Комитет по образованию Правительства Санкт-Петербурга

**САНКТ-ПЕТЕРБУРГСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОНЫХ ТЕХНОЛОГИЙ**

**Отчет по практической работе № 4**

**МДК 01.03 Разработка мобильных приложений**

**Тема: Разработка интерактивного приложения карт**

Выполнил

студент Группы 493

сидоров антон дмитриевич

Проверила Фомин А. В.

Оценка \_\_\_\_\_\_\_\_

Санкт-Петербург 2022

**Содержание**

[1. Цели работы 3](#_Toc125019376)

[2. Диаграмма базы данных 3](#_Toc125019377)

[3. Макеты экранов приложения и их описание 5](#_Toc125019378)

[4. Программный код 6](#_Toc125019379)

[5. Демонстрация работы приложения 7](#_Toc125019380)

[6. Вывод 9](#_Toc125019381)

[Приложение 10](#_Toc125019382)

# Цели работы

Разработать интерактивное приложение карт с использованием API

# Диаграмма базы данных

Ниже представлены макеты таблиц с их назначение и описанием данных.

Таблица для хранения URL-ссылки на API представлена на рисунке 1.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 1 – Диаграмма базы данных

Таблица для хранения параметров масштабирования карты представлена на рисунке 2.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 2 – Диаграмма базы данных

Таблица для хранения параметров кэша для хранения тайлов карты представлена на рисунке 3.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 3 – Диаграмма базы данных

Таблица для хранения параметров тайлов карты представлена на рисунке 4

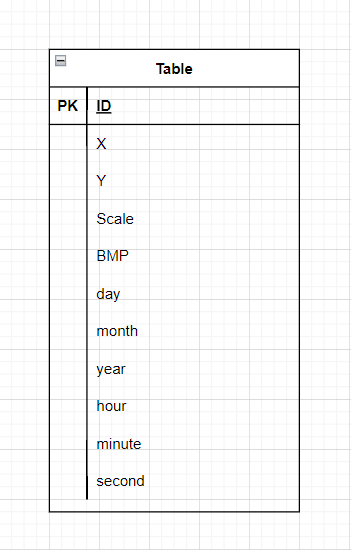


Рисунок 4 – Диаграмма базы данных

Описание таблицы представлено на рисунке 5.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 5 – Диаграмма базы данных

# Макеты экранов приложения и их описание

* 1. Примечания по макетам

Экраны приложений могут отличаться от своих макетов.

* 1. Макеты окон

Макеты окон для работы с пользователем показаны на рисунке 6.

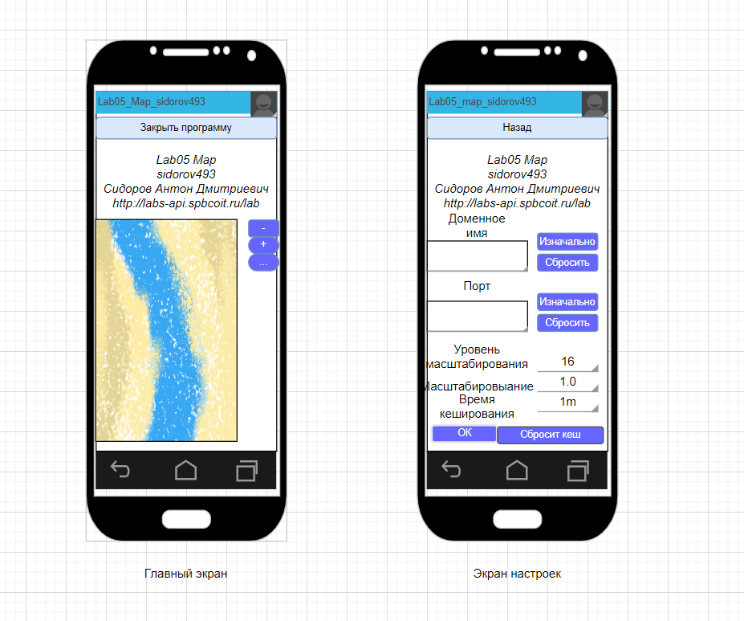


Рисунок 6 – Макеты окон

Макеты, которые связаны с API показаны на рисунке 4.

Описание макетов:

На главном экране находятся

* *Графическое изображение* – пользовательский компонент унаследованный от класса SurfaceView
* Кнопки («+» и «-») – соответственно, увеличение и уменьшение маштаба
* *Кнопка для перехода к настройкам*

На экране настроек находятся:

* *2 Текстовых поля* – для ввода адреса конечной точки API (одно для имени узла, другое для порта и пути).
* *Надписи* (Для каждого текстового поля) – его назначение
* *2 кнопки* (Для каждого текстового поля) – возврат к значению до открытия экрана настроек и сброс (очистка)
* *3 текстовых поля ввода с надписями* – Для ввода уровня маштабирования, значения маштабирования (в соотношении со сторонами компонента) и времени кэширования (хранения в базе данных)
* *2 Кнопки* – сохранение и сброс кэша

# Программный код

Программный код условно можно разбить на 6 частей:

1. База данных SQLite
2. Код работы окон приложения
3. Код компонентов интерфейса
4. Вспомогательный класс
5. Код тайла
6. Код работы с API

Далее всё буде рассмотрено подробнее.

* 1. База данных SQLite

Код работы с базой данных для хранения тайлов представлен в приложении 1.

Код работы с базой данных для хранения URL-ссылок представлен в приложении 2.

Код работы с базой данных для хранения данных маштабирования представлен в приложении 3.

* 1. Код работы окон приложения

Программный код начального окна представлен в приложении 4.

Программный код окна настроек приложения представлен в приложении 5.

* 1. Класс, визуальный компонент

Это – компонент для вывода карты, код которого представлен на рисунке 6.

* 1. Вспомогательный класс

Вспомогательный класс, представлен в приложении 7.

* 1. Код тайла

Этот код представлен в приложении 8.

* 1. Код работы с API

Этот код представлен в приложении 9.

# Демонстрация работы приложения

Начальный экран приложения представлен на рисунке 7.

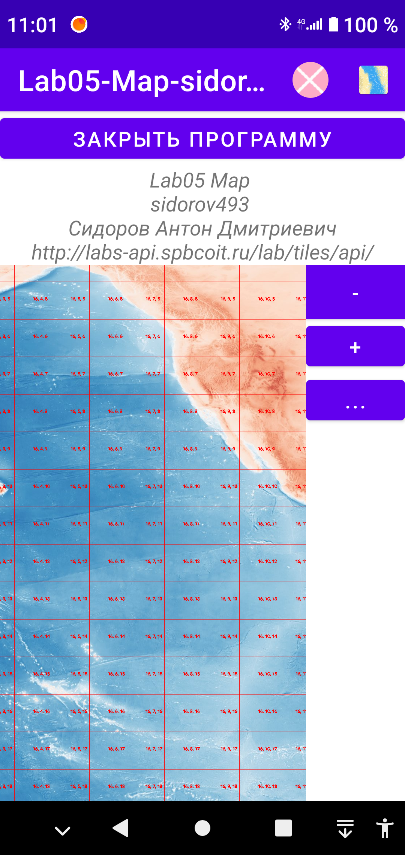


Рисунок 7 – Работа приложения

Окно настроек представлено на рисунке 8.

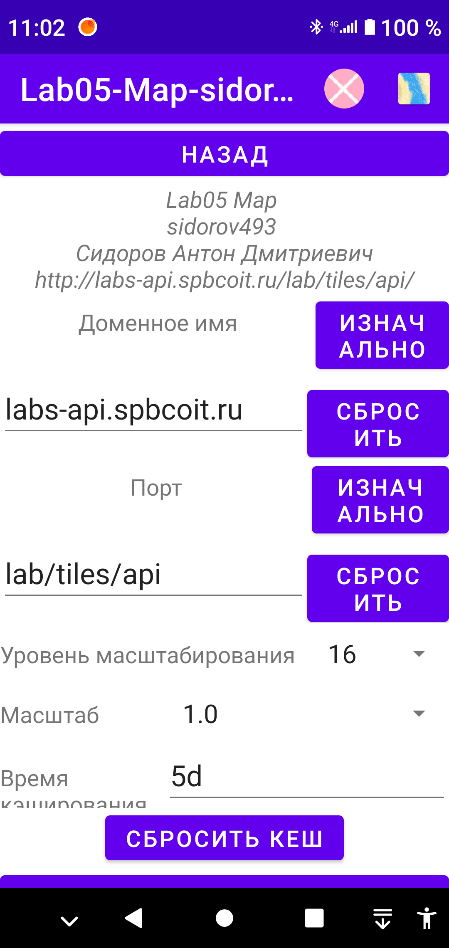


Рисунок 8 – Работа приложения

Продолжение окна настроек представлен на рисунке 9.



Рисунок 9 – Работа приложения

# Вывод

Освоена работа с картой в java.

# Приложение

**Приложение 1. Программный код SQLite для хранения тайлов**

package com.example.lab05\_map\_sidorov493;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import androidx.annotation.Nullable;

import java.io.ByteArrayOutputStream;

public class TilesDB extends SQLiteOpenHelper {

public TilesDB(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {

super(context, name, factory, version);

}

public static TilesDB GetDB(Context context)

{

return new TilesDB(context);

}

public TilesDB(Context context)

{

this(context, "Tiles.db", null, 1);

}

@Override

public void onCreate(SQLiteDatabase sqLiteDatabase) {

SQLiteDatabase db = sqLiteDatabase;

String cash = "CREATE TABLE Cash (\n" +

" id INTEGER NOT NULL,\n" +

" time varchar(255), \n"+

" PRIMARY KEY (id)\n" +

")";

db.execSQL(cash);

UpdateTimeCash(db);

String tileTable = "CREATE TABLE Tile (\n" +

" id INTEGER NOT NULL,\n" +

" x INTEGER NOT NULL,\n" +

" y INTEGER NOT NULL,\n" +

" scale INTEGER NOT NULL,\n" +

" bmp BLOB,\n" +

" day int NOT NULL,\n"+

" month int NOT NULL,\n"+

" year int NOT NULL,\n"+

" hour int NOT NULL,\n"+

" minute int NOT NULL,\n"+

" second int NOT NULL,\n"+

" PRIMARY KEY (id)\n" +

")";

db.execSQL(tileTable);

// UpdateUrl(db);

}

public Tile GetTile(Tile tile) throws Exception {

return GetTile(tile.x, tile.y, tile.scale);

}

public void ClearCash()

{

SQLiteDatabase db = getWritableDatabase();

String clearCash = "Delete from Tile";

db.execSQL(clearCash);

}

public Tile GetTile(int x, int y, int scale) throws Exception

{

try {

SQLiteDatabase db = getReadableDatabase();

Tile tile = new Tile(x, y, scale);

String TileInfo = "Select bmp, id, day, month, year, hour, minute, second From Tile where x=" + x + " and y=" + y + " and scale=" + scale;

Cursor cur = db.rawQuery(TileInfo, null);

if (cur.moveToFirst() == true) {

tile.bmp = getImage(cur.getBlob(0));

tile.ID = cur.getInt(1);

int day = cur.getInt(2);

int month = cur.getInt(3);

int year = cur.getInt(4);

int hour = cur.getInt(5);

int minute = cur.getInt(6);

int second = cur.getInt(7);

tile.SetDateTime(hour, minute, second, day, month, year);

cur.close();

}

else

{

throw new Exception();

}

if(!tile.YesTime())

{

db = getWritableDatabase();

String delete = "Delete From Tile where id="+tile.ID;

db.execSQL(delete);

tile = GetTile(tile);

}

return tile;

}

catch (Exception e)

{

throw e;

}

}

public Tile insert(Tile tile) {

try {

return GetTile(tile);

} catch (Exception e) {

e.printStackTrace();

try {

ContentValues content = createContentValues(tile.bmp);

content.put("id", CountTile() + 1);

content.put("x", tile.x);

content.put("y", tile.y);

content.put("scale", tile.scale);

content.put("day", tile.Days());

content.put("month", tile.Monthes());

content.put("year", tile.Years());

content.put("hour", tile.Hours());

content.put("minute", tile.Minutes());

content.put("second", tile.Seconds());

SQLiteDatabase db = getWritableDatabase();

db.insert("Tile", null, content);

}

catch (Exception ex)

{

ex.printStackTrace();

}

return tile;

}

}

public void UpdateTimeCash()

{

UpdateTimeCash(getWritableDatabase());

}

public void UpdateTimeCash(SQLiteDatabase db)

{

try {

if(Count() < 1)

throw new Exception();

ContentValues context = new ContentValues();

context.put("time", MapHelper.GetTime());

db.update("Cash", context, null, null);

}

catch (Exception ex)

{

ContentValues context = new ContentValues();

context.put("time", MapHelper.GetTime());

context.put("id", 1);

db.insert("Cash", null, context);

}

GetTimeCash();

}

public void GetTimeCash()

{

try {

try {

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT time FROM Cash;";

Cursor cur = db.rawQuery(sql, null);

if (cur.moveToFirst() == true) {

MapHelper.SetTime( cur.getString(0));

cur.close();

}

} catch (Exception ex) {

UpdateTimeCash();

}

}

catch(Exception ex)

{

}

}

public int Count()

{

try {

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT Count(\*) FROM Cash;";

Cursor cur = db.rawQuery(sql, null);

int count = 0;

if (cur.moveToFirst() == true) {

count = cur.getInt(0);

}

cur.close();

return count;

} catch (Exception ex) {

return 0;

}

}

public int CountTile()

{

try {

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT Count(\*) FROM Tile;";

Cursor cur = db.rawQuery(sql, null);

int count = 0;

if (cur.moveToFirst() == true) {

count = cur.getInt(0);

}

cur.close();

return count + Count();

} catch (Exception ex) {

return 0;

}

}

@Override

public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

}

// convert from bitmap to byte array

public static byte[] getBytes(Bitmap bitmap) {

ByteArrayOutputStream stream = new ByteArrayOutputStream();

bitmap.compress(Bitmap.CompressFormat.PNG, 0, stream);

return stream.toByteArray();

}

// convert from byte array to bitmap

public static Bitmap getImage(byte[] image) {

return BitmapFactory.decodeByteArray(image, 0, image.length);

}

public static ContentValues createContentValues(String column, byte[] image) {

ContentValues cv = new ContentValues();

cv.put(column, image);

return cv;

}

public static ContentValues createContentValues(byte[] image)

{

return createContentValues("bmp", image);

}

public static ContentValues createContentValues(String column, Bitmap image) {

return createContentValues(column, getBytes(image));

}

public static ContentValues createContentValues(Bitmap image) {

return createContentValues(getBytes(image));

}

}

**Приложение 2. Программный код SQLite для URL**

package com.example.lab05\_map\_sidorov493;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

public class UrlStorege extends SQLiteOpenHelper {

public UrlStorege(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {

super(context, name, factory, version);

}

public UrlStorege(Context context)

{

this(context, "Url.db", null, 1);

}

@Override

public void onCreate(SQLiteDatabase sqLiteDatabase) {

SQLiteDatabase db = sqLiteDatabase;

String urlTable = "CREATE TABLE urlTable (\n" +

" id INTEGER NOT NULL,\n" +

" domain VARCHAR(255) NOT NULL,\n" +

" port VARCHAR(255) NOT NULL,\n" +

" PRIMARY KEY (id)\n" +

")";

db.execSQL(urlTable);

UpdateUrl(db);

}

public void UpdateUrl()

{

UpdateUrl(getWritableDatabase());

}

public int Count()

{

try {

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT Count(\*) FROM urlTable;";

Cursor cur = db.rawQuery(sql, null);

int count = 0;

if (cur.moveToFirst() == true) {

count = cur.getInt(0);

}

cur.close();

return count;

} catch (Exception ex) {

return 0;

}

}

public void UpdateUrl(SQLiteDatabase db)

{

try {

if(Count() < 1)

throw new Exception();

ContentValues context = new ContentValues();

context.put("domain", MapHelper.DomainUrl);

context.put("port", MapHelper.PortUrl);

db.update("urlTable", context, null, null);

}

catch (Exception ex)

{

ContentValues context = new ContentValues();

context.put("domain", MapHelper.DomainUrl);

context.put("port", MapHelper.PortUrl);

context.put("id", 1);

db.insert("urlTable", null, context);

}

GetUrl();

}

public void GetUrl()

{

try {

try {

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT domain, port FROM urlTable;";

Cursor cur = db.rawQuery(sql, null);

if (cur.moveToFirst() == true) {

MapHelper.DomainUrl = cur.getString(0);

MapHelper.PortUrl = cur.getString(1);

cur.close();

}

} catch (Exception ex) {

UpdateUrl();

}

}

catch(Exception ex)

{

}

}

public static UrlStorege GetDB(Context context)

{

return new UrlStorege(context);

}

@Override

public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

}

}

**Приложение 3. Программный код SQLite для хранения данных маштабирования**

package com.example.lab05\_map\_sidorov493;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

public class ScaleDB extends SQLiteOpenHelper {

public ScaleDB(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {

super(context, name, factory, version);

}

public ScaleDB(Context context)

{

this(context, "ScaleMap.db", null, 1);

}

public static ScaleDB GetDB(Context context)

{

return new ScaleDB(context);

}

@Override

public void onCreate(SQLiteDatabase sqLiteDatabase) {

SQLiteDatabase db = sqLiteDatabase;

String urlTable = "CREATE TABLE Scale (\n" +

" id INTEGER NOT NULL,\n" +

" level Integer NOT NULL,\n" +

" value Integer NOT NULL,\n" +

" PRIMARY KEY (id)\n" +

")";

db.execSQL(urlTable);

UpdateScale(db);

}

public int Count()

{

try {

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT Count(\*) FROM Scale;";

Cursor cur = db.rawQuery(sql, null);

int count = 0;

if (cur.moveToFirst() == true) {

count = cur.getInt(0);

}

cur.close();

return count;

} catch (Exception ex) {

return 0;

}

}

public void UpdateScale(SQLiteDatabase db)

{

UpdateScale(db, 0, MapHelper.DefaultScale());

}

public void UpdateScale(int level, int value)

{

UpdateScale(getReadableDatabase(), level, value);

}

public void UpdateScale(MapView map)

{

UpdateScale(map.current\_level\_index, map.ScaleValue\_Index);

}

public void GetScale(MapView map)

{

try {

try {

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT level, value FROM Scale;";

Cursor cur = db.rawQuery(sql, null);

if (cur.moveToFirst() == true) {

map.SetScale(cur.getInt(0), cur.getInt(1));

cur.close();

}

} catch (Exception ex) {

//UpdateUrl();

}

}

catch(Exception ex)

{

}

}

public void UpdateScale(SQLiteDatabase db, int level, float scale)

{

try {

if(Count() < 1)

throw new Exception();

ContentValues context = new ContentValues();

context.put("level", level);

context.put("value", scale);

db.update("Scale", context, null, null);

}

catch (Exception ex)

{

ContentValues context = new ContentValues();

context.put("level", level);

context.put("value", scale);

context.put("id", 1);

db.insert("Scale", null, context);

}

//GetUrl();

}

@Override

public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

}

}

Приложение 4. Программный код начального окна

package com.example.lab05\_map\_sidorov493;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

Button exit;

TextView info;

MapView mv;

Boolean exiting;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

MapHelper.exit = false;

exit = findViewById(R.id.CloseProgram);

info = findViewById(R.id.AppInfo);

info.setText(MapHelper.AppInfoWidthURL(this));

exit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Exit\_Click(view);

}

});

MapHelper.ChangeTime(this);

mv = findViewById(R.id.map);

mv.SetScale();

// mv.invalidate();

}

public void ZoomOut\_Click(View v)

{

if(mv.current\_level\_index == 0) return;

mv.offset\_x += mv.width / 2.0f;

mv.offset\_y += mv.height / 2.0f;

mv.offset\_x /= 2.0f;

mv.offset\_y /= 2.0f;

mv.current\_level\_index--;

mv.invalidate();

}

public void ZoomIn\_Click(View v)

{

if(mv.current\_level\_index == mv.levels.length - 1) return;

mv.offset\_x \*= 2.0f;

mv.offset\_y \*= 2.0f;

mv.offset\_x -= mv.width / 2.0f;

mv.offset\_y -= mv.height / 2.0f;

mv.current\_level\_index++;

mv.invalidate();

}

@Override

public boolean onOptionsItemSelected(@NonNull MenuItem item) {

int id = item.getItemId();

switch (id)

{

case R.id.exit: {

View v = exit;

Exit\_Click(v);

}

break;

}

return super.onOptionsItemSelected(item);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main, menu);

return super.onCreateOptionsMenu(menu);

}

public void Exit\_Click(View v)

{

AlertDialog.Builder bld = new AlertDialog.Builder(this);

bld.setPositiveButton("Нет",

new DialogInterface.OnClickListener()

{

@Override

public void onClick(DialogInterface dialog, int which) {

dialog.cancel(); // Закрываем диалоговое окно

MapHelper.exit = false;

}

});

bld.setNegativeButton("Да", new DialogInterface.OnClickListener(){

@Override

public void onClick(DialogInterface dialog, int which) {

finish(); // Закрываем Activity

}

});

AlertDialog dlg = bld.create();

dlg.setTitle("Выход из приложения");

dlg.setMessage("Уважаемый пользователь \n" +

"Вы действительно хотите выйти из программы \n" +

"Вы, также, можете запустить программу снова \n" +

"С уважением и любовью, Создатель программы, Сидоров Антон Дмитриевич");

dlg.show();

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

super.onActivityResult(requestCode, resultCode, data);

MapHelper.ChangeTimeInDB(this);

UrlStorege.GetDB(this).UpdateUrl();

info.setText(MapHelper.AppInfoWidthURL(this));

mv.invalidate();

MapHelper.exit = false;

}

public void Settings\_Click(View v)

{

MapHelper.mv = mv;

Intent i = new Intent(this, Settings.class);

startActivityForResult(i, 200);

}

}

Приложение 5. Программный код настроек приложения

package com.example.lab05\_map\_sidorov493;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Spinner;

import android.widget.TextView;

import java.util.ArrayList;

public class Settings extends AppCompatActivity {

Button exit;

EditText domain, port;

TextView info;

MapView mv;

Spinner scales, levels;

EditText TextChash;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_settings);

exit = findViewById(R.id.buttonExit);

domain = findViewById(R.id.DomainNameText);

port = findViewById(R.id.PortText);

domain.setText(MapHelper.DomainUrl);

port.setText(MapHelper.PortUrl);

info = findViewById(R.id.AppInfo1);

info.setText(MapHelper.AppInfoWidthURL(this));

mv = MapHelper.mv;

levels = findViewById(R.id.spinnerLevels);

ArrayList<String> levaelTexts = new ArrayList<>();

int[] levelValues = mv.levels;

for(int i = 0; i < levelValues.length; i++)

{

levaelTexts.add(String.valueOf(levelValues[i]));

}

ArrayAdapter<String> levelAdapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1, levaelTexts);

levels.setAdapter(levelAdapter);

levels.setSelection(mv.current\_level\_index);

scales = findViewById(R.id.scales);

float []scaleValues = mv.scales;

ArrayList<String> scaleTexts = new ArrayList<>();

for(int i = 0; i < scaleValues.length; i++)

{

scaleTexts.add(String.valueOf(scaleValues[i]));

}

ArrayAdapter<String> scaleAdapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1, scaleTexts);

scales.setAdapter(scaleAdapter);

scales.setSelection(mv.ScaleValue\_Index);

TextChash = findViewById(R.id.TextCash);

TextChash.setText(MapHelper.GetTime());

}

public void DomainClear\_Click(View v)

{

domain.setText("");

}

public void DomainStart\_Click(View v)

{

domain.setText(MapHelper.DomainUrl);

}

public void PortStart\_Click(View v)

{

port.setText(MapHelper.PortUrl);

}

public void PortClear\_Click(View v)

{

port.setText("");

}

@Override

public boolean onOptionsItemSelected(@NonNull MenuItem item) {

int id = item.getItemId();

switch (id)

{

case R.id.exit: {

View v = exit;

Exit1\_Click(v);

}

break;

}

return super.onOptionsItemSelected(item);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main, menu);

return super.onCreateOptionsMenu(menu);

}

public void Exit1\_Click(View v)

{

MapHelper.exit = true;

Exit\_Click(v);

}

public void Exit\_Click(View v)

{

MapHelper.ChangeTime(TextChash.getText().toString());

mv.current\_level\_index = levels.getSelectedItemPosition();

mv.ScaleValue\_Index = scales.getSelectedItemPosition();

//GrapsParams.NowGraph = graphs.GetGraph();

MapHelper.DomainUrl = domain.getText().toString();

MapHelper.PortUrl = port.getText().toString();

Intent i = getIntent();

finish();

}

public void ClearCash\_Click(View v)

{

TilesDB.GetDB(this).ClearCash();

}

}

Приложение 6. Компонент для вывода карты

package com.example.lab05\_map\_sidorov493;

import android.app.Activity;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.util.AttributeSet;

import android.view.MotionEvent;

import android.view.SurfaceView;

import java.util.ArrayList;

public class MapView extends SurfaceView {

ArrayList<Tile> tiles = new ArrayList<>();

Activity ctx;

public Tile GetTile(int x, int y, int scale) {

for (int i = 0; i < tiles.size(); i++) {

Tile t = tiles.get(i);

if (t.x == x && t.y == y && t.scale == scale) {

//TilesDB.GetDB(ctx).insert(t);

if(t.bmp == null && MapHelper.api)

tiles.set(i, new Tile(x, y, scale, ctx));

t = tiles.get(i);

return t;

}

}

Tile nt = new Tile(x, y, scale, ctx);

if(nt.bmp != null) {

tiles.add(nt);

if (MapHelper.api) {

TilesDB.GetDB(ctx).insert(nt);

}

}

return nt;

}

float last\_x, last\_y;

public int current\_level\_index = 0;

public int[] levels = new int[]{16, 8, 4, 2, 1};

int[] x\_tiles = new int[]{54, 108, 216, 432, 864};

int[] y\_tiles = new int[]{27, 54, 108, 216, 432};

int tile\_width = 100;

int tile\_height = 100;

float offset\_x = 0.0f;

float offset\_y = 0.0f;

Paint p;

int width, height;

public MapView(Context context) {

super(context);

SetMap(context);

}

public MapView(Activity context) {

super(context);

SetMap(context);

}

public MapView(Context context, AttributeSet attrs) {

super(context, attrs);

SetMap(context);

}

public MapView(Activity context, AttributeSet attrs) {

super(context, attrs);

SetMap(context);

}

public void SetScale(int level, int value)

{

current\_level\_index = level;

ScaleValue\_Index = value;

invalidate();

}

public void SetScale(ScaleDB db)

{

db.GetScale(this);

}

public void SetScale()

{

SetScale(ScaleDB.GetDB(ctx));

}

float[] scales;

public void SetMap(Context ctx)

{

try {

this.ctx = (Activity) ctx;

}

catch (Exception ex)

{

}

p = new Paint();

p.setStyle(Paint.Style.STROKE);

p.setColor(Color.RED);

setWillNotDraw(true);

scales = MapHelper.ScaleValues();

//invalidate();

}

Boolean draw = false;

@Override

public boolean onTouchEvent(MotionEvent event) {

int act = event.getAction();

switch (act)

{

case MotionEvent.ACTION\_DOWN:

{

last\_x = event.getX();

last\_y = event.getY();

draw = true;

return true;

}

case MotionEvent.ACTION\_MOVE:

{

draw = false;

float x = event.getX();

float y = event.getY();

float dx = x - last\_x;

float dy = y - last\_y;

offset\_x += dx;

offset\_y += dy;

invalidate();

last\_x = x;

last\_y = y;

return true;

}

case MotionEvent.ACTION\_UP:

{

if(draw)

{

draw = false;

invalidate();

}

return true;

}

}

return false;

}

boolean rect\_interects\_rect(

int ax0, int ay0, int ax1, int ay1,

int bx0, int by0, int bx1, int by1)

{

if(!rect\_interects\_rect\_X(ax0, ax1, bx0, bx1)) return false;

if(!rect\_interects\_rect\_Y(ay0, ay1, by0, by1)) return false;

return true;

}

boolean rect\_interects\_rect\_X(

int ax0, int ax1,

int bx0, int bx1)

{

if(ax1 < bx0) return false;

if(ax0 > bx1) return false;

return true;

}

boolean rect\_interects\_rect\_Y(

int ay0, int ay1,

int by0, int by1)

{

if(ay1 < by0) return false;

if(ay0 > by1) return false;

return true;

}

public int ScaleValue\_Index = -1;

public void ErrorDraw()

{

}

float aMinesToB(float a, int b, float mines)

{

int a1 = (int)a;

int mines1 = (int)mines;

while (a1 > b + mines1)

{

a -= mines;

a1 = (int)a;

}

return a1;

}

float aPlusToB(float a, int b, float plus)

{

int a1 = (int)a;

int plus1 = (int)plus;

while (-a1 > b - plus1)

{

a += plus1;

a1 = (int)a;

}

return a1;

}

@Override

public void invalidate() {

setWillNotDraw(false);

super.invalidate();

}

@Override

protected void onDraw(Canvas canvas) {

try {

ApiHelper helper = new ApiHelper(ctx);

helper.SendNoParamsStop(MapHelper.GetUrl(ctx) + "raster", "");

MapHelper.api = helper.Ready;

width = canvas.getWidth();

height = canvas.getHeight();

float canvas\_scale;

try {

canvas\_scale = scales[ScaleValue\_Index];

} catch (Exception ex) {

canvas\_scale = 0.0f;

}

float canvas\_width = canvas\_scale, canvas\_height = canvas\_scale;

try {

if (canvas\_height == 0.0f || canvas\_width == 0.0f)

throw new Exception();

canvas.scale(canvas\_width, canvas\_height);

} catch (Exception ex) {

ScaleValue\_Index = MapHelper.DefaultScale();

canvas\_scale = scales[ScaleValue\_Index];

canvas\_width = canvas\_scale;

canvas\_height = canvas\_scale;

canvas.scale(canvas\_width, canvas\_height);

} finally {

}

width = canvas.getWidth();

height = canvas.getHeight();

canvas.drawColor(Color.WHITE);

/\*if(!helper.Ready)

{

ErrorDraw();

return;

}\*/

int screan\_x0 = 0;

int screan\_y0 = 0;

int screan\_x1 = width - 1;

int screan\_y1 = height - 1;

int scale = levels[current\_level\_index];

int w = x\_tiles[current\_level\_index];

int h = y\_tiles[current\_level\_index];

int xView = (int)offset\_x, yView = (int)offset\_y;

offset\_x = aMinesToB(offset\_x, screan\_x0, 400.0f);

offset\_y = aMinesToB(offset\_y, screan\_y0, 400.0f);

offset\_x = aPlusToB(offset\_x, w\*tile\_width, 400.0f);

offset\_y = aPlusToB(offset\_y, h\*tile\_height, 400.0f);

for (int y = 0; y < h; y++) {

int y0 = y \* tile\_height + (int) offset\_y;

int y1 = y0 + tile\_height;

if (y1 < screan\_y0) {

y = (screan\_y0 - (int) offset\_y) / tile\_height;

y--;

continue;

}

if (y0 > screan\_y1)

break;

if (!rect\_interects\_rect\_Y(screan\_y0, screan\_y1, y0, y1))

continue;

for (int x = 0; x < w; x++) {

int x0 = x \* tile\_width + (int) offset\_x;

int x1 = x0 + tile\_width;

if (x1 < screan\_x0) {

x = (screan\_x0 - (int) offset\_x) / tile\_width;

x--;

continue;

}

if (x0 > screan\_x1)

break;

if (!rect\_interects\_rect(screan\_x0, screan\_y0, screan\_x1, screan\_y1, x0, y0, x1, y1))

continue;

Tile t = GetTile(x, y, scale);

if (t.bmp != null)

canvas.drawBitmap(t.bmp, x0, y0, p);

canvas.drawRect(x0, y0, x1, y1, p);

canvas.drawText(String.valueOf(scale) + ", " + String.valueOf(x) +

", " + String.valueOf(y), (x0 + x1) / 2, (y0 + y1) / 2, p);

}

}

}

catch(Exception ex)

{

}

ScaleDB.GetDB(ctx).UpdateScale(this);

}

}

Приложение 7. Вспомогательный класс

package com.example.lab05\_map\_sidorov493;

import android.app.Activity;

import android.content.Context;

import androidx.annotation.Nullable;

public class MapHelper {

public static String DomainUrl="labs-api.spbcoit.ru";

public static String PortUrl = "lab/tiles/api";

public static String GetUrl(Context context)

{

UrlStorege.GetDB(context).GetUrl();

return Url(context);

}

public static String Url(@Nullable Context context) {

String url = "";

DomainUrl = DomainUrl.replace('\\', '/');

PortUrl = PortUrl.replace('\\', '/');

int end = 0;

String end1 = "";

if(PortUrl != "" && !PortUrl.equals("") && PortUrl != null) {

try {

if(DomainUrl.contains("/") || DomainUrl.contains(":") || DomainUrl.contains("?"))

throw new Exception();

String[] parts = PortUrl.split("/");

Integer.parseInt(parts[0]);

url = "http://" + DomainUrl + ":" + PortUrl;

} catch (Exception ex) {

end = DomainUrl.length() - 1;

end1 = DomainUrl.substring(end, end + 1);

String start = PortUrl.substring(0, 1);

if (end1 != "/" && !end1.equals("/"))

if (start != "/" && !start.equals("/"))

url = "http://" + DomainUrl + "/" + PortUrl;

else

url = "http://" + DomainUrl + PortUrl;

else

url = "http://" + DomainUrl + PortUrl;

}

}

else

url = "http://"+DomainUrl;

end = url.length()-1;

end1 = url.substring(end, end+1);

if(end1 != "/" && !end1.equals("/"))

url+="/";

UrlStorege.GetDB(context).UpdateUrl();

return url;

}

public static String AppInfo()

{

return "Lab05 Map \n" +

"sidorov493 \n" +

"Сидоров Антон Дмитриевич";

}

public static String AppInfoWidthURL(Activity ctx)

{

return AppInfo() + "\n"+GetUrl(ctx);

}

//static int length = 60;

static int div = 2;

static float length1 = 40f;

static int length()

{

return (int)(length1\*div);

}

public static int DefaultScale()

{

return ((int)(length()/length1)\*10)-1;

}

public static float[] ScaleValues() {

int length = length();

float[] values = new float[length];

float startValue = length1 / (float)length;

startValue /= 10f;

for (int i = 0; i < values.length; i++) {

values[i] = startValue \* (i + 1);

}

return values;

}

public static Boolean exit;

public static MapView mv;

public static String TimeCash = "1m";

public static int GetValueTime()

{

try {

String value = TimeCash.substring(0, TimeCash.length() - 1);

int number = Integer.parseInt(value);

if (GetMeasurementTime().equals("d") && number > 12) {

TimeCash = "12d";

return GetValueTime();

} else if (GetMeasurementTime().equals("h") && number >= 24) {

number /= 24;

TimeCash = number + "d";

return GetValueTime();

} else if (GetMeasurementTime().equals("m") && number >= 60) {

number /= 60;

TimeCash = number + "h";

return GetValueTime();

} else if (GetMeasurementTime().equals("s") && number >= 60) {

number /= 60;

TimeCash = number + "m";

return GetValueTime();

}

if(number < 1)

{

return 1;

}

return number;

}

catch (Exception ex)

{

return 1;

}

}

public static boolean Loaded = false;

public static void SetTime(String time)

{

TimeCash = time;

}

public static String GetTime()

{

String time = GetValueTime()+GetMeasurementTime();

SetTime(time);

return time;

}

public static String ChangeTime (String time)

{

SetTime(time);

String time1 = GetTime();

return time1;

}

public static String ChangeTime(String time, Context context)

{

String time1 = ChangeTime(time);

TilesDB.GetDB(context).GetTimeCash();

time1 = ChangeTime(GetTime());

return time1;

}

public static String ChangeTime(Context context)

{

ChangeTime(GetTime(), context);

Loaded = true;

return ChangeTimeInDB(context);

}

public static String ChangeTimeInDB(Context context)

{

TilesDB.GetDB(context).UpdateTimeCash();

return ChangeTime(GetTime(), context);

}

public static String GetMeasurementTime()

{

try

{

String measurement = TimeCash.substring(TimeCash.length() - 1, TimeCash.length());

if(measurement.equals("m") || measurement.equals("h") || measurement.equals("s") || measurement.equals("d"))

return measurement;

throw new Exception();

}

catch (Exception ex)

{

return "m";

}

}

public static Boolean api = false;

}

Приложение 8. Программный код тайла

package com.example.lab05\_map\_sidorov493;

import android.app.Activity;

import android.graphics.Bitmap;

import org.json.JSONException;

import java.text.SimpleDateFormat;

import java.util.Date;

public class Tile {

public int scale;

public int x;

public int y;

Bitmap bmp;

public int ID = -1;

public Bitmap GetBMP()

{

return bmp;

}

public void SetBMP(Bitmap bmp)

{

this.bmp = bmp;

}

public Tile(int x, int y, int scale, Bitmap bmp)

{

this(x, y, scale);

SetBMP(bmp);

}

public Tile(int x, int y, int scale)

{

this();

this.x = x;

this.y = y;

this.scale = scale;

nowTime = new Tile();

}

Tile nowTime;

public Tile()

{

SetDatetimeNow();

}

public void SetDatetimeNow()

{

SetTimeStamp(new Date());

}

public String TimeStamp;

public void SetdateTime(String date)

{

SetTimeStamp(new Date(date));

}

public void SetDateTime(String date)

{

TimeStamp = date;

}

public void SetTimeStamp(Date date)

{

TimeStamp = new SimpleDateFormat("dd.MM.yyyy hh:mm:ss").format(date);

}

public void SetDateTime(int hour, int minute, int second, int day, int month, int year)

{

//SetTimeStamp(new Date(hour, minute, second, day, month, year));

SetTimeStamp(new Date(year-1900, month, day, hour, minute, second));

}

public void SetTimeStamp(long date)

{

SetTimeStamp(new Date(date));

}

public String GetDatetime()

{

return TimeStamp;

}

public Date GetTimeStamp()

{

return new Date(Date.parse(TimeStamp));

}

public Date GetTimeStamp1()

{

return new Date(Years()-1900, Monthes()-1, Days(), Hours(), Minutes(), Seconds());

}

public int Days()

{

//return Integer.parseInt(GetDate().split(".")[0]);

return Integer.parseInt(GetDate1().split("/")[0]);

}

public String GetDate1()

{

return GetDate().replace('.', '/');

}

public int Monthes()

{

return Integer.parseInt(GetDate1().split("/")[1]);

}

public int Years()

{

return Integer.parseInt(GetDate1().split("/")[2]);

}

public int DeltaDays() {

nowTime.SetDatetimeNow();

int deltaDays = DeltaDays2();

if (deltaDays < 0)

deltaDays += 31;

return deltaDays;

}

public int DeltaDays2()

{

nowTime.SetDatetimeNow();

int deltaDays = DeltaDays1();

if(DeltaHours2() < 0)

deltaDays--;

if(DeltaHours2() > 24)

deltaDays++;

return deltaDays;

}

public int DeltaDays1()

{

nowTime.SetDatetimeNow();

return nowTime.Days() - Days();

}

public int DeltaHours()

{

int deltaHours = DeltaHours2();

if(deltaHours < 0)

deltaHours+=24;

if(deltaHours >= 24)

deltaHours-=24;

return deltaHours;

}

public int DeltaHours2()

{

nowTime.SetDatetimeNow();

int deltaHours = DeltaHours1();

if(DeltaMinutes2() < 0)

deltaHours--;

if(DeltaMinutes2() >= 60)

deltaHours++;

return deltaHours;

}

public int DeltaHours1()

{

nowTime.SetDatetimeNow();

return nowTime.Hours() - Hours();

}

public int Hours()

{

return Integer.parseInt(GetTime().split(":")[0]);

}

public int Minutes()

{

return Integer.parseInt(GetTime().split(":")[1]);

}

public int DeltaMinutes()

{

int deltaMinutes = DeltaMinutes2();

if(deltaMinutes < 0)

deltaMinutes+=60;

if(deltaMinutes >= 60)

deltaMinutes-=60;

return deltaMinutes;

}

public int DeltaMinutes2()

{

nowTime.SetDatetimeNow();

int deltaMinute = DeltaMinutes1();

if(DeltaSeconds1() < 0)

deltaMinute--;

if(DeltaSeconds1() >= 60)

deltaMinute++;

return deltaMinute;

}

public int DeltaMinutes1()

{

nowTime.SetDatetimeNow();

int minute1 = nowTime.Minutes();

int minute2 = this.Minutes();

int deltaMinute = minute1 - minute2;

return deltaMinute;

}

public int Seconds()

{

return Integer.parseInt(GetTime().split(":")[2]);

}

public int DeltaSeconds1()

{

nowTime.SetDatetimeNow();

int second1 = nowTime.Seconds();

int second2 = this.Seconds();

int deltaSecond = second1 - second2;

return deltaSecond;

}

public int DeltaSeconds()

{

int deltaSecond = DeltaSeconds1();

if(deltaSecond < 0)

deltaSecond += 60;

if(deltaSecond >= 60)

deltaSecond -= 60;

return deltaSecond;

}

public Boolean YesTime()

{

try {

int value = MapHelper.GetValueTime();

String measurement = MapHelper.GetMeasurementTime();

if (measurement.equals("s"))

return DeltaSeconds() < value;

if (measurement.equals("m"))

return DeltaMinutes() < value;

if (measurement.equals("h"))

return DeltaHours() < value;

if(measurement.equals("d"))

return DeltaDays() < value;

return false;

}

catch (Exception ex)

{

return false;

}

}

public String GetDate()

{

return GetDatetime().split(" ")[0];

}

public String GetTime()

{

return GetDatetime().split(" ")[1];

}

public Tile(int x, int y, int scale, Activity ctx)

{

this(x, y, scale);

try {

Tile tile = TilesDB.GetDB(ctx).GetTile(x, y, scale);

SetBMP(tile.bmp);

} catch (Exception e) {

e.printStackTrace();

if(!MapHelper.api)

{

return;

}

final Bitmap[] bmp = new Bitmap[1];

ApiHelper req = new ApiHelper(ctx) {

@Override

public void on\_ready(String res) {

if (!Ready)

return;

super.on\_ready(res);

try {

bmp[0] = BitmapFromJsonObject(res, "data");

} catch (JSONException e) {

e.printStackTrace();

}

}

};

req.SendNoParamsStop(MapHelper.GetUrl(ctx) + "raster", String.format("%d/%d-%d", scale, x, y));

SetBMP(bmp[0]);

}

}

}

Приложение 9. Программный код работы с API

package com.example.lab05\_map\_sidorov493;

import android.app.Activity;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.os.Build;

import androidx.annotation.RequiresApi;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.net.HttpURLConnection;

import java.net.URL;

import android.util.Base64;

public class ApiHelper

{

public Activity ctx;

public String Method = "GET";

public float DoubleFromJson(String res, String punkt, String Pole) throws JSONException {

try { JSONObject doc = new JSONObject(res);

JSONObject curr = doc.getJSONObject(punkt);

return (float) curr.getDouble(Pole);

}

catch(Exception e)

{

throw new JSONException(e.getMessage());

}

}

public float DoubleFromJsonObject(String res, String Pole) throws JSONException {

try { JSONObject doc = new JSONObject(res);

JSONObject curr = doc;

return (float) curr.getDouble(Pole);

}

catch(Exception e)

{

throw new JSONException(e.getMessage());

}

}

//@RequiresApi(api = Build.VERSION\_CODES.O)

public Bitmap BitmapFromJsonObject(String res, String Pole) throws JSONException {

try { JSONObject doc = new JSONObject(res);

JSONObject curr = doc;

String image = curr.getString(Pole);

//byte[] jpeg = Base64.getDecoder().decode(image);

byte [] jpeg = Base64.decode(image, Base64.DEFAULT);

Bitmap bmp = BitmapFactory.decodeByteArray(jpeg, 0, jpeg.length);

return bmp;

}

catch(Exception e)

{

throw new JSONException(e.getMessage());

}

}

public int IntFromJsonObject(String res, String Pole) throws JSONException {

try { JSONObject doc = new JSONObject(res);

JSONObject curr = doc;

return (int) curr.getInt(Pole);

}

catch(Exception e)

{

throw new JSONException(e.getMessage());

}

}

public String StringFromJson(String res, String punkt, String Pole) throws JSONException {

try { JSONObject doc = new JSONObject(res);

JSONObject curr = doc.getJSONObject(punkt);

return curr.getString(Pole);

}

catch(Exception e)

{

throw new JSONException(e.getMessage());

}

}

public String StringFromJsonObject(String res, String Pole) throws JSONException {

try { JSONObject doc = new JSONObject(res);

JSONObject curr = doc;

return curr.getString(Pole);

}

catch(Exception e)

{

throw new JSONException(e.getMessage());

}

}

public ApiHelper(Activity ctx)

{

this.ctx = ctx;

}

public void on\_ready(String res){

MessageOutput(GetMessageReady());

MessageReadyOutput(GetMessageReady());

Ready = true;

}

public void on\_fail()

{

ctx.runOnUiThread(() -> {

MessageOutput(GetMessageFatal());

MessageFatalOutput(GetMessageFatal());

Ready = false;

});

}

public String GetMessageFatal()

{

return "";

}

public void MessageFatalOutput(String message)

{

}

public void MessageOutput(String message)

{

}

public String GetMessageReady()

{

return "";

}

public void MessageReadyOutput(String message)

{

}

String http\_get(String req, String payload) throws IOException

{

return http\_get(req, payload, "GET");

}

String http\_put(String req, String payload) throws IOException

{

return http\_get(req, payload, "PUT");

}

String http\_post(String req, String payload) throws IOException

{

return http\_get(req, payload, "POST");

}

String http\_delete(String req, String payload) throws IOException

{

return http\_get(req, payload, "DELETE");

}

public Boolean JsonInput = false;

public Boolean Params = true;

String http\_get(String req, String payload, String method) throws IOException

{

return http\_get(req, payload,method,JsonInput);

}

String http\_get(String req, String payload, String method, Boolean JsonInput) throws IOException

{

return http\_get(req, payload, method, JsonInput, Params);

}

String http\_get(String req, String payload, String method, Boolean JsonInput, Boolean params) throws IOException

{

Ready = false;

String url1 = req;

if(!JsonInput) {

if(params) {

url1 += "?" + payload;

}

else if(!payload.equals("")) {

url1 += "/";

url1 += payload;

}

}

URL url = new URL(url1);

HttpURLConnection con = (HttpURLConnection) url.openConnection();

byte[] outmsg = payload.getBytes("utf-8");

con.setRequestMethod(method);

if(JsonInput) {

con.setRequestProperty("Content-Type", "application/json");

con.setRequestProperty("Content-Length", String.valueOf(outmsg.length));

con.setDoOutput(true);

con.setDoInput(true);

BufferedOutputStream out = new BufferedOutputStream(con.getOutputStream());

out.write(outmsg);

out.flush();

}

InputStream is = con.getInputStream();

BufferedInputStream inp = new BufferedInputStream(is);

byte[] buf = new byte[1024];

String res = "";

while (true)

{

int num = inp.read(buf);

if (num < 0) break;

res += new String(buf, 0, num);

}

con.disconnect();

Ready = true;

return res;

}

public String res, session;

public ApiHelper GetAPIHelper()

{

return this;

}

public class NetOp implements Runnable

{

public String req, payload;

public void run()

{

try

{

final String res = http\_get(req, payload, Method);

ctx.runOnUiThread(

new Runnable()

{

@Override

public void run() {

on\_ready(res);

}

}

);

GetAPIHelper().res = res;

}

catch (Exception ex)

{

on\_fail();

}

}

}

public Thread th;

public void send(String req, String payload)

{

NetOp nop = new NetOp();

nop.req = req;

nop.payload = payload;

th = new Thread(nop);

th.start();

}

public void send (String req, String payload, Boolean params)

{

Params = params;

send(req, payload);

}

public void sendNoParams(String req, String payload)

{

send(req, payload, false);

}

public Boolean Ready = false;

public void on\_ready()

{

on\_ready(res);

}

public void SendStop()

{

Ready = false;

Send();

try {

th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

go\_Ready();

}

public void go\_Ready()

{

if(Ready) {

on\_ready();

}

}

public void SendStop(String req, String payload)

{

Ready = false;

send(req, payload);

try {

th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

go\_Ready();

}

public void SendStop(String req, String payload, Boolean params)

{

Ready = false;

send(req, payload, params);

try {

th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

go\_Ready();

}

public void SendNoParamsStop(String req, String payload)

{

Ready = false;

sendNoParams(req, payload);

try {

th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

go\_Ready();

}

public void Send()

{

}

}