



Prolećni semestar, 2021/2022

# Food Order

PREDMET: CS330 – Razvoj mobilnih aplikacija

**Profesor:** Vladimir Milićević

**Studenti:** Aleksandar Ilić 3939,

Jovan Spasić4026

## Sadržaj

Opis projekta.....	3
Opis funkcionalnosti .....	4
Struktura aplikacije .....	5
Korisničko uputstvo .....	11
Zaključak .....	21
Literatura .....	21

## Opis projekta

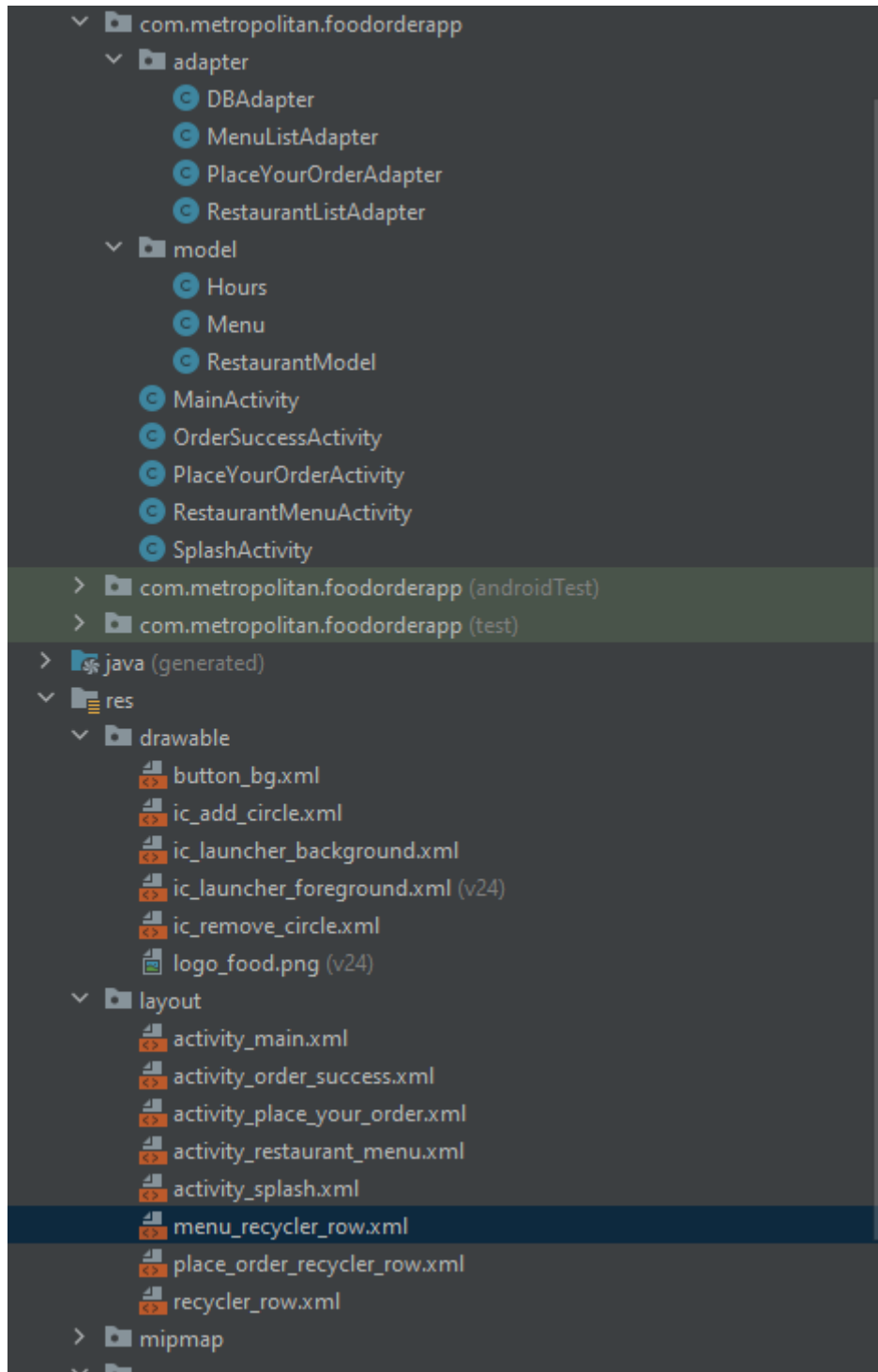
Kao temu projektnog zadatka odabrali smo aplikaciju za poručivanje hrane (poput aplikacije glovo). Osnovna zamisao aplikacije je da sadrži listu restorana od kojih korisnik bira onaj iz kog želi da poruči hranu, zatim biranje proizvoda iz odabranog restorana i u kojoj količini. Nakon dodavanja svih proizvoda u korpu, korisnik bira jednu od dve opcije za izvršavanje porudžbine. Ukoliko odaberemo opciju za dostavljanje (opcija delivery), uz osnovne podatke o korisniku (ime, adresa, broj kartice, itd.) u krajnju cenu se uračunavaju troškovi dostave. Ukoliko se odabere druga opcija (opcija pick up) od podataka su potrebni samo ime i broj kartice, a troškovi za dostavu su isključeni. Podaci koji su prikazani u aplikaciji biće kreirani uz pomoć API-a, a porudžbine korisnika upisane u bazu preko SQLite

## Opis funkcionalnosti

Aplikacija radi tako što se pri pokretanju aplikacija na početnom ekranu može videti prikaz restorana u vidu liste kartica od kojih korisnik može odabrati onaj koji mu odgovara. Nakon odabira ekrana korisniku se prikazuju proizvodi koje taj restoran nudi od kojih korisnik može odabrati koji proizvod želi da doda u korpu klikom na dugme Add to cart. Nakon klika na dugme Add to cart korisnik može odabrati i koju količinu određenog proizvoda želi da doda u korpu. Nakon toga, korisnik bira opciju Checkout gde može odabrati da li želi dostavu na kućnu adresu. Ukoliko korisnik želi dostavu na kućnu adresu potrebno je uneti ime, adresu, grad, državu, postanski broj, broj kartice, vreme do isteka kartice i pin, a u ukupnu cenu se uračunavaju i troškovi dostave. Ukoliko se korisnik ne opredeli za opciju dostave na kućnu adresu, onda unosi samo ime, broj kartice, vreme do isteka kartice i pin. Nakon ovog koraka korisniku se prikazuje ekran na kome se može videti da je porudžbina uspela. Klikom na dugme Done ispisuju se podaci porudžbine i vraća se na početni ekran.

## Struktura aplikacije

Osnovna podela strukture aplikacije se može videti na slici gde imamo podelu po aktivnostima, modelima, adapterima, drawable xml elementima i layout xml elementima.



DBAdapter je klasa koja nam služi za rad sa bazom podataka. U ovoj klasi su definisane metode za kreiranje baze, kreiranje tabele, i rad sa podacima. U nastavku se nalaze neke od metoda ove klase.

```
///--insert() food order in db--  
public long insertFoodOrder(String customer_name, String customer_address, String customer_city, String customer_state,  
    int customer_zip, String card_number, int card_expiry, int card_pin, int total_items,  
    String subtotal, String delivery_charge, String total) {  
    ContentValues initialValues = new ContentValues();  
    initialValues.put(KEY_NAME, customer_name);  
    initialValues.put(KEY_ADDRESS, customer_address);  
    initialValues.put(KEY_CITY, customer_city);  
    initialValues.put(KEY_STATE, customer_state);  
    initialValues.put(KEY_ZIP, customer_zip);  
    initialValues.put(KEY_CARD_NUMBER, card_number);  
    initialValues.put(KEY_CARD_EXPIRY, card_expiry);  
    initialValues.put(KEY_CARD_PIN, card_pin);  
    initialValues.put(KEY_TOTAL_ITEMS, total_items);  
    initialValues.put(KEY_SUBTOTAL, subtotal);  
    initialValues.put(KEY_DELIVERY_CHARGE, delivery_charge);  
    initialValues.put(KEY_TOTAL, total);  
    return db.insert(DATABASE_TABLE, nullColumnHack: null, initialValues);  
}  
  
///--findAll() food orders--  
public Cursor getAllFoodOrders() {  
    return db.query(DATABASE_TABLE, new String[] {KEY_NAME, KEY_ADDRESS, KEY_CITY, KEY_STATE, KEY_ZIP, KEY_CARD_NUMBER,  
        KEY_CARD_EXPIRY, KEY_CARD_PIN, KEY_TOTAL_ITEMS, KEY_SUBTOTAL, KEY_DELIVERY_CHARGE, KEY_TOTAL}, selection: null, selectionArgs: null, groupBy: null,  
    );  
}
```

```
///--deleteById() food ordered--  
public boolean deleteFoodOrdered(long food_ordered_id) {  
    return db.delete(DATABASE_TABLE, whereClause: KEY_INDEX + "=" + food_ordered_id, whereArgs: null) > 0;  
}  
  
public void clearDB() {  
    db.execSQL("DROP TABLE IF EXISTS foodOrdered;");  
    db.execSQL(DATABASE_CREATE);  
}
```

MenuListAdapter je klasa koja nam služi za listom proizvoda koju nudi neki restoran. U ovoj klasi su definisane metode za prikupljanje podataka i prikaza podataka u sklopi liste proizvoda, kao i metode za dodavanje u korpu i podešavanja količine proizvoda.

PlaceYourOrderActivity je klasa koja nam služi za rad sa podacima koji se unose na stranici gde korisnik završava porudžbinu, tj. unosi podatke za porudžbinu.

RestaurantListAdapter je klasa koja sadrži metode koje nam služe za rad sa podacima o restoranima koji se prikazuju na početnoj strani aplikacije.

Zatim imamo modele Hours, Menu i RestaurantModel i ovo su klase gde imamo definisane prikaze sata i dana u nedelji, prikaze proizvoda i prikaze restorana.

```

public String getTodaysHours() {
    Calendar calendar = Calendar.getInstance();
    Date date = calendar.getTime();
    String day = new SimpleDateFormat( pattern: "EEEE", Locale.ENGLISH).format(date.getTime());

    switch (day) {
        case "Sunday":
            return this.Sunday;
        case "Monday":
            return this.Monday;
        case "Tuesday":
            return this.Tuesday;
        case "Wednesday":
            return this.Wednesday;
        case "Thursday":
            return this.Thursday;
        case "Friday":
            return this.Friday;
        case "Saturday":
            return this.Saturday;
        default:
            return this.Sunday;
    }
}

```

Metoda za prikupljanje vremena i dana u nedelji. Klasa Hours

```

@Override
public void writeToParcel(Parcel parcel, int i) {
    parcel.writeString(name);
    parcel.writeFloat(price);
    parcel.writeInt(totalInCart);
    parcel.writeString(url);
}

```

Metoda za prikaz proizvoda. Klasa Menu

```

protected RestaurantModel(Parcel in) {
    name = in.readString();
    address = in.readString();
    image = in.readString();
    delivery_charge = in.readFloat();
    menus = in.createTypedArrayList(Menu.CREATOR);
}

public static final Creator<RestaurantModel> CREATOR = new Creator<RestaurantModel>() {
    @Override
    public RestaurantModel createFromParcel(Parcel in) { return new RestaurantModel(in); }

    @Override
    public RestaurantModel[] newArray(int size) { return new RestaurantModel[size]; }
};

@Override
public int describeContents() { return 0; }

@Override
public void writeToParcel(Parcel parcel, int i) {
    parcel.writeString(name);
    parcel.writeString(address);
    parcel.writeString(image);
    parcel.writeFloat(delivery_charge);
    parcel.writeTypedList(menus);
}

```

Metode za definisanje prikaza restorana. Klasa RestoranModel

Nakon ovih modela tu su i Activity klase koje rade zajedno sa prethodno navedenim klasama kako bi se omogućile funkcionalnosti na ekranima aplikacije.

MainActivity klasa – u ovoj klasi je definisan rad sa JSON objektima i pokretanja modela sa restoranima.



```

private List<RestaurantModel> getRestaurantData() {
    InputStream is = getResources().openRawResource(R.raw.restaurant);
    Writer writer = new StringWriter();
    char[] buffer = new char[1024];

    try {
        Reader reader = new BufferedReader(new InputStreamReader(is, Charset.forName("UTF-8")));
        int n;
        while ((n = reader.read(buffer)) != -1) {
            writer.write(buffer, 0, n);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    String jsonStr = writer.toString();
    Gson gson = new Gson();
    RestaurantModel[] restaurantModels = gson.fromJson(jsonStr, RestaurantModel[].class);
    List<RestaurantModel> restaurantModelList = Arrays.asList(restaurantModels);

    Log.d("jsonStr", "Print JsonStr " + jsonStr);
    Log.d("tag: RestaurantModelList", "Print restaurantModels " + restaurantModels);
    Log.d("tag: RestaurantModelList", "Print restaurantModelList " + restaurantModelList);
    Log.d("tag: RestaurantModelList", "Print restaurantModelList.get(0).getMenuList() " + restaurantModelList.get(0).getMenuList());

    return restaurantModelList;
}

@Override
public void onItemClick(RestaurantModel restaurantModel) {
    Intent intent = new Intent(getApplicationContext(), RestaurantMenuActivity.class);
    intent.putExtra("name", restaurantModel);
    startActivity(intent);
}

```

OrderSuccessActivity je klasa koja nam služi za prikaz ekrana kada korisnik obavi porudžbinu. Takođe u ovoj klasi se pozivaju metode iz DBAdaptera koje nam služe za upis podataka u bazu i prikaz podataka iz nje. Ove metode se izvršavaju klikom na dugme Done na ovoj stranici.

PlaceYourOrderActivity je klasa gde imamo definisane metode za rad sa ekranom gde korisnik unosi podatke kako bi završio porudžbinu. Ovde imamo definisane metode za izračunavanje ukupne cene (koja je prikazana ispit), metode za izbacivanja greške ukoliko korisnik nije uneo neki od podataka i metoda za prebacivanje na sledeći ekran programa.

```

private void calculateTotalAmount(RestaurantModel restaurantModel) {
    float amount = 0f;
    for (Menu m : restaurantModel.getMenuList()) {
        amount += m.getTotalInCart() * m.getPrice();
    }

    subTotalAmount.setText("$" + String.format("%.2f", amount));

    if (isDeliveryOn == true) {
        deliveryChargeAmount.setText("$" + String.format("%.2f", restaurantModel.getDelivery_charge()));
        amount += restaurantModel.getDelivery_charge();
    }

    totalAmount.setText("$" + String.format("%.2f", amount));
}

```

RestaurantMenuActivity je klasa koja sadrži metode koje nam služe za dodavanje proizvoda u korpu, odabira količine proizvoda kao i nastavka na sledeći ekran.

Na kraju imamo i SplashActivity klasu koja sadrži metodu gde se prilikom učitavanja aplikacije prikazuje logo aplikacije.

```

package com.merit.spezzian.restaurantapp;

import ...

public class SplashActivity extends AppCompatActivity {

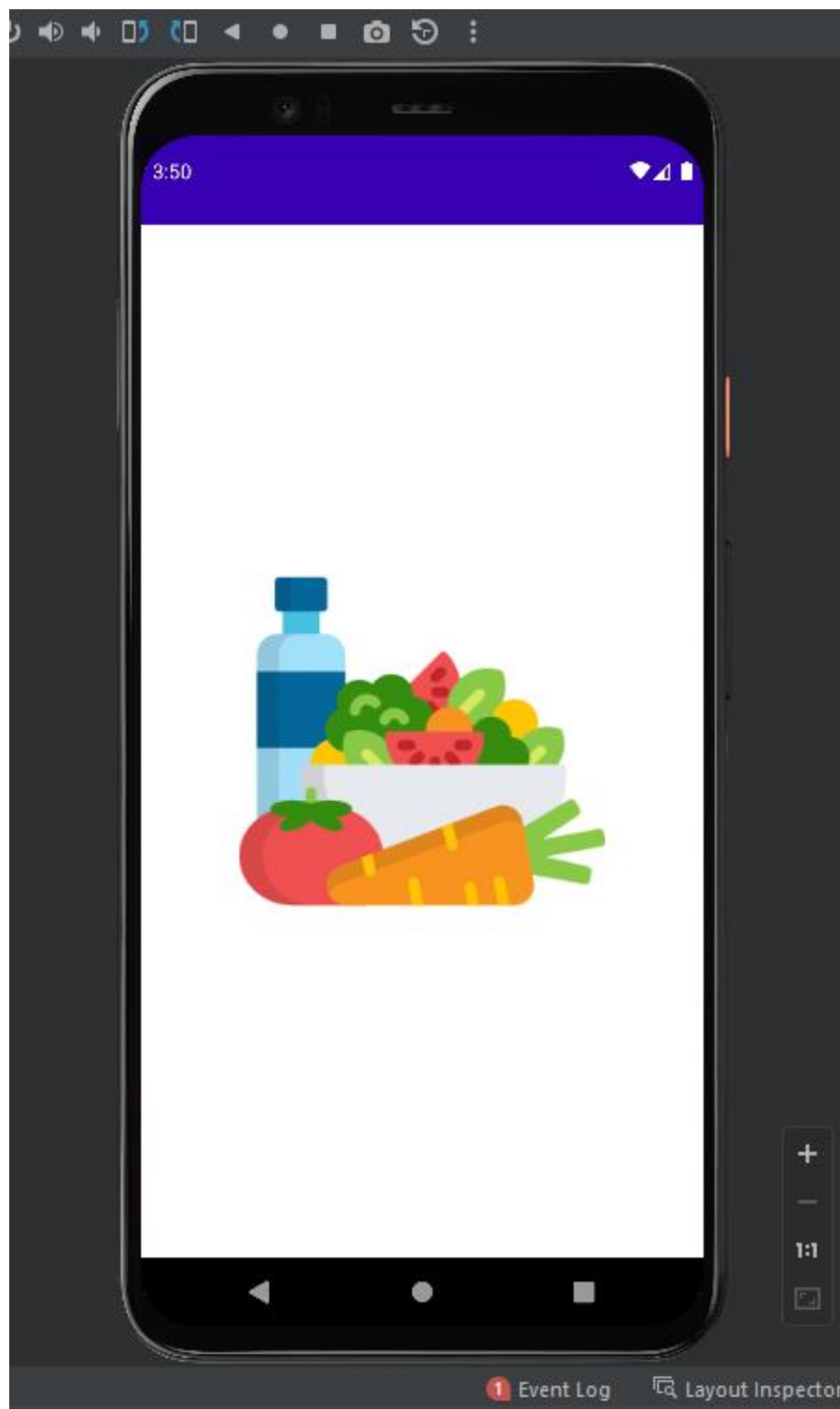
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);

        ActionBar actionBar = getSupportActionBar();
        actionBar.hide();

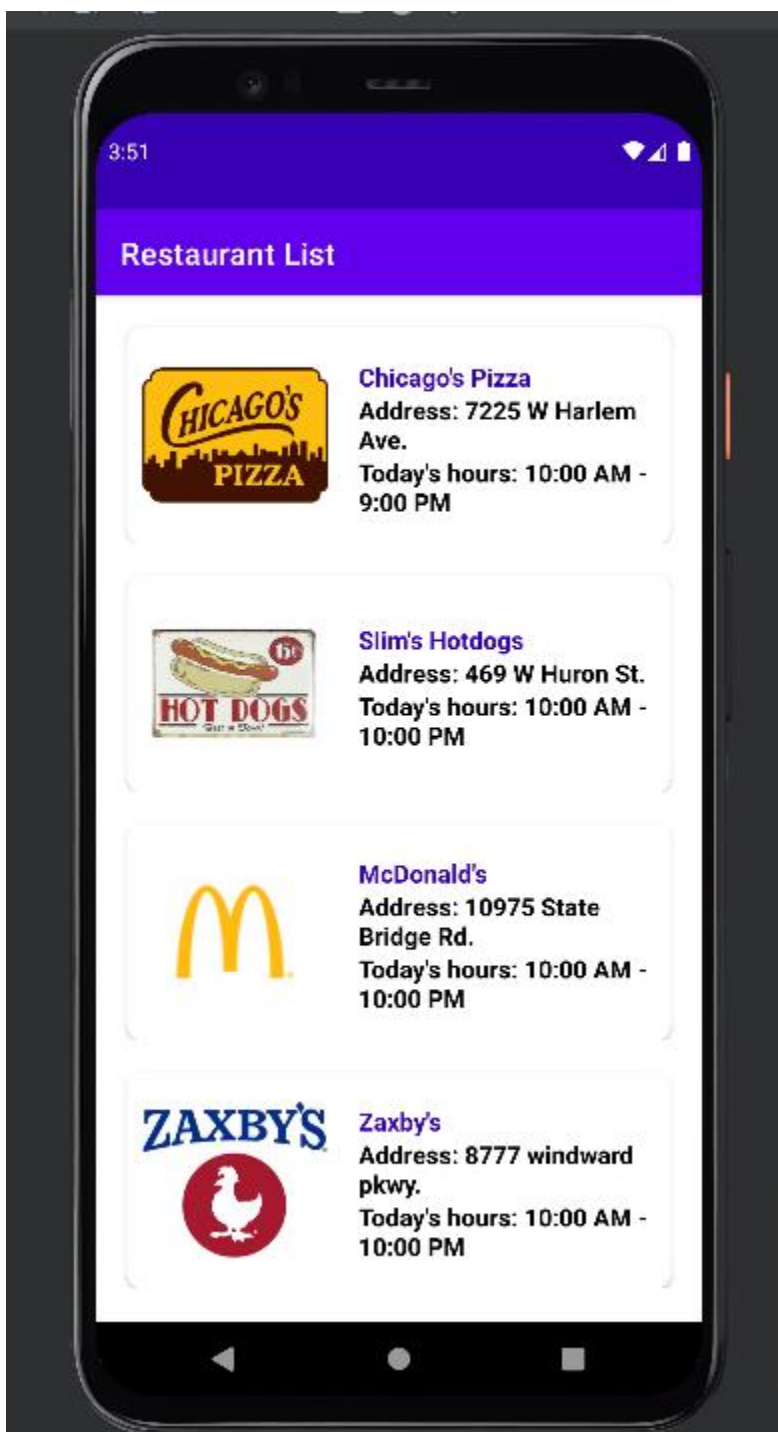
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                startActivity(new Intent( packageContext: SplashActivity.this, MainActivity.class));
                finish();
            }
        }, delayMillis: 2000);
    }
}

```

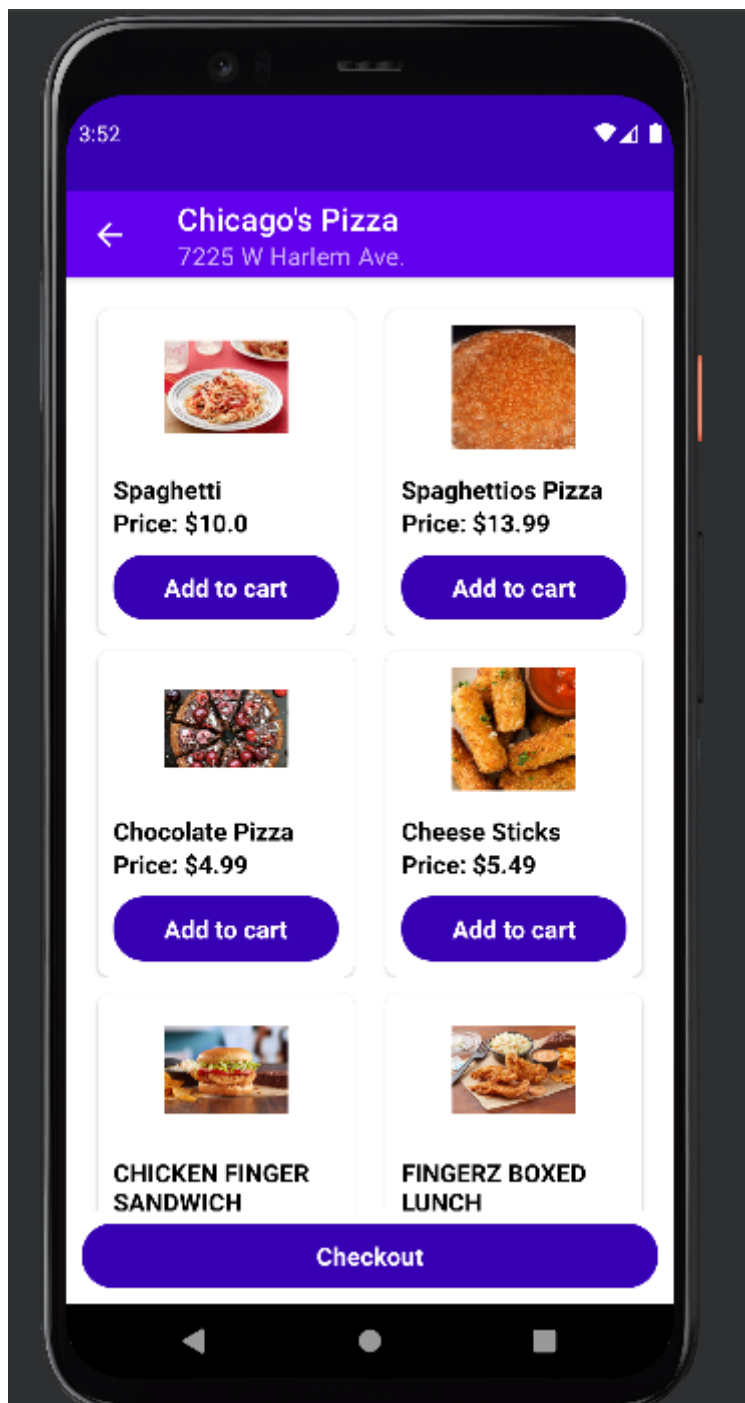
## Korisničko uputstvo



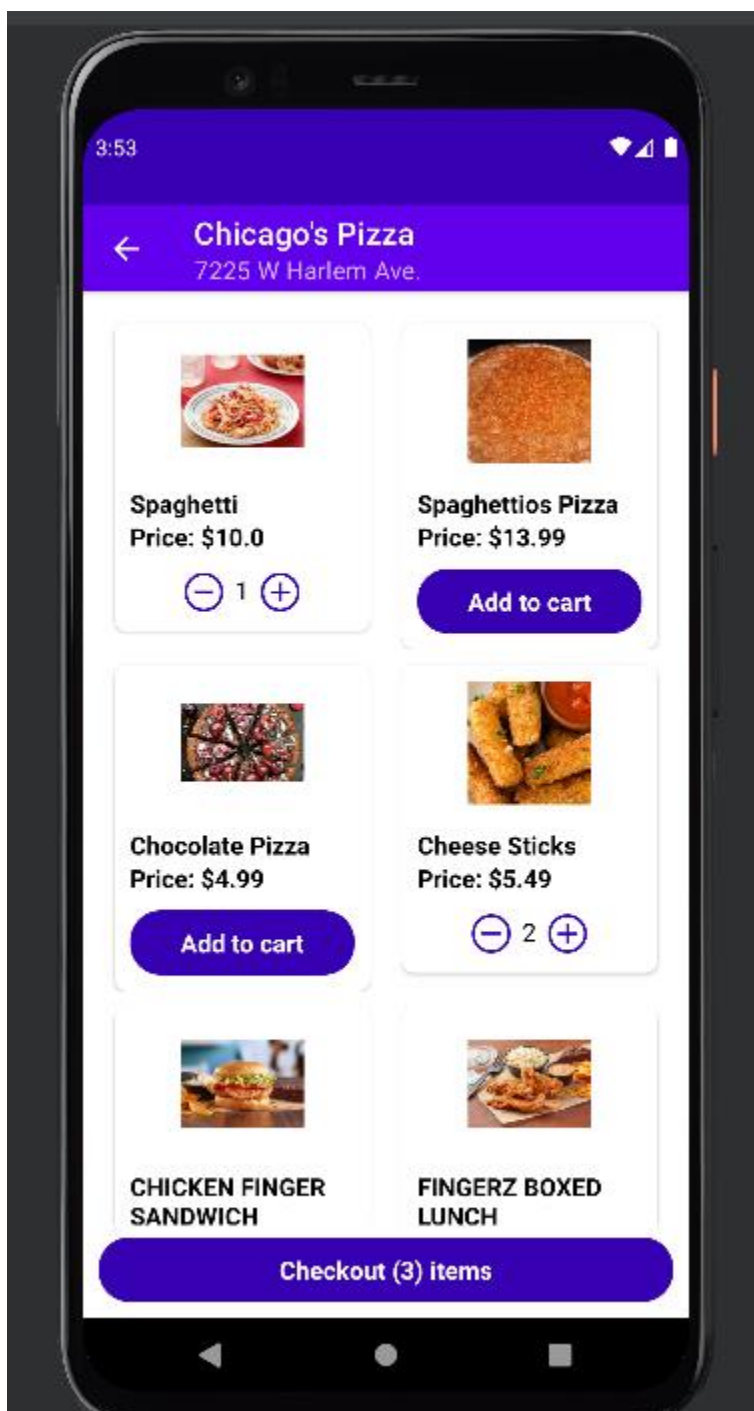
Prilikom pokretanja aplikacije možemo videti logo same aplikacije koji traje 2 sekunde. Nakon ovoga učitava se sledeći prikaz.



Ovde možemo videti listu restorana od kojih je moguće izabrati jedan iz kojeg želimo poručiti neki proizvod. Nakon odabira ekrana imamo sledeći prikaz.

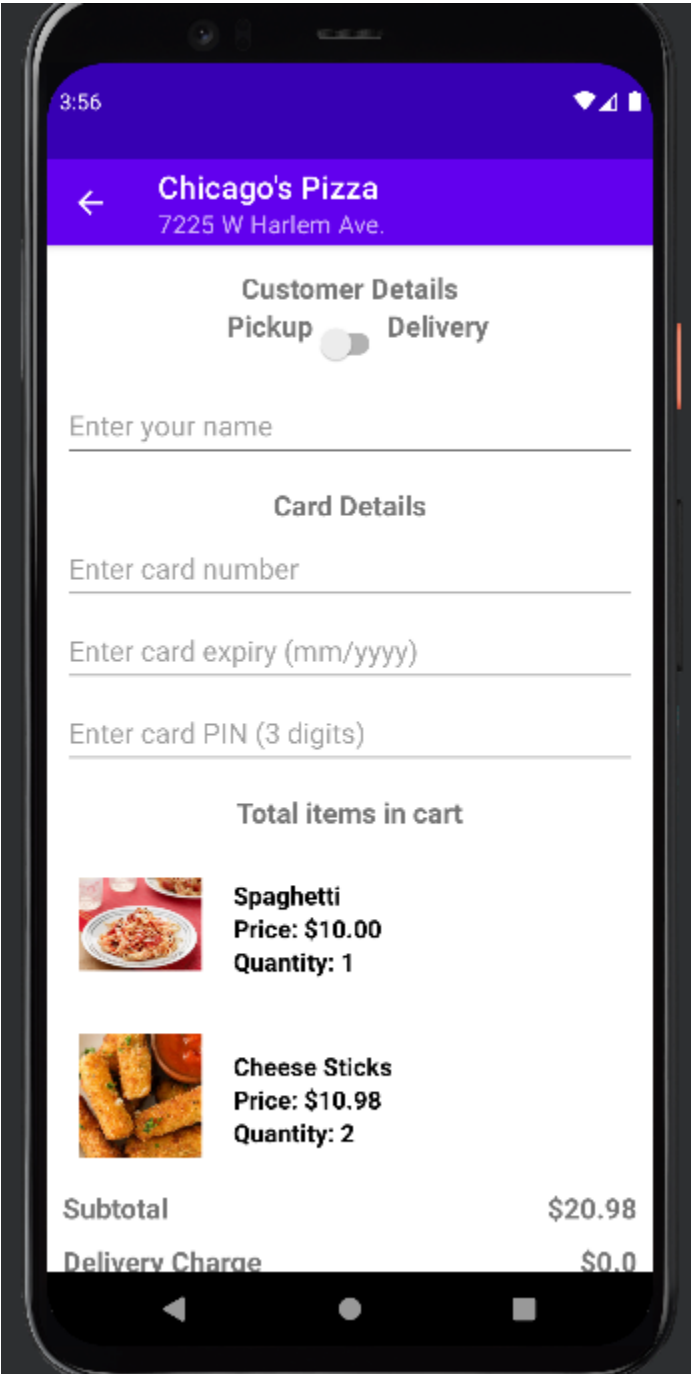


Na ovom ekranu vidimo listu proizvoda koje nam taj restoran nudi i možemo izabrati one koje ćemo dodati u korpu i u kojoj količini.



Kao što vidimo dodali smo 3 proizvoda u korpu ali korisnik je uvek u mogućnosti da doda još neki proizvod ili poveća ili smanji količinu nekog od odabranih proizvoda. Ukoliko korisnik smanji količinu nekog proizvoda na 0 pojavljuje se dugme Add to cart i taj proizvod se briše iz korpe.

Klikom na dugme Checkout imamo sledeći prikaz.



The screenshot shows a mobile application interface for 'Chicago's Pizza' at 7225 W Harlem Ave. The screen is divided into sections for Customer Details, Card Details, and a cart summary. The 'Customer Details' section has a toggle for 'Pickup' (selected) and 'Delivery'. Below it is a text input field for 'Enter your name'. The 'Card Details' section has three text input fields for 'Enter card number', 'Enter card expiry (mm/yyyy)', and 'Enter card PIN (3 digits)'. The 'Total items in cart' section lists two items: 'Spaghetti' (Price: \$10.00, Quantity: 1) and 'Cheese Sticks' (Price: \$10.98, Quantity: 2). At the bottom, the 'Subtotal' is \$20.98 and the 'Delivery Charge' is \$0.00.

3:56

Chicago's Pizza  
7225 W Harlem Ave.

Customer Details  
Pickup ☒ Delivery

Enter your name


Card Details


Enter card number

Enter card expiry (mm/yyyy)

Enter card PIN (3 digits)

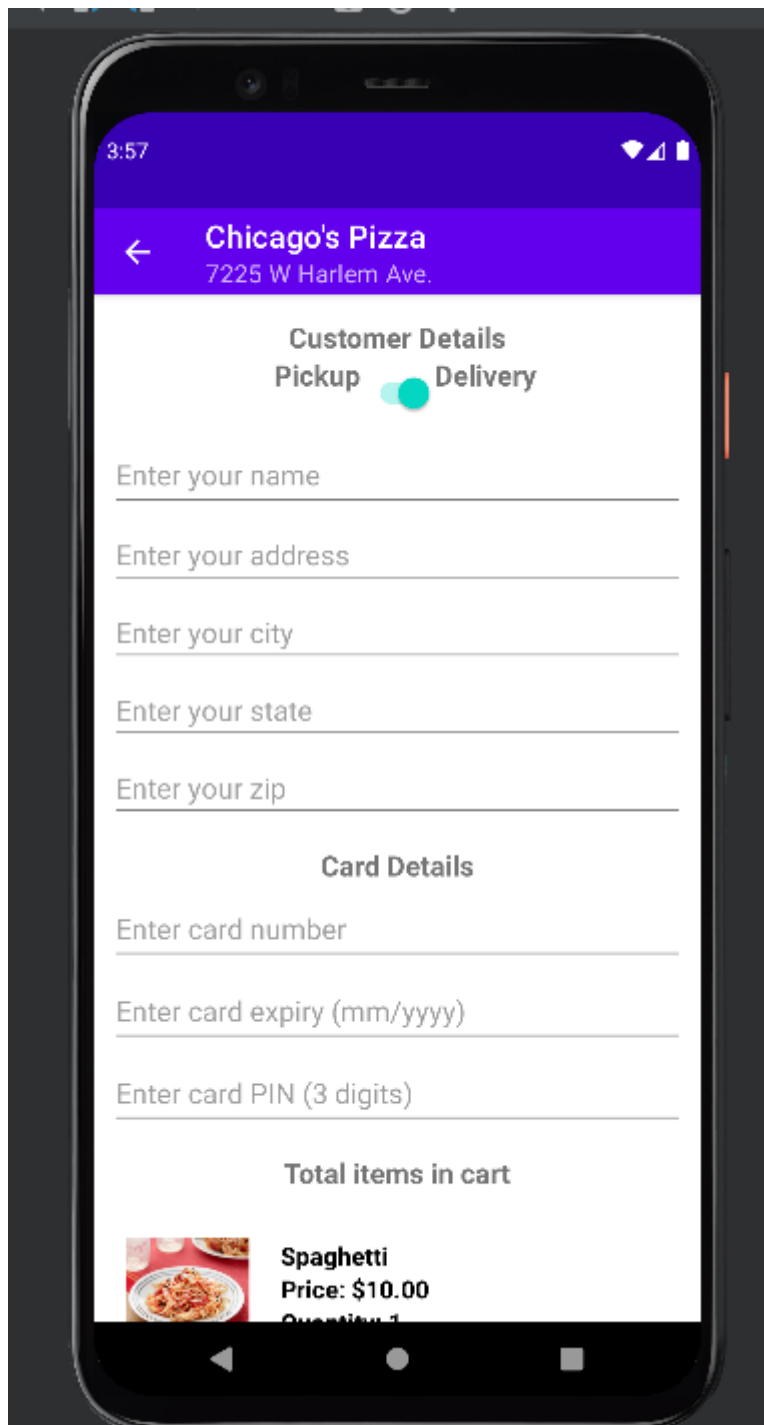
Total items in cart

 **Spaghetti**  
Price: \$10.00  
Quantity: 1

 **Cheese Sticks**  
Price: \$10.98  
Quantity: 2

Subtotal \$20.98

Delivery Charge \$0.00



Kao što se može videti postoji različit broj polja gde korisnik unosi svoje podatke a broj polja zavisi od toga da li je korisnik uključio opciju za dostavu na kućnu adresu ili ne. Nakon popunjavanja svih polja korisnik treba kliknuti na dugme Place your order.



3:59

Chicago's Pizza  
7225 W Harlem Ave.

Enter your name


Card Details


Enter card number

Enter card expiry (mm/yyyy)

Enter card PIN (3 digits)

Total items in cart

 **Spaghetti**  
Price: \$10.00  
Quantity: 1

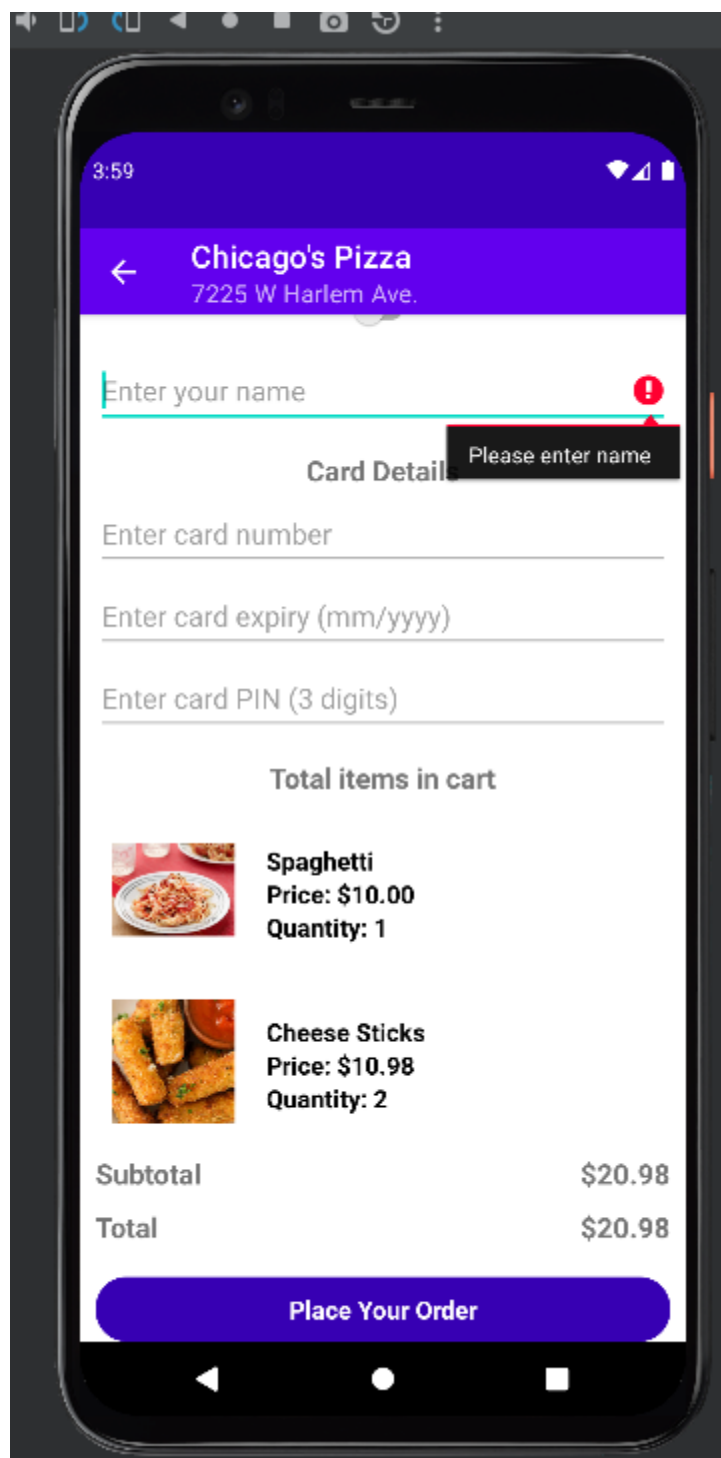
 **Cheese Sticks**  
Price: \$10.98  
Quantity: 2

Subtotal \$20.98

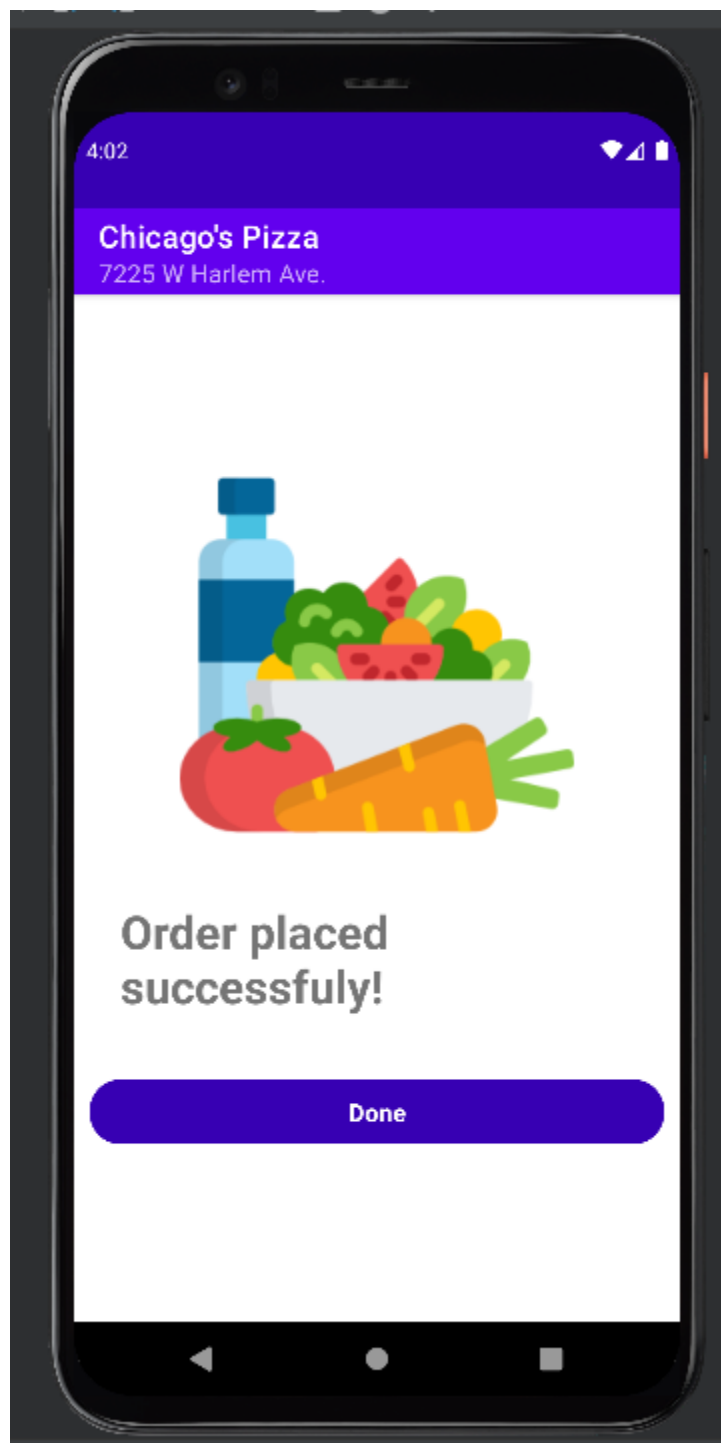
Total \$20.98

Place Your Order

Ukoliko neki od podataka nisu uneti javlja se opomena korisniku da unese podatke.



Nakon unosa podataka i klika na dugme Place your order možemo videti sledeći prikaz.



Klikom na dugme Done prikazuju nam se podaci koje smo uneli u okviru Toast-a i aplikacija nas odvodi na početni ekran.

4:02



## Chicago's Pizza

7225 W Harlem Ave.



Order placed  
successfully

customer\_name: js  
customer\_address:  
customer\_city:  
customer\_state:  
customer\_zip: 0  
card\_number: 123  
card\_expiry: 456  
card\_pin: 789  
total\_items: 2  
subtotal: \$20.98  
delivery\_charge: \$5.00  
total: \$20.98

## Zaključak

Cilj ovog projekta je bio da omogućimo korisnicima da vrše porudžbinu hrane preko mobilne aplikacije. Korišćenjem znanja stečenog na predmetu uspešno smo realizovali ovu ideju. Elementi sa predmeta koji su korišćeni u ovom projektu su: rad sa aktivnostima, rad sa dugmićima, omogućavanje prikaza, različite vrste prikaza itd.

## Literatura

Activity. (n.d.). Android Developers. Retrieved June 6, 2022, from <https://developer.android.com/reference/android/app/Activity>

ListActivity. (n.d.). Android Developers. Retrieved June 6, 2022, from <https://developer.android.com/reference/android/app/ListActivity>