ANDROID PROGRAMIRANJE

Vežba 4: RecycleView i priprema za 1. kolokvijum

Teorijske napomene (RecycleView)

RecyclerView je napredna i fleksibilna komponenta za prikaz podataka u Android aplikacijama. Naslednik je starijeg ListView-a i donosi značajne prednosti u performansama, posebno kod prikaza velikog broja stavki. RecyclerView je dizajniran da efikasno upravlja memorijom kroz **recikliranje** svojih elemenata dok korisnik skroluje kroz listu.

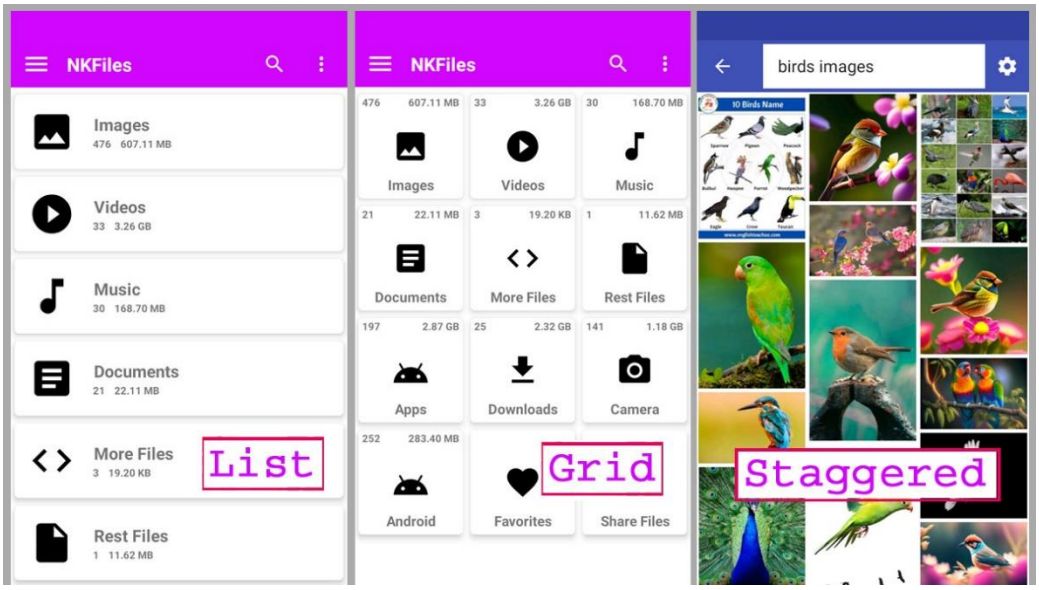
RecyclerView koristi **tri ključne komponente** za pravilno funkcionisanje:

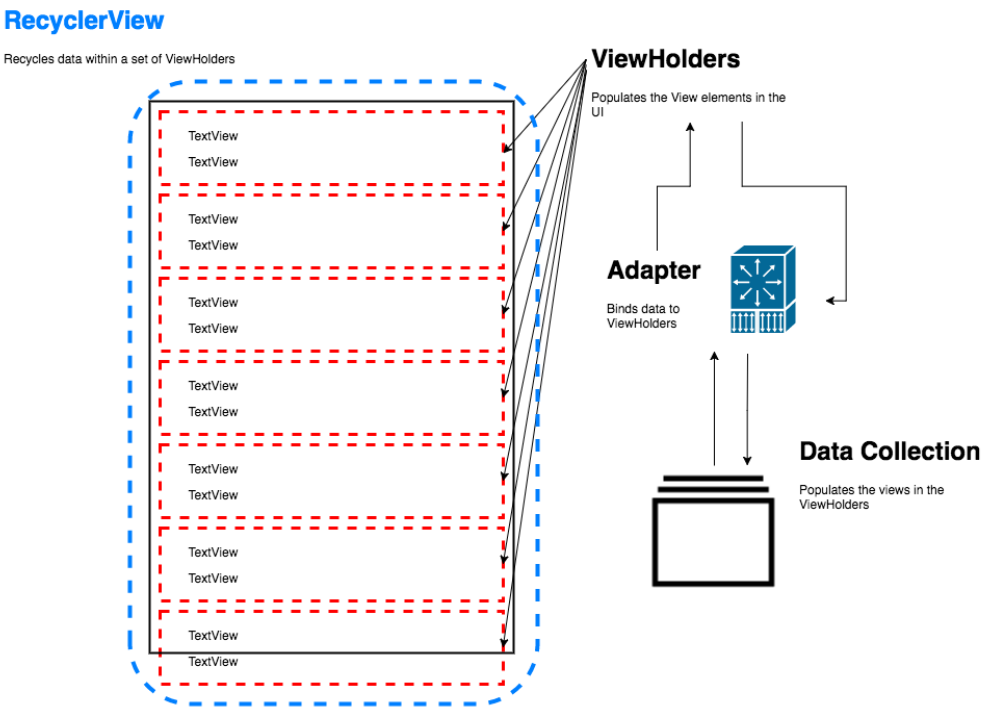
1. **Adapter**:
    - Adapter povezuje podatke sa RecyclerView-om i definiše kako će se svaka stavka prikazivati. Adapter učitava podatke (obično sa liste) i prenosi ih u ViewHolder.
    - onCreateViewHolder i onBindViewHolder metode u adapteru služe za kreiranje prikaza i postavljanje vrednosti za svaku stavku.
2. **ViewHolder**:
    - ViewHolder predstavlja jedan element liste i koristi se za čuvanje referenci na prikaze elemenata, čime se smanjuje potreba za stalnim pronalaženjem View-ova i povećava brzina prikaza.
    - ViewHolder klasa se definiše kao unutrašnja klasa unutar adaptera i sadrži elemente prikaza (npr. TextView, ImageView) za pojedinačnu stavku.
3. **LayoutManager**:
    - LayoutManager definiše raspored elemenata unutar RecyclerView-a i omogućava različite vrste prikaza, kao što su **LinearLayout** (vertikalni ili horizontalni prikaz), **GridLayout** (prikaz u mreži) ili **StaggeredGridLayout** (nestandardna mreža).
    - **LinearLayoutManager** prikazuje stavke u jednom redu ili jednoj koloni, dok **GridLayoutManager** prikazuje stavke u više kolona.

RecyclerView je osmišljen da poboljša performanse u odnosu na ListView, jer koristi princip recikliranja prikaza elemenata. Umesto da kreira nove prikaze svaki put kada korisnik skroluje kroz listu, RecyclerView ponovo koristi već postojeće prikaze (View-ove) i na taj način smanjuje opterećenje memorije. Ovo posebno dolazi do izražaja kada lista sadrži veliki broj stavki.

Osnovne razlike između RecyclerView i ListView:

ListView	RecyclerView
Manje fleksibilan, nema ugrađeni LayoutManager	Fleksibilniji i koristi LayoutManager za različite prikaze
Adapter ne zahteva ViewHolder klasu	Adapter koristi ViewHolder za recikliranje prikaza
Pogodniji za jednostavne liste	Pogodan za kompleksne liste i velike setove podataka
Podržava samo vertikalno listanje	Podržava horizontalno, vertikalno i grid prikaz





Za kreiranje RecyclerView-a potrebne su sledeće komponente i operacije:

1. **XML Layout za RecycleView**

   U activity_main.xml fajlu dodajemo RecyclerView:

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"/>
```

2. **XML Layout za stavke**

   Napravimo novi XML fajl (item_shopping.xml) koji će prikazivati svaku namirnicu sa njenom količinom i dugmetom za brisanje:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="8dp">
    <TextView
        android:id="@+id/itemName"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text="Item Name"
        android:textSize="18sp"
        android:padding="8dp"/>
    <TextView
        android:id="@+id/itemQuantity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Qty"
        android:textSize="16sp"
        android:padding="8dp"/>
    <Button
        android:id="@+id/deleteButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Remove"/>
</LinearLayout>
```

3. Model klasa za artikle (namirnice)

Ova klasa će predstavljati svaki pojedinačni predmet u listi, sa nazivom i količinom.

```
public class ShoppingItem {
    private String name;
    private int quantity;

    public ShoppingItem(String name, int quantity) {
        this.name = name;
        this.quantity = quantity;
    }

    public String getName() {
        return name;
    }

    public int getQuantity() {
        return quantity;
    }
}
```

4. Adapter klasa za RecycleView

Sada kreiramo ShoppingListAdapter klasu koja nasleđuje RecyclerView.Adapter i koristi ViewHolder za prikaz stavki.

```
import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import java.util.List;


// Adapter klasa za RecyclerView koja prikazuje listu namirnica za kupovinu

public class ShoppingListAdapter extends
RecyclerView.Adapter<ShoppingListAdapter.ShoppingViewHolder> {


    // Lista objekata koji predstavljaju stavke za prikaz u RecyclerView-u

    private List<ShoppingItem> shoppingList;


    // Konstruktor adaptera koji prima listu namirnica

    public ShoppingListAdapter(List<ShoppingItem> shoppingList) {

        this.shoppingList = shoppingList;

    }
```

```
// Kreiranje ViewHolder-a za prikaz stavke u RecyclerView-u
@NonNull
@Override
public ShoppingViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
     viewType) {
    // Uzima layout (item_shopping.xml) i kreira ViewHolder za svaku stavku
    View view = LayoutInflater.from(parent.getContext())
                              .inflate(R.layout.item_shopping, parent, false);
    return new ShoppingViewHolder(view);
}

// Vezuje podatke sa liste na odgovarajuće UI elemente u ViewHolder-u
@Override
public void onBindViewHolder(@NonNull ShoppingViewHolder holder, int position) {
    // Dohvata stavku sa liste na trenutnoj poziciji
    ShoppingItem item = shoppingList.get(position);

    // Postavlja naziv i količinu namirnice u odgovarajuće TextView-ove
    holder.itemName.setText(item.getName());
    holder.itemQuantity.setText(String.valueOf(item.getQuantity()));

    // Postavlja OnClickListener na dugme za brisanje stavke
    holder.deleteButton.setOnClickListener(v -> {
        // Uklanja stavku sa liste
        shoppingList.remove(position);

        // Obaveštava RecyclerView da je stavka uklonjena na toj poziciji
        notifyItemRemoved(position);

        // Obaveštava da su ostale pozicije izmenjene kako bi se ažurirao prikaz
        notifyItemRangeChanged(position, shoppingList.size());
    });
}

// Vraća broj stavki u listi (da RecyclerView zna koliko stavki treba da prikaže)
@Override
public int getItemCount() {
    return shoppingList.size();
}
```

```
    // ViewHolder klasa koja sadrži referencu na UI elemente svake pojedinačne stavke
    public static class ShoppingViewHolder extends RecyclerView.ViewHolder {
        // TextView za prikaz naziva namirnice
        TextView itemName;

        // TextView za prikaz količine namirnice
        TextView itemQuantity;

        // Button za brisanje stavke sa liste
        Button deleteButton;

        // Konstruktor ViewHolder-a koji pronalazi odgovarajuće UI elemente iz layout-a
        public ShoppingViewHolder(@NonNull View itemView) {
            super(itemView);
            itemName = itemView.findViewById(R.id.itemName);
            itemQuantity = itemView.findViewById(R.id.itemQuantity);
            deleteButton = itemView.findViewById(R.id.deleteButton);
        }
    }
}
```

5. **Povezivanje RecyclerView-a sa Adapterom u glavnoj aktivnosti**

   U MainActivity.java inicijalizujemo RecyclerView, kreiramo listu sa par stavki i dodeljujemo je adapteru.

```
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    private RecyclerView recyclerView;
    private ShoppingListAdapter adapter;
    private List<ShoppingItem> shoppingList;

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

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
```

```
        // Kreiranje početne liste za kupovinu
        shoppingList = new ArrayList<>();
        shoppingList.add(new ShoppingItem("Apples", 3));
        shoppingList.add(new ShoppingItem("Bananas", 5));
        shoppingList.add(new ShoppingItem("Milk", 2));

        // Inicijalizacija adaptera i postavljanje na RecyclerView
        adapter = new ShoppingListAdapter(shoppingList);
        recyclerView.setAdapter(adapter);
    }
}
```

Šta ovaj primer radi:

1. RecyclerView prikazuje listu sa početnim podacima o namirnicama.
2. ViewHolder prikazuje naziv namirnice i količinu, a Button omogućava brisanje sa liste.
3. Kada korisnik klikne na "Remove", stavka se uklanja, i RecyclerView se ažurira automatski.

Zadatak za samostalni rad (ulazi u ocenu ove vežbe)

Cilj ovog zadatka je da kroz konkretan primer naučimo kako se koristi RecyclerView u Androidu, kako da definišemo i prikažemo listu podataka u aplikaciji, ali i kako da koristimo osnovne Android funkcionalnosti za rad sa interaktivnim elementima.

Funkcionalnosti koje treba implementirati:

1. Kreiranje modela podataka za prikaz informacija o filmu.
2. Prikaz liste filmova korišćenjem RecyclerView.
3. Definisanje i prikaz detalja o svakom filmu sa liste, uključujući naziv, žanr, godinu i ocenu.
4. Implementacija dugmeta za uklanjanje filma sa liste.

Korak 1: Kreiranje novog projekta

- Otvorite **Android Studio**.
- Kreirajte novi projekat koristeći **Empty Activity** predložak.
- Dodelite projektu naziv, na primer, "MoviesVezba", i izaberite jezik koji preferirate.
- Postavite minimum API level na verziju koja podržava RecyclerView.

Korak 2: Dodavanje RecyclerView-a u projekat

- Dodajte zavisnost za RecyclerView u build.gradle (Module: app) fajlu.
- Otvorite build.gradle (Module: app) fajl. U dependencies sekciji, dodajte sledeću liniju ako već nije prisutna: implementation 'androidx.recyclerview:recyclerview:1.2.1
- Sinhronizujte projekat (klikom na Sync Now) kako bi Android Studio mogao da prepozna RecyclerView.

Korak 3: Definisanje modela podataka za film

- Napravite klasu **Movie** koja će predstavljati jedan film. Model treba da sadrži ove atribute:
    - **title** - naziv filma (tipa String).
    - **genre** - žanr filma (tipa String).
    - **year** - godina kada je film objavljen (tipa int).
    - **rating** - ocena filma (tipa double).
- Definišite konstruktor i metode za pristup podacima (getere) u okviru modela.

Korak 4: Priprema layout-a za MainActivity

- Otvorite XML fajl **activity_main.xml** koji definiše glavnu aktivnost aplikacije.
- Dodajte RecyclerView u glavnu aktivnost tako da zauzima celokupan ekran. RecyclerView će služiti za prikaz liste filmova.

Korak 5: Kreiranje layout-a za pojedinačne stavke u listi

- Kreirajte novi XML fajl pod nazivom **item_movie.xml** koji će definisati izgled svake stavke (filma) u RecyclerView-u.
- Svaka stavka treba da sadrži:
    - Tekstualni element za prikaz **naziva filma**.
    - Tekstualni element za prikaz **žanra filma**.
    - Tekstualni element za prikaz **godine objavljivanja**.
    - Tekstualni element za prikaz **ocene filma**.
    - Dugme za **brisanje filma** iz liste.
- Rasporedite ove elemente koristeći **LinearLayout** ili **ConstraintLayout**.

Korak 6: Kreiranje adaptera za RecyclerView

- Napravite novu klasu **MovieAdapter** koja će naslediti RecyclerView.Adapter.
- U okviru MovieAdapter klase implementirajte sledeće metode:
    - **onCreateViewHolder** - kreira i vrati novu instancu ViewHolder-a sa layout-om definisanim u item_movie.xml.
    - **onBindViewHolder** - vezuje podatke o svakom filmu za odgovarajuću poziciju u RecyclerView-u.
    - **getItemCount** - vraća broj stavki u listi filmova.
- Napravite unutrašnju klasu **MovieViewHolder** u MovieAdapter-u, koja će čuvati referencu na sve UI elemente definisane u item_movie.xml.

Primer koda (zajedno za korake 6 i 7) je dat na sledećoj strani.

```
public class MovieAdapter extends RecyclerView.Adapter<MovieAdapter.MovieViewHolder> {
    private List<Movie> movieList;

    public MovieAdapter(List<Movie> movieList) {
        this.movieList = movieList;
    }

    @Override
    public MovieViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
       View view = LayoutInflater.from(parent.getContext()).
                                       inflate(R.layout.item_movie, parent, false);
        return new MovieViewHolder(view);
    }

    @Override
    public void onBindViewHolder(MovieViewHolder holder, int position) {
        Movie movie = movieList.get(position);
        holder.title.setText(movie.getTitle());
        holder.genre.setText(movie.getGenre());
        holder.year.setText(String.valueOf(movie.getYear()));
        holder.rating.setText(String.valueOf(movie.getRating()));

        holder.deleteButton.setOnClickListener(v -> {
            movieList.remove(position);
            notifyItemRemoved(position);
            notifyItemRangeChanged(position, movieList.size());
        });
    }

    @Override
    public int getItemCount() {
        return movieList.size();
    }

    public class MovieViewHolder extends RecyclerView.ViewHolder {
        TextView title, genre, year, rating;
        Button deleteButton;
        public MovieViewHolder(View itemView) {
            super(itemView);
            title = itemView.findViewById(R.id.movieTitle);
```

```
            genre = itemView.findViewById(R.id.movieGenre);
            year = itemView.findViewById(R.id.movieYear);
            rating = itemView.findViewById(R.id.movieRating);
            deleteButton = itemView.findViewById(R.id.deleteButton);
        }
    }
}
```

Korak 7: Implementacija funkcionalnosti za uklanjanje filma iz liste

- U onBindViewHolder metodi, implementirajte **OnClickListener** za dugme za brisanje.
- Kada se dugme pritisne, uklonite odgovarajući film sa liste i obavestite adapter o promeni.
- Koristite metode **notifyItemRemoved** i **notifyItemRangeChanged** kako bi RecyclerView mogao ažurirati prikaz nakon uklanjanja stavke.

Korak 8: Inicijalizacija RecyclerView-a u MainActivity

- U MainActivity, pronađite RecyclerView pomoću findViewById metode.
- Kreirajte instancu MovieAdapter klase i prosledite joj listu filmova koju želite prikazati.
- Inicijalizujte RecyclerView koristeći **setLayoutManager** i postavite adapter pomoću **setAdapter** metode.

Primer koda:

```
public class MainActivity extends AppCompatActivity {
    private RecyclerView recyclerView;
    private MovieAdapter movieAdapter;
    private List<Movie> movieList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        movieList = new ArrayList<>();
        movieList.add(new Movie("Inception", "Sci-Fi", 2010, 8.8));
        movieList.add(new Movie("The Dark Knight", "Action", 2008, 9.0));
        movieAdapter = new MovieAdapter(movieList);
        recyclerView.setAdapter(movieAdapter);
    }
}
```

Korak 9: Testiranje i provera funkcionalnosti

- Pokrenite aplikaciju na emulatoru ili fizičkom uređaju.
- Proverite da li su svi filmovi prikazani u RecyclerView-u.
- Testirajte dugme za brisanje i proverite da li stavke nestaju sa liste nakon što kliknete tu.

Korak 10: Bonus zadatak

- Dodajte opciju za **dodavanje novog filma** u listu putem korisničkog unosa ili omogućite **sortiranje liste filmova** po godini, oceni ili žanru.

Priprema za kolokvijum (ne ulazi u ocenu ove vežbe)

Kreirajte aplikaciju pod nazivom **Trošak Putovanja** koja korisnicima omogućava unos informacija o troškovima putovanja i izračunava ukupne troškove. Aplikacija treba da sadrži sledeće grafičke komponente i pokriva određene opercije:

1. Layout i UI komponente:

- **GridLayout** sa sledećim elementima:
  - **EditText** za unos broja dana putovanja.
  - **EditText** za unos prosečnih dnevnih troškova (RSD ili EUR).
  - **EditText** za unos cene goriva (RSD ili EUR po litru).
  - **EditText** za unos procene ukupne kilometraže.
  - **Button** sa tekstom "Izračunaj" koji pokreće kalkulaciju ukupnih troškova.

2. Izračunavanje i prikaz rezultata:

- Kada korisnik unese sve podatke i pritisne dugme "Izračunaj":
  - Aplikacija treba da prikaže **Toast** poruku sa ukupnim troškom (broj dana x dnevni troškovi + kilometraža x cena goriva).
  - Aplikacija treba da prikaže rezultat u **TextView** elementu ispod dugmeta.

3. Navigacija i Opcije u Meniju:

- Kreirajte jednostavan **OptionsMenu** sa stavkama kao što su:
  - **"Resetuj"** – briše sve unesene podatke.
  - **"Info"** – koristeći **Toast** ili **AlertDialog**, prikazuje obaveštenje sa opisom aplikacije i podacima o timu ili individualnom studentu koji radi zadatak.

4. Notifikacije:

- Dodajte jednostavnu **notifikaciju** koja korisnika obaveštava kada se izračunavanje troškova uspešno završi.

5. Prikaz istorije izračunavanja:

- Kreirajte **ListView** koji prikazuje poslednjih pet unosa troškova. Kada korisnik klikne na neki od rezultata u listi, prikažite **Toast** sa detaljima tog unosa. Ovo se može realizovati i preko **RecyleView** ako ste dobro ispratili ovu vežbu.

Primer layout-a:

```
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="2"
    android:padding="16dp">

    <!-- Logo na vrhu ekrana -->
    <ImageView
        android:id="@+id/logoImageView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/logo" <!—Koristimo sliku iz drawable foldera -->
        android:contentDescription="Logo aplikacije"
        android:layout_columnSpan="2"
        android:layout_gravity="center"
        android:padding="8dp" />

    <!-- Preostali elementi -->
    <TextView android:text="Broj dana putovanja:"/>
    <EditText android:id="@+id/editDays" android:inputType="number" />

    <TextView android:text="Prosečni dnevni trošak:"/>
    <EditText android:id="@+id/editDailyCost" android:inputType="numberDecimal" />

    <TextView android:text="Cena goriva:"/>
    <EditText android:id="@+id/editFuelPrice" android:inputType="numberDecimal" />

    <TextView android:text="Kilometraža:"/>
    <EditText android:id="@+id/editDistance" android:inputType="number" />

    <Button
        android:id="@+id/buttonCalculate"
        android:layout_columnSpan="2"
        android:text="Izračunaj" />

    <TextView android:id="@+id/textResult" android:layout_columnSpan="2"
     android:text="Rezultat: 0" />
</GridLayout>
```

Primer menija:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/action_reset" android:title="Resetuj" />
    <item android:id="@+id/action_info" android:title="Info" />
</menu>
```

Primer glavne aktivnosti:

```
public class MainActivity extends AppCompatActivity {
    private EditText editDays, editDailyCost, editFuelPrice, editDistance;
    private TextView textResult;

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

        // Povezivanje komponenti
        editDays = findViewById(R.id.editDays);
        editDailyCost = findViewById(R.id.editDailyCost);
        editFuelPrice = findViewById(R.id.editFuelPrice);
        editDistance = findViewById(R.id.editDistance);
        textResult = findViewById(R.id.textResult);

        Button buttonCalculate = findViewById(R.id.buttonCalculate);
        buttonCalculate.setOnClickListener(v -> calculateCost());
    }

    private void calculateCost() {
        // Validacija unosa
        if (!isValidInput()) {
            return; // Ako unos nije validan, ne nastavljamo dalje
        }

        // Računanje na osnovu unosa
        int days = Integer.parseInt(editDays.getText().toString());
        double dailyCost = Double.parseDouble(editDailyCost.getText().toString());
        double fuelPrice = Double.parseDouble(editFuelPrice.getText().toString());
        int distance = Integer.parseInt(editDistance.getText().toString());
```

```
    // Formula za računanje
    double totalCost = days * dailyCost + (distance * fuelPrice);

    // Prikaz rezultata
    textResult.setText("Rezultat: " + totalCost);

    // Prikazivanje Toast-a
    Toast.makeText(this, "Ukupni trošak: " + totalCost, Toast.LENGTH_LONG).show();

    // Prikazivanje notifikacije
    NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
        "calc_channel")
        .setSmallIcon(R.drawable.ic_notification)
        .setContentTitle("Izračunavanje završeno")
        .setContentText("Vaš ukupni trošak je " + totalCost)
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);

    NotificationManagerCompat notificationManager =
           NotificationManagerCompat.from(this);
           notificationManager.notify(1, builder.build());
}

// Metoda za validaciju unosa
private boolean isValidInput() {
    // Provera da li su svi podaci uneti i da li su pozitivni brojevi
    if (editDays.getText().toString().isEmpty() ||
            editDailyCost.getText().toString().isEmpty() ||
            editFuelPrice.getText().toString().isEmpty() ||
            editDistance.getText().toString().isEmpty()) {
             Toast.makeText(this, "Svi podaci moraju biti uneseni.",
                                Toast.LENGTH_SHORT).show();
        return false;
    }
    try {
        int days = Integer.parseInt(editDays.getText().toString());
        double dailyCost = Double.parseDouble(editDailyCost.getText().toString());
        double fuelPrice = Double.parseDouble(editFuelPrice.getText().toString());
        int distance = Integer.parseInt(editDistance.getText().toString());
```

```
            // Provera da li su brojevi pozitivni
            if (days <= 0 || dailyCost <= 0 || fuelPrice <= 0 || distance <= 0) {
                Toast.makeText(this, "Svi brojevi moraju biti pozitivni.",
                                       Toast.LENGTH_SHORT).show();
                return false;
            }
        } catch (NumberFormatException e) {
            Toast.makeText(this, "Uneti podaci moraju biti validni brojevi.",
                                   Toast.LENGTH_SHORT).show();
            return false;
        }
        return true;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_reset:
                editDays.setText("");
                editDailyCost.setText("");
                editFuelPrice.setText("");
                editDistance.setText("");
                textResult.setText("Rezultat: 0");
                return true;
            case R.id.action_info:
                Toast.makeText(this, "Autor: Plavi tim", Toast.LENGTH_SHORT).show();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

Korisni resursi

1. Tutorijal za dinamičke liste sa RecycleView
   https://developer.android.com/develop/ui/views/layout/recyclerview
2. Detaljan tutorijal za RecycleView
   https://www.youtube.com/watch?v=Mc0XT58A1Z4
3. Urađen primer sa RecycleView, vredi pogledati i probati nešto sliično
   https://www.codeproject.com/Articles/1275740/Using-RecyclerView-in-Android
4. Odlična plejlista sa klipovima koji mogu biti od pomoći u pripremi kolokvijuma
   https://www.youtube.com/playlist?list=PLQkwcJG4YTCTq1raTb5iMuxnEB06J1VHXK
5. Korisna vežbanja u Androidu, možete da aktivirate free trial kako biste skinuli fajl, samo kasnije otkažite membership pre navedenog datuma ako ne želite više da koristite
   https://www.scribd.com/document/608223464/AndroidLabExercises