Пловдивски университет „Паисий Хилендарски”



Факултет по математика и информатика

Курсова работа

По дисциплина „Програмиране на приложения за мобилни устройства“

На тема: „Калкулатор за решаване на квадратно уравнение и обикновен калкулатор“

|  |  |
| --- | --- |
| Изготвил: Полина Белчева  Специалност: СТД редовно  Факултетен номер: 1701681053 | Проверил:  / доц. д‐р Н. Касъклиев/ |
|  |  |

Съдържание

[1. Увод 3](#_Toc42620079)

[2. Виртуализация на приложението 4](#_Toc42620080)

[I. Калкулатор за решаване на квадратно уравнение 4](#_Toc42620081)

[II. Обикновен калкулатор 4](#_Toc42620082)

[3. Създаване на функциалността на приложението 4](#_Toc42620083)

[I. Изчисления 5](#_Toc42620084)

[a. Калкулатор за решаване на квадратно уравнение 5](#_Toc42620085)

[b. Обикновен калкулатор 6](#_Toc42620086)

[II. Създаване на ново активити 6](#_Toc42620087)

[III. Добавяне на локална база 6](#_Toc42620088)

[a. Калкулатор за решаване на квадратно уравнение 6](#_Toc42620089)

[b. Обикновен калкулатор 7](#_Toc42620090)

[4. Записване на резултат от изчисления 9](#_Toc42620091)

[5. Изтриване на записите в базата данни 10](#_Toc42620092)

[6. Грешки 10](#_Toc42620093)

[7. Заключение 10](#_Toc42620094)

[8. Използвана литература 11](#_Toc42620095)

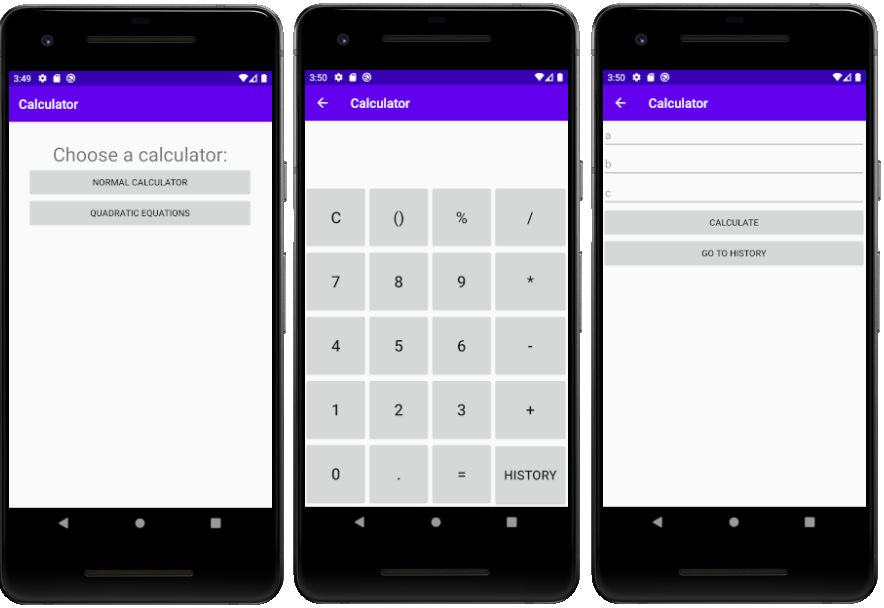
[9. Списък на фигурите 11](#_Toc42620096)

# Увод

Приложението съдържа два калкулатора, при които единия изчислява корените на едно квадратно уравнение по въведени реални коефициенти, а другия е обикновен калкулатор.

Приложението е написано на Java и е разработено в среда Android Studio, защото е гъвкава и удобна за разработване и тестване на всички устройства с Android. За база данни се използва SQLite, защото е по-лека и е подходяща за мобилни устройства.

От приложението се изисква да изчислява корените на квадратно уравнение от втора степен, да събира, да изважда, да дели, да умножава, да изчислява процент, да пази историята от последните изчисления и резултатите в БД, които да може да виртуализира и да могат да бъдат изтрити.



Фигура 1 - Графичен потребителски интерфейс на приложението

# Виртуализация на приложението

## Калкулатор за решаване на квадратно уравнение

За да може да работи приложението са необжодими полета за въвеждане на коефициентите, бутон, който да изчислява формуата, елемент за показване на текст, в което извеждаме резултата, списък със записаните резултати от БД и бутон за изтриване на списъка.

В главното активити се използва линейно оформление (Linear Layout) с вертикална ориентация, като се избутва съдържанието отдясно и отляво.

За полетата за въвеждане се използва <EditText> елемент с тип за десетично число.

<EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="numberDecimal|numberSigned"  
 android:ems="10"  
 android:hint="a"  
 android:id="@+id/et\_a"/>

Фигура 2 – Дизайн на поле за въвеждане <TextView>

За бутон се използва елемент <Button> и полето за текст се използва елемент <TextView>.

Във второстепенното активити се използва отново линейно оформление (Linear Layout) с вертикална ориентация, като се избутва съдържанието отдясно и отляво. За бутон се използва елемент <Button> и съответно за списъка със записите <ListView>.

## Обикновен калкулатор

Необходими са текстови полета за въвеждане и извеждане на текст, бутони за въвеждане на изчисленията и изчисляване на формулата, която сме задали, списък със записаните резултати от БД и бутон за изтриване на списъка.

В главното активити се използва <ScrollView>, <LinearLayout>, <RelativeLayout> с вертикална ориентация, която избутва съдържанието отдясно и отляво.

С помощта на <include> вкарваме в главното активити второстепенните активитита. Input\_layout.xml съдържа две <TextView> за въвеждане и извеждане на резултата, а buttons.xml - <GridLayout> и <Button>.

# Създаване на функциалността на приложението

След като интерфейсът е готов, се създава функциалността на приложението. Във всяко едно активити се включват референциите на елементите, които сме създали, за да могат да се използват. В MainActivity.java създаваме по един метод за всеки от бутоните за отваряне на нов прозорец.

## Изчисления

### Калкулатор за решаване на квадратно уравнение

На бутона за калкулация btnCalc задаваме click listener със setOnClickListener функция. Този click listener е функция, която ще бъде изпълнена при всеки натиск на бутона от потребителя. Изпълнява try … catch ... finally, с който прави няколко проверки. Първата е използване на условен оператор, с който правим валидация на данните в полетата. Функцията ще пресмята корените. За да пресметнем корените трябва да знаем математическата формула и как да я представим на програмен език.

if (!et\_a.getText().toString().equals("") && !et\_b.getText().toString().equals("")  
 && !et\_c.getText().toString().equals("")) {  
 A = Double.*parseDouble*(et\_a.getText().toString());  
 B = Double.*parseDouble*(et\_b.getText().toString());  
 C = Double.*parseDouble*(et\_c.getText().toString());  
  
 if (A == 0) {  
 x1 = -C/B;  
 tv\_result.setText("Only one root: X = " + x1 + "\n");  
 } else {  
  
 D = Math.*pow*(B, 2) - 4 \* A \* C;  
  
 if (D == 0) {  
 x1 = -B / (2 \* A);  
 tv\_result.setText("D = " + D + "\nx = " + x1 + "\n");  
 } else if (D < 0) {  
 tv\_result.setText("No real roots!" + "\n");  
 } else if (D > 0) {  
 x1 = (-B + Math.*sqrt*(D)) / (2 \* A);  
 x2 = (-B - Math.*sqrt*(D)) / (2 \* A);  
 tv\_result.setText("D = " + D + "\nx1 = " + x1 + "\nx2 = " + x2 + "\n");  
 }  
 }  
}

Фигура 3 – Пресмятане на корени от кв. у-е

В този секмент е изпълнена логиката на квадратно уравнение. Обходени са всички случаи. Накрая са пресметнати корените и са изведени на екрана.

### Обикновен калкулатор

На всеки бутон задаваме click listener със setOnClickListener функция. При всяко натискане на съответния бутон да се изписва дадения знак от бутона.

btnEqual.setOnClickListener(v -> {  
 process = tv\_Input.getText().toString();  
 process = process.replaceAll("%", "/100");  
  
 org.mozilla.javascript.Context rhino = org.mozilla.javascript.Context.*enter*();  
 rhino.setOptimizationLevel(-1);  
  
 String finalResult = "";  
  
 try {  
 Scriptable scriptable = rhino.initSafeStandardObjects();  
 finalResult = rhino.evaluateString(scriptable, process, "javascript", 1, null) .toString();  
 } catch (Exception e) {  
 finalResult = "0";  
 }  
  
 tv\_Output.setText(finalResult);

Фигура 4 – Бутон за изчисляване на обикновен калкулатор

В този сегмент е изпълнена логиката какво да се случи след натискане на бутона за равенство.

## Създаване на ново активити

Създаваме ново активити, с наименования RootsHistory.java и activity\_history.xml за калкулатора за решаване на квадратно уравнение и съответно CalculatorHistory.java и activity\_calculator\_history.xml за обикновения калкулатор, където ще се показва списък с изведени корени и изчисления. Включват се референциите на елементите, които сме създали, за да могат да се използват.

## Добавяне на локална база

За добавяне на локална база ще ни трябват два метода, тъй като ще се ползват на няколко места. Първият метод е initDB(), който прави референция на локална база данни.

### Калкулатор за решаване на квадратно уравнение

В него създаваме референция към нова SQLite база данни, във файл rootsDB.db, в която създаваме таблица с нужните полета – ID, който е самоувеличаващ се, коефициенти A, B, C, корени х1, х2 и подаваме като уникални A, B, C, с цел да избегнем дублиране на записи.

### Обикновен калкулатор

В него създаваме референция към нова SQLite база данни, но във файл resultsDB.db. В нея създаваме таблица с полета – ID, INPUT и OUTPUT.

protected void initDB() throws SQLException {

SQLiteDatabase database = null;

database = SQLiteDatabase.openOrCreateDatabase(

getFilesDir().getPath() + "/" + "resultsDB.db",

null

);

String q = "CREATE TABLE if not exists RESULT ( ";

q += " ID integer primary key AUTOINCREMENT, ";

q += " INPUT text not null, ";

q += " OUTPUT text not null, ";

q += " unique( INPUT, OUTPUT ) );";

database.execSQL(q);

database.close();

}

Фигура 5 – Създаване на локална БД на обикновен калкулатор

Създаваме ново активити, в което ще се съдържат данните на предишните записи. Ще се казват rootsListView.java и activity\_roots\_list\_view.xml и съответно CalcResultListView.java и activity\_calc\_result\_list\_view.xml. От него ще ни трябва само xml файла му, който е един и същ за **двете** активитита.

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="20sp"  
 android:id="@+id/textView"/>  
  
</LinearLayout>

Фигура 6 – Дизайн на activity\_list\_view.xml

Вторият метод е selectBD(), в който отново създаваме референция към локалната база от данни. Селектираме всички записи от базата подредени по ID, създаваме ArrayList от String-ове и с помощта на Cursor обект, обхождаме резултата от заявката като прикрепяме нов елемент към ArrayList-а.

След като Cursor-a няма какво повече да обхожда, се създава ArrayAdapter с помощта на новото активити activity\_roots\_list\_view.xml / activity\_calc\_result\_list\_view.xml. Това свързва данните от базата с елементите в графичната част и за всеки един елемент ще покаже на екрана текстов елемент с данни вътре.

ArrayList<String> listResults= new ArrayList<String>();  
String query = "SELECT \* FROM ROOTS ORDER BY ID;";  
Cursor cursor = db.rawQuery(query, null);  
  
while (cursor.moveToNext()) {  
 String A = cursor.getString(cursor.getColumnIndex("A"));  
 String B = cursor.getString(cursor.getColumnIndex("B"));  
 String C = cursor.getString(cursor.getColumnIndex("C"));  
 String x1 = cursor.getString(cursor.getColumnIndex("x1"));  
 String x2 = cursor.getString(cursor.getColumnIndex("x2"));  
 listResults.add("A:" + A + " / " + "B:" + B + " / " + "C:"  
 + C + " / " + "x1:" + x1 + " / " + "x2:" + x2);  
}  
ArrayAdapter<String> arrayAdapter =  
 new ArrayAdapter<String>(  
 getApplicationContext(),  
 R.layout.*activity\_roots\_list\_view*,  
 R.id.*textView*,  
 listResults );

Фигура 7 – Метод за селект от локалната БД на калкулатор за решаване на кв. у-е

След което се прави референция към локалната БД от главното активити и се създава заявка за добавяне на данните към запис в таблицата ROOTS / RESULT. Ако данните са коректни, записът се добавя.

Извежда се съобщение за успешно добавен запис. Изчисляваме променливата, което държи референция към базата данни, с цел да намалин т.нар. „garbage collection” или разхода на памет на мобилното устройство.

# Записване на резултат от изчисления

След натискане на бутона „Calculate“ (bntCalc от калкулатора за решаване на кв. у-е) / бутона „=“ (btnEqual от обикновения калкулатор), приложението трябва да записва резултатите в съответната локалната база. Това става с помощта на INSERT заявка, която трябва да бъде изпълнена след намиране на изчисленията и изписването им на дисплея.

SQLiteDatabase database = null;  
try {  
 database = SQLiteDatabase.*openOrCreateDatabase*(  
 getFilesDir().getPath() + "/" + "resultsDB.db",  
 null  
 );  
  
 String q = "INSERT INTO RESULT (INPUT, OUTPUT) ";  
 q += "VALUES(?, ?); ";  
 database.execSQL(q, new Object[]{process, finalResult});  
  
 Toast.*makeText*(getApplicationContext(),  
 "Successfully added new results to record!",  
 Toast.*LENGTH\_LONG*).show();  
} catch (Exception e) {  
 Toast.*makeText*(getApplicationContext(), "Record already added!",  
 Toast.*LENGTH\_LONG*).show();  
} finally {  
 if (database != null) {  
 database.close();  
 database = null;  
 }  
}

Фигура 8 – Записване на резултат от изчисления на обикновен калкулатор

Отново се създава референция към локалната база данни. Пише се SQLite заявка и се изпълнява. След изпълнение на записа в базата данни, ако не са възникнали никакви проблеми, се изпълнява selectDB(), за да може да се обновят текущите показани записи на дисплея.

# Изтриване на записите в базата данни

Създава се функциалността на бутона за изчисляване на записите btnClear / btnCalcClear като се добавя click listener, чрез който прави заявка към БД да изтрие записите от таблицата ROOTS / RESULT. Като завършек, отново се извиква selectDB(), за да покаже празен списък. След изтриване на записите връща потребителя в главното активити с помощта на CloseThisActivity().

db = SQLiteDatabase.*openOrCreateDatabase*(getFilesDir().getPath() + "/" + "rootsDB.db",null);  
String query = "DELETE FROM ROOTS";  
db.execSQL(query);  
Toast.*makeText*(getApplicationContext(),  
 "All records deleted!",  
 Toast.*LENGTH\_LONG*).show();

Фигура 9 – Изтриване на записите в БД на калрулатор за решаване на кв. у-е

# Грешки

При калкулатора за решаване на квадратно уравнение:

* когато уравнението няма реални корени като резултат се изписват нули.
* когато уравнението има само един реален корен втория се изписва като нула.

При обикновения калкулатор:

* има възможност за добавяне на няколко знака един след друг.
* когато няма изчислителни знаци и въведем примерно 18(2+3), резултата е нула.
* когато екрана е малък, бутоните от първата колона се разпъват.

# Заключение

Приложението е семпло и с ограничена функционалност. Използвани са похвати от основи на програмирането, работа с графични елементи, работа с методи, работа с цикли, основни оператори и локална база данни.

За тестване на приложението няма написани Unit тестове. Може да се тества от гледна точка на QA Developer, като се въвеждат различни стойности в полетата и се сверява крайния резултат.

За да стане функционално и да отговаря на стандартите за мобилни приложения от 2020 година, трябва да се добавят още функционалности и да се направи по изчистен и съвременнен дизайн.

# Използвана литература

1. Формули за квадратно уравнение - <https://www.matematika.bg/algebra/quadratic-equation.html>

2. Tihomir Radeff, „Develop Quadratic Equation solver in Android Studio“ (20.04.2016) - <https://www.youtube.com/watch?v=XfJuce2BDbI>

3. гл. ас. д-р Г. Пашев, „Задание № 2 - Sqlite3 Basics“ (13.04.2020) - <https://drive.google.com/file/d/1t82jPYwWFnYUtHxgmE9qRXWlXiAC_42C/view>

4. Coding in Flow, “ How to Open a New Activity and Pass Variables to It” (21.10.2017) - <https://www.youtube.com/watch?v=eL69kj-_Wvs>

5. CodingWithMe , „How to create SIMPLE CALCULATOR in Android Studio || create a calculator in android“ (15.10.2019) - <https://www.youtube.com/watch?v=H_NdNqdZa04&t=194s>

6.

# Списък на фигурите

[Фигура 1 - Графичен потребителски интерфейс на приложението 3](#_Toc42615374)

[Фигура 2 – Дизайн на поле за въвеждане <TextView> 4](#_Toc42615375)

[Фигура 3 – Пресмятане на корени от кв. у-е 5](#_Toc42615376)

[Фигура 4 – Бутон за изчисляване на обикновен калкулатор 6](#_Toc42615377)

[Фигура 5 – Създаване на локална БД на обикновен калкулатор 7](#_Toc42615378)

[Фигура 6 – Дизайн на activity\_list\_view.xml 7](#_Toc42615379)

[Фигура 7 – Метод за селект от локалната БД на калкулатор за решаване на кв. у-е 8](#_Toc42615380)

[Фигура 8 – Записване на резултат от изчисления на обикновен калкулатор 9](#_Toc42615381)

[Фигура 9 – Изтриване на записите в БД на калрулатор за решаване на кв. у-е 10](#_Toc42615382)