ELEKTROTEHNIČKA I EKONOMSKA ŠKOLA NOVA GRADIŠKA

Autori:

Toni Jelašac

Luka Kovačević

Roberto Gemeri

Mentor: Zoran Ivančić

Godina 2021.

Sadržaj dokumentacije

UVOD

Predstavljanje problema

Za naš zadatak je bilo potrebno izraditi aplikaciju sa funkcijom kalkulatora.

Bilo je potrebno izraditi kalkulator sa osnovnim funkcijama kao zbrajanje,oduzimanje,množenje I dijeljenje I znanstveni kalkulator.

Predstavljanje rješenja

Problem smo riješili aplikacijom koja koristi fragmente za dva različita prikaza tipki. Prvi prikaz je standardni kalkulator, a drugi je znanstveni.

0. Biblioteke/Dependencies

a. Javaluator

http://javaluator.sourceforge.net/en/home/

"Javaluator is a simple, but powerful, infix expression evaluator for Java."

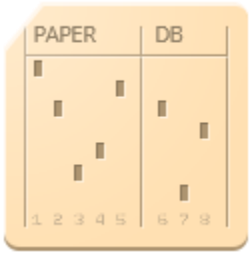
Javaluator je evaluator koji je napisan u Java programskom jeziku.

Vrlo je jednostavan za korisititi zbog svojih ugrađenih funkcija za evaluaciju String tipa podataka.

Primjer korištenja:

```
String expression = "(2^3-1)*sin(pi/4)/ln(pi^2)";
Double result = new DoubleEvaluator().evaluate(expression);
```

b. *Paper*



https://github.com/pilgr/Paper

“Paper's aim is to provide a simple yet fast object storage option for Android. It allows to use Java/Kotlin classes as is: without annotations, factory methods, mandatory class extensions etc. Moreover adding or removing fields to data classes is no longer a pain – all data structure changes are handled automatically.”

Paper je biblioteka koja se koristi za jednostavno spremanje mnogo tipova podataka preko key-value sistema. Može se koristiti za tipove podataka od primitivnih, do user-made klasa.

U ovoj aplikaciji se koristi za spremanje povijesti računanja I umemoriranih vrijednosti.

1. Splash screen

a. Front end

Prilikom pokretanja aplikacije se pojavi ovaj splash screen u kojem se izvrši animacija logotipa aplikacije. Tijekom animacije se u pozadini učitavaju I inicijaliziraju potrebne biblioteke.

b. Back end(Java)

```java
import io.paperdb.Paper;

import android.animation.AnimatorSet;
import android.animation.ObjectAnimator;
import android.animation.PropertyValuesHolder;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;


public class Splash extends AppCompatActivity {
    //Inicijalizacija varijabli i elemenata na zaslonu
    int logoDuration = 500;
    int textDuration = 200;
    int splashDelay = 800;
    private ImageView imageView;
    private TextView textView;
    private View view;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //Inicijalizacija Paper biblioteke i postavljanje podataka ako već ne postoje
        Paper.init( context: this);
        if (Paper.book().read( key: "calculatorHistory") == null){
            ArrayList<String> empty=new ArrayList<~>();
            Paper.book().write("calculatorHistory",empty);
        }
        if (Paper.book().read( key: "memory") == null){
            Paper.book().write("memory","");
        }
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);

        //UI/Fullscreen scaling fixes
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowMan

        imageView = (ImageView) findViewById(R.id.AnimiranLogo);
        textView = (TextView) findViewById(R.id.ezcalc);

        handleAnimation(view);
    }
```

-Importanje potrebnih biblioteka i klasa

-Inicijalizacija I definicija varijabli/zaslonskih elemenata

-Inicijalizacija Paper biblioteke I postavljanje početnih vrijednosti za calculatorHistory I memory varijable.

-Postavljanje zaslonskih elemenata

```
public void handleAnimation (View view) {

    PropertyValuesHolder logoMove = PropertyValuesHolder.ofFloat( propertyName: "y", ...values: 1000f,-75f);
    PropertyValuesHolder logoScaleX = PropertyValuesHolder.ofFloat(View.SCALE_X, ...values: 0.05f,0.7f);
    PropertyValuesHolder logoScaleY = PropertyValuesHolder.ofFloat(View.SCALE_Y, ...values: 0.05f,0.7f);

    ObjectAnimator logo = ObjectAnimator.ofPropertyValuesHolder(imageView,logoMove,logoScaleX,logoScaleY).setDuration(logoDuration);

    ObjectAnimator ezcalcNull = ObjectAnimator.ofFloat(textView,View.ALPHA, ...values: 1.0f, 0.0f).setDuration(0);
    ObjectAnimator ezcalcAlpha = ObjectAnimator.ofFloat(textView,View.ALPHA,  ...values: 0.0f, 1.0f).setDuration(textDuration);


    AnimatorSet animatorSet = new AnimatorSet();
    animatorSet.playSequentially(ezcalcNull,logo,ezcalcAlpha);
    animatorSet.start();

    new Handler().postDelayed(()-> { //Kod u ovoj funkciji se izvede nakon što zadano vrijeme splashDelay istekne.
        Splash.this.startActivity(new Intent( packageContext: Splash.this, MainActivity.class));
        Splash.this.finish();
    }, splashDelay);
}
```

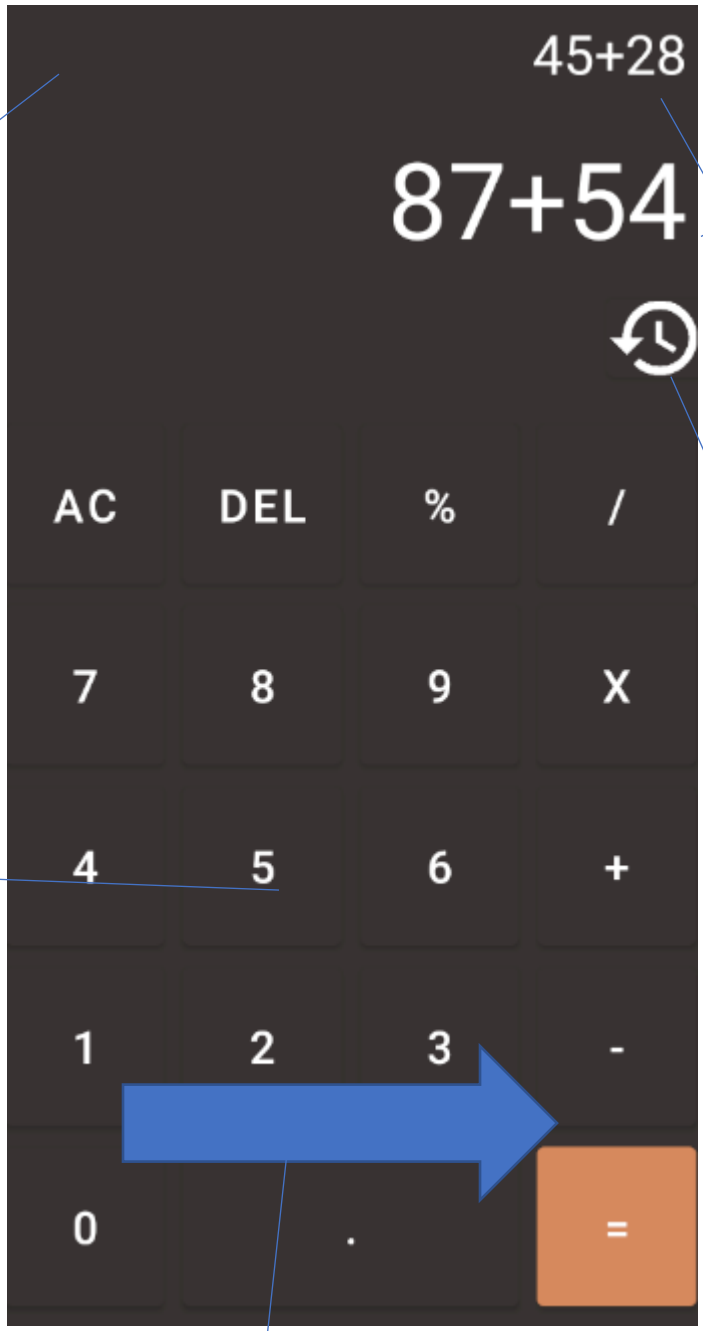
- Funkcija koja obavlja animaciju logotipa aplikacije.

- Funkcija koja zatvori Splash aktivnost nakon postavljenog delay-a

2. Glavni prikaz

a. Front end

Prikaz 1:



Indikator stanja memorije (trenutno sakriven; memorija prazna.)

Glavni prikaz za računanje/rezultate.

Prikaz povijesti računanja.

Tipke za unos

Tipka za prikaz pune povijesti računanja

Prijelazom prsta po tipkama prema desno se dolazi do znanstvenog kalkulatora.

Za to se koristi ViewPager2

Prikaz 2(Znanstveni):

Indikator stanja memorije(trenutno prikazan;u memoriji postoji zapis

Osnovne funkcije I prikazi ostanu u znanstvenom kalkulatoru.

Dodatne funkcije se pojave u ovom prikazu.

b. Back end

```
public class MainActivity extends AppCompatActivity {
    TextView results;
    TextView resultsHitsory;
    DoubleEvaluator evaluator;
    TextView memoryStatus;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Function SQRT = new Function( name: "sqrt",  argumentCount: 1);
        Parameters params = DoubleEvaluator.getDefaultParameters();
        params.add(SQRT);
        DoubleEvaluator evaluator = new DoubleEvaluator(params){
            @Override
            protected Double evaluate(Function function, Iterator arguments, Object evaluationContext) {
                if (function == SQRT) {
                    // Implements the new function
                    return Math.sqrt((Double) arguments.next());
                } else {
                    // If it's another function, pass it to DoubleEvaluator
                    return super.evaluate(function, arguments, evaluationContext);
                }
            }
        };


        results = findViewById(R.id.currentView);
        resultsHitsory = findViewById(R.id.historyMiniView);
        memoryStatus = findViewById(R.id.memoryStatus);
        results.setText("");
        ViewPager2 viewPager = findViewById(R.id.viewPager);
        FragmentStateAdapter pageAdapter = new ScreenSlidePageAdapter( fa: this);
        viewPager.setAdapter(pageAdapter);
        if (Paper.book().read( key: "memory") == ""){
            memoryStatus.setVisibility(View.INVISIBLE);
        }else{
            memoryStatus.setVisibility(View.VISIBLE);
        }
        //UI/Fullscreen scaling fixes
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams


        View decorView = getWindow().getDecorView();
        int uiOptions = SYSTEM_UI_FLAG_HIDE_NAVIGATION
```

-Inicijalizacija elemenata prikaza.

-Definicija custom funkcije za korijenovanje I inicijaliziranje evaluatora.

-Postavljanje elemenata, ViewPager-a, čitanje I postavljanje indikatora memorije.

-Postavljanje UI opcija.

```
private boolean isInsideDecimal(String str)
{

        int len = str.length();

        for (int i = len - 1; i >= 0; i--) {
            if (str.charAt(i) == '+' || str.charAt(i) == '-' || str.charAt(i) == '/' || str.charAt(i) =
                return false;
            }
            else if (str.charAt(i) == '.'){
                return true;
            }
        }
        return false;

}

private static class ScreenSlidePageAdapter extends FragmentStateAdapter {

    public ScreenSlidePageAdapter(FragmentActivity fa) { super(fa); }

    @NonNull
    @Override
    public Fragment createFragment(int position) {
        switch (position) {
            case 0:
                return new FragmentSimple();
            case 1:
                return new FragmentScientfic();
        }
        return null;
    }

    @Override
    public int getItemCount() { return 2; }
}
```

-Funkcija za provjeru stanja decimalnog broja.(Koristi se za prevenciju stavljanja više od jedne decimalne točke u jednom decimalnom broju)

-ScreenSlide adapter koji služi za prijelaz između fragmenata jednostavnog I znanstvenog kalkulatora.

```
/Definicija dugmadi
ublic void onClick(View view){

   int id = view.getId();
   if (id==R.id.SIMPLEbtn0){
       try {
           results.setText(results.getText()+"0");
       }
       catch(Exception e){
           Toast toast = Toast.makeText(getApplicationContext(), text: "Generic string error!",Toast.LENGT
           toast.show();
       }
   }
   else if(id==R.id.SIMPLEbtn1){
       try {
           results.setText(results.getText()+"1");
       }
       catch(Exception e){
           Toast toast = Toast.makeText(getApplicationContext(), text: "Generic string error!",Toast.LENGT
           toast.show();
       }
   }
   else if(id==R.id.SIMPLEbt2){
       try {
           results.setText(results.getText()+"2");
       }
       catch(Exception e){
           Toast toast = Toast.makeText(getApplicationContext(), text: "Generic string error!",Toast.LENGT
           toast.show();
       }
   }
   else if(id==R.id.SIMPLEbtn3){
       try {
           results.setText(results.getText()+"3");
       }
       catch(Exception e){
           Toast toast = Toast.makeText(getApplicationContext(), text: "Generic string error!",Toast.LENGT
           toast.show();
       }
   }
   else if(id==R.id.SIMPLEbtn4){
       try {
           results.setText(results.getText()+"4");
       }
```

-OnClick funkcija koja obavlja zadatke vezane za stiskanje određenih tipki.

-Kada je neka tipka pritisnuta u fragmentu, on vrati Clicked event i id pritisnute tipke. Preko toga znamo koja je tipka stisnuta I što trebamo odraditi.

-Funkcija obavlja zadatke I za jednostavni I za znanstveni kalkulator.

```
else if(id==R.id.SIMPLEequals) {
        Log.d( tag: "D", msg: "calculating:"+ results.getText());
        Double result = evaluator.evaluate(results.getText().toString());
        ArrayList<String> currentValues = Paper.book().read( key: "calculatorHistory");
        resultsHitsory.setText(results.getText());
        results.setText(results.getText() + "=");
        if (result % 1 == 0) {
            currentValues.add(results.getText() + " = " + (int) Math.round(result));
            results.setText(Integer.toString((int) Math.round(result)));
        } else {
            currentValues.add(results.getText() + " = " + result);
            results.setText(String.valueOf(result));
        }
        Paper.book().write("calculatorHistory", currentValues);

}
else if(id==R.id.SIMPLEdecimalPoint){
    try {
        int stringlen = results.length();
        if(results.getText().charAt(stringlen - 1) != '.'){
            if(results.getText().charAt(stringlen - 1) == '\0' || results.getText().charAt(stringle
                results.setText(results.getText()+"0.");
            }else{
                if(!isInsideDecimal(results.getText().toString())) {
                    results.setText(results.getText()+"0.");
                }
            }
        }
    }
    catch(Exception e){
        Toast toast = Toast.makeText(getApplicationContext(), text: "Generic string error!",Toast.LEN
        toast.show();
    }
}

else if(id==R.id.piButton){
    try {
        results.setText(results.getText()+"pi");
    }
```

-Naprednije funkcije se obavljaju na isti način.

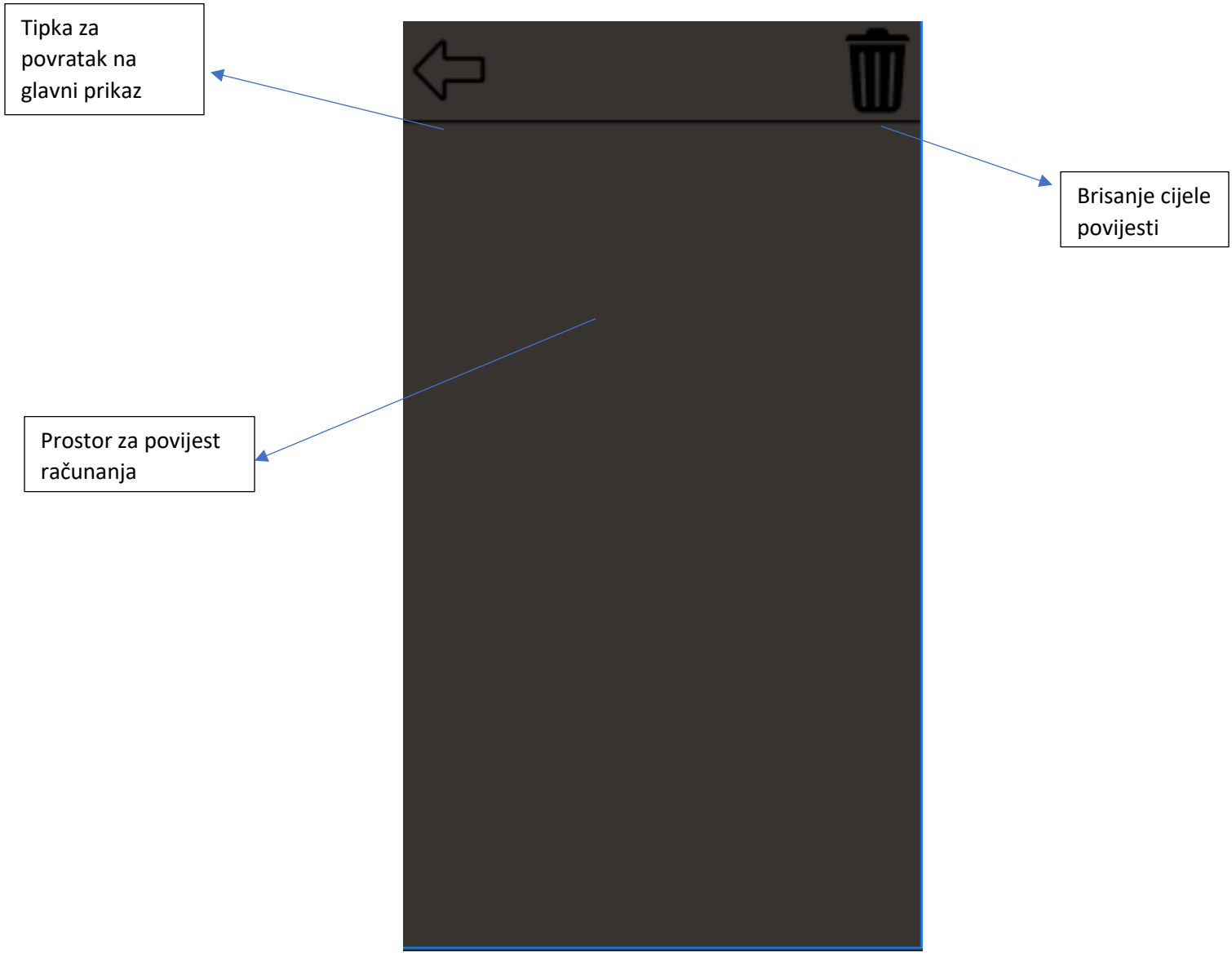
-Kod equals tipke se javaluatoru posalje string jednadžbe za riješit, koji se spremi u history I prikaže na glavnom prikazu.

-Kod ove funkcije aplikacija prvo provjeri valjanost postavljanja decimalne tocke na trenutnoj poziciji, pa ju onda postavi ako je moguće.

-Neke tipke se jednostavno dodaju na kraj jednadžbe.

3. History activity

a. Front end



b. Back end(Java)

```java
public class HistoryActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_history);
        //UI/Fullscreen scaling fixes
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,

        ImageButton back = findViewById(R.id.Back);
        ImageButton deleteHistory = findViewById(R.id.deleteHistory);

        setView();

        back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { HistoryActivity.this.finish(); }
        });

        deleteHistory.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ArrayList<String> empty=new ArrayList<~>();
                Paper.book().write("calculatorHistory",empty);
                setView();
            }
        });
    }
```

-Postavljanje UI/Fullscreen postavki

-Postavljanje tipki I pozivanje funkcija za inicijalizaciju

-OnClickListener za brisanje povijesti.

```
public void setView(){
    ListView list = findViewById(R.id.list);

    ArrayList<String> stringArrayValues = Paper.book().read( key: "calculatorHistory");
    ArrayAdapter<String> listAdapter = new ArrayAdapter<~>( context: this, android.R.layout.simple_list_item_1
        @Override
        public View getView(int position, View convertView, ViewGroup parent){
            // Get the Item from ListView
            View view = super.getView(position, convertView, parent);

            // Initialize a TextView for ListView each Item
            TextView tv = (TextView) view.findViewById(android.R.id.text1);

            // Set the text color of TextView (ListView Item)
            tv.setMaxWidth(200);
            tv.setTextColor(Color.WHITE);
            tv.setTextAlignment(View.TEXT_ALIGNMENT_TEXT_END);
            tv.setAutoSizeTextTypeUniformWithConfiguration( autoSizeMinTextSize: 1,  autoSizeMaxTextSize: 29,  autoSiz


            // Generate ListView Item using TextView
            return view;
        }
    };
    list.setAdapter(listAdapter);
}
}
```

-Učitavanje povijesti iz Paper baze podataka I potsavljenje ih na arraylist preko listAdapter-a.

-Postavljanje textView elemenata za prikaz prijašnjih računa.

4. Vanjske poveznice

Poveznica na GitHub stranicu EZCalc aplikacije:
https://github.com/TheUNkilled123/EZCalc

Poveznica na Javaluator SourceForge:
http://javaluator.sourceforge.net/en/home/

Poveznica na GitHub stranicu Paper biblioteke:

https://github.com/pilgr/Paper