Dokumentacija projekta

Moji recepti

1. Uvod

1.1. Opis projekta

Moji Recepti je Android aplikacija čiji je cilj da korisnicima omogući jednostavno upravljanje i pregled svojih omiljenih recepata. Aplikacija omogućava dodavanje novih recepata, pretragu po nazivu, razvrstavanje po vremenu dana (doručak, ručak, večera itd.), kao i pregled sastojaka i načina pripreme. Fokus aplikacije je na **preglednosti**, **lakoći korišćenja** i **personalizaciji sadržaja**.

1.2. Ciljevi aplikacije

Glavni ciljevi razvoja ove aplikacije su:

- **Omogućiti korisnicima da na jednom mestu čuvaju i organizuju svoje recepte.**
- **Pružiti intuitivan i vizuelno prijatan korisnički interfejs.**
- **Automatizovati izbor recepata prema delu dana** (npr. doručak ujutru).
- **Omogućiti brzo dodavanje, pregled i pretragu recepata.**

Aplikacija je **idealna za korisnike** koji žele da prate **sopstvene navike u ishrani, eksperimentišu sa novim jelima** ili jednostavno imaju **sve svoje recepte na dohvat ruke — uvek i svuda.**

1.3. Ključne funkcionalnosti

- **Prikaz recepata po kategorijama** (doručak, ručak, večera, užina, napici)
- **Filtracija recepata na osnovu doba dana**
- **Dodavanje novog recepta sa slikom, sastojcima i uputstvom**
- **Prikaz detalja o receptu:** kalorije i vreme pripreme
- **Označavanje recepata kao omiljenih** i njihovo kasnije pregledanje
- **Pretraga recepata po nazivu**

2. Korišćene tehnologije i alati

Aplikacija **Moji Recepti** je razvijena koristeći **savremene tehnologije i alate** koje su prilagođene za **razvoj mobilnih aplikacija na Android platformi**. U nastavku su opisani glavni alati i tehnologije korišćeni tokom izrade:

2.1. Android Studio

Za razvoj aplikacije korišćeno je **Android Studio**, **zvanično integrisano razvojno okruženje (IDE)** za Android aplikacije. Omogućava rad sa **Java kodom**, **XML fajlovima**, **emulaciju aplikacije** i **povezivanje sa bazom podataka**.

2.2. Java

Aplikacija je implementirana u programskom jeziku **Java**, koji je jedan od najzastupljenijih jezika u **Android razvoju**. Java je korišćena za:

- **logiku aplikacije**
- **obradu korisničkih akcija**
- **upravljanje bazom podataka**
- **komunikaciju između različitih delova aplikacije** (aktivnosti i fragmenti)

2.3. SQLite

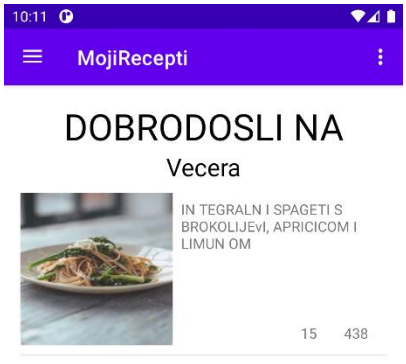
Za **lokalno skladištenje podataka** korišćena je **SQLite baza podataka**. **Recepti, sastojci i omiljene stavke** se čuvaju u **lokalnim tabelama**. Baza omogućava **brzu pretragu, dodavanje, brisanje i ažuriranje podataka** bez potrebe za **internet konekcijom**. Takođe i **malo prostora** zauzima na uređaju.

2.4. Material Design komponente

Aplikacija koristi principe **Material Design-a** za **moderan i intuitivan izgled**. Korišćeni su elementi poput **navigacionog menija, toolbar-a, karti sa zaobljenim ivicama, ikonica i animacija** kako bi **korisničko iskustvo** bilo što prijatnije i **prilagođenije mobilnim uređajima**.

3. Korisnički interfejs (UI)

3.1. Početna



Početni ekran predstavlja **centralno mesto u aplikaciji**. Ovde korisnik odmah vidi **listu recepata koji su automatski filtrirani na osnovu dela dana**:

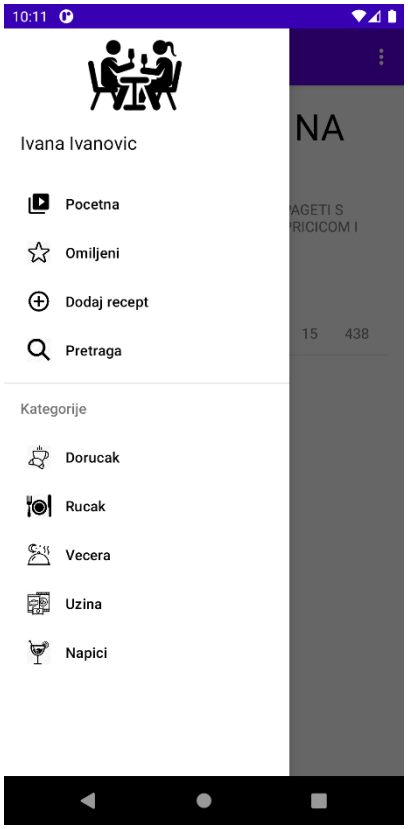
- **Ujutru** se prikazuju recepti iz kategorije **Doručak**
- **Popodne** – recepti za **Ručak**
- **Uveče** – recepti za **Večeru**

Svaki recept je prikazan u posebnoj kartici i sadrži:

- **Naziv recepta**
- **Broj kalorija**
- **Vreme potrebno za pripremu (u minutima)**

Ova početna selekcija **štedi vreme korisniku** i pomaže mu da **lako pronađe ideju za obrok** u skladu s trenutkom dana.

3.2. Navigacija



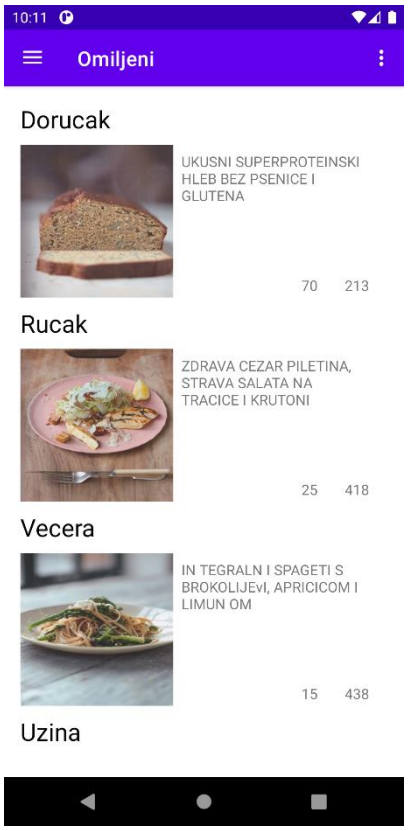
Početna stranica osim prikaza recepata nudi i **meni sa tri osnovne opcije**, koje korisniku omogućavaju **brz pristup najvažnijim funkcijama aplikacije**:

1.1. **Početna** – vraća korisnika na glavni ekran sa receptima
1.2. **Omiljeni** – prikazuje listu jela koja je korisnik označio kao omiljena
1.3. **Dodaj** – vodi korisnika na ekran za unos novog recepta

Pored ovog menija, korisniku su dostupne i **kategorije jela** (**Doručak, Ručak, Večera, Užina, Napici**), koje omogućavaju da se **recepti filtriraju po vrsti obroka, nezavisno od doba dana**. Kategorije su **vizuelno istaknute i lako dostupne**.

Klikom na bilo koju od kategorija, korisnik se preusmerava na **stranicu sa receptima izabrane kategorije**, što dodatno **olakšava navigaciju i pretragu sadržaja**.

3.3. Omiljeni recepti

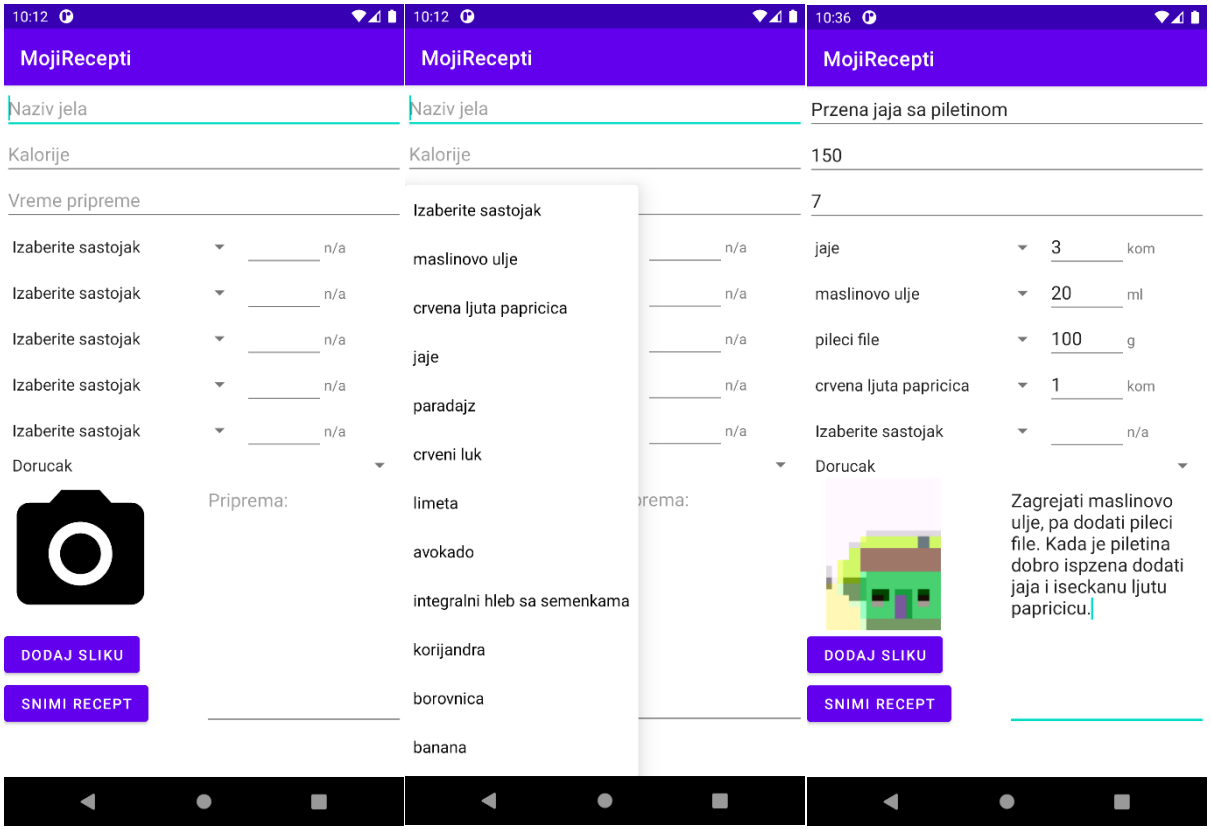


Sekcija sa omiljenim receptima služi za **brz pristup najdražim jelima**. Recepti su prikazani u istom formatu kao na početnoj stranici, sa svim relevantnim informacijama:

- **Naziv**
- **Kalorije**
- **Vreme pripreme**

Korisnik može klikom na bilo koji recept otvoriti njegov **detaljan prikaz**, kao i **ukloniti ga iz omiljenih jednostavnim klikom na zvezdicu**.

3.4. Dodavanje recepta

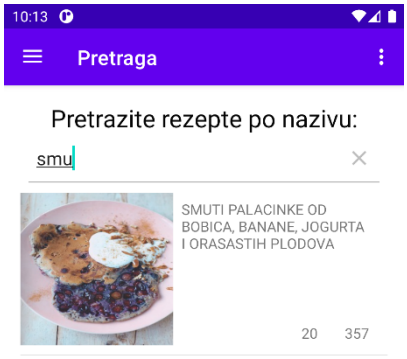


Ovaj ekran korisniku omogućava da **unese potpuno novi recept**. Forma je **detaljna, ali jednostavna za korišćenje**, i obuhvata:

- **Slika recepta**
- **Naziv recepta**
- **Sastojci** – unos teksta sa sastojcima i količinama
- **Priprema** – tekstualno polje za detaljan opis koraka (npr. *"Iseckati luk i propržiti..."*)
- **Broj kalorija**
- **Vreme pripreme u minutima**
- **Kategorija** – izbor između **Doručak, Ručak, Večera, Užina, Napici**

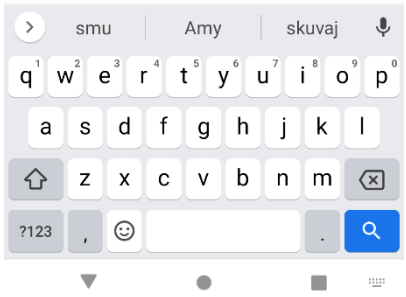
Na dnu se nalazi **dugme za potvrdu i čuvanje recepta**. Ova funkcionalnost omogućava korisniku da **personalizuje aplikaciju i doda sopstvene ideje**.

3.5. Pretraga



Korisnik u svakom trenutku može koristiti **pretragu po nazivu recepta**. Funkcija pretrage se nalazi kao **ikonica lupe** i omogućava **brzo filtriranje sadržaja**.

Unos teksta dinamički ažurira prikaz rezultata, čime korisnik dobija **brzu povratnu informaciju** i **lakše dolazi do željenog jela**.



3.6. Kategorije

Osim automatskog filtriranja po dobu dana, aplikacija omogućava korisniku da **direktno bira recepte prema kategorijama**. Prikazane su kategorije:

- **Doručak**
- **Ručak**
- **Večera**
- **Užina**
- **Napici**

Klikom na kategoriju, prikazuju se svi recepti koji joj pripadaju, u poznatom formatu **kartica sa osnovnim informacijama**. Ovo pruža korisniku **dodatnu fleksibilnost u pretraživanju i planiranju obroka**.

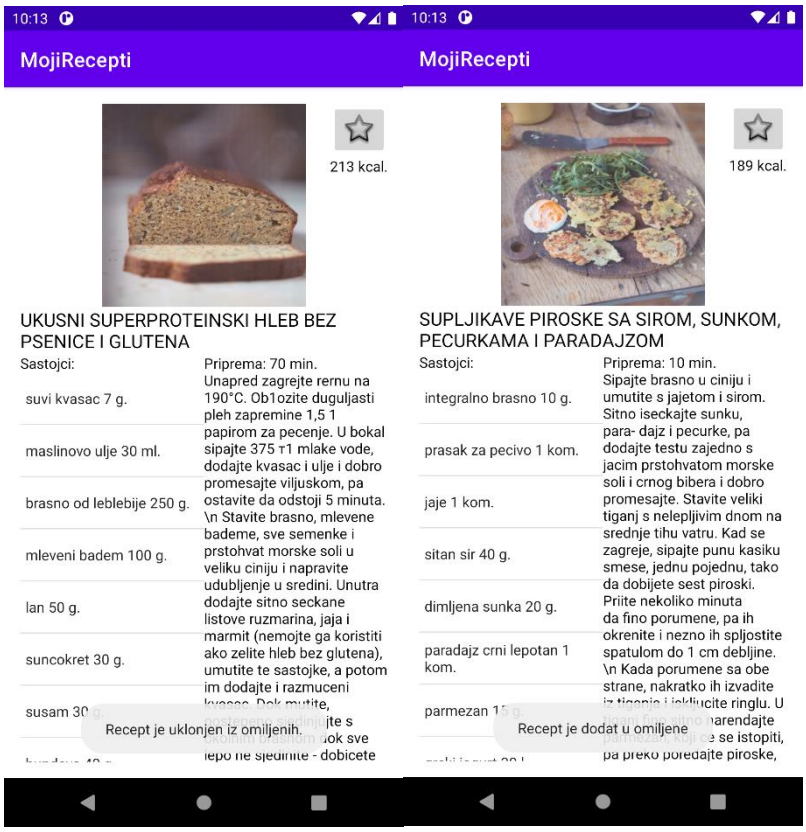
3.7. Detaljan prikaz recepta

Klikom na recept iz bilo koje liste otvara se ekran sa **svim detaljima recepta**:

- **Velika fotografija jela**
- **Naziv**
- **Lista sastojaka**
- **Opis pripreme**
- **Kalorije**
- **Vreme pripreme**
- **Ikonica za dodavanje/uklanjanje iz omiljenih**

Svi podaci su **pregledno prikazani**, a korisnik može **lako odlučiti da li želi da sačuva recept kao omiljeni**.

3.8. Recept dodat-uklonjen iz omiljenih

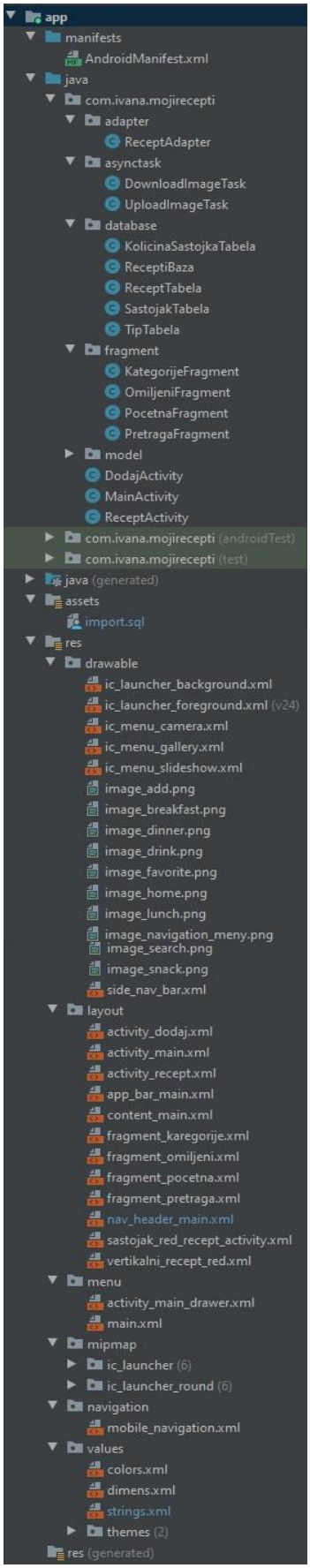


Korisnik klikom na zvezdicu u detaljnom prikazu može da **doda recept u omiljene** ili da ga **ukloni sa te liste**. Nakon svake akcije, prikazuje se **potvrdna poruka**:

- **„Recept je dodat u omiljene“**
- **„Recept je uklonjen iz omiljenih“**

Ova funkcionalnost omogućava korisniku da **organizira recepte prema sopstvenim preferencijama**.

4. Struktura projekta



4.1. Organizacija fajlova i foldera

Struktura projekta je organizovana tako da jasno odvaja različite komponente aplikacije po logičkim celinama:

- **activities/** – sadrži sve aktivnosti kao što su MainActivity, ReceptActivity, DodajActivity.
- **fragments/** – uključuje fragmente za prikaz početne stranice, omiljenih recepata i pretrage.
- **adapters/** – obuhvata klase adaptera, uključujući ReceptAdapter.
- **models/** – sadrži model klasu Recept, koja definiše strukturu recepta.
- **database/** – uključuje ReceptBaza za rad sa lokalnom SQLite bazom.
- **utils/** – pomoćne klase kao što su DownloadImageTask i UploadImageTask.
- **res/** – sadrži sve resurse (layout XML fajlovi, drawable slike, stringovi itd).

4.2. Glavne klase i njihova uloga

- **MainActivity** – centralna aktivnost koja hostuje fragmente i upravlja navigacijom kroz aplikaciju.
- **ReceptActivity** – prikazuje detaljan opis recepta nakon što korisnik klikne na karticu recepta.
- **DodajActivity** – omogućava korisniku unos novog recepta, uključujući sliku, sastojke i korake pripreme.
- **ReceptAdapter** – povezuje podatke sa korisničkim interfejsom u listama (RecyclerView).
- **ReceptBaza** – klasa koja upravlja SQLite bazom podataka i pristupom tabelama.
- **DownloadImageTask** ** / ****UploadImageTask** – omogućavaju preuzimanje i otpremanje slika sa/to servera u pozadini.

Primer: ****ReceptAdapter

ReceptAdapter nasleđuje ArrayAdapter<Recept> i koristi se za prikaz liste recepata u okviru ListView. Klasa implementira:

- getView() metodu kojom inflatira layout za svaki recept i prikazuje naziv, kalorije, vreme i sliku.
- getFilter() za pretragu po nazivu i kalorijama.

- Pomoćnu klasu DownloadImageTask za učitavanje slike recepta iz URL-a.

```
public class ReceptAdapter extends ArrayAdapter<Recept> {
    private int resourceLayout;
    Context mContext;
    private ArrayList<Recept> listaRecepata;
    private ArrayList<Recept> listaRecepataPuna;

    public ReceptAdapter(Context context, int resource, ArrayList<Recept> items) {
        super(context, resource, items);
        this.resourceLayout = resource;
        this.mContext = context;
        this.listaRecepataPuna = items;
        this.listaRecepata = items;
    }

    @Override
    public View getView(int position,  View convertView, ViewGroup parent) {
        View v = convertView;
        if (v == null) {
            LayoutInflater vi;
            vi = LayoutInflater.from(mContext);
            v = vi.inflate(resourceLayout,  root: null);
        }
        Recept p = getItem(position);
        if (p != null) {
            TextView labelNaziv = (TextView) v.findViewById(R.id.labelNazivVertikalni);
            TextView labelVreme = (TextView) v.findViewById(R.id.labelVremeVertikalni);
            TextView labelKalorije = (TextView) v.findViewById(R.id.labelKalorijeVertikalni);
            ImageView image = (ImageView) v.findViewById(R.id.imageVertikalni);
            if (labelNaziv != null) {
                labelNaziv.setText(p.getNaziv());
            }

            if (labelVreme != null) {
                labelVreme.setText(String.valueOf(p.getVremePripreme()));
            }

            if (labelKalorije != null) {
                labelKalorije.setText(String.valueOf(p.getKalorije()));
            }
            if (image != null) {
                new DownloadImageTask(image).execute(p.getSlika());
            }
        }
        return v;
    }

    @Override
    public int getCount() { return listaRecepata.size(); }

    @Override
    public Recept getItem(int position) { return listaRecepata.get(position); }

    @Override
    public Filter getFilter() {
        return new Filter() {
            @Override
            protected FilterResults performFiltering(CharSequence constraint) {
                FilterResults results = new FilterResults();
                if (constraint == null || constraint.length() == 0) {
                    results.count = listaRecepataPuna.size();
                    results.values = listaRecepataPuna;
                } else {//do the search
                    ArrayList<Recept> resultsData = new ArrayList<>();
                    String searchStr = constraint.toString().toUpperCase();
                    for (Recept s : listaRecepataPuna)
                        if (s.getNaziv().toUpperCase().contains(searchStr)
                                || String.valueOf(s.getKalorije()).contains(searchStr)) resultsData.add(s);
                    results.count = resultsData.size();
                    results.values = resultsData;
                }
                return results;
            }

            @Override
            protected void publishResults(CharSequence constraint, FilterResults results) {
                listaRecepata = (ArrayList<Recept>) results.values;
                notifyDataSetChanged();
            }
        };

    }
}
```

Primer: ****DownloadImageTask

DownloadImageTask je klasa koja nasleđuje AsyncTask<String, Void, Bitmap> i koristi se za asinhrono preuzimanje slika preko URL-a. Implementira:

- doInBackground() – otvara stream i konvertuje ga u bitmap.
- onPostExecute() – prikazuje sliku u odgovarajućem ImageView-u.

```
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.util.Log;
import android.widget.ImageView;

import java.io.InputStream;

public class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
    ImageView bmImage;

    public DownloadImageTask(ImageView bmImage) { this.bmImage = bmImage; }

    protected Bitmap doInBackground(String... urls) {
        String url = urls[0];
        Bitmap bitmap = null;
        try {
            InputStream in = new java.net.URL(url).openStream();
            bitmap = BitmapFactory.decodeStream(in);
        } catch (Exception e) {
            Log.e( tag: "Error", e.getMessage());
            e.printStackTrace();
        }
        return bitmap;
    }

    protected void onPostExecute(Bitmap result) { bmImage.setImageBitmap(result); }

}
```

4.3. Fragmenti i aktivnosti

Aplikacija koristi više fragmenta, što omogućava modularnost i bolju upotrebu memorije:

- **PocetnaFragment** – prikazuje recepte na osnovu doba dana.
- **KategorijeFragment** – prikazuje sve recept kategorije (Doručak, Ručak, itd.).
- **OmiljeniFragment** – lista svih recepata koje je korisnik označio kao omiljene.
- **PretragaFragment** – omogućava korisniku da pretražuje recepte po nazivu.
- Aktivnosti:
- **MainActivity** – glavni kontejner za fragmentaciju aplikacije.
- **DodajActivity** – ekran za dodavanje novog recepta.
- **ReceptActivity** – detaljan prikaz selektovanog recepta.

Primer: ****ReceptActivity

ReceptActivity prikazuje sve informacije o odabranom receptu: naziv, kalorije, vreme pripreme, listu sastojaka i sliku. Koristi DownloadImageTask za učitavanje slike i ReceptBaza za dohvat podataka iz baze. Takođe omogućava dodavanje ili uklanjanje recepta iz omiljenih.

```java
public class ReceptActivity extends AppCompatActivity implements View.OnClickListener {
    Recept r;
    ReceptiBaza baza;
    public static final String RECEPT_ID = "recept_id";

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

        int receptId = getIntent().getIntExtra(RECEPT_ID,  defaultValue: 0);

        Log.d( tag: "ReceptActivity",  msg: "receptId=" + receptId);

        baza = new ReceptiBaza( context: this);
        r = baza.getReceptPoID(receptId);

        ImageView slika = findViewById(R.id.slikaReceptA);
        ImageButton dugmeSlikaReceptA = findViewById(R.id.dugmeSlikaReceptA);
        TextView labelReceptANaziv = findViewById(R.id.labelReceptANaziv);
        TextView labelReceptAKalorije = findViewById(R.id.labelReceptAKalorije);
        ListView listReceptASastojci = findViewById(R.id.listReceptASastojci);
        TextView labelReceptAVremePripreme = findViewById(R.id.labelReceptAVremePripreme);
        TextView labelReceptAPriprema = findViewById(R.id.labelReceptAPriprema);

        new DownloadImageTask(slika).execute(r.getSlika());
        labelReceptANaziv.setText(r.getNaziv());
        labelReceptAKalorije.setText( String.valueOf(r.getKalorije()) + " kcal.");
        labelReceptAVremePripreme
                .setText("Priprema: " + String.valueOf(r.getVremePripreme()) + " min.\n" + String.valueOf(r.getPriprema()));
        labelReceptAPriprema.setText(r.getPriprema());

        ArrayList<String> listaSastojakaString = new ArrayList<>();
        for (Sastojak s: r.getSastojci()) {
            listaSastojakaString.add(s.toString());
        }
        ArrayAdapter<String> adapter =
                new ArrayAdapter<~>( context: this, R.layout.sastojak_red_recept_activity, listaSastojakaString);
        listReceptASastojci.setAdapter(adapter);

        dugmeSlikaReceptA.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        int o = r.getOmiljeni();
        if (o == 1) {
            baza.ukloniIzOmiljenog(r.getId());
            Toast.makeText( context: this, text: "Recept je uklonjen iz omiljenih.", Toast.LENGTH_LONG).show();
        } else if (o == 0) {
            baza.dodajUOmiljeni(r.getId());
            Toast.makeText( context: this, text: "Recept je dodat u omiljene", Toast.LENGTH_LONG).show();
        }
    }
}
```

Primer: ****KategorijeFragment

KategorijeFragment prikazuje recepte određene kategorije (npr. doručak). Koristi ReceptAdapter za prikaz liste i prosleđuje selektovani recept ka ReceptActivity uz pomoć Intent-a. Podaci se dohvataju iz baze pomoću metode getReceptiPoKategoriji().

```java
public class KategorijeFragment extends Fragment implements AdapterView.OnItemClickListener {
    private static Recept.Tip tipRecepta;
    private ReceptAdapter adapter;

    public KategorijeFragment(){}
    public static KategorijeFragment newInstance(Recept.Tip tip){
        KategorijeFragment fragment = new KategorijeFragment();
        tipRecepta = tip;
        return fragment;
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_karegorije, container,  attachToRoot: false);
        ListView lista = v.findViewById(R.id.listKategorije);
        ReceptiBaza db = new ReceptiBaza(getContext());
        adapter = new ReceptAdapter(getContext(), R.layout.vertikalni_recept_red, db.getReceptiPoKategoriji(tipRecepta));
        lista.setAdapter(adapter);
        lista.setOnItemClickListener(this);
        return v;
    }

    public static Recept.Tip getTipRecepta() { return tipRecepta; }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Recept recept = adapter.getItem(position);
        Intent intent = new Intent(getContext(), ReceptActivity.class);
        int o = recept.getId();
        intent.putExtra(ReceptActivity.RECEPT_ID, recept.getId());
        startActivity(intent);
    }
}
```

5. Baza podataka

5.1. Opis baze i svrha

Baza podataka u aplikaciji koristi **SQLite sistem za lokalno skladištenje podataka**. Glavna svrha baze je da:

- **Čuva sve recepte koje korisnik doda.**
- **Omogući brzo pretraživanje, filtriranje i prikaz podataka.**
- **Pamti koje recepte je korisnik označio kao omiljene.**

Baza je enkapsulirana kroz klasu **ReceptBaza**, koja implementira **pristup i izvršavanje SQL upita.**

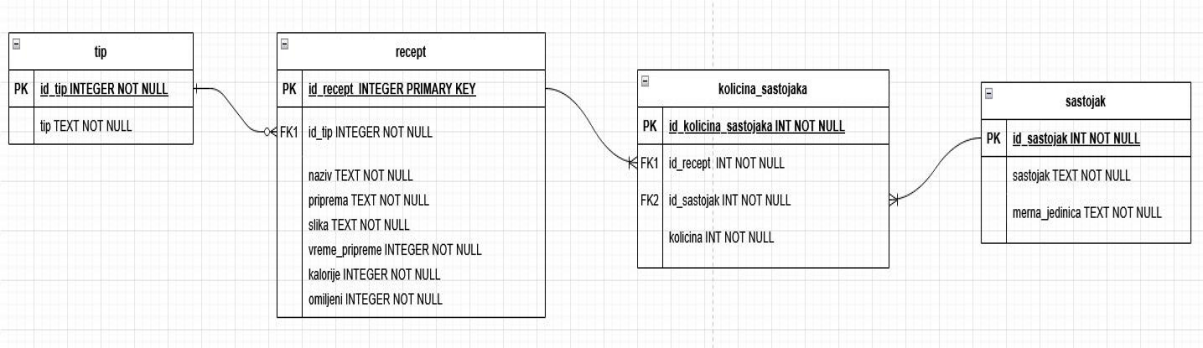
5.2. Tabele i relacije

Baza se sastoji od četiri međusobno povezana entiteta:

- **tip** – čuva tipove recepata (npr. doručak, ručak, večera).
- **recept** – osnovna tabela sa podacima o receptima.
- **sastojak** – definicija sastojaka, uključujući naziv i meru.
- **kolicina_sastojaka** – povezuje recepte i sastojke i čuva količinu za svaki par.

Relacije:

- Svaki recept pripada jednom tipu (id_tip kao strani ključ).
- Svaki recept može imati više sastojaka (relacija preko kolicina_sastojaka).
- Svaki sastojak može se koristiti u više recepata.



5.3. Kreiranje baze

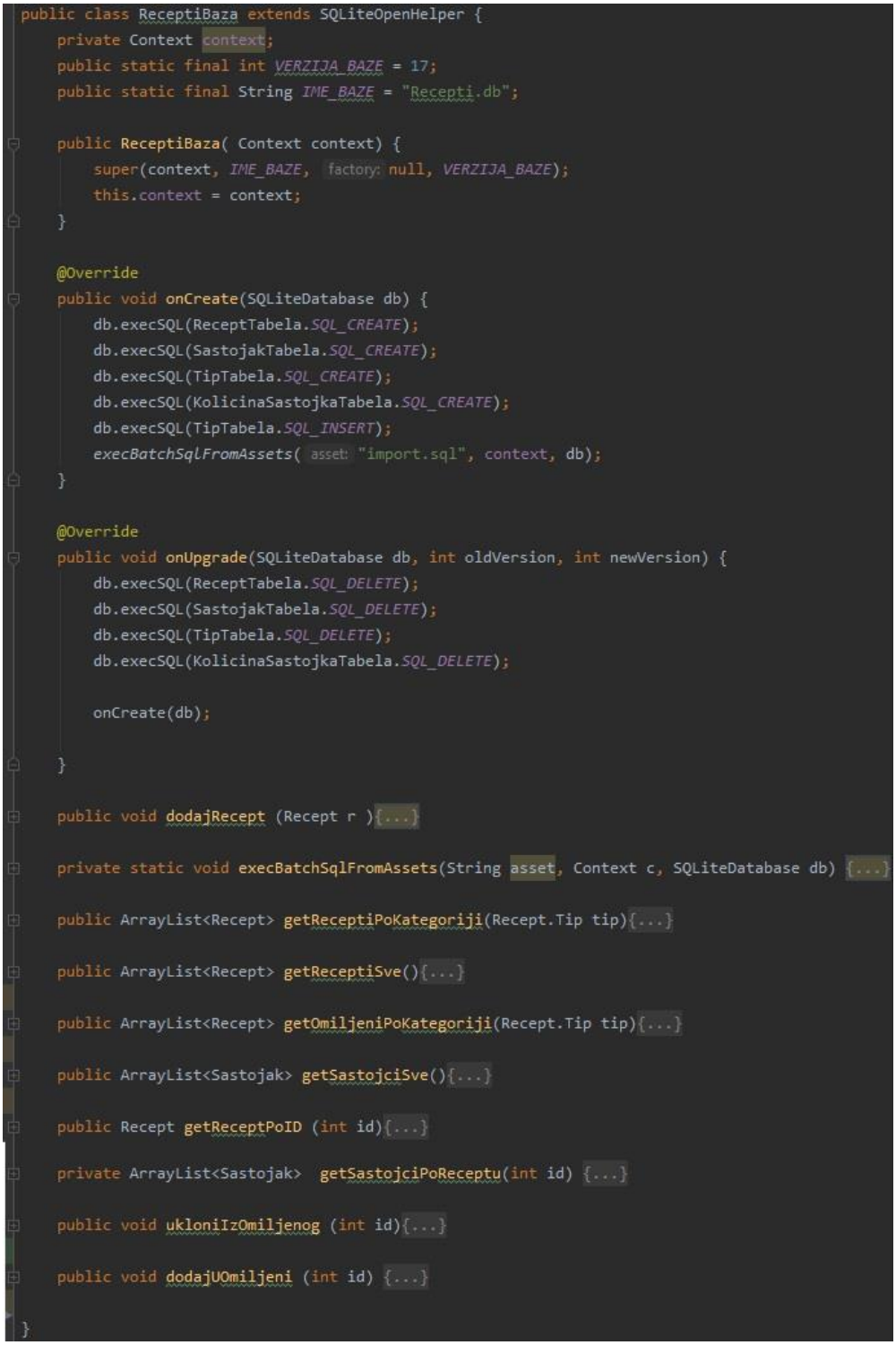
Za kreiranje baze koristi se **SQL skripta** koja se izvršava prilikom prve inicijalizacije baze. Kreiranje i brisanje tabela je enkapsulirano u pomoćnim klasama, kao što je **KolicinaSastojkaTabela**, prikazanoj na sledećoj slici:

```
public class KolicinaSastojkaTabela {
    private KolicinaSastojkaTabela() {
    }
    public static class KolicinaSastojkaKolone implements BaseColumns {
        public static final String TABLE_NAME = "kolicina_sastojaka";
        public static final String COLUMN_RECEPT_ID = "id_recept";
        public static final String COLUMN_SASTOJAK_ID = "id_sastojak";
        public static final String COLUMN_KOLICINA = "kolicina";
    }
    public static final String SQL_CREATE =
            "CREATE TABLE "+ KolicinaSastojkaKolone.TABLE_NAME + " (" +
                    KolicinaSastojkaKolone._ID + " INTEGER PRIMARY KEY, " +
                    KolicinaSastojkaKolone.COLUMN_RECEPT_ID + " INTEGER NOT NULL, " +
                    KolicinaSastojkaKolone.COLUMN_SASTOJAK_ID + " INTEGER NOT NULL, " +
                    KolicinaSastojkaKolone.COLUMN_KOLICINA + " INTEGER NOT NULL" +
                    " )";

    public static final String SQL_DELETE =
            "DROP TABLE IF EXISTS "+ KolicinaSastojkaKolone.TABLE_NAME;

}
```

Inicijalizacija baze i izvršavanje SQL skripti se odvija u klasi **ReceptiBaza**, koja proširuje SQLiteOpenHelper. U metodi **onCreate()** pozivaju se sve **SQL naredbe za kreiranje tabela**, kao i metoda **execBatchSqlFromAssets()** za izvršavanje dodatnih SQL upita iz fajla import.sql:



```
public class ReceptiBaza extends SQLiteOpenHelper {
    private Context context;
    public static final int VERZIJA_BAZE = 17;
    public static final String IME_BAZE = "Recepti.db";

    public ReceptiBaza( Context context) {
        super(context, IME_BAZE,  factory: null, VERZIJA_BAZE);
        this.context = context;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(ReceptTabela.SQL_CREATE);
        db.execSQL(SastojakTabela.SQL_CREATE);
        db.execSQL(TipTabela.SQL_CREATE);
        db.execSQL(KolicinaSastojkaTabela.SQL_CREATE);
        db.execSQL(TipTabela.SQL_INSERT);
        execBatchSqlFromAssets( asset: "import.sql", context, db);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL(ReceptTabela.SQL_DELETE);
        db.execSQL(SastojakTabela.SQL_DELETE);
        db.execSQL(TipTabela.SQL_DELETE);
        db.execSQL(KolicinaSastojkaTabela.SQL_DELETE);

        onCreate(db);

    }

    public void dodajRecept (Recept r ){...}

    private static void execBatchSqlFromAssets(String asset, Context c, SQLiteDatabase db) {...}

    public ArrayList<Recept> getReceptiPoKategoriji(Recept.Tip tip){...}

    public ArrayList<Recept> getReceptiSve(){...}

    public ArrayList<Recept> getOmiljeniPoKategoriji(Recept.Tip tip){...}

    public ArrayList<Sastojak> getSastojciSve(){...}

    public Recept getReceptPoID (int id){...}

    private ArrayList<Sastojak>  getSastojciPoReceptu(int id) {...}

    public void ukloniIzOmiljenog (int id){...}

    public void dodajUOmiljeni (int id) {...}

}
```

Ova skripta omogućava **inicijalno popunjavanje baze podacima**. Sve SQL komande definišu se u stringovima i izvršavaju kroz **execSQL()** metodu.

6. Zaključak

Aplikacija *Moji Recepti* je pažljivo strukturirana kako bi obezbedila jasnoću, modularnost i lako održavanje. Organizacija fajlova po funkcionalnim celinama (aktivnosti, fragmenti, adapteri, modeli, baza, pomoćne klase) omogućava jednostavno snalaženje. Korišćenje fragmenta i adaptera doprinosi modernom i dinamičnom korisničkom interfejsu, dok ReceptBaza obezbeđuje stabilno i brzo upravljanje podacima kroz SQLite.

Kroz pregled glavnih klasa i njihove upotrebe, kao i kroz primere korišćenja aktivnosti i fragmenta, može se uočiti jasno definisana arhitektura aplikacije, fokusirana na korisničko iskustvo i performanse. Baza podataka je modelovana tako da podrži fleksibilno dodavanje recepata i sastojaka, sa jasno definisanim relacijama i podrškom za import inicijalnih podataka.

Sve ove komponente zajedno čine aplikaciju pouzdanom, proširivom i spremnom za dalje unapređenje, kao što su podrška za korisničke naloge, online sinhronizaciju i dodatne funkcionalnosti poput automatskog planera obroka ili notifikacija.