukuje s *Java* odnosno *Kotlin* objektima.

Dodatno *OkHttp* pruža mogućnosti poput presretanja *HTTP* zahtjeva te je moguća njihova dodatna obrada prije slanja. Ova funkcionalnost je iznimno korisna kada je potrebno u zaglavlje zahtjeva ubaciti autorizacijski token ili ga osvježiti.

3.6. *Mutable state*

Mutable state [13] varijable (varijable promjenjivog stanja) pojavljuju se kao sastavni dio *Jetpack Composea*. Njihovo je ponašanje slično ponašanju dosadašnje *Live Datae*, ali se one koriste zbog bolje sinergije s *Jetpack Composeom*.

Mutable state varijable služe kako bi *Composable* funkcije sačuvale stanje varijabli prilikom rekompozicije te kako bi se osvježio trenutni prikaz na sučelju. Ako su *Composable* funkcije dobro napisane, svaka promjena *mutable state* varijabli bit će popraćena promjenom podataka na sučelju.

```
val menuId : MutableState<Int> = mutableStateOf(0)
```

Slika 3.6. *Primjer inicijalizacije mutable state varijable*

Na slici 3.6. vidljiv je primjer stvaranja *mutable state* varijable tipa *Int*. Ova varijabla inicijalno je postavljena na vrijednost 0 (nula) što nije nužno, ali je praksa pokazala da je bolje imati inicijaliziranu varijablu nego *null* vrijednost. Ako bi se varijabla iz primjera u *Composable* funkciji povezala s nekom komponentom i napisala logika što ta varijabla predstavlja u prikazu, svaka promjena dane varijable ažurirala bi prikaz na sučelju.

```
menuId.value = 7
```

Slika 3.7. *Postavljanje vrijednosti mutable state varijable*

Slika 3.7. prikazuje način ažuriranja vrijednosti *mutable state* varijable, u konkretnom primjeru varijabla *menuId* poprimit će vrijednost 7.

3.7. *Dependency Injection (Hilt)*

Dependency injection [14] je postupak u objektno orijentiranom programiranju kada se za kreiranje nekog objekta u njegov konstruktor ubacuju instance objekata o kojima on ovisi.
Na primjer: objekt *Automobil* ovisan je o objektu *Motor*. Kada se kreira objekt *Automobil* te se u konstruktor automobila ubaci instanca kreiranog objekta *Motor* napravljen je *dependency injection*.

Prilikom razvoja mobilnih aplikacija postoji velik broj slučajeva kada jedan objekt ovisi o nekoliko različitih objekata te ti objekti ovise o još nekolicini drugih objekata. Uzmemo li u obzir da je takav kompleksan objekt potreban na više mjesta u aplikaciji pojavljuje se situacija pojave gomile ponavljajućeg koda. Ovo je vrlo kompleksan problem kojeg *dependency injection* biblioteke rješavaju na elegantan način.

Hilt [15] je biblioteka razvijena od strane *Google-a* koja pomaže u rješavanju *dependency injection* problematike. Način funkcioniranja *Hilta* je da se na jednom mjestu u aplikaciji definiraju pravila kreiranja nekog objekta i svih objekata o kojima on i ostali objekti ovise. Kada je kasnije u aplikaciji potreban neki od tih objekata može ga se pozvati, a *Hilt* se u pozadini brine o kreiranju neophodnih objekata o kojima taj objekt ovisi. Također je moguće kod pisanja pravila kreiranja nekog objekta definirati ga kao *Singleton* te će se kod zahtijevanja istog dodijeliti uvijek ista instanca objekta.

Ovaj je način instanciranja vrlo koristan posebno kod mrežnih poziva jer uvedena arhitektura u aplikaciji stvara nekoliko slojeva kroz koje se podatak mora propagirati. Ti su slojevi najčešće objekti koje je prethodno potrebno inicijalizirati. Kada bi za svaki mrežni poziv inicijalizirali sve neophodne objekte to bi bio mukotrpan posao i zbog toga se primjenjuju biblioteke, kao što je *Hilt*, koje nam omogućuju lakše rukovanje *dependency injectionom*.

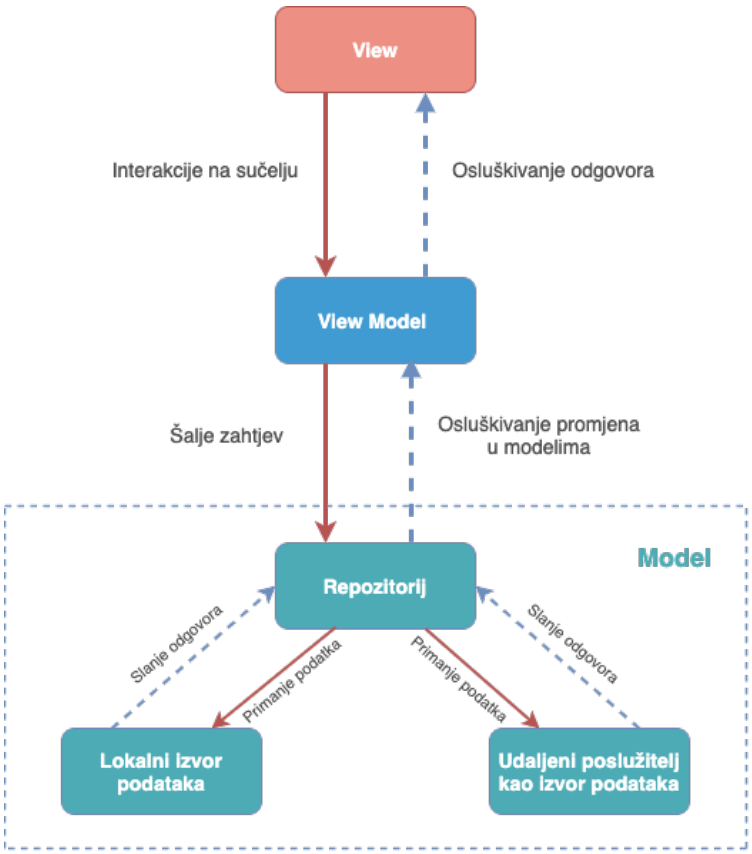
4. REALIZACIJA APLIKACIJE

U ovom poglavlju bit će opisani procesi izrade aplikacije *Menuely*. Izrada je podijeljena u više odvojenih etapa u kojima su stvarani slojevi aplikacije te funkcionalnosti za njihovu interakciju.

4.1. Strukturiranje *Android* projekta

Kvalitetna struktura projekta je stavka koja se ne smije zanemariti kod izrade aplikacija bilo koje veličine. Ovisno o njoj aplikacija će biti više ili manje skalabilna te otvorena za uvođenje dodatnih funkcionalnosti. Između ostalog, dobra struktura omogućit će novim razvojnim programerima lakše snalaženje i bržu prilagodbu na projekt te samim time povećati njihovu produktivnost.

Prilikom izrade aplikacije uveden je *MVVM* (engl. *Model-View–View Model* - *model-pogled-pogledmodel*) [16] strukturni obrazac zato što uvodi jasno odvajanje prezentacijskog od datotečnih slojeva.



Slika 4.1. *MVVM strukturni obrazac*

Na slici 4.1. prikazan je način strukturiranja projekta prema *MVVM* obrascu te tok podataka od izvora do prikaza na sučelju. Ovaj strukturni obrazac dijeli aplikaciju u tri razine od kojih je samo jedna zaslužna za prezentaciju podataka i interakciju s korisnikom dok su druge dvije razine korištene za pozadinske procese rukovanja s podacima.

View je razina aplikacije koja se bavi prikazom dobivenih podataka na sučelje. Osim toga svrha *View* sloja je pratiti korisnikove interakcije na sučelju te ovisno o njima slati *View Modelu* zahtjeve za podatke koji su potrebni za prikazivanje.

View Model je posrednički sloj između *viewa* i modela. Njegova je zadaća podatke dobivene iz modela prilagoditi najboljem formatu za prikaz korisniku. Najčešće se *mutable state* varijable i *live data* varijable nalaze u *View* modelima jer uz pomoć njih sučelje uvijek prati promjenu podataka.

Model je najdublji sloj aplikacije. Svrha modela je prikupiti podatke s izvora podataka. Izvor podataka može biti ili udaljeni poslužitelj ili lokalna baza podataka. Sastavni dio svakog modela su repozitoriji jer se u njima pišu funkcije za pristupanje udaljenom poslužitelju odnosno lokalnoj bazi.

4.2. Mrežni modul

Mrežni modul (engl. *Network Module*) aplikacije zadužen je za uspostavu umrežavanja aplikacije (engl. *networking*). Sadrži pravila za instanciranje biblioteka koje rukuju mrežnim zahtjevima kao što su *Retrofit* i *OkHttp*. Osim ove dvije spomenute biblioteke, u mrežnom modulu napisana su pravila za instanciranje drugih neophodnih stvari vezanih za umrežavanje aplikacije.

Prilikom pokretanja aplikacije *Hilt* instancira modul te on prati životni ciklus (engl. *lifecycle*) aplikacije što bi značilo da mrežni modul živi koliko i sama aplikacija.

```
@Provides
@Singleton
fun provideRetrofit(okHttpClient: OkHttpClient): Retrofit =
    Retrofit.Builder()
        .baseUrl(BASE_URL)
        .client(okHttpClient)
        .addConverterFactory(GsonConverterFactory.create())
        .build()
```

Slika 4.2. *Funkcija pravila za instanciranje Retrofita*

Na slici 4.2. dan je primjer pisanja funkcije pravila za instanciranje biblioteke *Retrofit* u mrežnom modulu. Vidljivo je kako se ovoj funkciji kroz parametar predaje instanca *OkHttp* biblioteke što bi značilo da će prilikom instanciranja *Retrofita Hilt* pronaći funkciju za instanciranje *OkHttpa* te instancu predati danoj funkciji.

Kako bi cijeli mrežni modul bio funkcionalan potrebno je napisati još nekolicinu neophodnih funkcija za umrežavanje aplikacije.

4.3. Modul baze podataka

Lokalna baza podataka neophodni je element svake aplikacije kojoj je potrebno da nakon gašenja određeni podaci ostanu sačuvani. Konkretno u aplikaciji *Menuely* baza je korištena kako bi se sačuvali podaci o prijavljenom korisniku kao što su: ime, prezime, e-*mail*, hiperveza profilne i naslovne fotografije. Osim podataka o korisniku nužno je spremiti i autorizacijske tokene s kojima se potpisuju mrežni zahtjevi. Izvedba baze podataka obavljena je korištenjem biblioteke *Room* koja na uređaju stvara lokalnu *SQL* (engl. *Structured Query Language*) [17] bazu.

Modul baze podataka sadrži funkcije pravila za kreiranje *Room* baze podataka kao i funkcije za kreiranje *DAO* (engl. *Data Access Object*) sučelja (engl. *interface*) za pojedine tablice iz baze. *DAO* sučelja sadržavaju funkcije za *CRUD* (engl. *Create–Read–Update-Delete*) operacije nad pojedinim entitetima iz baze.

```
@InstallIn(SingletonComponent::class)
@Module
object DatabaseModule {
    @Provides
    @Singleton
    fun provideDatabase(@ApplicationContext context: Context): MenuelyDatabase {
        return Room.databaseBuilder(context, MenuelyDatabase::class.java,  name: "menuely_database")
            .fallbackToDestructiveMigration()
            .allowMainThreadQueries()
            .build()
    }
```

Slika 4.3. *Funkcija pravila za instanciranje lokalne baze podataka*

Na slici 4.3. dan je primjer funkcije pravila za instanciranje lokalne baze podataka iz modula baze podataka. Isto kao mrežni i ovaj se modul instancira prilikom pokretanja aplikacije te dijeli njezin životni ciklus.

```
@Query( value: "SELECT * from user_table")
fun getUser(): UserTableModel?
```

Slika 4.4. *Primjer funkcije iz DAO sučelja.*

Na slici 4.4. vidljiv je primjer funkcije iz *DAO* sučelja za dohvaćanje korisnika iz lokalne baze. Može se primjetiti kako funkcija ima anotaciju *Query* s dodijeljenim odgovarajućim *SQL* upitom na bazu.

4.4. Modul repozitorija

Repozitoriji su mjesta u aplikaciji u kojima se nalaze prikupljeni podaci s određenih izvora kao i funkcionalnosti za njihovo dohvaćanje. Moguće je kombiniranje više izvora podataka u repozitoriju, bitno je da iduća apstrakcija odnosno *View Model* ne mora voditi računa o tome kako će se ti podaci dohvatiti. Dobra je praksa kreirati više repozitorija ovisno o vrsti podataka koje će dohvaćati. Ovime se postiže preglednost i lakše snalaženje u projektu.

Moguće je da određeni repozitorij mora imati pristup nekom drugom repozitoriju, tada se mora dodatno voditi računa o slučajnom stvaranju kružne ovisnosti (engl. *circular dependency).* Kružna ovisnost se pojavljuje ako istovremeno dva repozitorija ovise jedan o drugom. Modul repozitorija sadrži funkcije pravila za instanciranje svih repozitorija aplikacije, instancira prilikom pokretanja aplikacije te dijeli njezin životni ciklus

```
@Provides
@Singleton
fun provideRestaurantRepo(menuelyApi: MenuelyApi, responseHandler: ResponseHandler) =
    RestaurantRepo(menuelyApi, responseHandler)
```

Slika 4.5. *Primjer funkcije pravila stvaranja repozitorija*

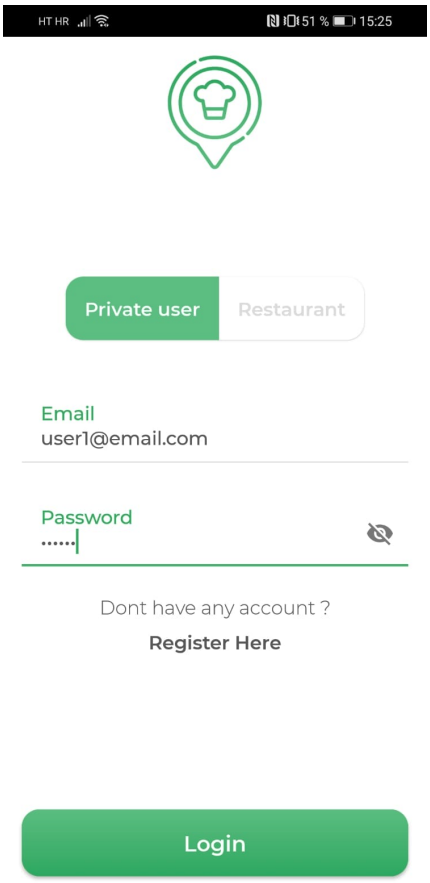
Na slici 4.5. prikazan je primjer funkcije pravila stvaranja repozitorija iz modula repozitorija.

5. IZGLED I NAČIN FUNKCIONIRANJA APLIKACIJE

U ovom poglavlju bit će opisane funkcionalnosti aplikacije *Menuely,* te će svaka od njih biti popraćena odgovarajućim slikovnim prikazima.

5.1. Prijava u aplikaciju

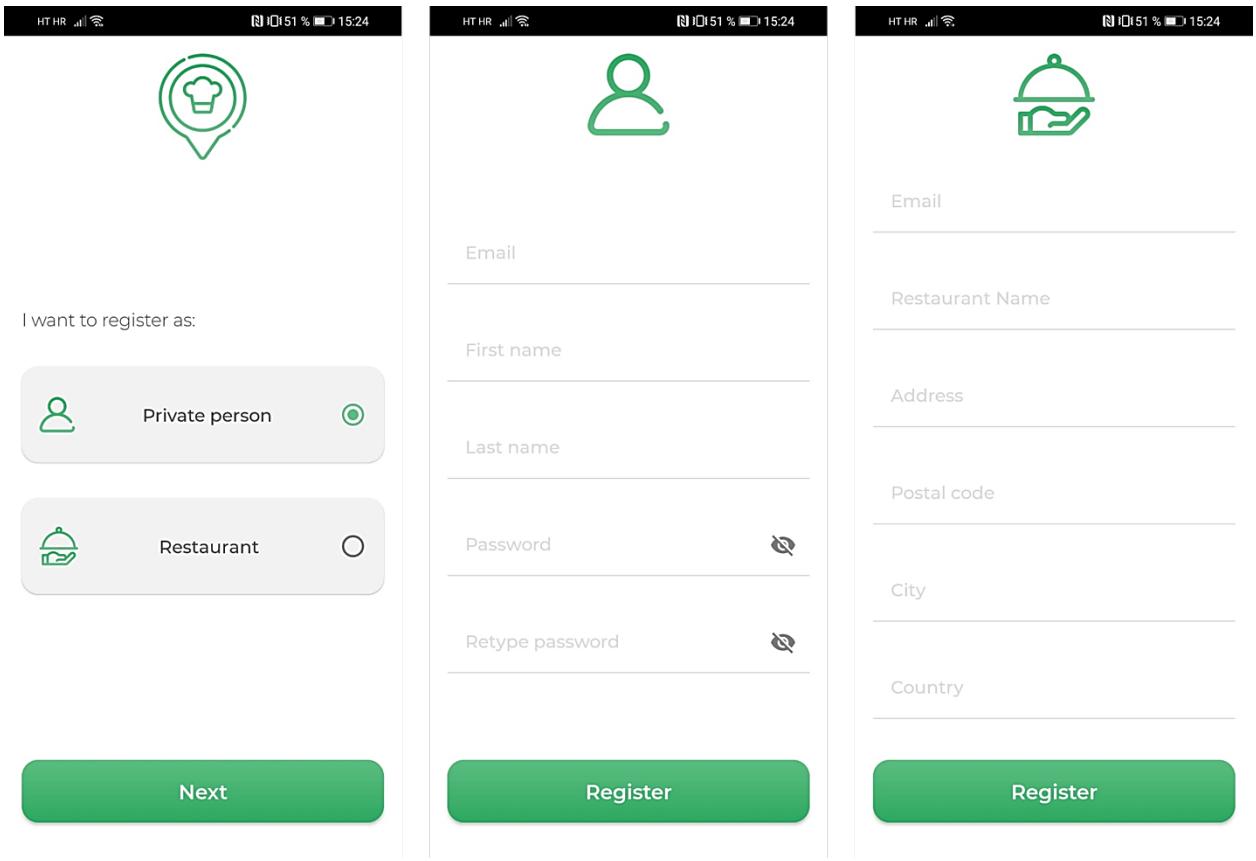
Na slici 5.1. prikazan je ekran prijave u aplikaciju. Korisnik se može prijaviti sa svojim osobnim ili s računom ugostiteljskog obrta ako ga posjeduje. Kako bi prijava bila uspješna potrebno je upisati odgovarajuću *e-mail* adresu i zaporku računa s kojim se prijavljuje.



Slika 5.1. *Prijava u aplikaciju*

5.2. Izrada korisničkog računa

Ukoliko korisnik nema izrađen račun potrebno ga je izraditi. Putem korisničkog sučelja moguće je kreirati račun kako privatne osobe tako i ugostiteljskog obrta. Kada se odlučimo za kreiranje korisničkog računa aplikacija postavlja upit kakav račun želimo izraditi što je vidljivo na slici 5.2. lijevo.



Slika 5.2. *Odabir vrste korisničkog računa prilikom njegove izrade i forme*

Nakon odabrane vrste korisničkog računa potrebno je popuniti formu s traženim podacima. Slika 5.2. sredina prikazan je primjer forme za izradu računa privatne osobe dok slika 5.2. desno forme za izradu računa ugostiteljskog obrta.

Kada je forma uspješno popunjena i predana, na *e-mail* adresu korisnika doći će poruka za verifikaciju računa. Račun je potrebno verificirati kako bi se korisnik kasnije mogao prijaviti u aplikaciju s istim.

5.3. Skeniranje *QR* koda

Kada se korisnik u aplikaciju prijavi putem računa privatne osobe, otvara se ekran *QR* skenera (slika 5.4.). Ovaj skener koristi se kako bi korisnik mogao otvoriti jelovnik restorana u kojem se nalazi te kreirati svoju narudžbu.

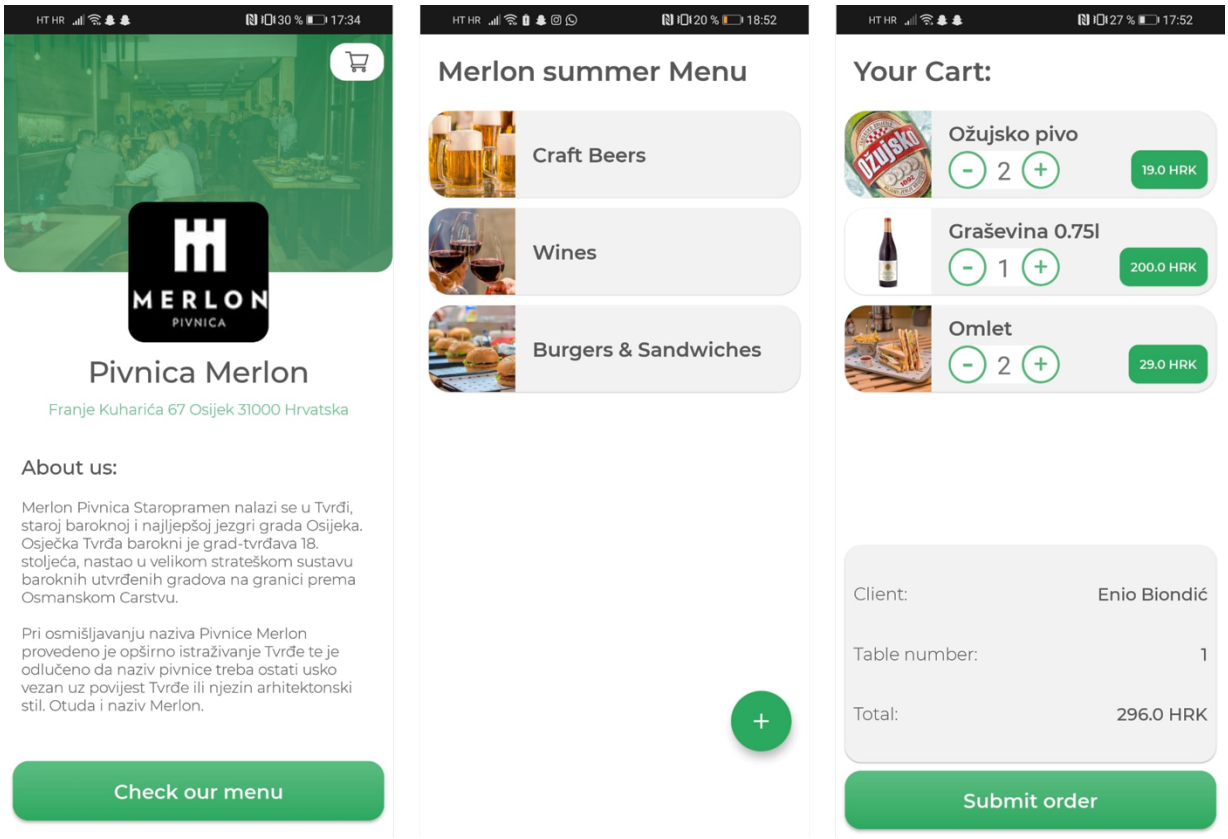
Slika 5.4. *Ekran QR skenera*

Ukoliko je skenirani *QR* kod kreiran od platforme *Menuely* aplikacija će korisnika preusmjeriti na profil restorana te za njega kreirati praznu košaricu, u suprotnom će korisnik biti obaviješten kako je skenirani kod nepoznat aplikaciji.

5.4. Kreiranje narudžbe

Kada se nakon skeniranja koda otvori profil restorana (slika 5.5. lijevo) pruža se mogućnost pregleda jelovnika te dodavanje proizvoda u kreiranu košaricu. Proizvodi se u košaricu dodaju tako da se klikom na proizvod odabere opcija za dodavanje u košaricu.

Predaja narudžbe restoranu odvija se s ekrana košarice (slika 5.5. desno) odabirom opcije *submit order.*

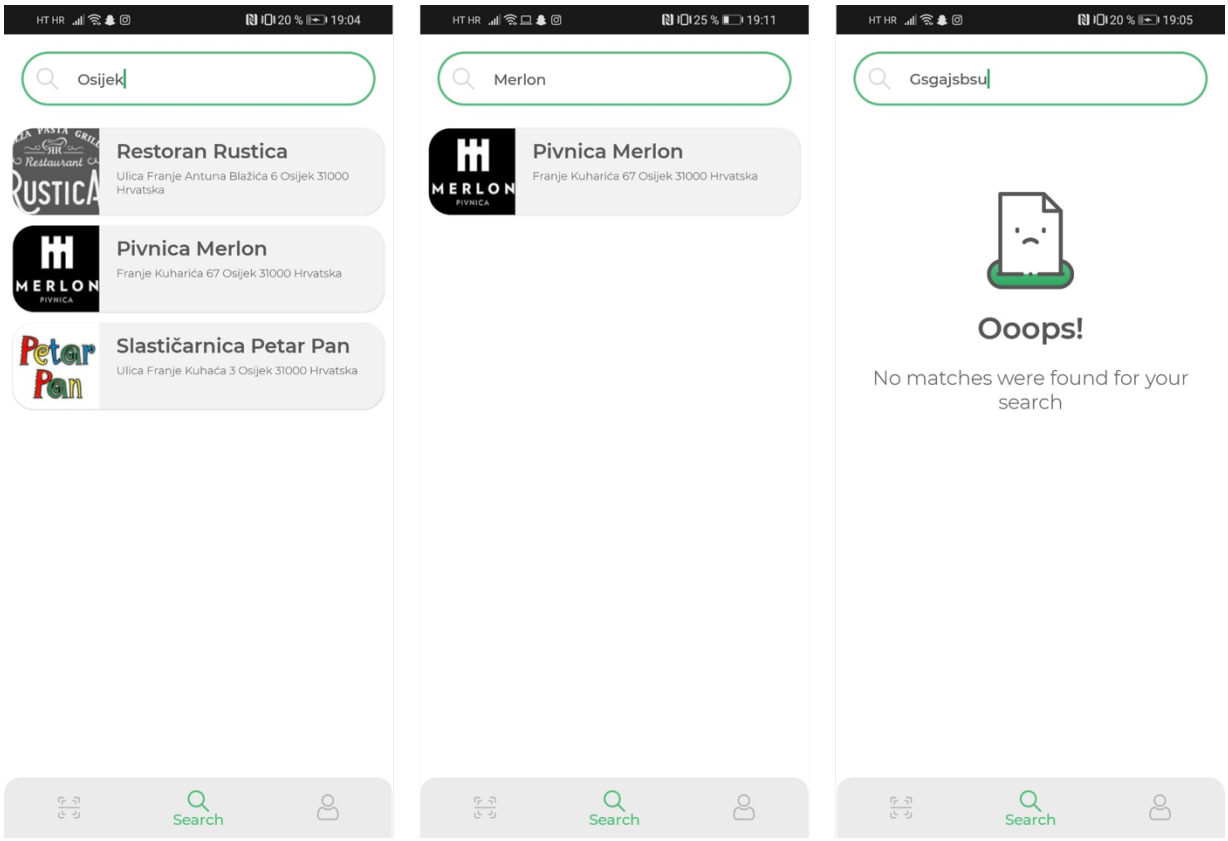


Slika 5.5. *Profil ugostiteljskog obrta, njegov jelovnik i košarica*

Slika 5.5. sredina prikazuje otvoreni jelovnik ugostiteljskog obrta. Proizvodi su u jelovniku grupirani u nekoliko kategorija. Svaku od prikazanih kategorija korisnik može otvoriti te iz liste proizvoda odabrati onaj koji želi dodati u košaricu. Unutar košarice moguće je mijenjati količinu proizvoda te ih po potrebi izbaciti. Iznad opcije za predaju, korisnik može vidjeti detalje svoje narudžbe kao što su cijena i broj stola s kojeg je skeniran *QR* kod.

5.5. Pretraživanje ugostiteljskih obrta

Privatni korisnici imaju mogućnost pretraživanja ugostiteljskih obrta po imenu i adresi. Ovoj funkcionalnosti pristupa se preko navigacijske trake u donjem dijelu ekrana odabirom srednje ikonice. Na slici 5.6. lijevo vidljiv je ekran za pretraživanje ugostiteljskih obrta s rezultatima pretraživanja za uneseni pojam.



Slika 5.6. *Pretraživanje ugostiteljskih obrta*

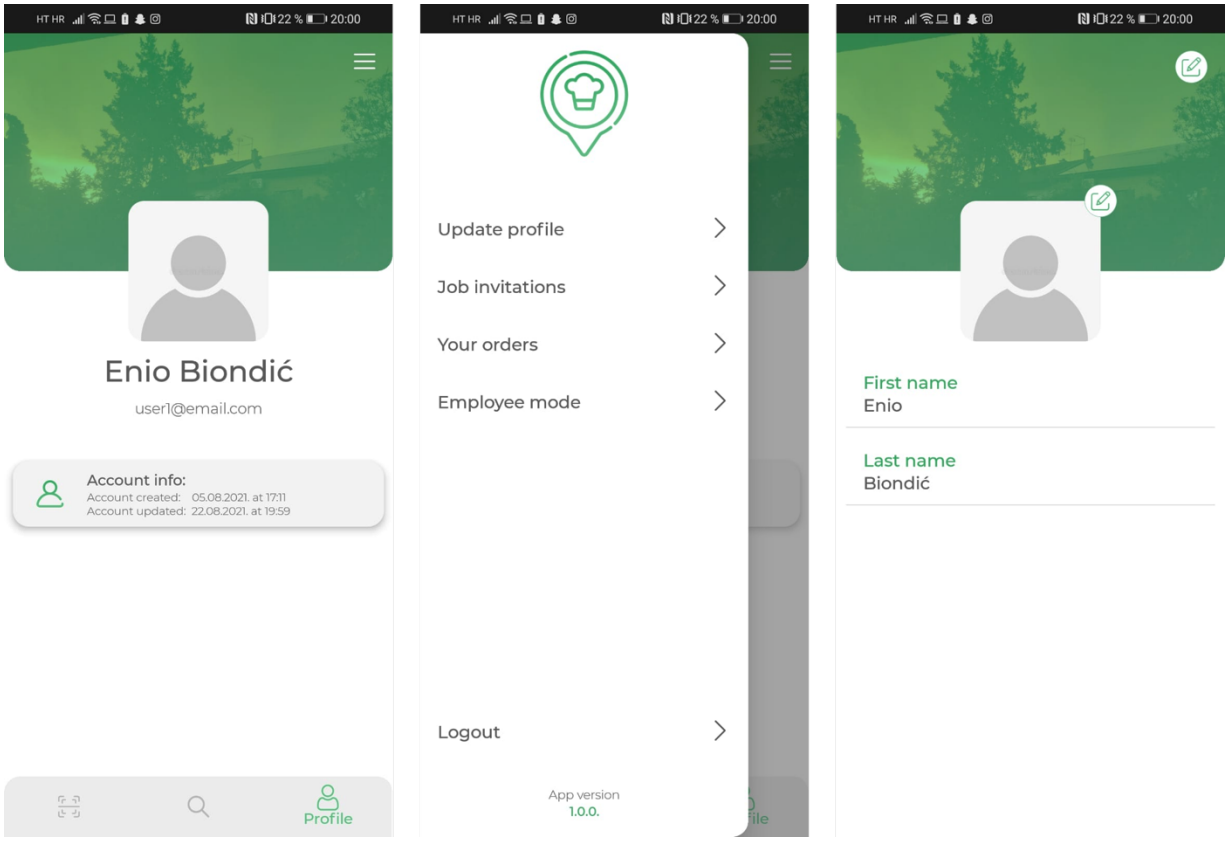
Slika 5.6. desno prikazuje slučaj kada aplikacija ne može pronaći niti jedan rezultat za uneseni pojam, u tom slučaju je zbog boljeg *UX* (engl. *user experience – korisničko iskustvo*) potrebno prikazati odgovarajuću poruku.

5.6. Korisnički profil privatne osobe

Stvaranjem računa u aplikaciji *Menuely* kreira se korisnički profil osobe. Vlastiti profil unutar aplikacije nužna je stavka aplikacija ovakve vrste. Jedan od važnijih razloga postojanja profila je mogućnost pregleda povijesti narudžbi. Osim toga putem profila korisnik može mijenjati

svoje osobne podatke koje je dao prilikom registracije, kao i uređivati svoju profilnu i naslovnu fotografiju.

Na slici 5.7. lijevo prikazan je izgled korisničkog profila. Gornji desni kut profila sadrži ikonicu opcija koje su dostupne korisniku na korištenje (slika 5.6. sredina). Odabirom opcije za uređivanje profila (engl. *Update profile*) otvara se ekran za uređivanje korisničkih podataka (slika 5.7. desno)



Slika 5.7. *Korisnički profil, njegove opcije i uređivanje profila*

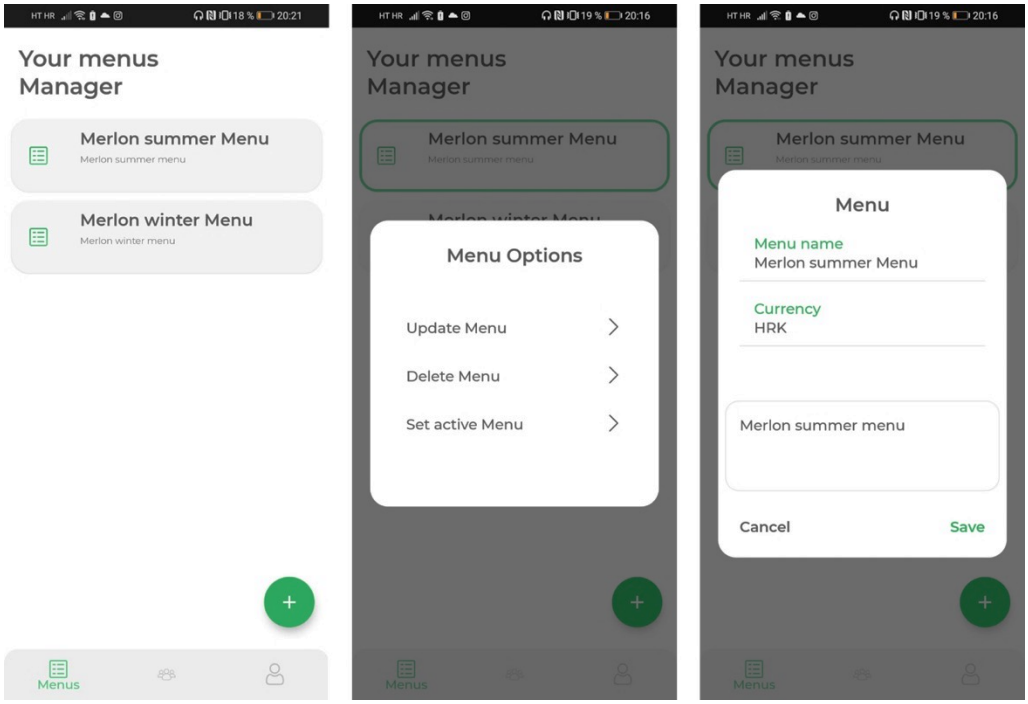
5.7. Upravitelj jelovnika

Kada se korisnik u aplikaciju prijavi putem računa ugostiteljskog obrta otvara se upravitelj jelovnicima (slika 5.8. lijevo). Upravitelj jelovnicima dopušta korisniku vršiti određene radnje nad jelovnikom (slika 5.8. sredina) kao što su uređivanje jelovnika, brisanje i postavljanje aktivnog jelovnika. Uređivanje jelovnika (engl. *Update menu*) omogućuje mijenjanje naziva, opisa i platne valute za proizvode iz tog jelovnika (slika 5.8. desno).

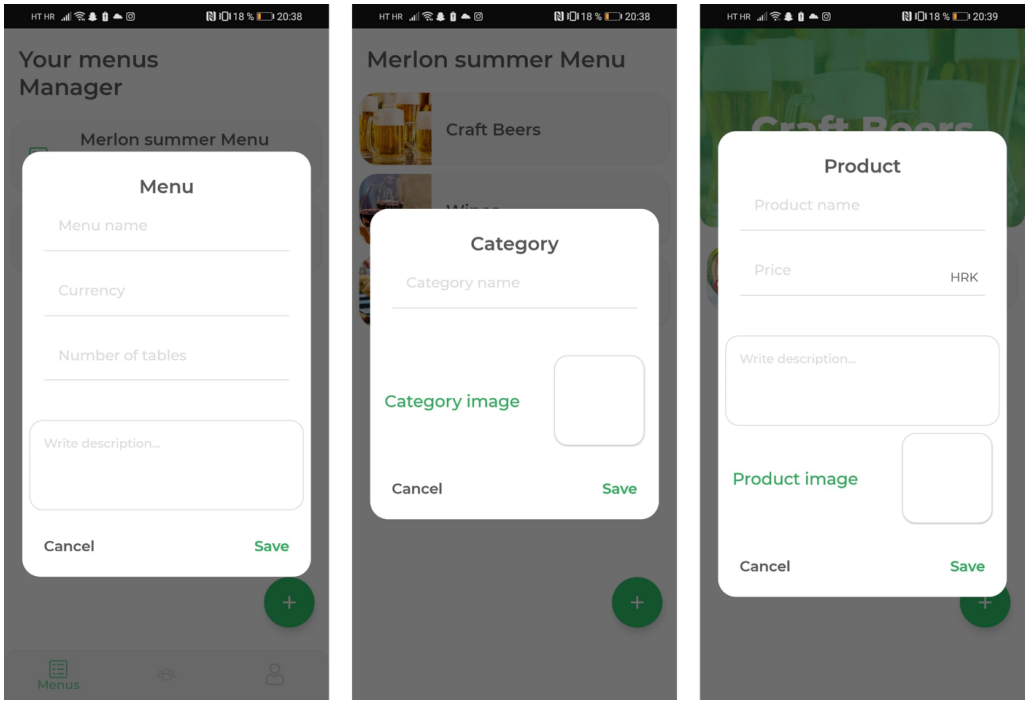
Svaki jelovnik sastoji se od kategorija, a u svakoj od njih nalaze se proizvodi. Sve *CRUD* operacije nad kategorijama i proizvodima omogućene su putem ovog upravitelja. Ako se neki

jelovnik proglasi aktivnim tada će se on otvarati kada korisnik skenira *QR* kod tog ugostiteljskog obrta.

Na slici 5.9. (s lijeva na desno) prikazane su forme za kreiranje jelovnika, kategorije proizvoda i proizvoda.



Slika 5.8. *Upravitelj jelovnika, opcije nad jelovnicima i uređivanje jelovnika*

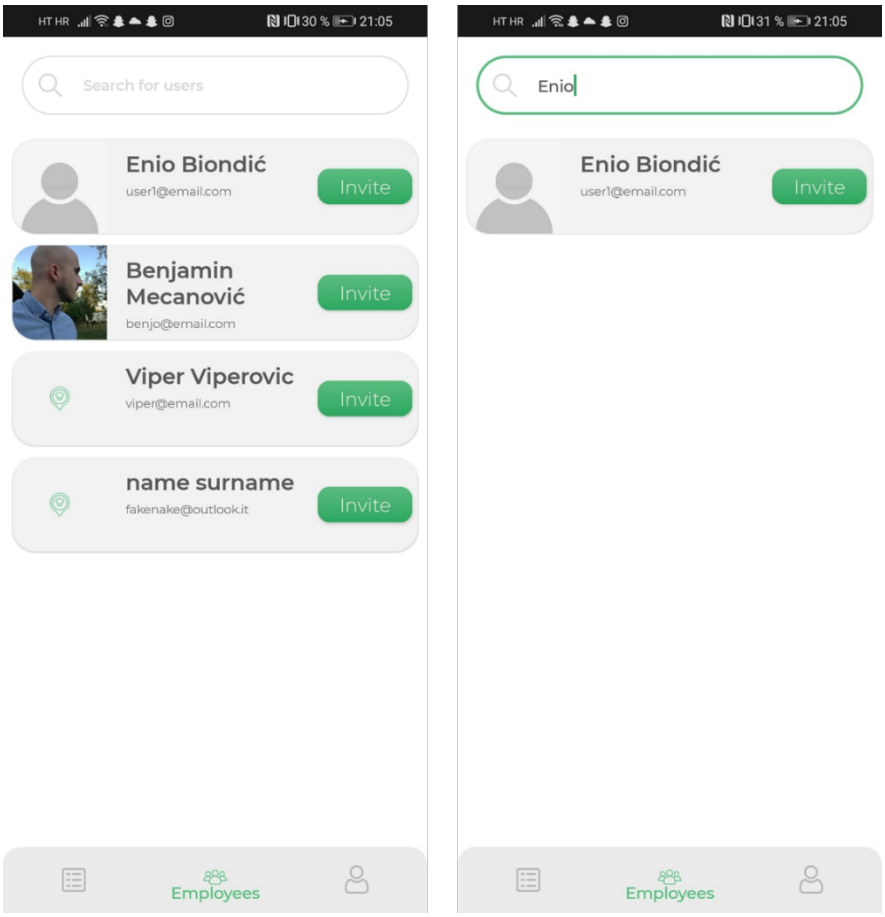


Slika 5.9. *Kreiranje jelovnika, kategorije proizvoda i proizvoda*

5.8. Pretraživanje korisnika aplikacije

Ugostiteljskim obrtima omogućeno je da putem aplikacije pretražuju njene korisnike. Ova funkcionalnost je zamišljena kako bi olakšala obrtima praćenje pristiglih narudžbi. Naime, narudžbe koje se naprave u nekom restoranu ne šalju se na profil restorana već na profile konobara koji rade u tom restoranu.

Kako bi restoran mogao dodati konobara na svoj profil potrebno je prvo pronaći korisnika putem tražilice, a zatim poslati mu zahtjev za zapošljavanje. Ukoliko korisnik prihvati zahtjev za zapošljavanje on postaje konobarom tog restorana i u svakom trenutku može vidjeti i prihvatiti pristigle narudžbe. Razlog iz kojeg se pristigle narudžbe ne šalju na profil restorana je da se uslužno osoblje odvoji od profila restorana kako bi se izbjegle slučajne i neželjene izmjene jelovnika.

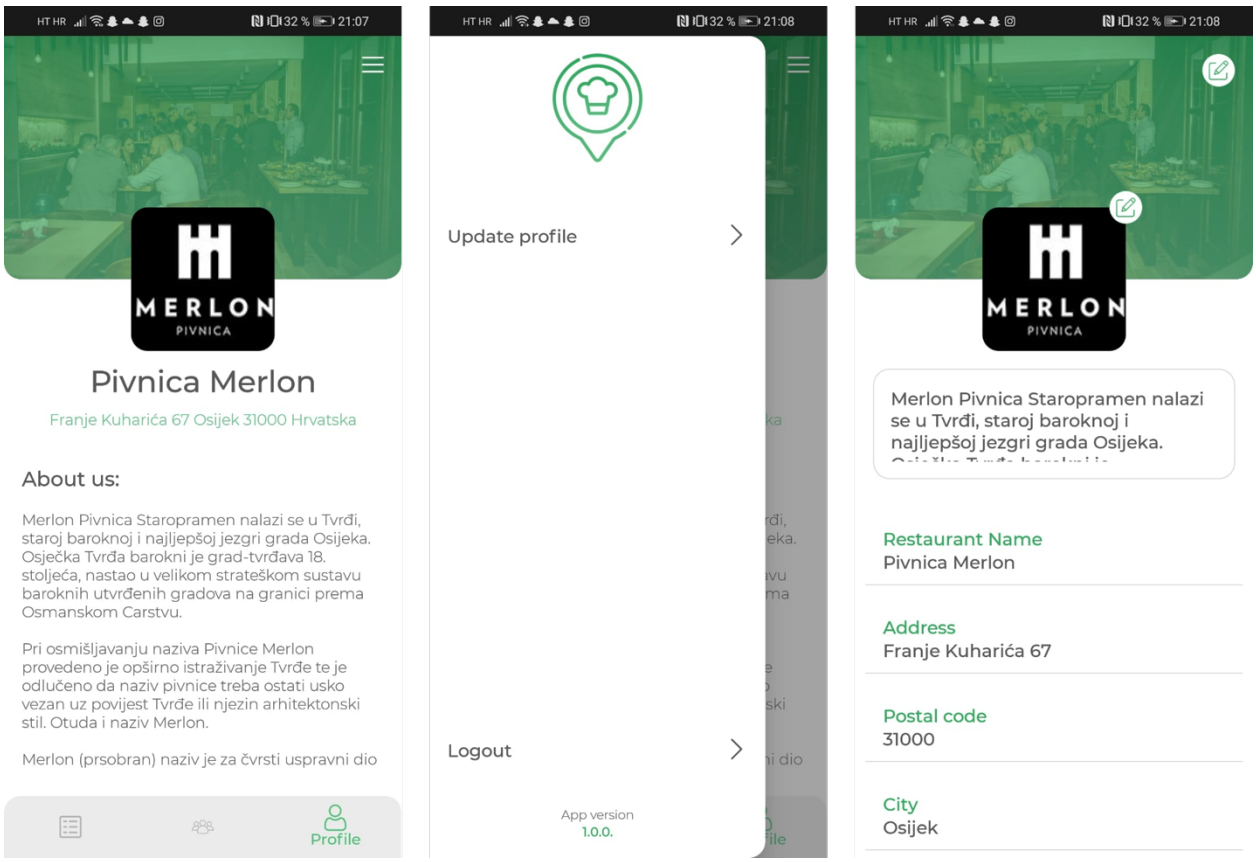


Slika 5.10. *Pretraživanje privatnih korisnika aplikacije*

5.9. Korisnički profil ugostiteljskog obrta

Kreiranjem računa ugostiteljskog obrta stvara se i njegov profil s popunjenim podacima danim prilikom registracije. Vlasnik profila ima mogućnost uređivanja tih informacija kao i mogućnost mijenjanja profilne i naslovne fotografije. Profilu se pristupa putem navigacijske trake na dnu ekrana odabirom treće ikonice.

Profil ugostiteljskog obrta dostupan je na pregled svim privatnim korisnicima aplikacije, stoga je potrebno da vlasnik profila ima mogućnost uređivanja informacija na njemu



Slika 5.11. *Profil ugostiteljskog obrta i njegovo uređivanje*

Na slici 5.11. lijevo vidljiv je prikaz ekrana profila ugostiteljskog obrta. Ovaj prikaz je pretpregled onog što privatni korisnici vide kada posjete ovaj profil. Slika 5.11. desno prikazuje izgled ekrana za uređivanje informacija danih prilikom registracije korisničkog računa kao i opcije za promjenu profilne i naslovne fotografije.

6. ZAKLJUČAK

Procesi modernizacije zahvaćaju raznolike segmente suvremenog života. Usprkos težnji pojednostavljivanja svakodnevnih obveza, pojedine prakse nisu mijenjane desetljećima. Iako smo kao društvo tromi u prihvaćanju novih radnih disciplina lako ćemo prihvatiti i naviknuti se na ono što nam život čini jednostavnijim.

Fokusirajući se na procese u ugostiteljstvu, cilj ovog rada je bio pronaći neke od njih i ponuditi tržištu moderniziranu alternativu. Kroz rad pokazano je kako se godinama ustaljeni procesi poput izrade, izmjene i pregleda jelovnika te prikupljanje, prihvaćanje i obrada narudžbi daju unaprijediti.

Ukoliko se prihvati modernizacija ovih procesa ona će za sobom povući brojne beneficije. Neke od glavnih bile bi smanjenje troškova prilikom izrade odnosno mijenjanja sadržaja jelovnika. Razlog tomu je što izostaju financijska davanja potrebna za tiskanje ažurnih verzija. Osim toga zadovoljstvo krajnjih korisnika se povećava jer bi moderni procesi smanjili vrijeme potrebno od trenutka prikupljanja do serviranja narudžbe. Uslužno osoblje bi imalo jasan pregled svih narudžbi te bi im raspodjela poslova bila jednostavnija.

Spomenute prednosti rezultat su modernizacije uskog spektra aktivnosti iz cjelokupnog ugostiteljstva. Prostora za napredak ima i u ostalim područjima poslovne djelatnosti kako ugostiteljstva tako i ostalih srodnih branši.

LITERATURA

[1] Glovo, https://about.glovoapp.com/en vrijeme pristupa (kolovoz 2021.)

[2] Wolt, https://wolt.com/en/about vrijeme pristupa (kolovoz 2021.)

[3] Developer Android: Android's Kotlin-first approach
Dostupno na: https://developer.android.com/kotlin/first vrijeme pristupa (kolovoz 2021.)

[4] S.Bjarne. What is object-oriented programming?. IEEE software, 1988, 5.3: 10-20.
Dostupno na: https://ieeexplore.ieee.org/abstract/document/2020

[5] A Arnold, K., Gosling, J., & Holmes, D. (2005). The Java programming language. Addison Wesley Professional. Dostupno na: https://www.acs.ase.ro/Media/Default/documents/java/ClaudiuVinte/books/ArnoldGoslingHolmes06.pdf

[6] A Studio, Android. "Android studio." The Official IDE for Android (2017).
Dostupno na: https://sistemasuni.edu.pe/pdfs/workshops/2019/Android.pdf

[7] Mule Soft: " What is an API? (Application Programming Interface)" Dostupno na : https://www.mulesoft.com/resources/api/what-is-an-api

[8] B. Lee, R. Fielding, and H. Frystyk. "Hypertext transfer protocol--HTTP/1.0." (1996). Dostupno na: https://www.hjp.at/doc/rfc/rfc1945.html

[9] Developer Andrpod: Jetpack Compose Tutorial
Dostupno na: https://developer.android.com/jetpack/compose/tutorial vrijeme pristupa (kolovoz 2021.)

[10] Square Retrofit : A type-safe HTTP client for Android and Java
Dostupno na: https://square.github.io/retrofit/ vrijeme pristupa (kolovoz 2021.)

[11] Square OkHttp: OkHttp Overview
Dostupno na: https://square.github.io/okhttp/ vrijeme pristupa (kolovoz 2021.)

[12] A Nurseitov, Nurzhan, et al. "Comparison of JSON and XML data interchange formats: a case study." Caine 9 (2009): 157-162. Dostupno na: https://www.cs.montana.edu/izurieta/pubs/caine2009.pdf

[13] Developer Android: Mutable State, Dostupno na: https://developer.android.com/reference/kotlin/androidx/compose/runtime/MutableState vrijeme pristupa (kolovoz 2021.)

[14] A Yang, H. Y., Tempero, E., & Melton, H. (2008, March). An empirical study into use of dependency injection in java. In 19th Australian Conference on Software Engineering (aswec 2008) (pp. 239-247). IEEE. Dostupno na: https://ieeexplore.ieee.org/abstract/document/4483212

[15] Developer Android: Dependency injection with Hilt:
Dostupno na : https://developer.android.com/training/dependency-injection/hilt-android vrijeme pristupa (kolovoz 2021.)

[16] L. Tian. "A comparison of Android Native App Architecture MVC, MVP and MVVM." Eindhoven University of Technology (2016).
Dostupno na: https://pure.tue.nl/ws/portalfiles/portal/48628529/Lou_2016.pdf

[17] Melton, J., & Simon, A. R. (2001). SQL: 1999: understanding relational language components. Elsevier. Dostupno na: https://books.google.hr/books?hl=hr&lr=&id=wyhXvU0Eyg0C&oi=fnd&pg=PP2&dd=SQL&ots=MvQNTpOOef&sig=dbm1Oh53z86_k5MAkGaRAicEcxg&redir_esc=y#v=onepage&q=SQL&f=false

SAŽETAK

Ovaj diplomski rad obrađuje temu izrade *Android* mobilne aplikacije koja pomaže u modernizaciji pojedinih procesa u ugostiteljstvu. Predstavljeno je moderno rješenje u pogledu rukovanja jelovnicima i narudžbama u ugostiteljskim obrtima. Prihvaćanje ovog rješenja najviše će se očitovati u zadovoljstvu klijenata ugostiteljskih obrta, ali i u boljoj koordinaciji uslužnog osoblja. Iako aplikacija rješava uzak spektar poslovnih procesa u ugostiteljstvu, beneficije koje rezultira su uistinu velike.

Ključne riječi: *Android*, aplikacija, ugostiteljstvo

ABSTRACT

ANDROID APPLICATION FOR ORDERS AND MENU DIGITIZATION IN HOSPITALITY FACILITIES

This paper discusses the topic of creating an Android mobile application that helps in the modernization of certain processes in the hospitality industry. A modern solution was presented in terms of handling menus and orders in hospitality facilities. The acceptance of this solution will be most reflected in the satisfaction of the hospitality facilities' clients of and the better coordination of the service staff. Although the application solves a narrow range of business processes in the hospitality industry, the benefits that result are truly great.

Keywords: Android, application, hospitality

ŽIVOTOPIS

Enio Biondić rođen je 28. studenoga 1997. godine u Virovitici. Nakon završenog osnovnoškolskog obrazovanja u Osnovnoj školi Ivane Brlić Mažuranić u Virovitici 2012. godine upisuje Srednju tehničku školu, smjer elektrotehničar, također u Virovitici. U svome srednjoškolskom obrazovanju sudjeluje u natjecanjima iz matematike i raznim projektima, a 2016. odlazi na Erasmus+ projekt razmjene studenata u Veliku Britaniju. Nakon državne mature, iste godine, upisuje tadašnji Fakultet elektrotehnike i računarstva u Osijeku, preddiplomski smjer računarstvo. Godine 2019., nakon završenog preddiplomskog studija, upisuje diplomski studij računalnog inženjerstva na spomenutom fakultetu.

Enio Biondić