Android klijentska aplikacija za centralizirano naručivanje hrane u Osijeku

Katić, Valentin

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij

ANDROID KLIJENTSKA APLIKACIJA ZA CENTRALIZIRANO NARUČIVANJE HRANE U OSIJEKU

Završni rad

Valentin Katić

Osijek, 2016

Sadržaj

1. UVOD

Današnji svijet iz dana u dan biva brži i to rezultira sve većom potražnjom za olakšanim pristupom svakodnevnim potrebama života. Jako veliku ulogu u današnjici imaju „pametni telefoni“ koji se koriste u razne svrhe. Nekad stvar luksuza, a danas život bez njih nije zamisliv. Praktičnost i mogućnosti pametnih telefona uzrokovale su nezaustavljivi rast u razvoju aplikacija na pokretnim uređajima, a tržište aplikacija je postalo jedno od najbrže rastućih grana industrije.

Iako postoji jako velika ponuda aplikacija, i dalje postoji potražnja za novim i inovativnim idejama. Kako je hrana jedna od osnovnih ljudskih potreba, glavni fokus aplikacije je prikupiti podatke restorana na području Osijeka i predstaviti ih sve s jednog mjesta omogućujući korisniku lakši pregled jelovnika i narudžbe.

U drugom poglavlju se upoznaje s tehnologijama koje su potrebne pri izradi Android aplikacije i bez kojih ne bi moglo biti moguće realizirati problematiku projekta.

Treće poglavlje pokazuje samu ideju projekta te upoznaje s njegovom problematikom, osnovnim elementima zaslona i njihovim funkcijama.

Kroz zadnje poglavlje (Programska izvedba aplikacije) se prolazi čitavim procesom izrade aplikacije, od njezinog osnovnog dizajna na početku te konačne verzije na samom kraju poglavlja s pripadajućim slikama zaslona (engl. *screenshot*) i dijelovima kôda.

1.1. Zadatak rada

Zadatak ovog rada je objasniti način rada centraliziranog naručivanja hrane u nekom gradu, u ovom primjeru grada Osijeka, izrada aplikacije za Android uređaj pomoću koje registrirani korisnik može naručiti hranu sa poslužiteljske aplikacije, realizacija komunikacije s poslužiteljem i testiranje i opis rada aplikacije.

2. RAZVOJNA OKOLINA I TEHNOLOGIJE ZA MOBILNE APLIKACIJE

2.1.Android (operacijski sustav)

Google Android je prvi otvoreni operacijski sustav za mobilne uređaje (mobilni telefoni, tableti, netbook računala, Google TV) kojeg je pokrenuo Google Inc. i vodila ga je Open Handset Alliance - grupa koja danas broji preko 80 tehnoloških kompanija između kojih se nalaze T-Mobile, HTC, Intel, Motorola, Qualcomm, i drugi, čiji je cilj ubrzati inovacije na području mobilnih operacijskih sustava, a samim time ponuditi krajnjim kupcima bogatije, jeftinije i bolje iskustvo uporabe.

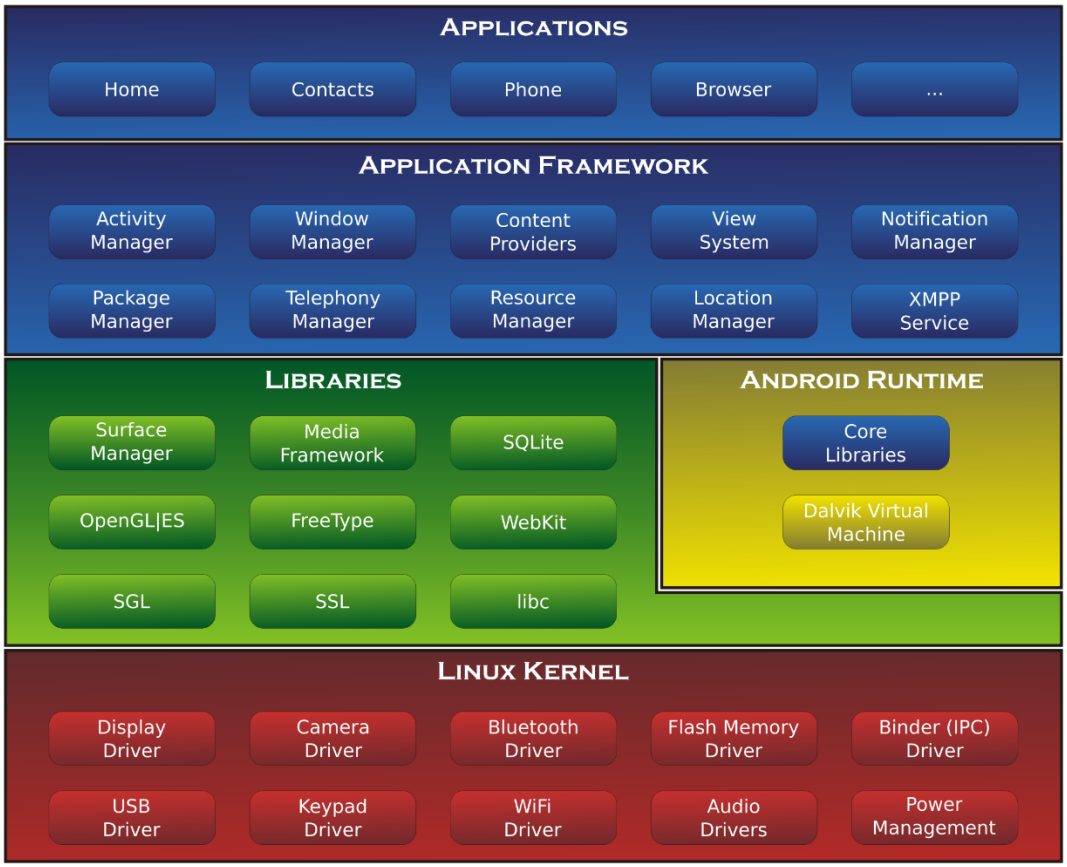
Android je modularan i prilagodljiv pa tako postoje slučajevi njegovog prenošenja na razne uređaje kao što su čitači elektronskih knjiga, mobilni telefoni, prijenosnici, te multimedijski izvođač [1].

U samim počecima Android je zamišljen kao projekt otvorenog koda (engl. *open source project*) te je od 21. listopada 2008. godine dostupan cjeloviti kôd pod Apache licencom. S druge strane, proizvođačima uređaja nije dopuštena uporaba Android zaštićenog imena ako Google ne certificira uređaj kao kompatibilan prema Compatibility Definition Document (CDD).

Sl. 2.1: Prikaz sustava Android 5.0 Lollipop.

2.1.1. Arhitektura

Android je zasnovan na jezgri Linux 2.6 i napisanom u C/C++ programskom jeziku. Obzirom na otvorenost izvornog programskog koda, aplikacije putem middleware-a imaju mogućnost komuniciranja i pokretanja drugih aplikacija primjerice za ostvarivanje poziva, slanje SMS poruka, pokretanja kamere i slično. Iako su C i C++ programski jezici primjenjivani za radno okružje (engl. *framework*), većina aplikacija pisana je u Java programskom jeziku rabeći Android Software Development Kit (SDK). Postoji mogućnost pisanja aplikacija i u C/C++ programskom jeziku, no tada se upotrebljava Android-ov razvijateljski kit u izvornom kodu (engl. *Android Native Code Development Kit*) (NDK). Ovakvim postupkom omogućuje se bolje raspolaganje resursima i uporaba knjižnica programa iz jezgre i radnog okružja. Ovakvim postupkom aplikacije se ubrzavaju i do 10 puta, no pisanje samog programa je puno složenije.



Sl. 2.2: Arhitektura Android sustava (izvor: http://simpledeveloper.com).

Arhitekturu Androida (Slika 2.2) možemo promatrati kao jedan programski stog koji sadrži nekoliko razina.

Na dnu stoga nalazi se Linux 2.6 jezgra koji sadrži drivere od kojih su najvažniji driver za među procesnu komunikaciju (IPC - *Inter-process communication*) koji služi za izmjenu podataka između različitih procesa ili niti unutar istog procesa te driver za upravljanje napajanjem (Power Management).

Iznad jezgre nalaze se knjižnice koje su pisane u C/C++ programskom jeziku:

- *Surface Manager* – knjižnica koja nadzire iscrtavanje grafičkog sučelja
- OpenGL | ES – knjižnica za sklopovsko ubrzavanje 3D prikaza (ako je moguća) te za visoko optimiziranu 3D softversku rasterizaciju
- SGL – 2D knjižnica upotrebljavana za većinu aplikacija
- Media Framework – knjižnica temeljena na OpenCORE koja podržava snimanje i reproduciranje poznatih audio/video formata
- FreeType – knjižnica namijenjena iscrtavanju fontova
- SSL (*Secure Sockets Layer*) - knjižnica za sigurnosnu komunikaciju putem interneta
- SQLite – knjižnica za upravljanje bazama podataka dostupna svim aplikacijama
- WebKit – *engine* za web preglednike
- libc – sistemska C knjižnica prilagođena za ugradbene sustave zasnovane na Linux OS-u

Slijedi Android Runtime odnosno sloj koji služi pokretanju aplikacija. Sastoji se od dvije važne komponente. Prva su tzv. "*Core libraries*" odnosno knjižnice koje sadrže većinu jezgrenih knjižnica programskog jezika Java. Druga komponenta je *Dalvik Virtual Machine* koji pokreće aplikacije kao zasebne procese odnosno kao instance virtualnog stroja. DVM pretvara Java class datoteke u svoj vlastiti format (.dex), kako bi bile optimizirane za minimalni utrošak memorije.

Nakon knjižnica dolazi aplikacijski okvir (engl. *Application Framework*) koji se sastoji od mehanizama koji pomažu pisanje aplikacija. Aplikacijski okvir dozvoljava upotrebu svih API-ja (*Application Programming Interface*) koji su upotrebljavani za bazne aplikacije. Tako je omogućeno upravljanje programskim paketima, aktivnostima aplikacije (odnosi se na životni ciklus aplikacije), pozivima, prozorima, resursima (pohrana komponenti aplikacija koje nisu sami kôd, primjerice slike),

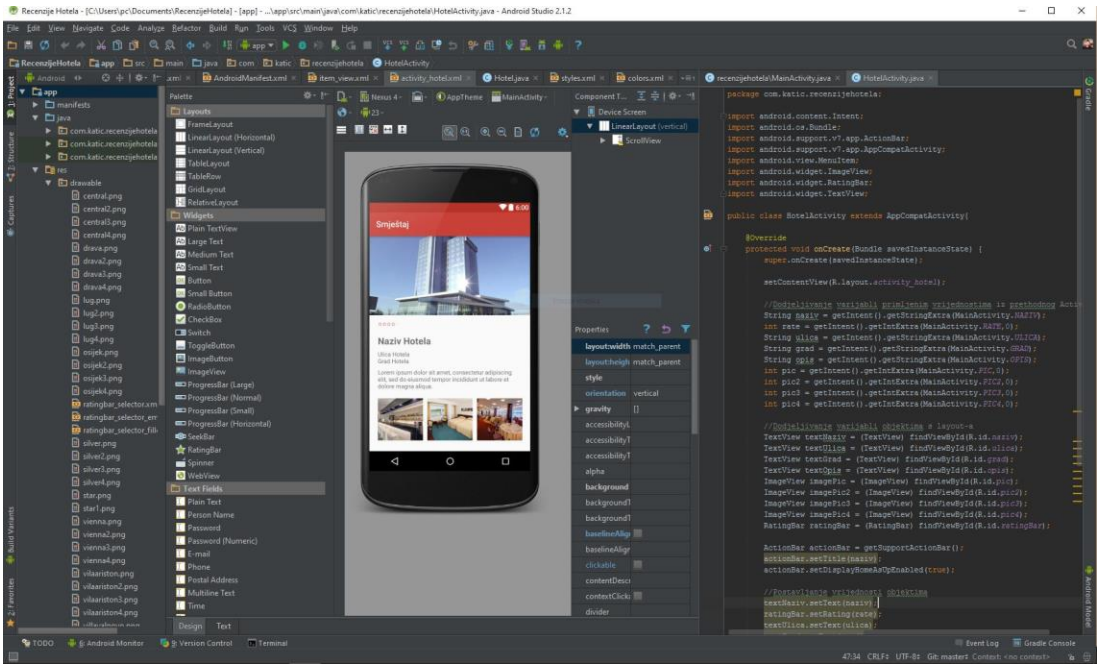
uporaba podataka od više različitih aplikacija, dohvaćanje i uporaba trenutne lokacije korisnika, prikaz obavijesti te baza pogleda i objekata koji se mogu upotrebljavati za dizajn aplikacije.

Na vrhu se nalaze same aplikacije. Ovaj sloj je vidljiv krajnjem korisniku i sastoji se kako od osnovnih, ugrađenih aplikacija poput klijenta elektroničke pošte, SMS programa, kalendara, web preglednika pa sve do aplikacija koje se mogu naći unutar Android Trgovine, kojih danas ima preko 250 000.

2.2.Android Studio

Android Studio je službeno integrirano razvojno okruženje (engl. *Integrated Development Environment,* IDE) za razvoj Android aplikacija, temeljen na IntelliJ IDEA. Na vrhu sa snažnim IntelliJ uređivačima koda i razvojnim alatima, Android Studio nudi još više mogućnosti koje poboljšavaju produktivnost prilikom izgradnje Android aplikacije, kao što su: [2]

- Fleksibilna Gradle-bazirana izgradnja sustava
- Brz i bogat značajkama emulator
- Jedinstvena okolina u kojoj možete razvijati za sve Android uređaje
- *Instant Run* za „guranje“ promjena u svoj pokrenutu aplikaciju bez izgradnje novog APK-a
- Predlošci koda i GitHub integracija koja će Vam pomoći izgraditi opće aplikacije i uvesti primjer koda.
- Opsežni alati za testiranje i okviri
- *Lint* alat koji će uhvatiti probleme izvođenja, iskoristivosti, kompatibilnosti i drugih
- C ++ i NDK podrška
- Ugrađena podrška za Google Cloud Platform, čime ga je lako integrirati s Google Cloud Messaging i App Engine



Sl. 2.3: Sučelje aplikacije Android Studio.

2.3.Android SDK

Android Software Development Kit (SDK) uključuje sveobuhvatan skup razvojnih alata. To uključuje program za pronalaženje pogrešaka, knjižnice, emulator mobilnog uređaja temeljen na QEMU, dokumentaciju, uzorak koda i vodiče. Trenutno podržane razvojne platforme uključuju računala sa sustavom Linux (bilo koja moderna Linux distribucija za stolno računalo), Mac OS X 10.5.8 ili noviji i Windows XP ili noviji. Od ožujka 2015. SDK nije dostupna na samom Androidu, ali razvoj softvera je moguć pomoću specijaliziranih aplikacija za Android. [3]

Poboljšanja Android SDK idu ruku pod ruku s ukupnim razvojem Android platforme. SDK također podržava starije verzije Android platformi, u slučaju da programeri žele razvijati aplikacije za starije uređaje. Razvojni alati su preuzete komponente, tako da nakon što ste jednom preuzeli najnoviju verziju i platformu, starije platforme i alati se također mogu preuzeti za testiranje kompatibilnosti.

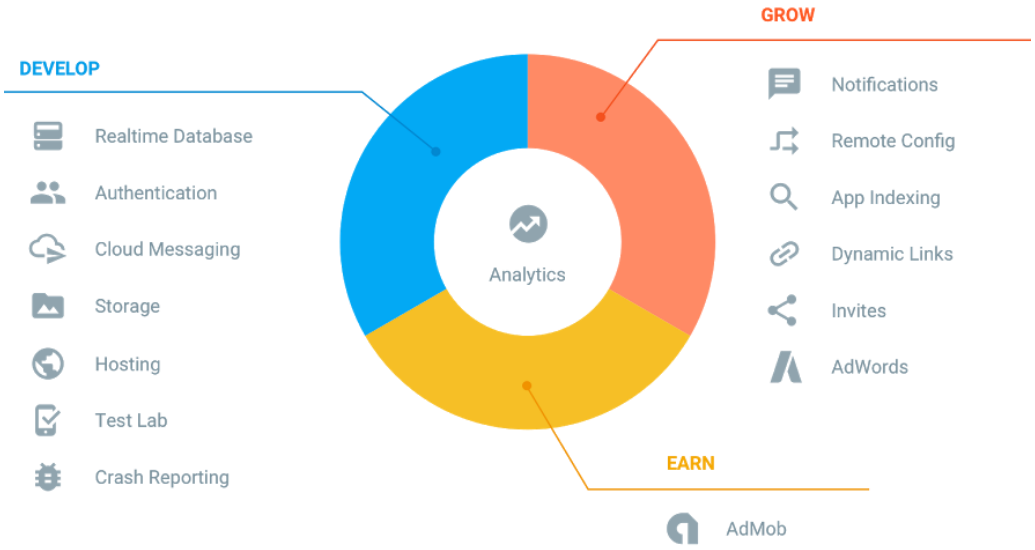
Prilikom instalacije Android Studio-a, automatski se postavlja i Android SDK što nije slučaj u ostalim razvojnim okruženjima za Android.

2.4. Google Firebase

Firebase je mobilna i web platforma s alatima i infrastrukturom osmišljenim kako bi pomogla programerima izgradnju visokokvalitetnih aplikacija. Sastoji se od komplementarnih značajki koje programeri mogu koristiti ovisno o njihovim potrebama. Primarni produkt Firebase je bila baza podataka u stvarnom vremenu koja pruža API koji omogućava programerima pohranu i sinkronizaciju podataka preko više klijenata. Tijekom vremena, proširila je svoju liniju proizvoda kako bi pružala cijeli paket za razvoj aplikacija. Tvrtku je preuzeo Google u listopadu 2014. godine, a značajan broj novih mogućnosti su objavljene u svibnju 2016. godine na Google I/O.

Neki od servisa koje platforma nudi su (Slika 2.4):

- Analitika (engl. *Analytics*) – Besplatan alat za mjerenje aplikacije koja pruža uvid u korištenje aplikacije i angažman korisnika.
- Poruke u oblacima (engl. *Cloud Messaging*) – Rješenje za međusobnu komunikaciju između različitih platformi kao što su Android, iOS i web aplikacije.
- Autentifikacija (engl. *Authentication*) – Usluga koja može autentificirati korisnike koristeći samo kôd s klijentske strane. Podržava i prijavu preko raznih društvenih davatelja usluga kao što su Facebook, GitHub, Twitter i Google. Osim toga omogućava i provjeru korisnika putem elektroničke pošte i zaporke pohranjene u Firebase.
- Baza podataka u stvarnom vremenu (engl. *Real time Database*) – Firebase pruža bazu podataka u stvarnom vremenu servis za rad u pozadini. Servis pruža razvijačima aplikacije API koji omogućava sinkronizaciju podataka aplikacije između klijenata i spremanje na Firebase-ov oblak.
- Skladištenje (engl. *Storage*) – Omogućava siguran prijenos datoteka i preuzimanje za Firebase aplikacije, bez obzira na kvalitetu mreže. Programer može koristiti servis za spremanje slike, zvuka, videozapisa ili drugih korisnički generiranih sadržaja.
- Usluge poslužitelja (engl. *Hosting*) – Podržava posluživanje statične datoteke, kao što su CSS, HTML, JavaScript i druge datoteke koje se ne mijenjaju dinamički.
- Obavijesti (engl. *Notifications*) – Servis koji omogućuje ciljane korisničke obavijesti za programere mobilnih aplikacija.



Sl. 2.4: Servisi Google Firebase platforme (izvor: https://firebase.google.com).

2.5.Osnovna struktura aplikacije

Aplikacija se sastoji od četiri ključna dijela:

- aktivnost (engl. *activity*) - Aktivnosti su javne klase koje nasljeđuju osobine od klase *android.app.MainActivity* te su same odgovorne za čuvanje svog stanja u životnom ciklusu aplikacije. Predstavljaju dio aplikacije koji se uglavnom može promatrati kao jedan konkretan prozor aplikacije gdje je korisnik u mogućnosti izvršavanja određene radnje. Aplikacija može sadržavati jednu ili više definiranih aktivnosti, pri čemu je jedna od aktivnosti uvijek definirana kao primarna. Prijelaz između aktivnosti odvija se tako što aktualna aktivnost invocira novu. Iako više aktivnosti tvori jedno kompaktno korisničko sučelje treba imati na umu da su one međusobno nezavisne;
- namjera (engl. *intent*) - Omogućava prijelaz između zaslona aplikacija te predstavlja namjeru za obavljanjem određene radnje. Dva najvažnija dijela namjere u strukturi podataka su akcije i podaci na kojima treba poduzeti te akcije;
- usluga (engl. *service*) - Usluga je zapravo kôd koji predstavlja proces bez vidljive korisničke interakcije te se uglavnom izvršava u pozadini neodređeni vremenski period. Služi za obnavljanje podatkovnih resursa, vidljivih aktivnosti i signalizacijskih obavijesti. Svaka usluga nasljeđuje klasu Service;

- pružatelj sadržaja (engl. *Content Provider*) - Aplikacije svoje podatke mogu pohraniti u datoteke, baze podataka ili na neka druga mjesta. Pružatelj usluga omogućava uzajamno korištenje podataka između različitih aplikacija i njihovih procesa. Pružatelj sadržaja nasljeđuje klasu *ContentProvider*.

Aplikacija ne mora sadržavati sve spomenute dijelove, ali također može sadržavati i neke druge, koje nisu navedene. Svi korišteni dijelovi se trebaju nalaziti u datoteci *AndroidManifest.xml,* jednoj od bitnijih dijelova Android projekta. To je XML[1] datoteka u kojoj se trebaju deklarirati dijelovi aplikacije koji će se koristiti u projektu te njihove predispozicije [4].

2.6.Grafičko korisničko sučelje

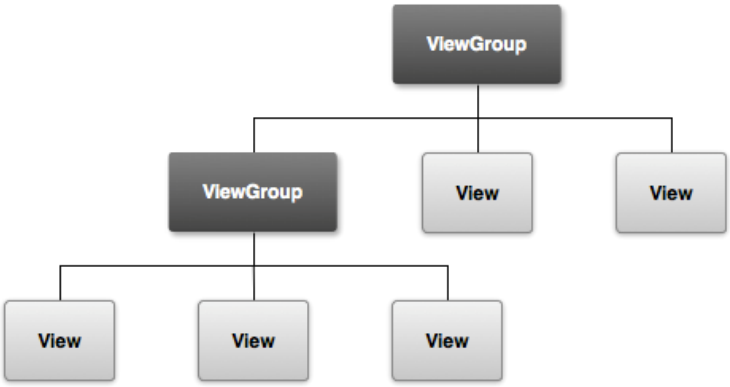
Grafičko korisničko sučelje[2] u Android aplikacijama može biti ostvareno na dva načina: proceduralno i deklarativno. Proceduralni dizajn odnosi se na pisanje Java kôda, a deklarativni na pisanje XML kôda. U praksi se za dizajniranje grafičkog korisničkog sučelja uglavnom koristi XML.

Kreiranjem sučelja aktivnosti dobivaju funkcionalnost, tj. vidljivost na zaslonu uređaja i na taj se način omogućava interakcija s korisnikom. Osnovni dijelovi korisničkog sučelja su:

- pogled (engl. *View*) - objekt čija se podatkovna struktura sastoji od zapisa izgleda i sadržaja određenog pravokutnog područja na zaslonu. Upravlja iscrtavanjem elemenata, pomicanjem sadržaja na zaslonu (engl. *scrolling*) i ostalim faktorima koji utječu na izgled aplikacije. Android raspolaže s već gotovim skupovima objekata ove vrste kao što su tipke, kvadratići za odabir (engl. *checkbox*), korištenje pomicanja sadržaja i slično. Ovi objekti nasljeđuju klasu *View*;
- grupa pogleda (engl. *ViewGroup*) - posebna vrsta objekta pogled koja sadrži i upravlja skupinom zavisnih objekata pogleda i grupa pogleda čime je omogućena kompleksnost prikaza korisničkog sučelja. Objekti ove vrste su instance klase *ViewGroup* [4].

Hijerarhijska ovisnost objekata pogleda i grupa pogleda prikazana je slikom 2.5.

[1] XML (engl. EXtensible Markup Language) - jezik za označavanje podataka
[2] Korisničko sučelje (engl. *Graphical User Interface*, skraćeno GUI) - način interakcije čovjeka s računalom kroz manipulaciju grafičkim elementima i dodacima uz pomoć tekstnih poruka i obavijesti.



Sl. 2.5: Hijerarhija elemenata grafičkog korisničkog sučelja (izvor: http://stackoverflow.com).

2.6.1. Datoteka XML

XML (engl. *eXtensible Markup Language*) je proširivi jezik za označavanje podataka i dokumenata. Ideja je bila stvoriti jedan jezik koji će biti jednostavno čitljiv i ljudima i računalnim programima. Princip realizacije je vrlo jednostavan: odgovarajući sadržaj treba se uokviriti odgovarajućim oznakama koje ga opisuju i imaju poznato, ili lako shvatljivo značenje. Danas je XML jezik vrlo raširen i koristi se za različite namjene: odvajanje podataka od prezentacije, razmjenu podataka, pohranu podataka, povećavanje dostupnosti podataka i izradu novih specijaliziranih jezika za označavanje. XML je standardizirani jezik i za njegovu standardizaciju brine se World Wide Web Consortium [5].

XML dokument se sastoji od 2 dijela. Prvi dio je zaglavlje ili deklaracija i nalazi se na početku XML-a. U njemu se navode podaci koji opisuju XML dokument kao što su inačica XML preporuke prema čijim pravilima je dokument napravljen te kôdna stranica. Drugi dio je sadržaj dokumenta u kojem se nalazi korisni sadržaj omeđen XML oznakama. Svaki XML dokument mora imati jedan korijenski ili root element koji uokviruje kompletan sadržaj. Kôd 1 prikazuje jedan osnovni primjer takvog dokumenta. Oznake počinju sintagmom „<tip oznake>“, a završavaju s „<tip oznake/>“ . Tekst između početnih i završnih znakova oznaka je sadržaj. Sadržaj i pripadne oznake čine element. Par ime/vrijednost se naziva atribut. Sadržaji se sastoje od atributa. U ovom primjeru, atributi su name, mobile i email.

```
<?xml version="1.0" encoding="UTF-8" ?>
<memo>
     <contact type="person">
             <name>Petar Perić</name>
             <mobile>098 1234567</mobile>
             <email>petar.peric@etfos.hr</email>
     </contact>
</memo>
```

Programski kôd 2.1: Primjer XML dokumenta.

Definiranje korisničkog sučelja pomoću XML datoteka izrazito je korisno iz više razloga: jednostavnost, hijerarhijska struktura, promjenjivost, povezanost pod elemenata sučelja i nepodložnost promjenama izvornog kôda. Referenciranje elemenata korisničkog sučelja (engl. *widget*) u Java kôdu preko jedinstvenog android:id atributa omogućava jednostavno programiranje funkcija sučelja. Zbog navedenih razloga XML postaje sve korišteniji u definiciji grafičkog korisničkog sučelja [6].

2.6.2. Datoteka *AndroidManifest.xml*

Svaka aplikacija mora imati datoteku *AndroidManifest.xml* koja ju opisuje. Ova se datoteka nalazi u glavnom direktoriju paketa te sadrži ključne informacije o aplikaciji, informacije koje Android sustav mora imati da bi mogao pokrenuti bilo koji dio kôda. Te informacije su:

- naziv paketa koji služi kao jedinstveni identifikator aplikacije;
- opis komponenti- aktivnosti, usluge, pružatelji sadržaja, filtere namjera (engl. *Intent Filters*), *Broadcast Receivers*, itd.;
- odredbe o tome koji će procesi sadržavati programske komponente;
- deklaracija dozvola za pristup aplikacijskim komponentama od drugih aplikacija;
- popis klasa koje osiguravaju oblikovanje i ostale informacije dok je aplikacija aktivna (te deklaracije su prisutne u datoteci *AndroidManifest.xml* tijekom razvoja i testiranja, a izbacuju se prije njenog objavljivanja);
- popis biblioteka koje aplikacija koristi;
- minimalna razina Android API-ja [5].

3. ZAHTJEVI PREMA APLIKACIJI

Temeljna zamisao ovog rada, kao što je navedeno u uvodu je izrada aplikacije za centralizirano naručivanje hrane na području Osijeka. Centralizirano naručivanje hrane je proces naručivanja hrane iz lokalnih restorana ili hrane zadruge putem web ili mobilne aplikacije. Slično kao i naručivanje robe široke potrošnje na Internetu, mnogi od njih omogućuju korisnicima kreiranje korisničkog računa s njima kako bi olakšali česta naručivanja. Kupac će tražiti svoj omiljeni restoran, obično filtriran po vrsti kuhinje, odabrati nešto u ponudi i odabrati isporuku ili *pick-up*. Plaćanje se može izvršiti gotovinom.

Glavni uvjeti za rad aplikacije su posjedovanje Android pametnog telefona ili tableta s minimalnom inačicom operativnog sustava Android 4.4 *KitKat* te mogućnosti povezivanja s Internetom.

Prilikom pokretanja aplikacije su ponuđene opcije za prijavu korisnika, registraciju korisnika ili pokretanje aplikacije bez korisničkog računa. Upisivanjem svojih podataka i prijavom korisnika, taj izbornik se više ne pojavljuje, izborom registracije korisnika otvora se zaslon s obrascem za registraciju, a izborom pokretanja aplikacije bez računa otvara se zaslon aplikacije ali bez mogućnosti naručivanja hrane nego samo s mogućnosti poziva restorana.

Račun korisnika sadržava podatke kao što su ime, prezime, adresa elektroničke pošte te adresa prebivališta i broj mobitela za kontakt. Korisnički račun se ne kreira na samom uređaju nego se aplikacija povezuje s online bazom podataka gdje su pohranjeni svi podaci radi mogućnosti pristupa podacima putem web aplikacije. Postoji više vrsta korisničkih računa, ovisno je li korisnik aplikacije krajnji korisnik, administrator ili moderator podataka za restoran.

Zaslon aplikacije nudi izbor restorana preko liste a ima i dodatnu mogućnost filtera s izborom restorana po kuhinji, pri tom je moguće da jedan restoran bude u više kategorija i mogućnost dodavanja restorana u oznake kako bi mogao olakšano pristupiti omiljenim restoranima. Izbor se potvrđuje pritiskom na određeni restoran i vodi na idući zaslon koji prikazuje ponudu izabranog restorana.

Ponuda restorana se kategorizira ovisno o restoranu a prikaz je u obliku naziv jela, njegova cijena, ikona za dodavanje u košaricu u slučaju da korisnik koji pregledava je registriran, inače ima samo mogućnost poziva na kontakt telefon restorana te skrivenog prikaza sastojaka jela koji postaje vidljiv pritiskom na jelo.

Nakon popunjene košarice, korisnik ima izbor načina isporuke (dostava ili pick-up). U slučaju odabira *pick-up* način isporuke, može putem Google Maps aplikacije naći upute do restorana s obzirom na

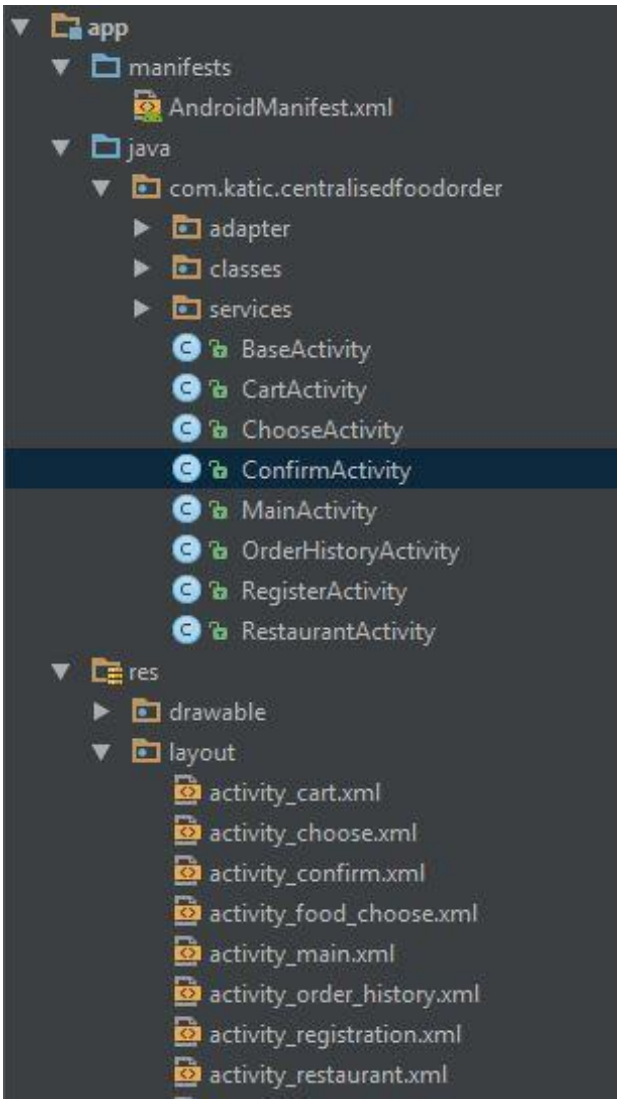
njegovu trenutnu lokaciju. U slučaju dostave, aplikacija traži potvrdu adrese za dostavu i pri tom ispisuje korisnikovu zadanu adresu koju je unio prilikom registracije i nudi mogućnost unosa druge adrese te nudi mogućnost unosa dodatnih elemenata kao što su kat i broj stana u slučaju da je korisnikovo prebivalište unutar stambene zgrade, te kontakt broj telefona.

4. PROGRAMSKA IZVEDBA APLIKACIJE

4.1. Struktura projekta

Rješenje problematike ovog završnog rada ostvareno je objektno orijentiranim programskim jezikom Java. Korišten je Android Studio sa svim alatima koji su potrebni za uspješno programiranje aplikacija s operativnim sustavom Android. Projekt se sastoji od:

- java/ : sadrži izvorne kodove u Javi;
- manifests/ : sadrži informacije o sustavu;
- res/ : sadrži ikone, definicije grafičkih korisničkih sučelja i datoteke XML;



Sl. 4.1: Sadržaj projekta Centralizirana dostava hrane.

4.2. Grafičko korisničko sučelje aplikacije

Grafičko korisničko sučelje aplikacije *Centralized Food Order* se sastoji od nekoliko vrsta tipa prikaza (engl. *layout*). Početno korisničko grafičko sučelje aplikacije *Centralized Food Order* je definirano u datoteci *activity_main.xml*, koja će biti opisana. Općenito, tip prikaza služi grupiranju više pod elemenata korisničkog sučelja u cjelinu te sugerira u kakvom su odnosu objekti djeca. Klase tipa *Layout* su potklase klase ViewGroup. Programski kôd 4.1 prikazuje koje tipove prikaza koristi *Centralized Food Order*. *RelativeLayout* omogućava postavljanje relativnog odnosa među objektima djecom i roditeljima.

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background">
...
</RelativeLayout>
```

Programski kôd 4.1: *RelativeLayout* u *activity_main.xml.*

Projekt sadrži i *LinearLayout-e.*. *LinearLayout* slaže objekte jedno do drugog, odnosno jedno ispod drugog ovisno o postavljenoj orijentaciji.

Layout-i u ovom projektu sadrže i *EditText, TextView, Button widgete, TabHost, RecyclerView widget, ImageView* te različite prilagođene *View-e. TextView* služi kako bi prikazao tekst korisniku te u nekim slučajevima dopušta i uređivanje teksta. Upravo je zbog toga povezan s *EditText*, s kojim u suradnji služi uređivanju teksta. Datoteka *activity_main.xml* sadrži dva *TextView-a* jer aplikacija sadrži dva polja za upis elektroničke pošte i zaporke, što se može vidjeti na slikama u Poglavlju 4.4. Primjer jednog *TextView-a* prikazuje Programski kôd 4.2, gdje se od korisnika traži da upiše elektroničku poštu s kojom se prijavljuje.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/mail"
    android:id="@+id/mail"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="35dp" />
```

Programski kôd 4.2: Primjer gdje korisnik upisuje svoju elektroničku poštu.

Kako bi se napravila funkcionalna tipka koju korisnik može pritisnuti, i da se pritom prijavi, u tipu prikaza se koristi *Button* (Programski kôd 4.3). U ovom slučaju, unutar tipke se nalazi i poveznica za tekst u *strings.xml* datoteci pod varijablom *login* i sadrži tekst „Prijava“. Povezivanje teksta s datotekom *strings.xml* omogućava prevođenje aplikacije na različite jezike.

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/login"
    android:id="@+id/btnLogin"
    android:layout_below="@+id/passwordText"
    android:layout_marginTop="30dp"
    android:layout_centerHorizontal="true" />
```

Programski kôd 4.3: Tipka „Prijava“.

TabHost služi kao okvir za različite kartice od kojih svaka ima svoj izgled koji se prikazuje izborom na jednu od kartica (Slika. 4.4). Za prikaz velikih skupova podataka koji su grafički jednaki ili slični koristi se *RecylcerView widget* jer on učitava u sebe veću količinu podataka ali prikazuje samo onaj dio koliki se može prikazati na zaslonu uređaja te štedi radnu memoriju konstantnim recikliranjem sadržaja. *ImageView* služi za prikaz slika ili ikona na zaslonu i za njega vrijede jednaka pravila kao i za ostale View-e. Primjer *ImageView-a* koji se nalazi u *activity_restaurant.xml* prikazuje Programski kôd 4.4.

```
<ImageView
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:src="@drawable/restaurant"
    android:id="@+id/restaurantImageView"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:scaleType="centerCrop"/>
```

Programski kôd 4.4: Primjer *ImageView-a.*

4.3. Android Manifest

Centralized Food Order u ovoj datoteci sadržava nekoliko bitnih stvari o svojstvima aplikacije koji određuju njezinu funkcionalnost. Jedna od bitnih stvari je informacija koje dozvole sadrži aplikacija. Bez navođenja ovih dozvola ne bi se moglo uspješno implementirati rješenje aplikacije (Programski kôd 4.5). Osim navedenoga, datoteka sadrži još informacija i osnovnih elemenata, koji su opisani u poglavlju 2.5.2.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
```

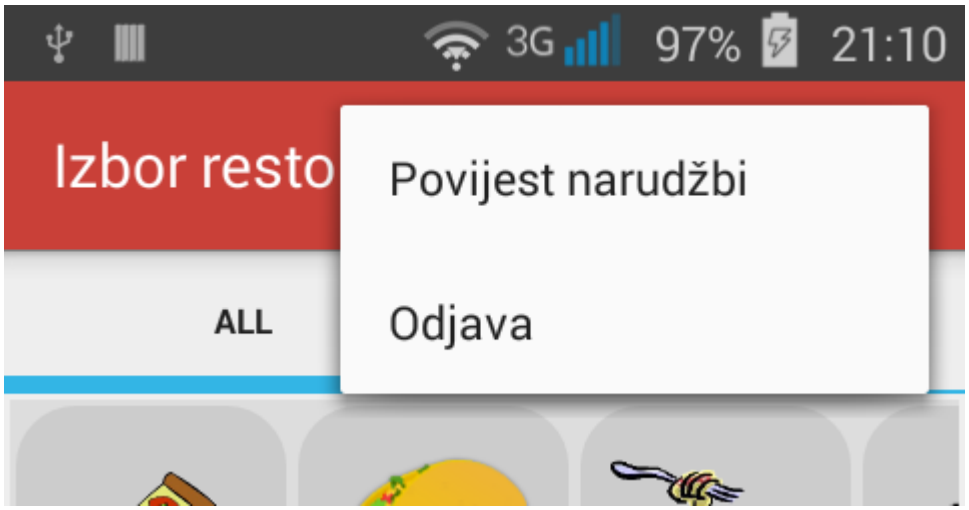
Programski kôd 4.5: Dozvole koje aplikacija koristi.

4.4. Korištenje aplikacije

Otvaranjem aplikacije prikazuje se početni zaslon koji se sastoji od 3 dijela: polje za upis korisnikove elektroničke pošte koja kasnije služi za autorizaciju, polje za upis lozinke te tri gumba: gumb za prijavu s unesenim podacima, gumb koji otvara formu za registraciju i gumb za ulaz u aplikaciju bez prijave. Početni zaslon je dizajniran po kôd-u prikazan u poglavlju 4.2. Slika 4.2 prikazuje izgled početnog zaslona.

Sl. 4.2: Izgled početnog zaslona.

Odabirom jednog od polja za unos podataka, pojavljuje se virtualna tipkovnica koja omogućava unos podataka. Gumb „Pokreni bez prijave“ započinje registraciju novog korisnika u Firebase bazi podataka ali anonimnog tipa. Anoniman tip korisnika je privremen korisnički račun koji može sadržavati samo spremljene oznake restorana i nepovratno se gube spremljeni podaci prilikom odjave korisnika. Odjava se izvršava na tri načina: registracijom novog korisnika, prijavom registriranog korisnika ili pritiskom na stavku „Odjava“ u izborniku (Slika 4.3).



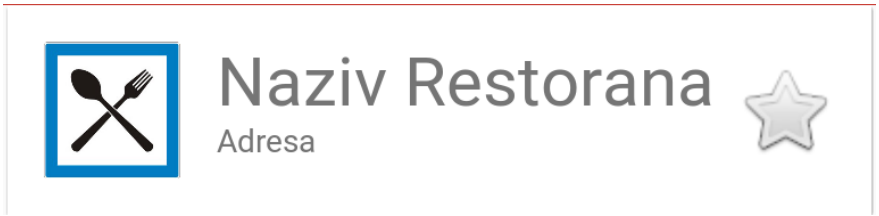
Sl. 4.3: Prikaz izbornika.

Pritiskom na gumb „Registriraj se“ otvara se obrazac za registraciju i ona sadrži obavezna polja za upis prilikom registracije, elektroničku poštu i zaporku, i neobavezna polja za upis prezimena, adrese, kućnog broja, grada odnosno naselja i kontakt broja mobitela. Neobavezna polja se mogu naknadno upisati u koraku u kojem korisnik potvrđuje adresu na koju naručuje izabrana jela. Izgled obrasca za registraciju je vidljiv na Slici 4.4. Klikom na gumb „Registriraj se“ pokreće se metoda s*ignUp()* vidljiva u *RegisterActivity.java* i ona prvo izvršava provjeru je li trenutni korisnik anoniman. U slučaju da je korisnik anoniman, odjavljuje ga i zatim metodom *validateForm()* provjerava jesu li popunjena polja elektroničke pošte i zaporke. U slučaju uspješne provjere, metoda sprema vrijednosti iz polja za upisivanje u varijable te pokreće metodu Firebase servisa za kreiranje korisnika s elektroničkom poštom i zaporkom kao ulaznim parametrima.

U slučaju pogreške, metoda vraća poruku obavještenja ili u suprotnom završava trenutni *Activity* i pokreće *ChooseActivity* koji je početni zaslon registriranih korisnika i u kojem se bira između restorana u ponudi (Slika 4.6.).

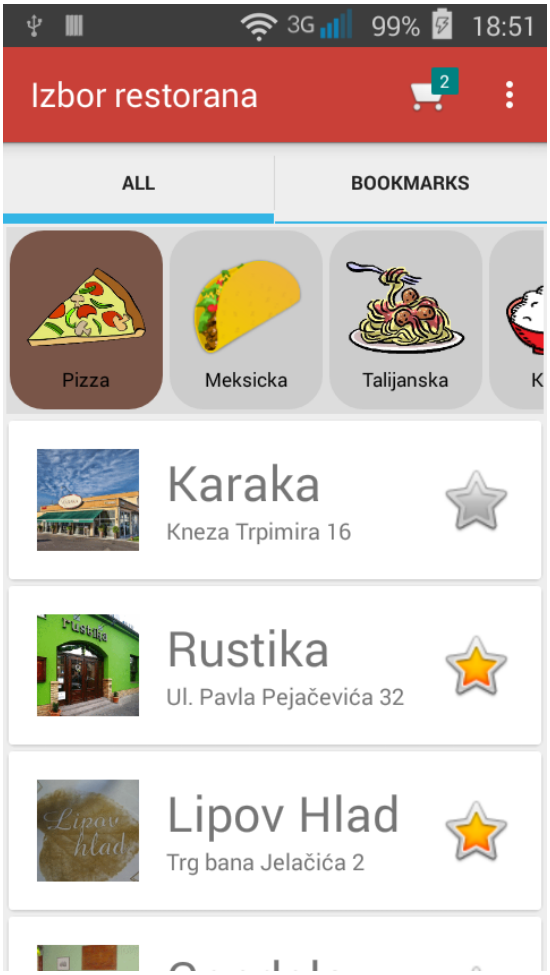
Sl. 4.4.: Izgled obrasca za registraciju.

U zaglavlju zaslona je prikazana akcijska traka (engl. *Action Bar*) u kojoj registriran korisnik vidi ikonu košarice kraj koje se prikaže broj s obzirom koliko stavki se nalazi u košarici (Slika 4.6). Ispod akcijske trake je vidljiv *TabHost widget* u kojem postoji izbor između svih i označenih restorana. Podaci o restoranima i filterima se nalaze u Firebase bazi podataka te se učitavaju prilikom kreiranja *Activity-a*. Kôd za učitavanje restorana i filtera vidljiv je u *ChooseActivity.java*



Sl. 4.5: Izgled kostura jednog restorana.

Prikaz svih filtera i restorana je omogućen *RecyclerView widgetom* u čiji adapter šaljemo listu svih filtera, tj. restorana a on pomoću ranije generiranog izgleda kostura za jedan filter i restoran (Slika 4.5) dinamički prikazuje sve filtere i restorane.



Sl. 4.6: *ChooseActivity* s označenim filtrom za prikaz restorana koji nude pizze.

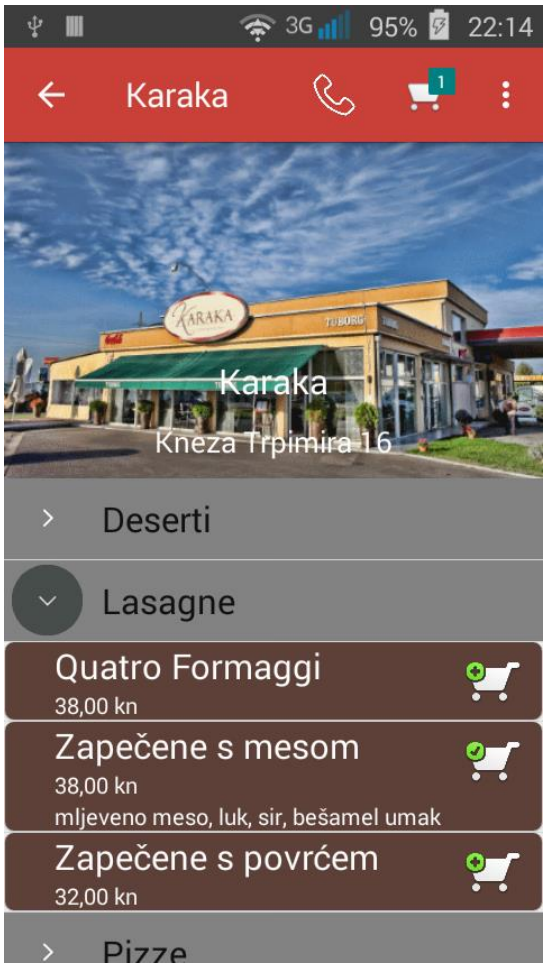
Označivanjem jednog od restorana pokreće se *RestaurantActivity.java*. koji u sebi sadrži sliku ili logo restorana, njegov naziv i adresu te kategorije jela sadržane u *ExpandableListView widgetu* koje reagiraju na pritisak pri kojem se raširе i prikazuju jela te kategorije također sadržane unutar niza (engl. *Array list*) i koje se dinamički prikazuju prema kosturu vidljivom na sa slici 4.7.

Sl. 4.7: Izgled kostura prikaza jela.

TextView Sastojci je po zadanoj postavci nevidljiv te se prikazuje dodirom na bilo koji dio *layout-a* kostura osim na ikonu košarice i može se vidjeti u programskom kôdu 4.6. Jela dodajemo u košaricu pritiskom na ikonu košarice, pri čemu ona mijenja ikonu. Primjer prikaza restorana gdje je jedno jelo dodano u košaricu je vidljivo u slici 4.8.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/ingredients"
    android:textColor="@android:color/white"
    android:text="@string/ingredients" />
```

Programski kôd 4.6: *TextView* Sastojci.

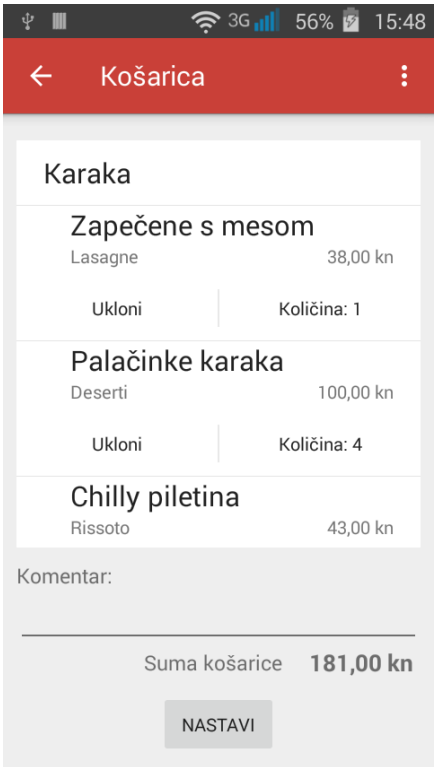


Sl. 4.8: *RestaurantActivity* s podacima restorana Karaka.

Ako kategorija jela sadrži više veličina jela i za nju se ne može definirati fiksna cijena, umjesto cijene se prikazuje poseban tekst sadržaja „Izaberi veličinu“ i klikom na košaricu se otvara *Dialog* s tekstom u naslovu naziva jela, u kojem se dodaje navedeno jelo u košaricu s obzirom na veličinu (Slika 4.9).

Sl. 4.9: *Prikaz Dialog View-a* za jelo koje ima više veličina.

Završavanjem dodavanja stavki u košaricu pritiskom na ikonicu košaricu u akcijskog traci otvara se prozor košarice (Slika 4.10) koji prikazuje jela dodana u košaricu u formatu naziv restorana kao naslovu rastezljive liste i s obzirom na broj stavki u košarici, toliki broj djece rastezljive liste. Format djece je jednak kao i u prikazu restorana s dodanim funkcijama za uklanjanje i mijenjanje količine izabranog jela.

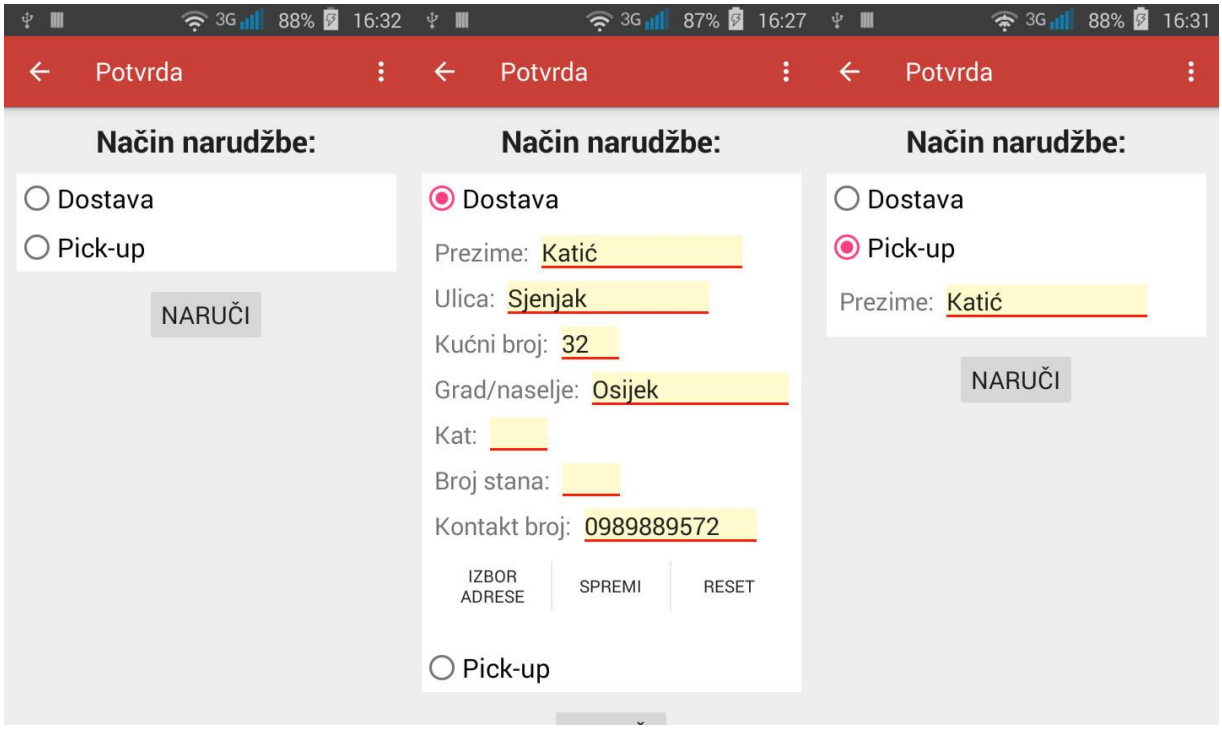


Sl. 4.10: *CartActivity* sa stavkama u košarici iz restorana Karaka.

Uklanjanje jela daje prozor upozorenja za potvrdu akcije, dok biranje količine jela je formatirano pomoću *NumberPicker widget-a* koji omogućava izbor broja, tj. količine izabranog jela iz klizajućeg izbornika koji se može vidjeti na slici 4.11. Završetkom izbora količine, može se dodati komentar za narudžbu primatelju narudžbi u restoranu u slučaju da korisnik želi različite izmjene na jelu koje nisu navedene u prikazu jela. Ispod prostora za komentar aplikacija automatski računa ukupnu sumu košarice s obzirom na sva jela i njihovu količinu. Klikom na gumb „Nastavi“ otvara se *ConfirmActivity* u kojem se izabire način narudžbe. U slučaju da su u košarici stavke iz više restorana, klikom na gumb otvara se upozorenje kako je moguće naručiti samo iz jednog restorana te za nastavak procesa narudžbe je potrebno obrisati stavke iz drugih restorana.

Sl. 4.11: Prikaz *NumberPicker widget-a.*

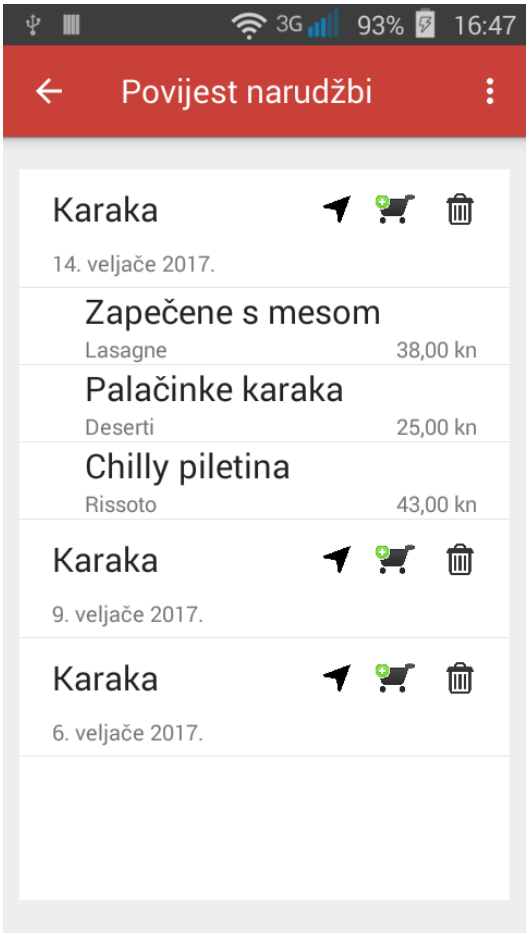
U *ConfirmActivity-u* način narudžbe se bira načinom pritiska na *RadioButton* kraj kojeg je naziv metode vidljivo u slici 4.12. Prvi gumb otvara *layout* u kojem se potvrđuje adresa dostave s mogućnošću biranja između već spremljenih adresa, spremanja nove adresa i brisanja svih polja u *layout-u*. Izbor adrese prikazuje sve spremljene adrese u bazi podataka i izabire se klikom na određenu adresu pri čemu je i moguće izbrisati spremljenu adresu pritiskom na ikonu košarice vidljivo na slici 4.13. Izborom *Pick-up* metode potrebno je dodati prezime uz koje će se vezati narudžbu. Klik na gumb „Naruči“ otvara prozor upozorenja na kojem je potrebno potvrditi narudžbu ili izbacuje grešku ako nisu unesena obavezna polja.



Sl. 4.12.: *ConfirmActivity* bez izabrane metode, s izabranom metodom dostave i *Pick-up* metodom.

U slučaju uspješnog postavljanja narudžbe, narudžba zajedno s žetonom (engl. *Token*) uređaja šalje se u bazu podataka te će se ista objavljivati restoranu u web aplikaciji. Prilikom potvrde narudžbe će se putem žetona uređaja s kojeg je narudžba postavljena uređaju dostaviti obavijest o potvrdi.

Postavljanjem narudžbe na uređaju se otvara *OrderHistoryActivity.java* koji prikazuje povijest narudžbi poslaganu kronološki, od najnovije prema najstarijom. Prikaz pojedine narudžbe omogućava korisniku otvaranje *Google Maps* aplikacije za prikaz uputa do restorana nakon potvrđene obavijesti upozorenja, omogućavanje dodavanje stavki iz izvršene narudžbe u košaricu i brisanje narudžbe iz povijesti. *Activity* s tri izvršene narudžbe, pri kojoj je zadnja raširena i prikazuje stavke iz narudžbe je vidljiv na slici 4.13.



Sl. 4.13: *OrderHistoryActivity* s jednom raširenom narudžbom

Metoda za primanje tijela poruke iz obavijesti dok je aplikacija otvorena na uređaju vidljiva je u kôdu 4.7. Metoda prima atribut vrste *String*, stvara novi *Intent* koji se pokreće pritiskom na obavijest, postavlja zadanu melodiju obavijesti kao zvučnu oznaku obavijesti i dodaje u naslov obavijesti naziv aplikacije, te u tijelo obavijesti postavlja tekst iz ulaznog *String-a*.

```
private void sendNotification(String messageBody) {
    Intent intent = new Intent(this, OrderHistoryActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,
      intent, PendingIntent.FLAG_UPDATE_CURRENT);

    Uri defaultSoundUri=
      RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    NotificationCompat.Builder notificationBuilder = new
      NotificationCompat.Builder(this)
            .setSmallIcon(R.drawable.ic_checkout)
            .setContentTitle(getText(R.string.app_name))
            .setContentText(messageBody)
```

```
            .setAutoCancel(true)
            .setSound(defaultSoundUri)
            .setContentIntent(pendingIntent);

    NotificationManager notificationManager =
            (NotificationManager)
      getSystemService(Context.NOTIFICATION_SERVICE);

    notificationManager.notify(0, notificationBuilder.build());
}
```

Programski kôd 4.7: Metoda *sendNotification(String);*

5. ZAKLJUČAK

U sklopu završnog rada je prikazan proces izrade i načina rada usluge za centralizirano naručivanje hrane unutar nekog grada na operacijskom sustavu Android. Razvojem aplikacije usvojena su vrijedna znanja objektno-orijentiranog programiranja, razvoja Android aplikacija i korištenja različitih programskih alata za povezivanje s poslužiteljem.

Google Firebase je, kao besplatan servis, omogućio pohranu svih podataka na poslužitelj i dohvaćanje podataka na različite operacijske sustave bez čega ovaj rad ne bi bilo moguće izvesti.

Aplikacija je napravljena tako da bude jednostavna za kasnije dorade podataka i implementaciju novih elemenata. Među prvim stvarima kojima će aplikacija biti poboljšana su prevođenje aplikacije na engleski jezik, mogućnost upisa recenzija restorana i ocjenjivanja, umetanje mogućnosti prijave preko računa društvenih mreža, sigurnosna provjera adrese elektroničke pošte, mogućnost resetiranja zaporke te mogućnost unaprijednog plaćanja pomoću servisa za online plaćanja kao što je *PayPal*. Pored toga, u aplikaciji će se izmijeniti dizajn cijele aplikacije i ubaciti jednostavne animacije zbog ugodnije interakcije s korisnicima.

S većim mogućnostima ali jednako jednostavnom interakcijom s korisnicima, aplikacija može približiti korisnike s restoranima te povećati kvalitetu usluge u restoranima s obzirom na njihove recenzije.

LITERATURA

[1] http://www.informatika.buzdo.com/pojmovi/mobile-3.htm , lipanj 2016.

[2] https://developer.android.com/studio/intro/index.html , lipanj 2016.

[3] https://developer.android.com/studio/releases/sdk-tools.html , lipanj 2016.

[4] Gramlich, N. 2008. - ''andbook!'', http://andbook.anddev.org/files/andbook.pdf , lipanj 2016.

[5] Extensible Markup Language (XML), http://www.w3.org/XML/ , lipanj 2016.

[6] Kirasić, D.''XML Tehnologija i Primjena u Sustavima Procesne Informatike'', http://www.fer.unizg.hr/_download/repository/mipro_xml_tekst.pdf , lipanj 2016.

SAŽETAK

U završnom radu opisan je postupak izrade, instalacije i pokretanje aplikacije za pametne pokretne telefone koji koriste operacijski sustav Android uz korištenje svih potrebnih alata za razvoj. Objašnjeni su temeljni dijelovi Android aplikacija te njihove funkcije.

Aplikacija *Centralized Food Order* je usluga koja prikuplja podatke restorana na području Osijeka, objedinjuje ih na jedno mjesto i zatim omogućava narudžbu hrane bez potrebe pozivanja restorana. Baza podataka koja se nalazi na mrežnom poslužitelju se spaja s aplikacijom i s obzirom na pohranjene podatke prikazuje restorane i njihovu ponudu. Baza podataka također sadrži i informacije o korisnicima čime se olakšava samo korištenje aplikacije i povećava sigurnost od lažnih narudžbi. Detaljno su opisani koraci rješavanja problematike zadatka, te su kroz slike zaslona prikazani koraci do narudžbe hrane. Za razvoj usluge koristio se programski jezik Java.

Ključne riječi: operacijski sustav Android, aplikacija, pametni telefon, dostava hrane

ABSTRACT

Android client application for centralized ordering food in Osijek

This paper describes the process of making, installing and launching application for smart mobile phones that use Android operating system with the use of all necessary tools for development. Fundamentals of Android applications and their functions are explained. Application *Centralized Food Order* is a service that collects restaurant data in the Osijek area, combines them into one place and offers food delivery order without the need to make a call. The database, located on a network server is connected with the application and with regard to the stored data shows restaurants and their offer. The database also contains information about the users making the application easier to use and increases the security of fraudulent orders. There are detailed step-by-step instructions regarding the completion of the task and step-by-step screen images how to order food. Java programming language was used for development.

Keywords: Android operating system, applications, smartphone, food

ŽIVOTOPIS

Valentin Katić rođen je 09. studenog 1993. u Vinkovcima. 2000. godine započinje svoje školovanje u Osnovnoj školi Matije Antuna Reljkovića u Cerni, nakon završetka osnovne škole, 2008.godine upisuje Gimnaziju Matije Antuna Reljkovića u Vinkovcima, Prirodoslovno-matematički smjer. Položenom državnom maturom, 2012.godine upisuje Elektrotehnički fakultet u Osijeku, stručni studij elektrotehnike, smjer informatika.