Mobilne Aplikacije

Nositelj: doc. dr. sc. Nikola Tanković
Izvođač: dr. sc. Robert Šajina
Asistent: mag. inf. Alesandro Žužić

Ustanova: Sveučilište Jurja Dobrile u Puli, Fakultet informatike u Puli

Fakultet informatike u Puli

[3] - Interakcija s elementima (FoodTracker)

Posljednje ažurirano: 19. studenog 2025.

Sadržaj

Dohvaćanje referenci UI elemenata

U prošloj skripti smo naučili kako definirati izgled aplikacije pomoću XML datoteka. Dodavali smo različite UI elemente (*gumbi, tekstualna polja, liste itd.*) u datoteku `activity_main.xml`. Međutim, samo definiranje izgleda nije dovoljno za interakciju s tim elementima. Potrebno je povezati te elemente s logikom aplikacije u Java ili Kotlin kôdu.

Za interakciju s UI elementima (*kao što su gumbi, tekstualna polja, liste itd.*), potrebno je dohvatiti njihove reference u kôdu. To se obično radi pomoću metode `findViewById()`, koja omogućuje pristup elementima definiranim u XML datotekama izgleda (*layout*).

Primjer XML definicije gumba s ID atributom:

```
...
<Button
    ...
    android:id="@+id/my_button"
    android:text="Klikni me!" />
...
```

Primjer dohvaćanja reference na gumb u `MainActivity.java` unutar `onCreate()` metode:

```
Button myButton = findViewById(R.id.my_button);
```

> Potrebno je osigurati da se `findViewById()` poziva nakon `setContentView()`, jer se na taj način postavlja sadržaj aktivnosti i omogućuje pristup elementima izgleda.

> Isto tako treba uvesti odgovarajuću klasu za UI elemente koje koristimo, kao što je `import android.widget.Button;` za gumb.

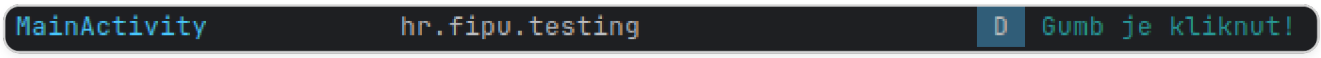
Pa cijeli MainActivity.java može izgledati ovako:

```
package hr.fipu.testing;

import android.os.Bundle;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

        Button myButton = findViewById(R.id.my_button);
    }
}
```

Dodavanje slušatelja događaja (*Event Listener*)

Sada možemo koristiti `myButton` objekt za postavljanje slušatelja događaja (*event listener*), kao što je `setOnClickListener()`, kako bismo definirali što se događa kada korisnik klikne na gumb:

```
myButton.setOnClickListener(v -> {
    // Radnja koja se izvršava kada se gumb klikne
    Log.d("MainActivity", "Gumb je kliknut!");
});
```



Logcat: Gumb je kliknut!

Također možemo koristiti druge metode za interakciju s elementima, kao što su `setOnLongClickListener()`, `setOnFocusChangeListener()`, itd., ovisno o vrsti elementa i željenoj interakciji.

Logiranje poruka

Za ispis poruka u *Logcat* konzolu, koristi se klasa `Log`. Ovo je korisno za praćenje događaja i grešaka tijekom razvoja aplikacije. Primjer ispisivanja poruke u *Logcat* kada se gumb klikne:

```
import android.util.Log;
...
myButton.setOnClickListener(v -> {
    Log.d("MainActivity", "Gumb je kliknut!");
});
```

> Logcat se nalazi u donjem dijelu Android Studija, a može se otvoriti putem izbornika: `View` -> `Tool Windows` -> `Logcat`.

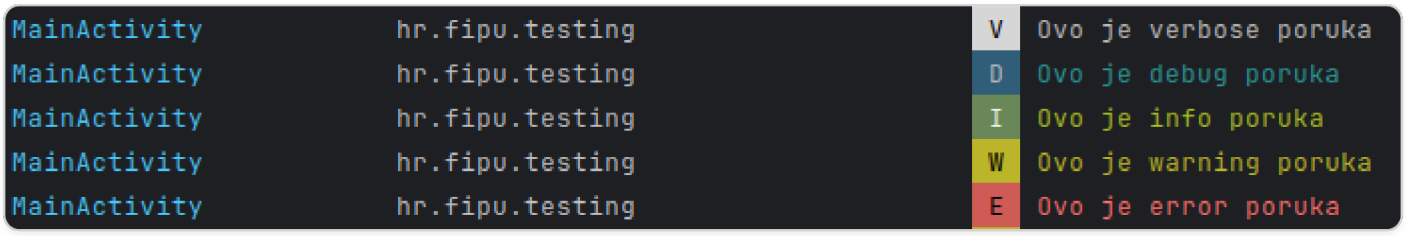
Logcat ikona

Postoje različite razine logiranja:

- `Log.v()` - Verbose
- `Log.d()` - Debug
- `Log.i()` - Info
- `Log.w()` - Warning
- `Log.e()` - Error

Primjer korištenja različitih razina logiranja:

```
Log.v("MainActivity", "Ovo je verbose poruka");
Log.d("MainActivity", "Ovo je debug poruka");
Log.i("MainActivity", "Ovo je info poruka");
Log.w("MainActivity", "Ovo je warning poruka");
Log.e("MainActivity", "Ovo je error poruka");
```



Logcat: Razine logiranja

Toast poruke

Toast poruke su kratke obavijesti koje se pojavljuju na ekranu kako bi informirale korisnika o nekoj radnji ili događaju. One su korisne za pružanje povratnih informacija bez ometanja korisničkog iskustva.

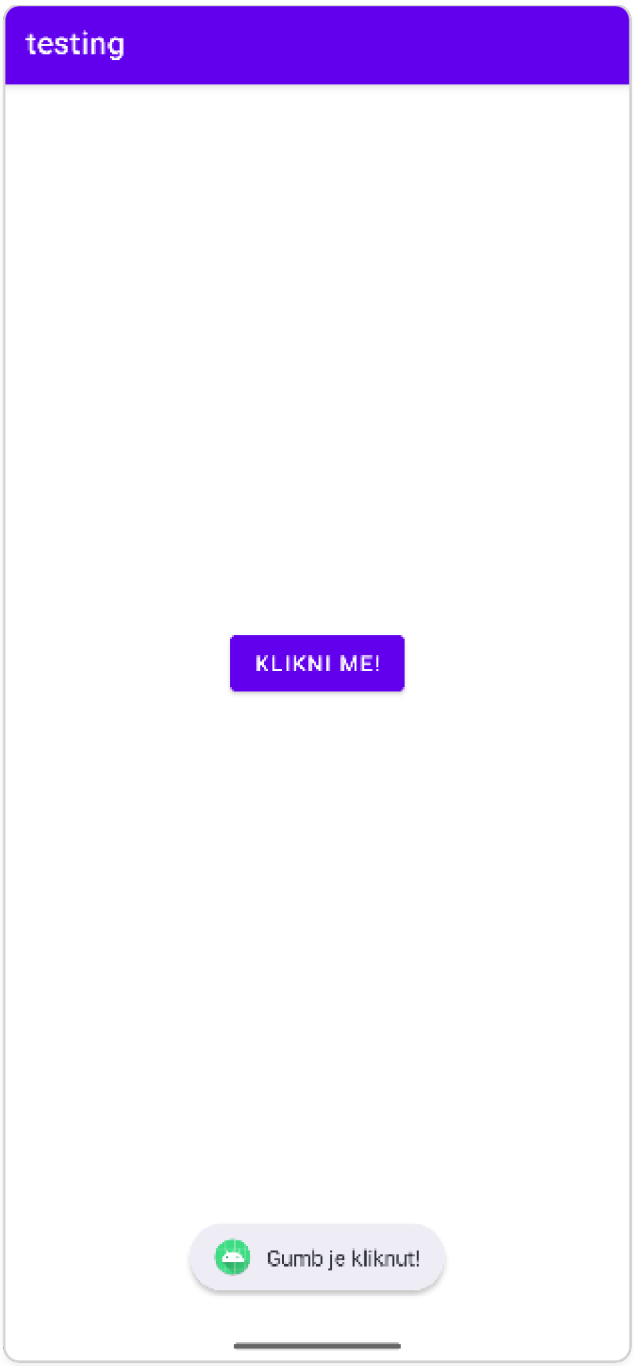
Za prikazivanje Toast poruke, koristi se metoda `Toast.makeText()`, koja prima tri argumenta:

1. **Kontekst** - obično se koristi `this` ili `getApplicationContext()`.
2. **Tekst poruke** - niz znakova koji predstavlja poruku koju želite prikazati.
3. **Trajanje** - može biti `Toast.LENGTH_SHORT` ili `Toast.LENGTH_LONG`, ovisno o tome koliko dugo želite da poruka ostane vidljiva.

Evo primjera kako prikazati Toast poruku kada se gumb klikne:

```
myButton.setOnClickListener(v -> {
    Toast.makeText(MainActivity.this, "Gumb je kliknut!",
Toast.LENGTH_SHORT).show();
});
```

> Potrebno je uvesti klasu `import android.widget.Toast;` za prikazivanje Toast poruka.



Toast

Druge metode elementa

Osim `setOnClickListener()`, postoje i druge metode za interakciju s UI elementima, ovisno o njihovoj vrsti. Na primjer:

- **EditText**:
  - `getText()` - dohvaća uneseni tekst
  - `setText(String text)` - postavlja tekst u polje
- **TextView**:
  - `setText(String text)` - postavlja tekst u prikaz
- **Button**:
  - `setEnabled(boolean enabled)` - omogućuje ili onemogućuje Gumb
- **CheckBox**:
  - `isChecked()` - provjerava je li označeno
  - `setChecked(boolean checked)` - postavlja stanje označenosti

Primjer korištenja `EditText` i `TextView`:

```
Button myButton = findViewById(R.id.my_button);
EditText inputField = findViewById(R.id.input_field);
TextView displayText = findViewById(R.id.display_text);

myButton.setOnClickListener(v -> {
    String userInput = inputField.getText().toString();
    displayText.setText(userInput);
});
```

getText() & setText() metode

> Potrebno je uvesti klase `import android.widget.EditText;` i `import android.widget.TextView;` za rad s tim elementima.

Potrebno je obratiti pažnju na konverziju tipova podataka prilikom dohvaćanja unosa iz `EditText`, jer se tekst dohvaća kao `String`. Ako je potreban broj, potrebno je izvršiti odgovarajuću konverziju, npr. `Integer.parseInt()` za cijele brojeve.

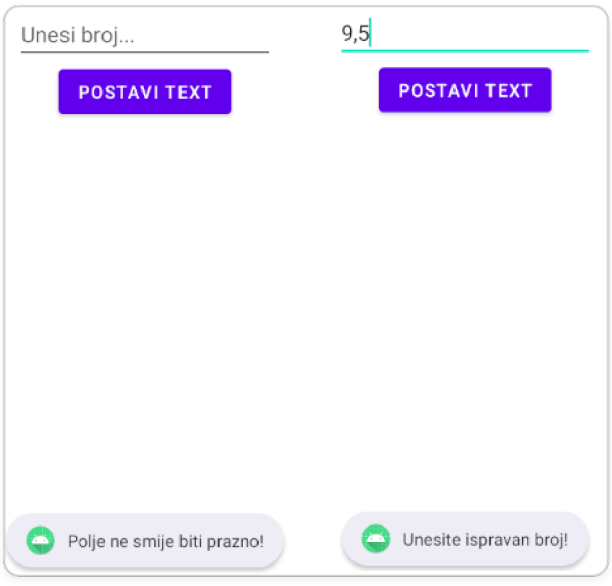
Primjer konverzije unosa iz `EditText` u `int`:

```
EditText numberInput = findViewById(R.id.number_input);
myButton.setOnClickListener(v -> {
    int number = Integer.parseInt(numberInput.getText().toString());
    // Daljnja obrada broja
});
```

U slučaju da korisnik unese neispravan broj ili ostavi polje prazno, može doći do iznimke `NumberFormatException`. Stoga je preporučljivo dodati provjeru unosa prije konverzije.

Primjer provjere unosa:

```
myButton.setOnClickListener(v -> {
    String inputText = numberInput.getText().toString();
    if (!inputText.isEmpty()) {
        try {
            int number = Integer.parseInt(inputText);
            // Daljnja obrada broja
        } catch (NumberFormatException e) {
            Toast.makeText(MainActivity.this, "Unesite ispravan broj!",
Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(MainActivity.this, "Polje ne smije biti prazno!",
Toast.LENGTH_SHORT).show();
    }
});
```



NumberFormatException

FoodTracker Zadatak

Potrebno je napraviti jednostavnu aplikaciju za praćenje kalorija unesenih prehrambenih proizvoda.

1. Model podataka Kreirajte novu klasu `FoodItem.java` koja sadrži:

- naziv proizvoda (`String`)
- broj kalorija (`int`)
- konstruktor koji postavlja oba podatka
- metode `getFoodName()` i `getKalorije()`

2. UI elementi (activity_main.xml) U prikaz dodajte:

- tekstualni unos za naziv (`EditText`)
- numerički unos za kalorije (`EditText`)
- dva gumba: "Dodaj stavku" i "Resetiraj"
- `TextView` za prikaz *Ukupno kalorija: 0*
- `TextView` za prikaz liste ili poruke *Prazna lista...*

3. Klasa FoodTracker Napravite novu klasu `FoodTracker.java` sa sljedećim funkcionalnostima:

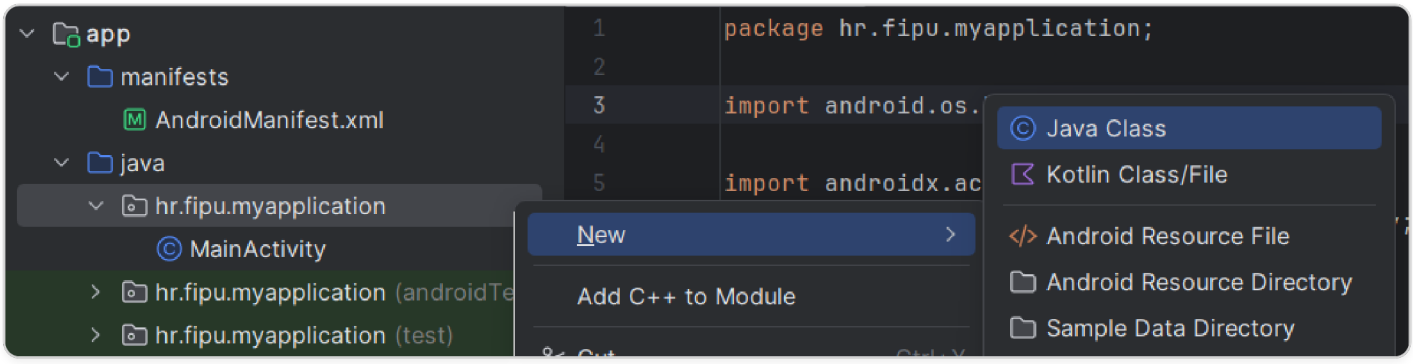
- lista `List<FoodItem>`
- metoda `addFoodItem()` za dodavanje stavke (*uz prikaz Toast poruke*)
- metoda `getTotalCalories()` za izračun ukupnih kalorija
- metoda `getFoodItems()` za dohvat liste
- metoda `clearAll()` za brisanje svih unosa

4. Spajanje UI-a i logike (MainActivity.java)

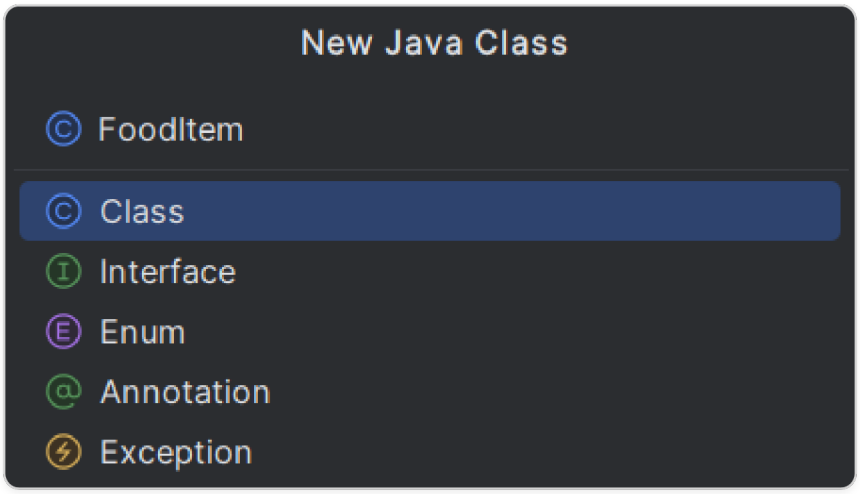
- kreirajte objekt `FoodTracker`
- dohvatite reference svih UI elemenata pomoću `findViewById()`
- na klik gumba *Dodaj*:
    - pročitajte unos
    - stvorite novi `FoodItem`
    - dodajte ga u `FoodTracker`
    - osvježite prikaz ukupnih kalorija
    - ispišite listu svih stavki
- na klik gumba *Resetiraj*:
    - obrišite sve stavke
    - postavite ukupne kalorije na 0
    - ispraznite prikaz liste

Primjer rješenja

Napraviti ćemo aplikaciju za praćenje kalorija. Prvo ćemo kreirati novu klasu `FoodItem.java`:



Nova Java klasa



Naziv nove klase

Zatim ćemo dodati sljedeći kôd u klasu:

```
public class FoodItem {

    private String FoodName;
    private int Kalorije;

    FoodItem(String foodName, int kalorije) {
        this.FoodName = foodName;
        this.Kalorije = kalorije;
    }

    public String getFoodName() {
        return this.FoodName;
    }

    public int getKalorije() {
        return this.Kalorije;
    }

}
```

Klasa `FoodItem` predstavlja model podataka za prehrambeni proizvod:

- **Atributi:**
  - `name` - naziv prehrambenog proizvoda (tipa `String`)
  - `calories` - broj kalorija (tipa `int`)
- **Konstruktor:**
  - Prima naziv i broj kalorija te inicijalizira oba atributa
  - Atributi su označeni kao `final`, što znači da se mogu postaviti samo jednom - prilikom stvaranja objekta
- **Metode:**
  - `getName()` vraća naziv proizvoda
  - `getCalories()` vraća broj kalorija

> Ova klasa je **nepromjenjiva (*immutable*)** jer nakon stvaranja objekta njegovi podaci se više ne mogu mijenjati.

Zatim ćemo u `activity_main.xml` dodati dva polja za unos:

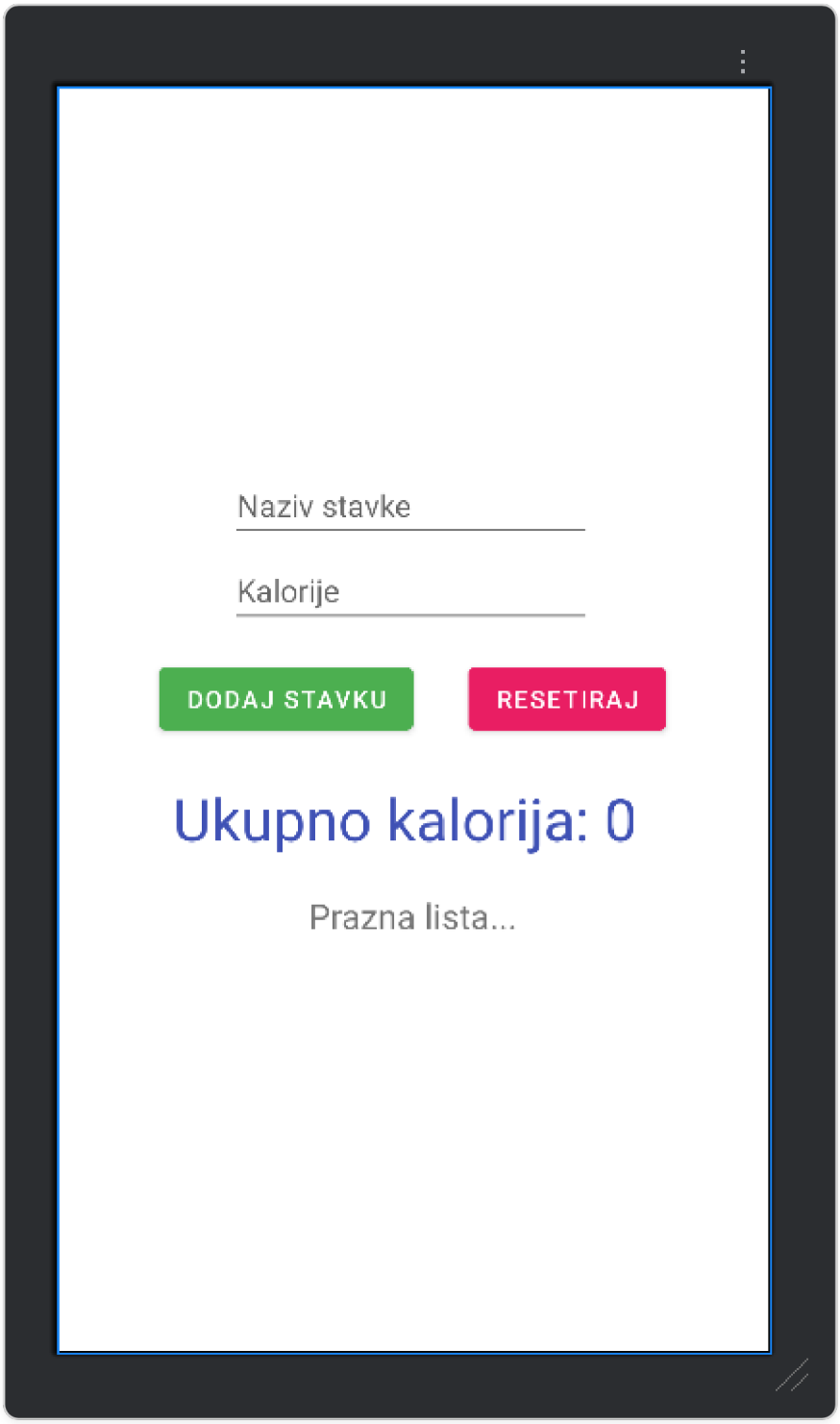
- polje za textualni input - `Plain Text` `<EditText>` element
- polje za brojčani input - `Number` `<EditText>` element

Dodat ćemo dva dugma `<Button>` elementa:

- jedno s nazivom "Dodaj stavku"
- drugo s nazivom "Resetiraj"

Dodati text `<TextView>` elemente "*Ukupno kalorija: 0*" i "*Prazna lista...*".

Tako da izgleda ovako:



Izgled

Zatim ćemo napraviti novu klasu `FoodTracker.java` i dodati sljedeći kôd:

```
import java.util.ArrayList;
import java.util.List;

public class FoodTracker {
    private List<FoodItem> foodItems = new ArrayList<>();

    public void addFoodItem(FoodItem foodItem, Context context) {
        foodItems.add(foodItem);
        Toast.makeText(context, ("Stavka " + foodItem.getFoodName() + "
uspješno dodana"), Toast.LENGTH_SHORT).show();
    }

    public int getTotalCalories() {
        int totalCalories = 0;

        for (FoodItem foodItem : foodItems) {
            totalCalories += foodItem.getKalorije();
        }
        return totalCalories;
    }

    public List<FoodItem> getFoodItems() {
        return foodItems;
    }

    public void clearAll() {
        foodItems.clear();
    }

}
```

Klasa `FoodTracker` služi za praćenje unosa hrane i zbrajanje ukupnog broja kalorija.

- **Atribut:**
  - `foodItems` - lista objekata tipa `FoodItem`, u kojoj se pohranjuju svi dodani prehrambeni proizvodi
- **Metode:**
  - `addFood(String name, int calories)` - dodaje novi prehrambeni proizvod u listu, stvarajući novi objekt klase `FoodItem`
  - `getTotalCalories()` - prolazi kroz cijelu listu i zbraja kalorije svih proizvoda; vraća ukupni broj kalorija
  - `getFoodItems()` - vraća cijelu listu prehrambenih proizvoda
  - `clearAll()` - briše sve proizvode iz liste (prazni praćenje hrane)

U `MainActivity.java` dodat ćemo objekt **FoodTracker**:

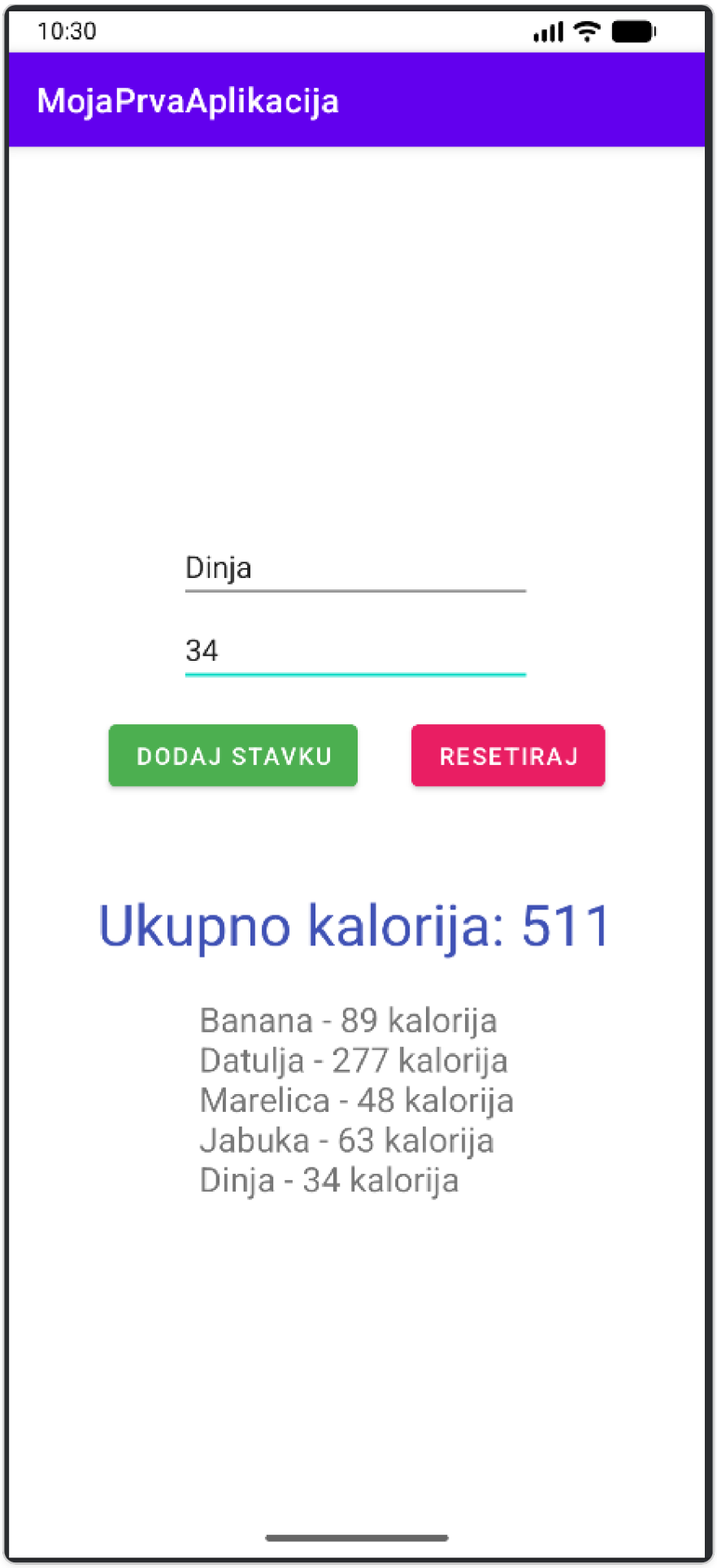
```
public class MainActivity extends AppCompatActivity {
  private FoodTracker foodTracker;
  ...
}
```

Sada kada imamo klase trebamo dohvatiti *reference* na elemente kosriteći njihov **ID** atribut. Pa ćemo dodati sljedeće nakon `setContentView()` funkcije:

```
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

...
setContentView(R.layout.activity_main);

foodTracker = new FoodTracker();

TextView textUkupno = findViewById(R.id.ukupno);
TextView textLista = findViewById(R.id.lista);
EditText inputNaziv = findViewById(R.id.inputNaziv);
EditText inputKalorije = findViewById(R.id.inputKalorije);
Button dodaj = findViewById(R.id.btnDodaj);
Button resetiraj = findViewById(R.id.btnResetiraj);
...
```

Nakon čega možemo dodati interakciju s elementima:

```
dodaj.setOnClickListener(view -> {
    String naziv = inputNaziv.getText().toString();
    int kalorije = Integer.parseInt(inputKalorije.getText().toString());
    FoodItem noviItem = new FoodItem(naziv, kalorije);
    foodTracker.addFoodItem(noviItem,this);

    int ukupnoKalorija = foodTracker.getTotalCalories();
    textUkupno.setText("Ukupno kalorija: " + ukupnoKalorija);

    String stavke = "";
    for (FoodItem item : foodTracker.getFoodItems()) {
        stavke += item.getFoodName() + " - " +
                  item.getKalorije() + " kalorija\n";
    }
    textLista.setText(stavke);

});

resetiraj.setOnClickListener(view -> {
    foodTracker.clearAll();
    textUkupno.setText("Ukupno kalorija: 0");
    textLista.setText("");
});
```


10:30
MojaPrvaAplikacija
Dinja
34
DODAJ STAVKU
RESETIRAJ
Ukupno kalorija: 511
Banana - 89 kalorija
Datulja - 277 kalorija
Marelica - 48 kalorija
Jabuka - 63 kalorija
Dinja - 34 kalorija

Primjer