Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рыбинский государственный авиационный технический университет

имени П. А. Соловьева»

Факультет радиоэлектроники и информатики

Кафедра вычислительных систем

**ОТЧЕТ**

по дисциплине

«Мобильные системы»

на тему «Разработка мобильного приложения для операционной системы Android»

Студенты группы ИВМ-20……………………….…………………Соловьев И. А.

Соловьева М. А.

Преподавать к.т.н., доцент ..….….….………….….….……………Павлов Р. В.

Рыбинск 2020

Введение

Мобильные приложения используются повсеместно и их функции охватывают практически все сферы деятельности. Использование мобильных приложений является более удобным вариантом по сравнению с веб-версиями сайтов, так как приложения могут быть более гибкими и используют ресурсы системы оптимальнее. В рамках данной лабораторной работы необходимо разработать мобильное приложение для платформы Android.

1 Выбор программы разработки и версий библиотек

Работа над программным кодом для операционной системы *Android* велась в программе *Android Studio Arctic Fox 2020.3.1 Patch 4*, разработанной компанией *Google LLC*. Данное ПО является стандартной программой для разработки под Android и позволяет использовать все возможности данной операционной системы.

Поскольку разработанная программа под *Android* будет работать с распределенными вычислениями посредством *REST*-запросов, то для этого была выбрана библиотека *Retrofit* версии 2.5.0. В связи с тем, что для обмена *REST*-запросами необходимо упаковывать информацию в *JSON* и извлекать её оттуда, была выбрана библиотека *converter-gson* версии 2.5.0 входящая в пакет *Retrofit.*

2 Описание функционала программы

Данная программа реализует функционал электротехнического калькулятора, который позволяет рассчитывать реактивное сопротивление катушки индуктивности и реактивное сопротивление конденсатора. Интерфейс основного экрана изображен на рисунке 1.

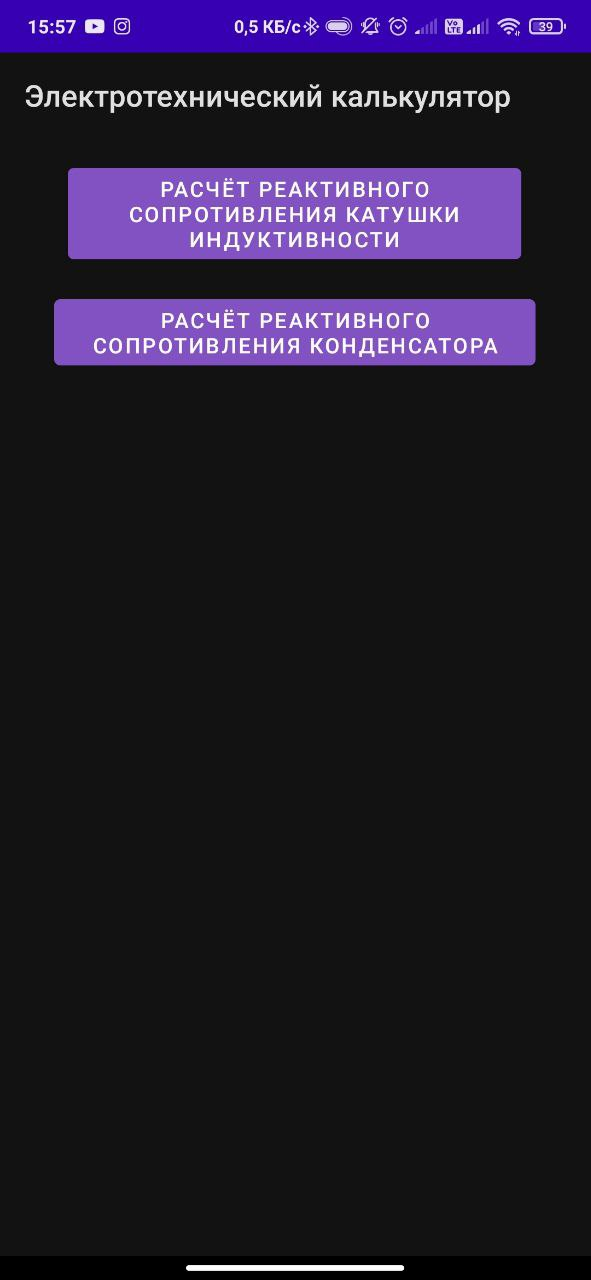


Рисунок 1 – Интерфейс основного экрана приложения

Для расчёта реактивного сопротивления катушки индуктивности необходимо ввести частоту и индуктивность, указав размерность величины. Частоту можно указывать в герцах, килогерцах и мегагерцах, а индуктивность можно указывать в Генри, миллигенри и микрогенри. Выбор величины указывается при помощи выпадающего меню. Интерфейс данного экрана изображен на рисунке 2.

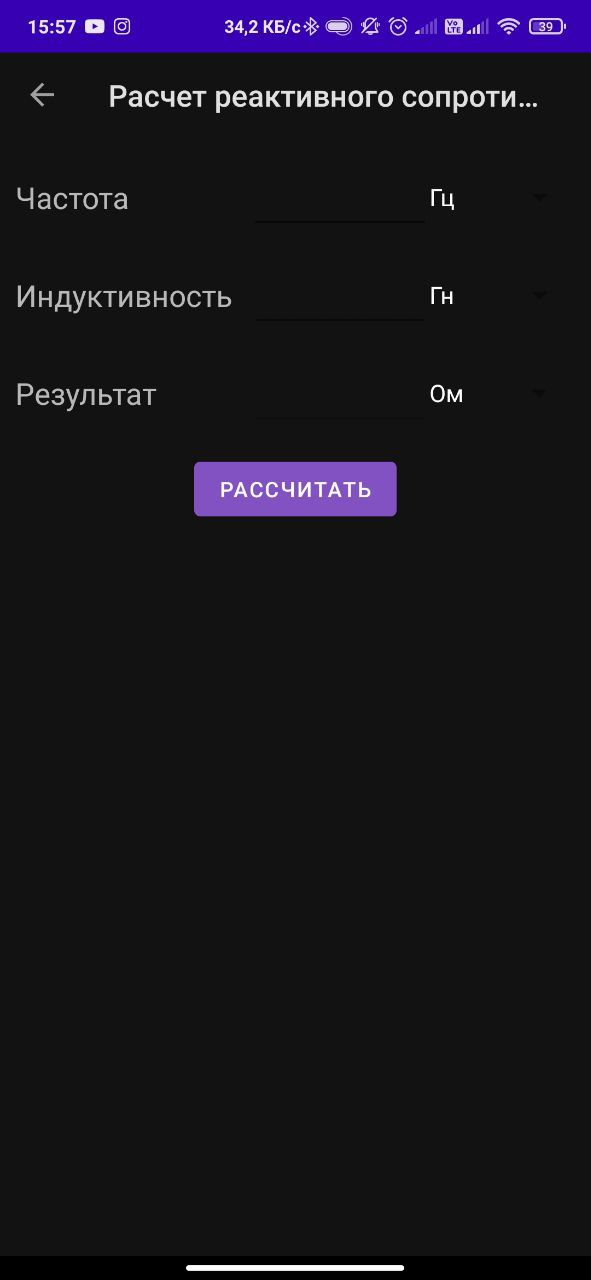


Рисунок 2 – Интерфейс экрана расчёта реактивного сопростивления катушки индуктивности

Для расчёта реактивного сопротивления конденсатора необходимо ввести частоту и ёмкость, указав размеры величины. Частота указывается так же, как и в расчёте реактивного сопротивления катушки индуктивности, а ёмкость в микрофарадах, нанофарадах и пикофарадах. Выбор величины указывается так же при помощи выпадающего меню. Интерфейс данного экрана изображен на рисунке 3.

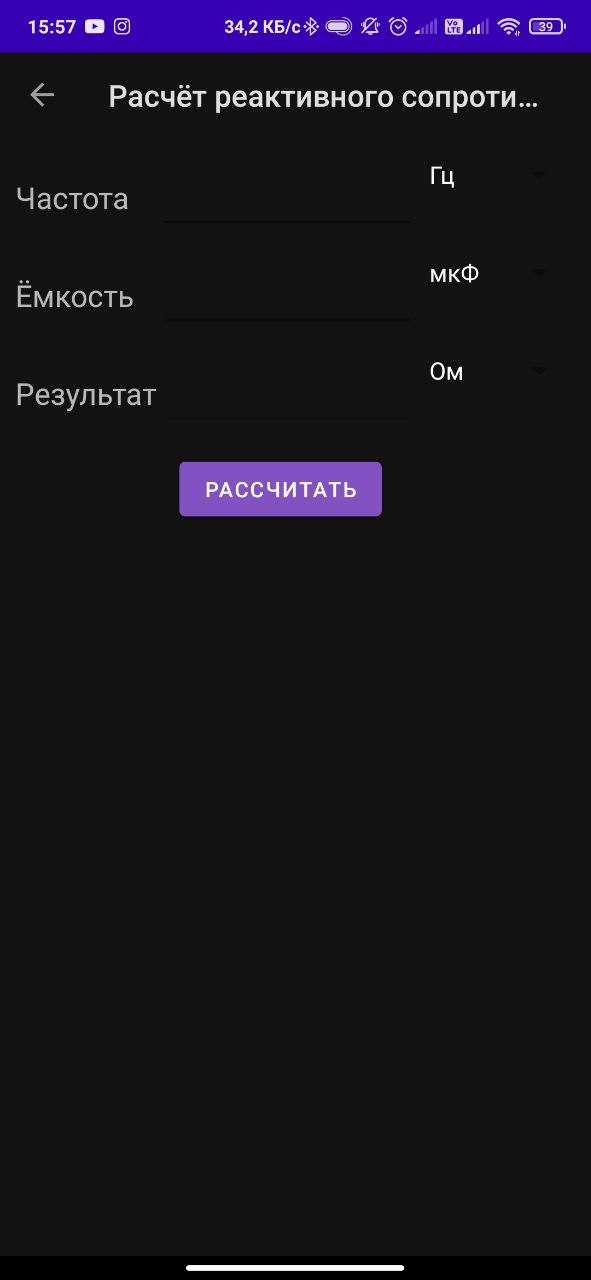


Рисунок 3 – Интерфейс экрана расчёта реактивного сопротивления конденсатора

После ввода значений нужно нажать кнопку «Рассчитать» и, после некоторой задержки, возникающей из-за отправки запроса на удалённый сервер и получения результата, в поле «Результат» отобразится сопротивление в омах. Для того, чтобы отобразить результат в величине отличной от выбранной, необходимо в выпадающем меню выбрать омы, килоомы или мегаомы. Преобразование будет произведено мгновенно, так как результат расчёта уже получен и необходимо лишь произвести небольшие вычисления с полученным результатом.

Для возврата к основному экрану необходимо нажать на стрелку в левом верхнем углу.

3 Алгоритм работы программы

При запуске приложения открывается экран, изображённый на рисунке 1. На данном экране размещены две кнопки, которые позволяют перейти на экраны расчёта реактивного сопротивления катушки индуктивности и расчёта реактивного сопротивления конденсатора. После нажатия на кнопку с нужным расчётом открывается экран 2 или 3, изображенные на рисунке 2 и 3 соответственно.

Для того, чтобы выполнить расчёт необходимо ввести в поле частота значение частоты и выбрать единицы измерения, то же самое для ёмкости и для индуктивности, если вы находитесь на экране расчёта реактивного сопротивления катушки индуктивности.

На сервер передаются данные, состоящие из введённых значений и числа, соотнесенного с единицами измерения, например, 1 – Гц; 2 – КГц; 3 – МГц.

После обработки данных сервером, возвращается *JSON*-строка с результатом в омах, который помещается в поле «Результат» и далее значение можно перевести в килоомы или мегаомы.

Исходный код программы находится в приложении А.

Вывод по результатам лабораторной работы

В результате выполнения лабораторной работы было разработано мобильное приложение для операционной системы *Android* реализующее функционал электротехнического калькулятора. Была изучена работа с библиотекой *REST*-запросов *Retrofit* и библиотекой converter-gson для работы с *JSON*-строками на *Android*.

Приложение A

Исходный код программы на Android

Код основного экрана приложения

package com.example.pavlovrest;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;

import android.content.Intent;

import android.net.ConnectivityManager;

import android.net.NetworkInfo;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import java.util.List;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

import retrofit2.Retrofit;

import retrofit2.converter.gson.GsonConverterFactory;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Button trBtn = (Button)findViewById(R.id.transformatorBtn);

Button reactResistInd = (Button)findViewById(R.id.reactiveResistanceBtn);

Button reactResistCap = (Button)findViewById(R.id.reactResistanceCapBtn);

reactResistInd.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(MainActivity.this, ReactiveResistanceInductionActivity.class);

startActivity(intent);

}

});

reactResistCap.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(MainActivity.this, ReactiveResistanceCapacitor.class);

startActivity(intent);

}

});

}

protected boolean isOnline() {

String cs = Context.CONNECTIVITY\_SERVICE;

ConnectivityManager cm = (ConnectivityManager)

getSystemService(cs);

if (cm.getActiveNetworkInfo() == null) {

return false;

} else {

return true;

}

};

}

Код экрана расчёта реактивного сопротивления катушки индуктивности

package com.example.pavlovrest;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Spinner;

import java.math.BigDecimal;

import java.text.DecimalFormat;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class ReactiveResistanceInductionActivity extends AppCompatActivity {

int check = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_reactive\_resistance\_induction);

final Button calculateResistanceBtn = (Button)findViewById(R.id.calculateResistanceBtn);

final EditText frequencyText = (EditText)findViewById(R.id.frequencyNumber);

final EditText inductionText = (EditText)findViewById(R.id.inductionsNumber);

final EditText resultField = (EditText)findViewById(R.id.resultField);

final Spinner freqSpinner = (Spinner)findViewById(R.id.freqSpinner);

final Spinner indSpinner = (Spinner)findViewById(R.id.indSpinner);

final Spinner resistSpinner = (Spinner)findViewById(R.id.resistSpinner);

final String[] lastResult = {""};

resultField.setEnabled(false);

calculateResistanceBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if(frequencyText.getText().toString() == null || inductionText.getText().toString() == null){

System.out.println("field is null");

}

else {

NetworkService.getInstance()

.getJsonApi()

.getReactResistInd(Integer.parseInt(frequencyText.getText().toString()), freqSpinner.getSelectedItemPosition(), Integer.parseInt(inductionText.getText().toString()),indSpinner.getSelectedItemPosition())

.enqueue(new Callback<Post>() {

@Override

public void onResponse(Call<Post> call, Response<Post> response) {

Post post = response.body();

System.out.println("success");

lastResult[0] = post.getValue();

//String val = new DecimalFormat("#").format(lastResult[0]);

BigDecimal bigDecimal = new BigDecimal(Double.toString(Double.parseDouble(lastResult[0]) / (Math.pow(1000, resistSpinner.getSelectedItemPosition()))));

resultField.setText(bigDecimal.toString());

}

@Override

public void onFailure(Call<Post> call, Throwable t) {

System.out.println("error");

t.printStackTrace();

}

});

}

}

});

resistSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

@Override

public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {

if(check++ > 0 && !lastResult[0].isEmpty()) {

String formattedDouble = new DecimalFormat("#0.0000").format(Double.parseDouble(lastResult[0]) / (Math.pow(1000, i))).replace(",",".");

BigDecimal bigDecimal = new BigDecimal(formattedDouble);

resultField.setText(bigDecimal.toString());

}

}

@Override

public void onNothingSelected(AdapterView<?> adapterView) {

}

});

}

}

Код экрана расчёта реактивного сопротивления конденсатора

package com.example.pavlovrest;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Spinner;

import java.math.BigDecimal;

import java.text.DecimalFormat;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class ReactiveResistanceCapacitor extends AppCompatActivity {

int check = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_reactive\_resistance\_capacitor);

final Button calculateResistanceCapBtn = (Button)findViewById(R.id.calculateResistanceCapBtn);

final EditText frequencyCapText = (EditText)findViewById(R.id.frequencyCapNumber);

final EditText capacitorText = (EditText)findViewById(R.id.CapNumber);

final EditText resultCap = (EditText)findViewById(R.id.resultCap);

final Spinner freqCapSpinner = (Spinner)findViewById(R.id.freqCapSpinner);

final Spinner capSpinner = (Spinner)findViewById(R.id.capSpinner);

final Spinner resistCapSpinner = (Spinner)findViewById(R.id.resistCapSpinner);

final String[] lastResult = {""};

resultCap.setEnabled(false);

calculateResistanceCapBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if(frequencyCapText.getText().toString() == null || capacitorText.getText().toString() == null){

System.out.println("field is null");

}

else {

NetworkService.getInstance()

.getJsonApi()

.getReactResistCap(Integer.parseInt(frequencyCapText.getText().toString()), freqCapSpinner.getSelectedItemPosition(), Integer.parseInt(capacitorText.getText().toString()),capSpinner.getSelectedItemPosition())

.enqueue(new Callback<Post>() {

@Override

public void onResponse(Call<Post> call, Response<Post> response) {

Post post = response.body();

System.out.println("success");

lastResult[0] = post.getValue();

//String val = new DecimalFormat("#").format(lastResult[0]);

BigDecimal bigDecimal = new BigDecimal(Double.toString(Double.parseDouble(lastResult[0]) / (Math.pow(1000, resistCapSpinner.getSelectedItemPosition()))));

resultCap.setText(bigDecimal.toString());

}

@Override

public void onFailure(Call<Post> call, Throwable t) {

System.out.println("error");

t.printStackTrace();

}

});

}

}

});

resistCapSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

@Override

public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {

if(check++ > 0 && !lastResult[0].isEmpty()) {

String formattedDouble = new DecimalFormat("#0.000000").format(Double.parseDouble(lastResult[0]) / (Math.pow(1000, i))).replace(",",".");

BigDecimal bigDecimal = new BigDecimal(formattedDouble);

resultCap.setText(bigDecimal.toString());

}

}

@Override

public void onNothingSelected(AdapterView<?> adapterView) {

}

});

}

}

Код класса *POST*-запроса

package com.example.pavlovrest;

import com.google.gson.annotations.Expose;

import com.google.gson.annotations.SerializedName;

public class Post {

@SerializedName("value")

@Expose

private String value;

public String getValue() {

return value;

}

public void setValue(String value) {

this.value = value;

}

}

Код класса сетевого подключения

package com.example.pavlovrest;

import retrofit2.Retrofit;

import retrofit2.converter.gson.GsonConverterFactory;

public class NetworkService {

private static NetworkService mInstance;

// private static final String BASE\_URL = "https://e58e-128-75-140-209.ngrok.io";

private static final String BASE\_URL = "https://fb59-128-75-140-209.ngrok.io";

private Retrofit mRetrofit;

private NetworkService() {

mRetrofit = new Retrofit.Builder()

.baseUrl(BASE\_URL)

.addConverterFactory(GsonConverterFactory.create())

.build();

}

public static NetworkService getInstance() {

if (mInstance == null) {

mInstance = new NetworkService();

}

return mInstance;

}

public JsonPlaceHolderApi getJsonApi(){

return mRetrofit.create(JsonPlaceHolderApi.class);

}

}

Код интерфейса формирования и отправки *GET*-запроса

package com.example.pavlovrest;

import java.util.List;

import retrofit2.Call;

import retrofit2.http.GET;

import retrofit2.http.Path;

import retrofit2.http.Query;

public interface JsonPlaceHolderApi {

@GET("/reactResistInd")

Call<Post> getReactResistInd(@Query("freq") int freq, @Query("freqMul") int fMul, @Query("ind") int ind, @Query("indMul") int iMul);

@GET("/reactResistCap")

Call<Post> getReactResistCap(@Query("freq") int freq, @Query("freqMul") int fMul, @Query("cap") int cap, @Query("capMul") int iMul);

}