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 графического приложения с использованием REST API**

Выполнил

студент Группы 493

сидоров антон дмитриевич

Проверила Фомин А. В.

Оценка \_\_\_\_\_\_\_\_

Санкт-Петербург 2022

**Содержание**

[1. Цели работы 3](#_Toc122277254)

[2. Диаграмма базы данных 3](#_Toc122277255)

[3. Макеты экранов приложения и их описание 4](#_Toc122277256)

[4. Программный код 5](#_Toc122277257)

[5. Демонстрация работы приложения 8](#_Toc122277258)

[6. Вывод 20](#_Toc122277259)

[Приложение 21](#_Toc122277260)

# Цели работы

Разработать интерактивное приложение для работы с графами. В качестве хранилища данных использовать предоставленное API.

# Диаграмма базы данных

На рисунках с диаграммами, также представлен словарь данных

Таблицы с ключём сессии и с URL-строкой для API представлены на рисунке 1.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 1 – Диаграмма базы данных

Остальные диаграммы представлены на рисунке 2.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 2 – 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 3.



Рисунок 3 – Макеты окон

Макеты, которые связаны с API показаны на рисунке 4.

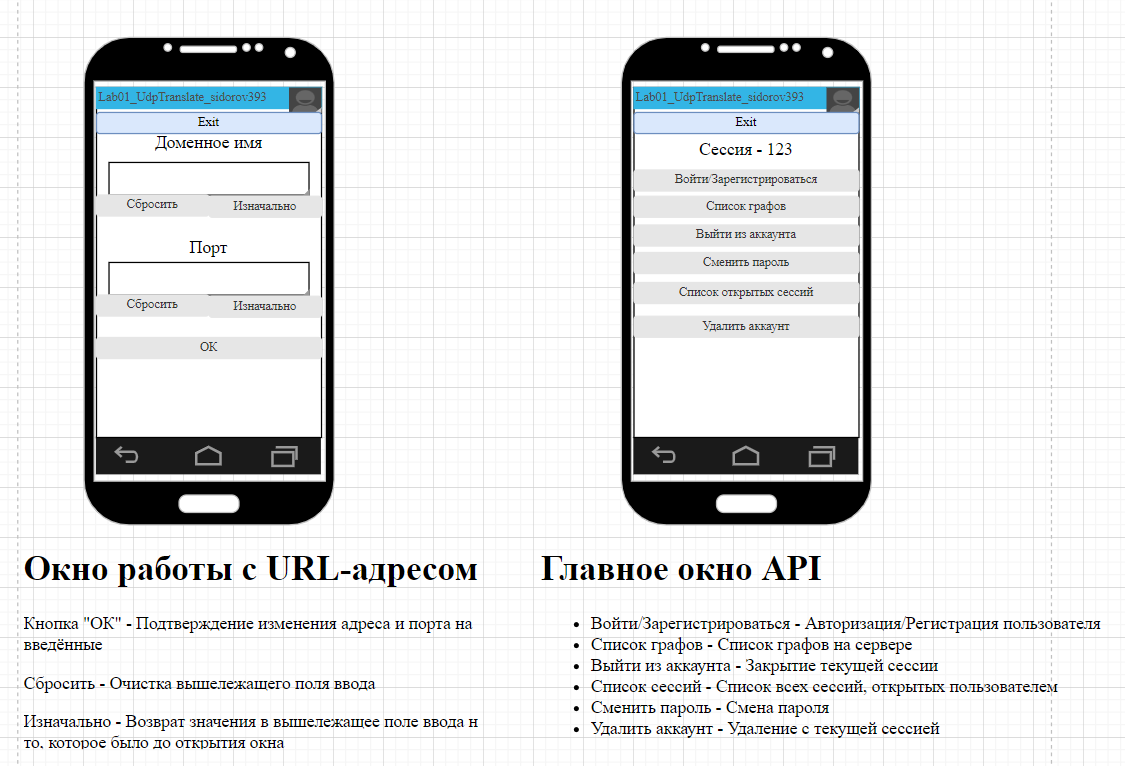


Рисунок 4 – Макеты окон

Остальные макеты окон представлены на рисунке 5.

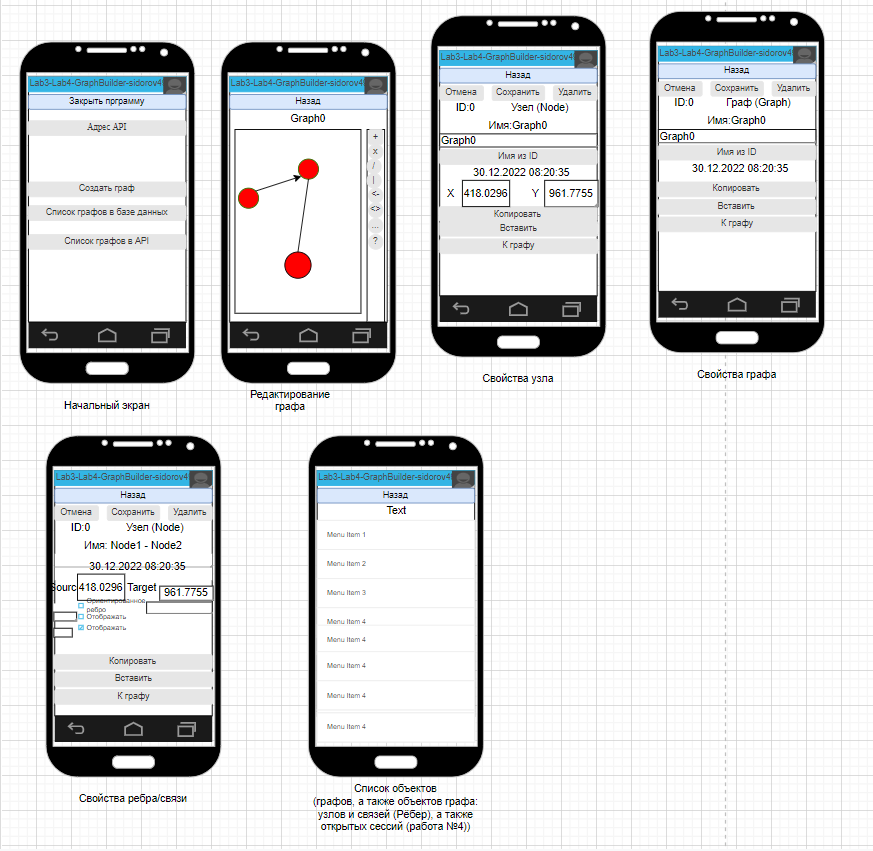


Рисунок 5 – Макеты окон

# Программный код

Программный код условно можно разбить на 6 частей:

1. База данных SQLite
2. Код работы окон приложения
3. Классы объектов графа
4. Буфер обмена для объекта графа
5. Код компонентов интерфейса
6. Код работы с API

Далее всё буде рассмотрено подробнее.

* 1. База данных SQLite

Код работы с базой данных графов представлен в приложении 1.

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

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

* 1. Код работы окон приложения

Программный код начального окна представлен в приложении 4.

Программный код окна редактора графов представлен в приложении 5.

Программный код окна свойств объектов графа (графа, узла, ребра) представлен в приложении 6.

Программный код окна списка объектов графа (графа, узла, ребра), а также списка сессий в приложении 7.

Программный код окна редактирования URL-ссылки представлен в приложении 8.

Программный код окна авторизации, регистрации и смены пароля представлен в приложении 9.

Программный код Начального окна API представлен в приложении 10.

* 1. Классы объектов графа

Один из этих классов – список объектов графа, представленный в приложении 11

Следующие 3 класса объектов графа, которые используются в визуальном отображении, наследуются от одного базового абстрактного класса *GraphElement*, представленного на рисунке 12. Это классы:

* *Собственно, Граф*, представленный в приложении 13
* *Узел графа*, представленный в приложении 14.
* *Ребро графа (Или, связь графа)*, представленный в приложении 15.

Ещё, вспомогательный класс для отображения узлов и рёбер – цвет элементов графа, представленный в приложении 16.

Также, есть перечисление для типов элемента графа:

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

public enum GraphElementName {

Graph, // Собственно, Граф

Node, // Узел

Link // Связь или ребро

}

* 1. Буфер обмена для объектов графа

Класс *Буфер обмена для объектов графа*, представлен в приложении 17.

* 1. Код компонентов интерфейса

Рисовальная панель для графа представлена на рисунке 18.

Текст с надписью представлен в приложении 19.

Текст с флажком представлен на рисунке 20

* 1. Код работы с API

Для этих классов есть перечисление для выбора «Graph»/«ID» в ссылках, где есть ID графа:

public enum GraphIdView {

graph,

id

}

Базовый класс для работы с API (От него унаследованы все остальные) представлен в приложении 21.

Класс для открытия сессии представлен в приложении 22.

Класс для закрытия сессии представлен в приложении 23.

Класс для просмотра списка сессий представлен в приложении 24.

Класс для создания аккаунта представлен в приложении 25.

Базовый класс для графов в API представлен в приложении 26.

Класс для создания графов в API представлен в приложении 27.

Класс для просмотра списка графов в API представлен в приложении 28.

Базовый класс для узлов графа в API представлен в приложении 29.

Класс для создания узлов графа в API представлен в приложении 30.

Класс для просмотра списка узлов графа в API представлен в приложении 31.

Базовый класс для связей графа в API представлен в приложении 32.

Класс для создания связей графа в API представлен в приложении 33.

Класс для просмотра списка связей графа в API представлен в приложении 34.

Также имеется вспомогательный класс «Сессия»:

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import java.text.SimpleDateFormat;

import java.util.Date;

public class Session {

public String Key = "";

public String TimeStamp = "";

public int ID = -1;

public void SetDatetimeNow()

{

SetTimeStamp(new Date());

}

public void SetTimeStamp(long date)

{

SetTimeStamp(new Date(date));

}

public void SetDateTime(String date)

{

SetTimeStamp(new Date(date));

}

public void SetTimeStamp(String date)

{

SetDateTime(date);

}

public String GetDatetime()

{

return TimeStamp;

}

public Date GetTimeStamp()

{

return new Date(TimeStamp);

}

public void SetTimeStamp(Date date)

{

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

}

# Демонстрация работы приложения

Данное приложение будет тестироваться эмуляторе Android Studio.

На начальном экране присутствуют кнопки:

* «Закрыть программу» – Выход из программы
* «Адрес API» – Переход на окно для редактирование URL-ссылки для подключения к API
* «Создать граф» – Переход на окно для редактирования нового графа
* «Список графов в базе данных» – Переход к списку графов, которые хранятся в базе данных
* «Система API» – Переход на окно с функциями с API (Главное окно с API).

Это окно представлено на рисунке 6.

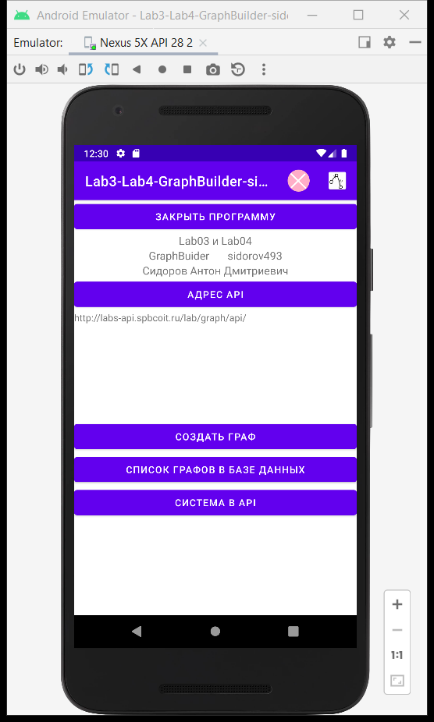


Рисунок 6 – Запущенная программа

Окно для редактирования URL-ссылки имеет 2 текстовых поля ввода:

* «Доменное имя» – Адрес Web-сервера, может быть вместе с путём.
* «Порт» – Порт сервера или путь к ресурсу

Теоретическая схема (Например, Сервер «Google.com» и порт «900») должна выглядеть следующим образом Домен:Порт (То есть, «Google.com:900»). Но возможны и другие варианты (Приведены примеры):

1. Сервер «Google.com» и Порт «900/a» – Google.com:900/a
2. Сервер «Google.com:90» и Порт «900/a» – Google.com:90/900/a
3. Сервер «Google.com/90» и Порт «900/a» – Google.com/90/900/a
4. Сервер «Google.com» и Порт «anton» – Google.com/anton
5. Сервер «Google.com/» и Порт «900/a» – Google.com/900/a
6. Сервер «Google.com» и Порт «:900/a» – Google.com/:900/a
7. Сервер «Google.com» и Порт «/900/a» – Google.com/900/a

Так сделано для гибкой и, в то же время, стабильной обработки URL-ссылок, состоящих из данных значений.

Также, возле каждого текстового поля ввода есть 2 кнопки:

* «Сбросить» – Очищает текстовое поле ввода
* «Изначально» – Устанавливает значение, которое было до открытия окна (оно хранится в базе данных).

Также, присутствуют кнопки «ОК» и «Назад» – Они обе закрывают окно, возвращают на начальный экран и сохраняют URL-строку в базу данных.

В приведённом ниже примере сервер «labs-api.spbcoit.ru» и порт «lab/graph/api», то есть URL-ссылка – labs-api.spbcoit.ru/ lab/graph/api, которая будет использоваться дальше. Окно с данным примером представлено на рисунке 7.

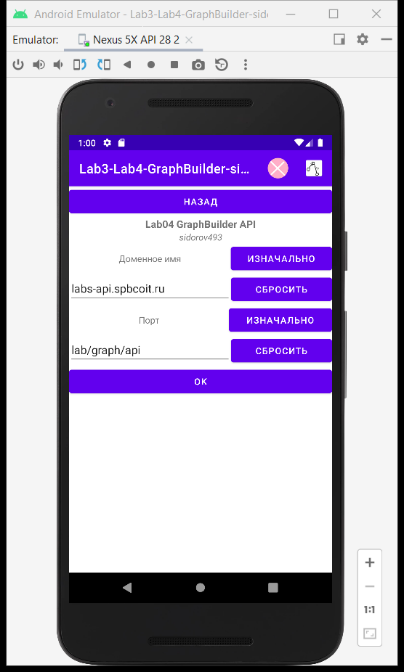


Рисунок 7 – Запущенная программа

На главном окне с API присутствует выведенный ключ открытой сессии и кнопки:

* «Назад» – Возврат на главное окно
* «Войти или зарегистрироваться» – Переход на окно авторизации пользователя.
* «Список графов» – Переход на окно со списком графов, который выводится по ссылке: http://labs-api.spbcoit.ru/lab/graph/api/session/list?token=riqp4ztz0f.
* «Выйти из аккаунта» – Закрытие текущей сессии по ссылке: http://labs-api.spbcoit.ru/lab/graph/api/session/close?token=riqp4ztz0f.
* «Сменить пароль» – Переход на окно смены пароля
* «Список открытых сессий» – Переход на окно со списком открытых сессий по ссылке: http://labs-api.spbcoit.ru/lab/graph/api/session/list?token=g58hp7fh16.
* «Удалить аккаунт» – Удаление аккаунта для которого открыта текущая сессия по ссылке: http://labs-api.spbcoit.ru/lab/graph/api/account/delete?token=g58hp7fh16.

Это окно представлено на рисунке 8.

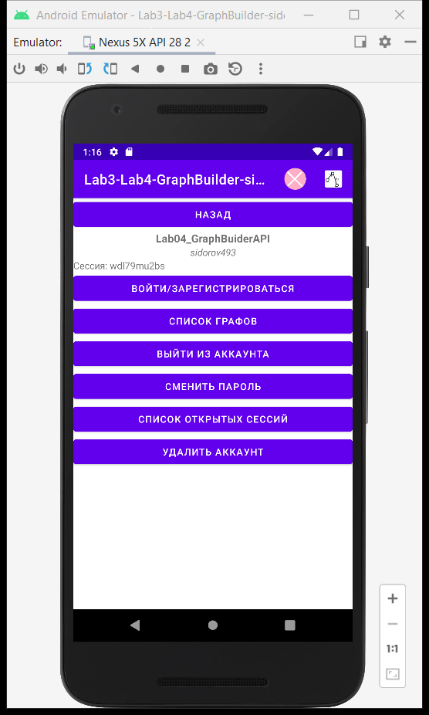


Рисунок 8 – Запущенная программа

Окно авторизации имеет 2 текстовых поля ввода: «Логин» и «Пароль». Также имеет флажок «Показывать пароль» для установки надо ли показывать символы в пароле, или надо показывать вместо них точки. Данное окно имеет несколько кнопок:

* «Назад» – Возврат на главный экран с API
* «Зарегистрироваться» – Регистрация пользователя с введёнными логином и паролем по ссылке: http://labs-api.spbcoit.ru/lab/graph/api/account/create?name=AntonSidorov493&secret=password.
* «Войти» – Открытие сессии пользователя с введёнными логином и паролем по ссылке: http://labs-api.spbcoit.ru/lab/graph/api/session/open?name=AntonSidorov493&secret=password.

Это окно представлено на рисунке 9.

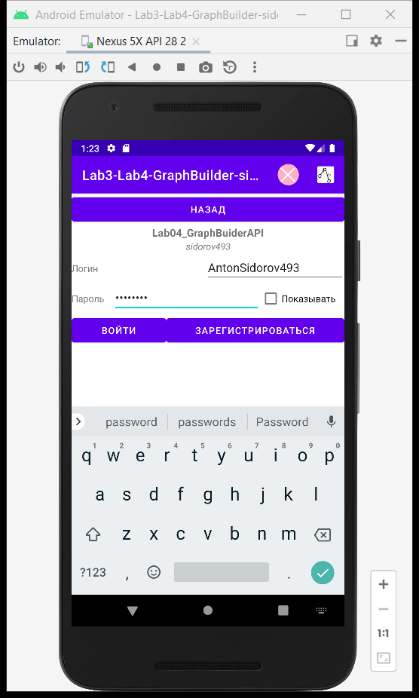


Рисунок 9 – Запущенная программа

Окно смены пароля имеет текстовое поле ввода: «Пароль». Также имеет флажок «Показывать пароль» для установки надо ли показывать символы в пароле, или надо показывать вместо них точки. Данное окно имеет несколько кнопок:

* «Назад» – Возврат на главный экран с API
* «Установить пароль» – Смена пароля на введённый: http://labs-api.spbcoit.ru/lab/graph/api/account/update?token=6dn09kvdoi&secret=password.

Это окно представлено на рисунке 10.

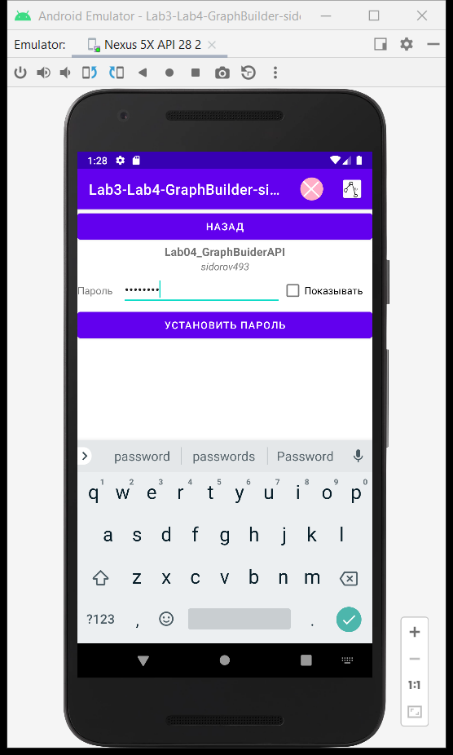


Рисунок 10 – Запущенная программа

Окно списка открытых сессий имеет кнопку «Назад», возвращающую на главный экран с API и перечисленный список сессий. Можно выбрать любую из этих сессий – откроется диалоговое окно с данными сессий и возможностью закрыть сессию.

Данное окно представлено на рисунке 11.

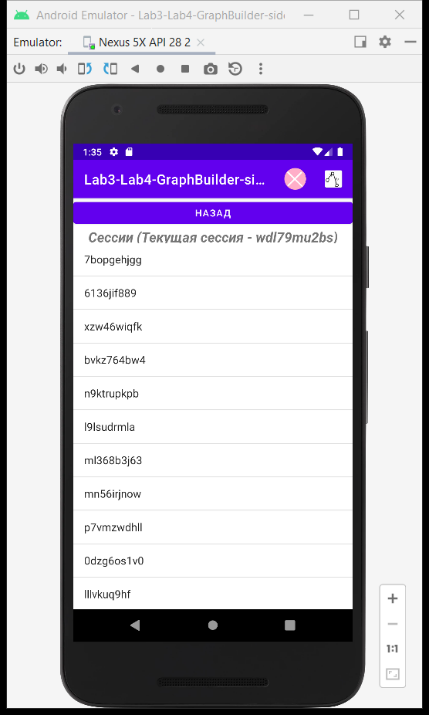


Рисунок 11 – Запущенная программа

Просмотр выбранной сессии показан на рисунке 12.

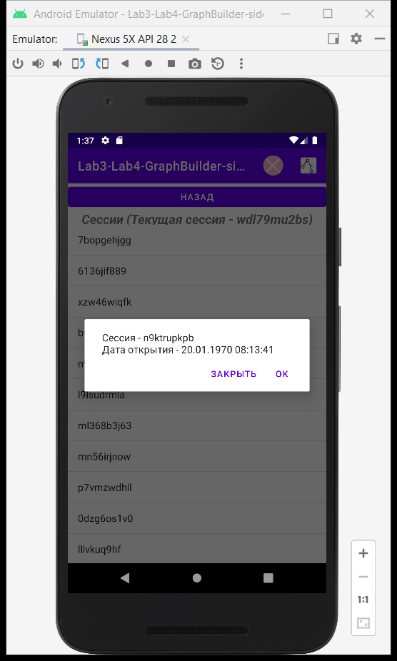


Рисунок 12 – Запущенная программа

Окно списка графов имеет 2 кнопки:

* «Назад» – Возврат на экран, с которого окно было запущено.
* «Добавить» – Создание нового графа. В случае с API по ссылке: http://labs-api.spbcoit.ru/lab/graph/api/graph/create?token=6dn09kvdoi&name=name. Переход на окно редактирования графа

Также, окно имеет список графов. При выборе любого из нах открывается окно редактирования графов.

Данное окно представлено на рисунке 13.

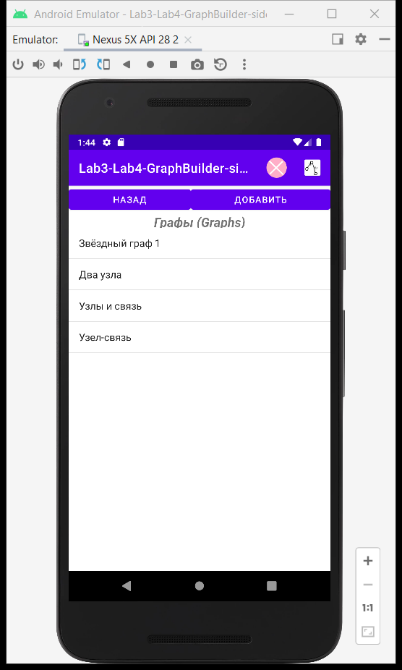


Рисунок 12 – Запущенная программа

Экран редактирования графа имеет надпись с названием графа и кнопку «Назад». Также это окно имеет Панель для рисования графа. И, наконец, эта панель имеет несколько кнопок:

* Кнопка создания графа. В случает с API по ссылке «http://labs-api.spbcoit.ru/lab/graph/api/node/create?token=6dn09kvdoi&пкфзр=105&x=123&y=123&name=node» или «http://labs-api.spbcoit.ru/lab/graph/api/node/create?token=6dn09kvdoi&id=105&x=123&y=123&name=node» (по какой получится)
* Кнопки создания связи (Одна для ориентированного графа, другая для неориентированного). В случае с API по ссылке: http://labs-api.spbcoit.ru/lab/graph/api/link/create?token=6dn09kvdoi&source=552&target=554&value=0
* Кнопка изменения ориентации связи. В случае с API – сначала удаление, потом создание заново
* Кнопка изменения узла/связи (Переход на окно свойств узла/связи). Изменения происходят после закрытия окна свойств. В случае с API по ссылке «http://labs-api.spbcoit.ru/lab/graph/api/node/update?token=6dn09kvdoi&id=552&x=552&y=250&name=node3» для узлов и «http://labs-api.spbcoit.ru/lab/graph/api/link/update?token=6dn09kvdoi&id=131&value=14.5» для связей.
* Кнопка изменения свойств графа с переходом на окно изменения свойств графа. В случае с API изменения происходят по ссылке «http://labs-api.spbcoit.ru/lab/graph/api/graph/update?token=6dn09kvdoi&graph=105&name=graph» или «http://labs-api.spbcoit.ru/lab/graph/api/graph/update?token=6dn09kvdoi&id=105&name=graph» (По какой получится)
* Кнопка просмотра списка узла/связи (Переход на окно списка узла/связи через диалоговое окно). В случае с API по ссылке для узлов «http://labs-api.spbcoit.ru/lab/graph/api/node/list?token=6dn09kvdoi&graph=105» иди «http://labs-api.spbcoit.ru/lab/graph/api/node/list?token=6dn09kvdoi&id=105», для связей «http://labs-api.spbcoit.ru/lab/graph/api/link/list?token=6dn09kvdoi&graph=105» иди «http://labs-api.spbcoit.ru/lab/graph/api/link/list?token=6dn09kvdoi&id=105».
* Кнопка удаления узла/связи. В случае с API по ссылке «http://labs-api.spbcoit.ru/lab/graph/api/node/delete?token=6dn09kvdoi&id=552» для узлов и «http://labs-api.spbcoit.ru/lab/graph/api/link/delete?token=6dn09kvdoi&id=131» для связей.

Это окно представлено на рисунке 13.

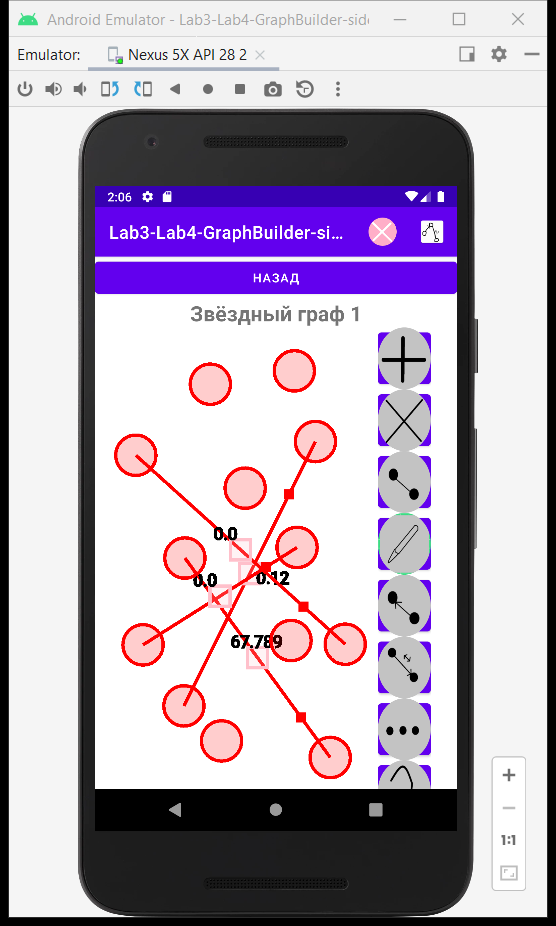


Рисунок 13 – Запущенная программа

Окно свойств графа имеет поле ввода «Имя» (Название графа) и кнопку «Имя из ID» (Вывод названия графа из ID в коллекции), а также кнопу «Назад» и надпись с датой создания графа. Также данное окно имеет кнопки:

* «Копировать» – Копирование графа в буфер обмена
* «Вставить» – Получение графа из буфера обмена
* «Сохранить в API» (Только для графов в базе данных) – сохраняет граф на сервере по URL-ссылке (Необходимо авторизироваться)
* «Сохранить на устройстве» (Только для графов на сервере API) – Сохраняет граф в базе данных
* «Удалить» – Удаление графа. В случае с API по ссылке «http://labs-api.spbcoit.ru/lab/graph/api/graph/delete?token=6dn09kvdoi&graph=105» или «http://labs-api.spbcoit.ru/lab/graph/api/graph/delete?token=6dn09kvdoi&id=105» (По какой получится)

Данное окно для графов в базе данных показано на рисунке 14.

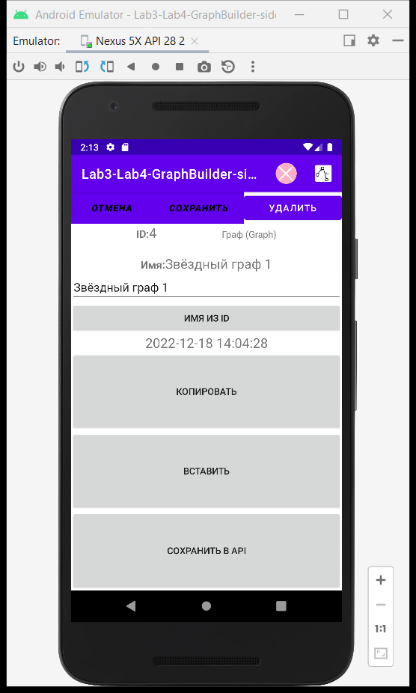


Рисунок 14 – Запущенная программа

Данное окно для графов на сервере API показано на рисунке 14.

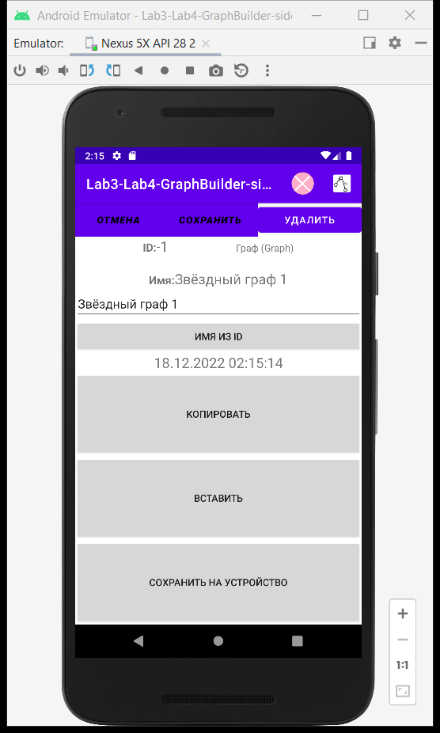


Рисунок 15 – Запущенная программа

Окна списка узлов и связей имеют одинаковый вид, причём тот же вид, что и окно списка графов, поэтому особых комментарии здесь не прилагаются. Единственная особенность – при выборе узла/связи в списке происходит переход к свойствам.

Окно списка узлов показано на рисунке 16.

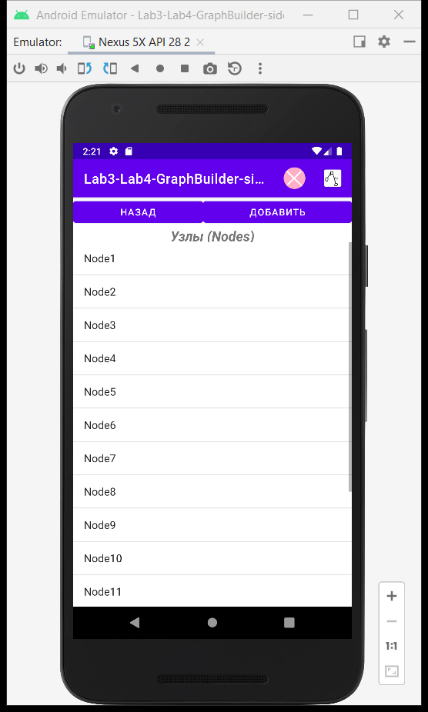


Рисунок 16 – Запущенная программа

Окно списка узлов показано на рисунке 17.

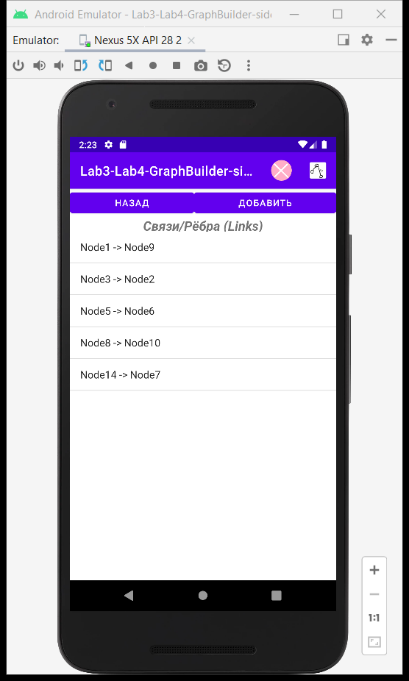


Рисунок 17 – Запущенная программа

Окно свойств узла имеет поле ввода «Имя» (Название узла) и кнопку «Имя из ID» (Вывод названия узла из ID в коллекции), а также кнопу «Назад» и надпись с датой создания узла. Также данное окно имеет текстовые поля ввода «X» и «Y» – для ввода координат узла. Данное окно представлено на рисунке 18.

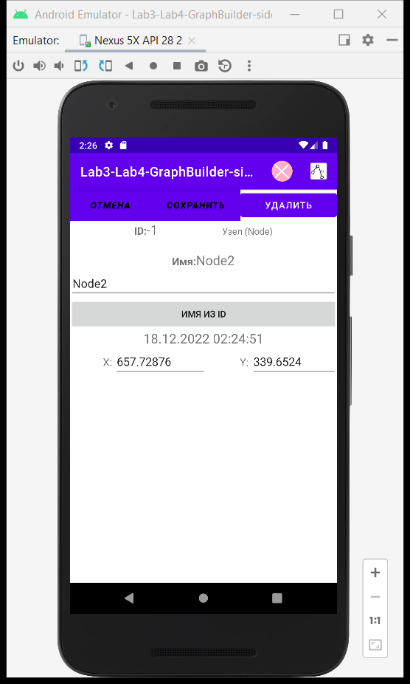


Рисунок 18 – Запущенная программа

Окно свойств связи имеет поле ввода «Имя» (Название связи) и кнопку «Имя из ID» (Вывод названия связи из ID в коллекции), а также кнопу «Назад» и надпись с датой создания связи. Также данное окно имеет текстовые поля ввода «Source» и «Target» для ID начального и конечного узла связи, соответственно. Присутствует флажок, указывающий, ориентировано ли ребро (это связь, по-другому) и кнопка для смены направления связи. Присутствует поля для ввода текстовой надписи и числового значения, а возле них – флажок, указывающий, нужно ли отображать значение этих полей. Данное окно представлено на рисунке 19.

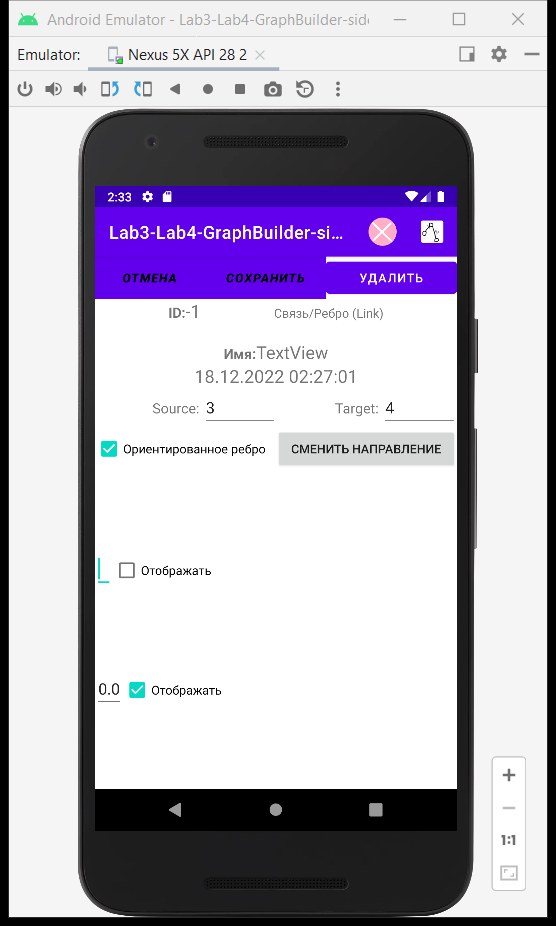


Рисунок 18 – Запущенная программа

# Вывод

Освоено рисование графов в java.

# Приложение

**Приложение 1. Программный код SQLite для графов**

package com.example.lab3\_lab4\_graphbuilder\_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;

import java.util.ArrayList;

public class DB\_Graphs extends SQLiteOpenHelper {

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

super(context, name, factory, version);

}

@Override

public void onCreate(SQLiteDatabase db) {

//String sql = "CREATE TABLE notes (id INT, txt Text);";

//db.execSQL(sql);

String graphTable = "CREATE TABLE graph (\n" +

" id INTEGER NOT NULL,\n" +

" name VARCHAR(30) NOT NULL,\n" +

" timestamp DATETIME DEFAULT (CURRENT\_TIMESTAMP) NOT NULL,\n" +

" PRIMARY KEY (id)\n" +

")";

db.execSQL(graphTable);

String nodeTable = "CREATE TABLE node (\n" +

" id INTEGER NOT NULL,\n" +

" graph INTEGER NOT NULL,\n" +

" x FLOAT NOT NULL,\n" +

" y FLOAT NOT NULL,\n" +

" name VARCHAR(30) NOT NULL,\n" +

" PRIMARY KEY (id),\n" +

" FOREIGN KEY(graph) REFERENCES graph (id) ON DELETE CASCADE\n" +

")";

db.execSQL(nodeTable);

String linkTable = "CREATE TABLE link (\n" +

" id INTEGER NOT NULL,\n" +

" source INTEGER NOT NULL,\n" +

" target INTEGER NOT NULL,\n" +

" orientatin Integer Not null, \n" +

" value FLOAT NOT NULL,\n" +

" valueVisible Integer Not null, \n" +

" Text VARCHAR(200) NOT NULL,\n" +

" textVisible Integer Not null, \n" +

" PRIMARY KEY (id),\n" +

" UNIQUE (source, target),\n" +

" FOREIGN KEY(source) REFERENCES node (id) ON DELETE CASCADE,\n" +

" FOREIGN KEY(target) REFERENCES node (id) ON DELETE CASCADE\n" +

")";

db.execSQL(linkTable);

}

public static DB\_Graphs CreateDB(@Nullable Context context, @Nullable String name)

{

return CreateDB(context, name, null, 1);

}

public static DB\_Graphs CreateDB(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version)

{

DB\_Graphs graphs = new DB\_Graphs(context, name, factory, version);

graphs.GetGraphs();

return graphs;

}

public void GetGraphs()

{

SQLiteDatabase db = getReadableDatabase();

String sql = "Select \* From graph;";

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

}

public int getMaxGraphId()

{

SQLiteDatabase db = getReadableDatabase();

String sql = "Select Max(id) From graph;";

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

if(cur.moveToFirst() == true) return cur.getInt(0);

return 0;

}

public int getCountGraphId(int id)

{

SQLiteDatabase db = getReadableDatabase();

String sql = "Select Count(id) From graph where id="+id+";";

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

if(cur.moveToFirst() == true) return cur.getInt(0);

return 0;

}

public int getMaxNodeId()

{

SQLiteDatabase db = getReadableDatabase();

String sql = "Select Max(id) From node;";

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

if(cur.moveToFirst() == true) return cur.getInt(0);

return 0;

}

public int getCountLinkId(int id)

{

SQLiteDatabase db = getReadableDatabase();

String sql = "Select Count(id) From link where id="+id+";";

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

if(cur.moveToFirst() == true) return cur.getInt(0);

return 0;

}

public int getMaxLinkId()

{

SQLiteDatabase db = getReadableDatabase();

String sql = "Select Max(id) From link;";

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

if(cur.moveToFirst() == true) return cur.getInt(0);

return 0;

}

public int getCountNodeId(int id)

{

SQLiteDatabase db = getReadableDatabase();

String sql = "Select Count(id) From node where id="+id+";";

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

if(cur.moveToFirst() == true) return cur.getInt(0);

return 0;

}

@Override

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

}

public void AddNote(int id, String stxt)

{

String sid = String.valueOf(id);

SQLiteDatabase db = getWritableDatabase();

String sql = "INSERT INTO notes VALUES ("+sid+", '"+stxt+"');";

db.execSQL(sql);

}

public String getNote(int id)

{

String sid = String.valueOf(id);

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT txt FROM notes WHERE id = "+sid+";";

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

if(cur.moveToFirst() == true) return cur.getString(0);

return "";

}

public void getAllNotes (ArrayList<Graph> lst)

{

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT id, txt FROM notes;";

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

if(cur.moveToFirst() == true)

{

do {

Graph n = new Graph();

//n.id = cur.getInt(0);

//n.txt = cur.getString(1);

lst.add(n);

}

while(cur.moveToNext() == true);

}

}

public void AlterNote(int id, String stxt)

{

String sid = String.valueOf(id);

SQLiteDatabase db = getWritableDatabase();

String sql = "UPDATE notes SET txt = '"+stxt+"' WHERE id = "+sid+";";

db.execSQL(sql);

}

public void DeleteAllNodes(Graph graph)

{

for(int i = 0; i < graph.LinkCount(); i++)

{

delete\_node(graph.GetNode(i).Get\_API\_ID());

}

}

public void delete\_graph(Graph graph)

{

DeleteAllNodes(graph);

SQLiteDatabase db = getWritableDatabase();

String sql = "DELETE FROM graph where id="+graph.Get\_API\_ID()+";";

db.execSQL(sql);

}

public void delete\_node(int node)

{

SQLiteDatabase db = getWritableDatabase();

String sql = "DELETE FROM node where id="+node+";";

db.execSQL(sql);

db = getWritableDatabase();

sql = "DELETE FROM link where source="+node+" or target="+node+";";

db.execSQL(sql);

}

public void delete\_link(int link)

{

SQLiteDatabase db = getWritableDatabase();

String sql = "DELETE FROM link where id="+link+";";

db.execSQL(sql);

}

public void upload\_graph(Graph graph)

{

int api\_id = graph.Get\_API\_ID();

int max = getMaxGraphId();

int count = getCountGraphId(api\_id);

if(count != 0)

{

SQLiteDatabase db = getWritableDatabase();

String sql = "Update graph set name='"+graph.GetName()+"' where id="+graph.Get\_API\_ID()+";";

db.execSQL(sql);

}

else

{

SQLiteDatabase db = getWritableDatabase();

graph.Set\_API\_ID(max+1);

String sql = "INSERT INTO graph (id, name) VALUES ("+graph.Get\_API\_ID()+", '"+graph.GetName()+"');";

db.execSQL(sql);

}

api\_id = graph.Get\_API\_ID();

ArrayList<Integer> IDs = new ArrayList<Integer>();

IDs.clear();

for(int i = 0; i < graph.NodeCount(); i++)

{

Node node = graph.GetNode(i);

upload\_node(node);

IDs.add(node.Get\_API\_ID());

}

ArrayList<Node> nodes = GetListNodes(api\_id);

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

{

Node node = nodes.get(i);

int id = node.Get\_API\_ID();

if(!IDs.contains(id))

{

delete\_node(id);

}

}

IDs.clear();

for(int i = 0; i < graph.LinkCount(); i++)

{

Link node = graph.GetLink(i);

upload\_link(node);

IDs.add(node.Get\_API\_ID());

}

ArrayList<Link> links = GetListLinks(api\_id);

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

{

Link node = links.get(i);

int id = node.Get\_API\_ID();

if(!IDs.contains(id))

{

delete\_link(id);

}

}

}

public void upload\_node(Node graph)

{

int api\_id = graph.Get\_API\_ID();

int max = getMaxNodeId();

int count = getCountNodeId(api\_id);

if(count != 0)

{

SQLiteDatabase db = getWritableDatabase();

String sql = "Update node set name='"+graph.GetName()+"' where id="+graph.Get\_API\_ID()+";";

//db.execSQL(sql);

ContentValues values = new ContentValues();

//values.put("id", graph.Get\_API\_ID());

values.put("name", graph.GetName());

values.put("x", graph.X);

values.put("y", graph.Y);

//values.put("graph", graph.GetGraph().Get\_API\_ID());

db.update("node", values, "id="+graph.Get\_API\_ID(), null);

}

else

{

SQLiteDatabase db = getWritableDatabase();

graph.Set\_API\_ID(max+1);

//String sql = "INSERT INTO graph (id, name, x, y) VALUES ("+graph.Get\_API\_ID()+", '"+graph.GetName()+"', "+graph.X+", "+graph.Y+");";

//db.execSQL(sql);

ContentValues values = new ContentValues();

values.put("id", graph.Get\_API\_ID());

values.put("name", graph.GetName());

values.put("x", graph.X);

values.put("y", graph.Y);

values.put("graph", graph.GetGraph().Get\_API\_ID());

db.insert("node", null, values);

}

}

public void upload\_link(Link graph)

{

int api\_id = graph.Get\_API\_ID();

int max = getMaxLinkId();

int count = getCountLinkId(api\_id);

if(count != 0)

{

SQLiteDatabase db = getWritableDatabase();

String sql = "Update node set name='"+graph.GetName()+"' where id="+graph.Get\_API\_ID()+";";

//db.execSQL(sql);

ContentValues values = new ContentValues();

values.put("source", graph.Source().Get\_API\_ID());

values.put("target", graph.Target().Get\_API\_ID());

int orientation = 0;

if(graph.Orientation)

orientation = 1;

values.put("orientatin", orientation);

orientation = 0;

if(graph.TextVisible)

orientation = 1;

values.put("textVisible", orientation);

orientation = 0;

if(graph.ValueVisible)

orientation = 1;

values.put("valueVisible", orientation);

values.put("Text", graph.Text);

values.put("value", graph.Value);

db.update("link", values, "id="+graph.Get\_API\_ID(), null);

}

else

{

SQLiteDatabase db = getWritableDatabase();

graph.Set\_API\_ID(max+1);

//String sql = "INSERT INTO graph (id, name, x, y) VALUES ("+graph.Get\_API\_ID()+", '"+graph.GetName()+"', "+graph.X+", "+graph.Y+");";

//db.execSQL(sql);

ContentValues values = new ContentValues();

values.put("id", graph.Get\_API\_ID());

//values.put("name", graph.GetName());

values.put("source", graph.Source().Get\_API\_ID());

values.put("target", graph.Target().Get\_API\_ID());

int orientation = 0;

if(graph.Orientation)

orientation = 1;

values.put("orientatin", orientation);

orientation = 0;

if(graph.TextVisible)

orientation = 1;

values.put("textVisible", orientation);

orientation = 0;

if(graph.ValueVisible)

orientation = 1;

values.put("valueVisible", orientation);

values.put("Text", graph.Text);

values.put("value", graph.Value);

db.insert("link", null, values);

}

}

public ArrayList<Graph> GetListGraphs()

{

ArrayList<Graph> lst = new ArrayList<>();

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT id, name, timestamp FROM graph;";

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

if(cur.moveToFirst() == true)

{

do {

Graph n = new Graph();

//n.id = cur.getInt(0);

//n.txt = cur.getString(1);

n.Set\_API\_ID(cur.getInt(0));

n.SetName(cur.getString(1));

n.TimeStamp = cur.getString(2);

lst.add(n);

}

while(cur.moveToNext() == true);

}

return lst;

}

public void GetListNodes(Graph graph)

{

ArrayList<Node> lst = GetListNodes(graph.Get\_API\_ID());

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

{

graph.AddNode(lst.get(i));

}

}

public ArrayList<Node> GetListNodes(int id)

{

ArrayList<Node> lst = new ArrayList<>();

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT id, name, x, y FROM node where graph="+id+";";

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

if(cur.moveToFirst() == true)

{

do {

Graph graph = new Graph();

Node n = new Node(graph);

//n.id = cur.getInt(0);

//n.txt = cur.getString(1);

n.TimeStamp = graph.TimeStamp;

n.Set\_API\_ID(cur.getInt(0));

n.SetName(cur.getString(1));

n.X = cur.getFloat(2);

n.Y = cur.getFloat(3);

lst.add(n);

}

while(cur.moveToNext() == true);

}

return lst;

}

public ArrayList<Link> GetListLinks(Graph graph)

{

ArrayList<Link> lst = new ArrayList<>();

for(int i = 0; i< graph.NodeCount(); i++)

{

lst.addAll(GetListLinks(graph.GetNode(i)));

}

return lst;

}

public ArrayList<Link> GetListLinks(int graph)

{

Graph graph1 = GetGraph(graph);

GetListNodes(graph1);

return GetListLinks(graph1);

}

public ArrayList<Link> GetListLinks(Node node)

{

ArrayList<Link> lst = new ArrayList<>();

int id = node.Get\_API\_ID();

SQLiteDatabase db = getReadableDatabase();

Graph graph = node.GetGraph();

String sql = "SELECT id, target, text, value, textVisible, valueVisible, orientatin FROM link where source="+id+";";

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

if(cur.moveToFirst() == true)

{

do {

Link n = new Link(graph);

//n.id = cur.getInt(0);

//n.txt = cur.getString(1);

n.TimeStamp = graph.TimeStamp;

n.Set\_API\_ID(cur.getInt(0));

int source = node.ID();

int target = cur.getInt(1);

target = graph.IdNodeFromApi(target);

n.SetNodes(source, target);

n.Orientation = cur.getInt(6) == 1;

n.Text = cur.getString(2);

n.Value = cur.getFloat(3);

n.TextVisible = cur.getInt(4) == 1;

n.ValueVisible = cur.getInt(5) == 1;

lst.add(n);

graph.AddLink(n);

}

while(cur.moveToNext() == true);

}

return lst;

}

public Graph GetGraph(int id)

{

ArrayList<Graph> lst = new ArrayList<>();

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT id, name, timestamp FROM graph where id="+id+";";

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

if(cur.moveToFirst() == true)

{

do {

Graph n = new Graph();

//n.id = cur.getInt(0);

//n.txt = cur.getString(1);

n.Set\_API\_ID(cur.getInt(0));

n.SetName(cur.getString(1));

n.TimeStamp = cur.getString(2);

lst.add(n);

}

while(cur.moveToNext() == true);

}

return lst.get(0);

}

}

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

package com.example.lab3\_lab4\_graphbuilder\_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", GrapsParams.DomainUrl);

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

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

}

catch (Exception ex)

{

ContentValues context = new ContentValues();

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

context.put("port", GrapsParams.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) {

GrapsParams.DomainUrl = cur.getString(0);

GrapsParams.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 ключей сессий**

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.content.ContentValues;

import androidx.activity.result.contract.ActivityResultContracts;

import androidx.annotation.Nullable;

public class DB\_Sessions extends SQLiteOpenHelper {

public DB\_Sessions(@Nullable Activity context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {

super(context, name, factory, version);

ctx = context;

}

public DB\_Sessions(Activity context)

{

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

}

Activity ctx;

public Activity GetActivity()

{

return ctx;

}

@Override

public void onCreate(SQLiteDatabase sqLiteDatabase) {

SQLiteDatabase db = sqLiteDatabase;

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

" "+sessionColumn+" VARCHAR(255) NOT NULL\n" +

");";

db.execSQL(urlTable);

UpdateSession(db);

}

@Override

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

}

String sessionTable = "sessionTable";

String sessionColumn = "session";

public int Count()

{

try {

SQLiteDatabase db = getReadableDatabase();

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

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 UpdateSession(SQLiteDatabase db)

{

try {

if(Count() < 1)

throw new Exception();

ContentValues context = new ContentValues();

context.put(sessionColumn, GrapsParams.GetSession(ctx));

db.update(sessionTable, context, null, null);

}

catch (Exception ex)

{

ContentValues context = new ContentValues();

context.put(sessionColumn, GrapsParams.GetSession(ctx));

db.insert(sessionTable, null, context);

}

GetSession();

}

public void UpdateSession()

{

UpdateSession(getWritableDatabase());

}

public void GetSession()

{

try {

try {

SQLiteDatabase db = getReadableDatabase();

String sql = "SELECT "+sessionColumn+" FROM "+sessionTable+";";

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

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

GrapsParams.OpenSession( cur.getString(0));

cur.close();

}

} catch (Exception ex) {

UpdateSession();

}

}

catch(Exception ex)

{

}

}

public static DB\_Sessions GetDB(Activity context)

{

return new DB\_Sessions(context);

}

public static String GetSession(Activity ctx)

{

GetDB(ctx).GetSession();

return GrapsParams.GetSession(ctx);

}

public static String SetSession(Activity ctx)

{

DB\_Sessions db = GetDB(ctx);

db.UpdateSession();

return GetSession(ctx);

}

}

Приложение 4. Программный код начального окна

package com.example.lab3\_lab4\_graphbuilder\_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 url;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

exit = findViewById(R.id.ExitButton);

url = findViewById(R.id.textUrl);

url.setText(GrapsParams.GetUrl(this));

exit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Exit\_Click(view);

}

});

GrapsParams.DB = DB\_Graphs.CreateDB(this, "graps.db");

GrapsParams.graphs = new GraphElement\_List(GrapsParams.DB.GetListGraphs());

}

public void ApiAddress\_onClick(View v)

{

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

startActivityForResult(i, 100);

}

public void GoAPI\_Click(View v)

{

GraphListAPI(v);

}

@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

}

});

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();

}

public void GraphCreate(View v)

{

GrapsParams.API=false;

Intent i = new Intent(this, GraphEdit2.class);

startActivity(i);

}

public void GraphListAPI(View v)

{

GrapsParams.API = true;

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

startActivityForResult(i, 100);

}

public void GraphList(View v)

{

GrapsParams.graphList = new GraphElement\_List(GrapsParams.graphs);

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

startActivityForResult(i, 100);

}

@Override

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

url.setText(GrapsParams.Url(this));

GrapsParams.API = false;

GrapsParams.Registration = false;

super.onActivityResult(requestCode, resultCode, data);

}

}

Приложение 5. Программный код редактора графов

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import androidx.activity.result.contract.ActivityResultContracts;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import androidx.constraintlayout.widget.ConstraintSet;

import android.app.Activity;

import android.content.Context;

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.view.ViewGroup;

import android.widget.Button;

import android.widget.LinearLayout;

import android.widget.TextView;

public class GraphEdit2 extends AppCompatActivity {

Graph graphCopy;

Button exit;

GraphView graphs;

LinearLayout panelGraphs;

TextView GraphNameView;

Boolean run = true, run1 = true, run2 = true;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_graph\_edit2);

exit = findViewById(R.id.ButtonBack1);

exit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Exit\_Click(view);

}

});

GraphNameView = findViewById(R.id.GraphNameView);

//graphs = findViewById(R.id.GraphsPanel);

graphs = new GraphView(this)

{

@Override

public void NameView() {

GraphNameView.setText(GetName());

}

@Override

public void Save() {

if(!GrapsParams.API) {

GrapsParams.DB.upload\_graph(GetGraph());

GrapsParams.graphs = new GraphElement\_List(GrapsParams.DB.GetListGraphs());

GrapsParams.graphList = new GraphElement\_List(GrapsParams.graphs);

}

}

@Override

public void BeforeEditNode(GraphElement n) {

graphCopy = graphs.GetGraph().CopyElement().Graph();

run = false;

}

@Override

public void AfterEditNode(GraphElement element, String method) {

if(!GrapsParams.API)

{

run = true;

return;

}

if(method == "select")

{

if(SelectedNowNode() < 0 && SelectedNowLink() < 0)

run = true;

else

run = false;

return;

}

ApiHelper helper = new ApiHelper(GetActivity())

{

@Override

public void send(String req, String payload) {

super.send(req, payload);

}

};

String url = "";

String keys = "";

Boolean inserting = method == "insert" || method.equals("insert");

if(!inserting) {

if (element.IsLink()) {

Link link = element.Link();

Link n = link;

url = GrapsParams.GetUrl(GetActivity()) + "link/" + method;

helper.Method = "POST";

if (method == "update") {

keys = "token=" + GrapsParams.Session + "&id=" + n.IDinAPI + "&value=" + n.ValueWithCheck();

} else if (method == "delete") {

keys = "token=" + GrapsParams.Session + "&id=" + n.IDinAPI;

}

} else {

Node n = element.Node();

url = GrapsParams.GetUrl(GetActivity()) + "node/" + method;

helper.Method = "POST";

if (method == "update") {

keys = "token=" + GrapsParams.Session + "&id=" + n.IDinAPI + "&x=" + n.X + "&y=" + n.Y + "&name=" + n.GetName();

} else if (method == "delete") {

keys = "token=" + GrapsParams.Session + "&id=" + n.IDinAPI;

}

}

}

if(inserting)

{

if(element.IsNode()) {

CreateNode.NodeCreate(GetActivity(), this.GetGraph(), element.Node());

}

else

{

CreateLink.LinkCreate(GetActivity(), GetGraph(), element.Link());

}

}

else {

helper.SendStop(url, keys);

if (!helper.Ready) {

helper.Method = "DELETE";

helper.SendStop(url, keys);

if (!helper.Ready) {

GrapsParams.NowGraph = graphCopy.CopyElement().Graph();

graphs.SetGraph(GrapsParams.NowGraph);

graphs.invalidate();

}

}

}

if(SelectedNowNode() < 0 && SelectedNowLink() < 0)

run = true;

else

run = false;

}

};

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, LinearLayout.LayoutParams.MATCH\_PARENT);

panelGraphs = findViewById(R.id.GraphPanel);

panelGraphs.addView(graphs);

graphs.setLayoutParams(params);

if(GrapsParams.Run\_Graph)

{

graphs.SetGraph(GrapsParams.NowGraph);

GrapsParams.Run\_Graph = false;

}

Runnable runnable = new Runnable() {

@Override

public void run() {

upload\_graph();

}

};

Thread thread = new Thread(runnable);

if(GrapsParams.API)

thread.start();

}

void upload\_graph()

{

while (run1)

{

if(run)

{

try {

Graph graph = graphs.GetGraph();

graphs.GetGraph().ClearNodes();

graphCopy = graph.CopyElement().Graph();

GraphElement\_List list = GraphListHelper.GetGraphs(GetActivity());

int api = graph.IDinAPI;

Graph graph1 = list.ElementFromAPI(api).Graph();

GrapsParams.CreateGraph(GetActivity(), graph1);

graph = graph1.CopyElement().GetGraph();

Graph finalGraph = graph;

GetActivity().runOnUiThread(() ->

{

try {

graphs.SetGraph(finalGraph);

} catch (Exception ex) {

}

});

}

catch (Exception ex)

{

GetActivity().runOnUiThread(() ->

{

try {

graphs.SetGraph(graphCopy);

graphs.invalidate();

} catch (Exception ex1) {

}

});

}

try {

Thread.sleep(4000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

else

{

}

}

}

@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)

{

GrapsParams.NowGraph = graphs.GetGraph();

run1 = false;

finish();

}

public void EditNode(View v)

{

if(!graphs.Selection())

return;

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

GrapsParams.GraphElement = graphs.GetSelected();

if(GrapsParams.API) {

GrapsParams.GraphElement = GrapsParams.GraphElement.CopyElement();

}

startActivityForResult(i, 100);

}

public void AddNode(View v)

{

graphs.AddNode();

}

public void DeleteNode(View v)

{

graphs.Delete();

}

public void SetLink(View v) {graphs.SetLink(false);}

public void SetOrientationLink(View v) {graphs.SetLink(true);}

public Activity GetActivity()

{

return this;

}

@Override

public void finish() {

run = false;

run1 = false;

run2 = false;

super.finish();

}

@Override

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

GrapsParams.Run\_Graph = false;

GrapsParams.ElementName = GraphElementName.Graph;

if (GrapsParams.API && !GrapsParams.HaveSession(GetActivity())) {

Intent intent1 = getIntent();

setResult(500, intent1);

run1 = false;

finish();

return;

}

//run = true;

if(requestCode==554 || resultCode == 554)

{

graphs.invalidate();

run = run2;

}

else if (requestCode==555 || resultCode == 555 || requestCode==550 || resultCode == 550) // Проверяем код результата (2-ая Activity была запущена с кодом 555)

{

if (data != null) // Вернула ли значение вторая Activity нам Intent с данными, или, просто, закрылась

{

GraphElement element = GrapsParams.GraphElement;

if(element.IsNode() || element.IsLink())

graphs.SetGraphElement(element);

else if(element.IsGraph()) {

graphs.SetGraph(element.ToGraph());

if(!GrapsParams.API) {

int api = element.Get\_API\_ID();

if (api > -1) {

GrapsParams.DB.upload\_graph(element.Graph());

} else {

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

}

});

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

@Override

public void onClick(DialogInterface dialog, int which) {

GrapsParams.DB.upload\_graph(element.Graph());

}

});

AlertDialog dlg = bld.create();

dlg.setTitle("Сохранять граф?");

dlg.setMessage("Сохранять граф?");

dlg.show();

}

}

else

{

String url = GrapsParams.GetUrl(this)+"graph/update";

ApiHelper helper = new ApiHelper(this)

{

@Override

public void send(String req, String payload) {

Method = "POST";

super.send(req, payload);

}

};

helper.send(url,"token="+GrapsParams.Session+"&graph="+element.IDinAPI+"&name="+element.GetName());

try {

helper.th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

if(!helper.Ready)

{

helper.send(url,"token="+GrapsParams.Session+"&id="+element.IDinAPI+"&name="+element.GetName());

try {

helper.th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

}

else if(requestCode==556|| resultCode == 556)

{

if(data != null)

{

int id = GrapsParams.elementID();

if(GrapsParams.GraphElement.IsNode())

{

if(!GrapsParams.API)

graphs.DeleteNode(id);

else

graphs.DeleteNode();

}

else if(GrapsParams.GraphElement.IsLink())

{

if(!GrapsParams.API)

graphs.DeleteLink(id);

else

graphs.DeleteLink();

}

else

{

Intent intent = getIntent();

setResult(556, intent);

run1 = false;

finish();

}

}

}

graphs.invalidate();

super.onActivityResult(requestCode,resultCode,data);

}

public void ChangeOrientationLink(View v)

{

graphs.ChangeOrientationLink();

}

public Context getActivity()

{

return this;

}

public void List\_Click(View v)

{

run2 = run;

run = false;

GrapsParams.GraphAPI = graphs.GetGraph();

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

GrapsParams.GraphElement = graphs.GetGraph();

bld.setPositiveButton("Узлы графа", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

GrapsParams.graphList = new GraphElement\_List(graphs.GetGraph(), GraphElementName.Node);

GrapsParams.ElementName = GraphElementName.Node;

StartList(v);

}

});

bld.setNegativeButton("Рёбра (связи) графа", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

GrapsParams.graphList = new GraphElement\_List(graphs.GetGraph(), GraphElementName.Link);

GrapsParams.ElementName = GraphElementName.Link;

StartList(v);

}

});

bld.setCancelable(true);

bld.setOnCancelListener(new DialogInterface.OnCancelListener() {

@Override

public void onCancel(DialogInterface dialogInterface) {

run = run2;

}

});

AlertDialog dlg = bld.create();

dlg.setTitle("Список элементов графа");

dlg.setMessage("Список элементов графа");

dlg.show();

}

public void StartList(View v)

{

GrapsParams.Run\_Graph = true;

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

GrapsParams.GraphElement = graphs.GetSelected();

startActivityForResult(i, 100);

}

public void GraphProperty(View v)

{

GrapsParams.Run\_Graph = true;

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

GrapsParams.GraphElement = graphs.GetGraph();

startActivityForResult(i, 100);

}

@Override

public void startActivityForResult(Intent intent, int requestCode) {

run = false;

super.startActivityForResult(intent, requestCode);

}

}

Приложение 6. Свойства объектов графа

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;

import android.content.Intent;

import android.os.Bundle;

import android.text.Editable;

import android.text.TextWatcher;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.CheckBox;

import android.widget.Checkable;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.TextView;

import java.time.LocalDateTime;

public class GraphElementEdit extends AppCompatActivity {

Button exit, buttonNameID, ChangeOrientation;

TextView id, nameLabel, dateTime, elementType;

EditText nameEdit;

GraphElement graphElement;

LinearLayout nameLayout, nameEditLayout, attributesPanel;

LinearLayout xyPanel, stPanel, mainPanel, OrientationPanel;

LayoutPoleInput xPole, yPole;

CheckBox OrientationGraph;

TextVisibleView LinkText, LinkValue;

Button copy, past, toGraph;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_graph\_element\_edit);

mainPanel = findViewById(R.id.MainPanel);

exit = findViewById(R.id.cancelButton);

exit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Exit\_Click(view);

}

});

graphElement = GrapsParams.GraphElement;

elementType = findViewById(R.id.ElementType);

elementType.setText(graphElement.TypeText());

id = findViewById(R.id.TextElementID);

id.setText(String.valueOf(graphElement.ID()));

nameLayout = findViewById(R.id.NameLayout);

nameLabel = findViewById(R.id.NameLabel);

try{

nameLabel.setText(graphElement.GetName());

}

catch (Exception ex) {

}

nameEditLayout = new LinearLayout(this);

nameEditLayout.setOrientation(LinearLayout.VERTICAL);

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, LinearLayout.LayoutParams.WRAP\_CONTENT, 1);

LinearLayout.LayoutParams params1 = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.WRAP\_CONTENT, LinearLayout.LayoutParams.WRAP\_CONTENT, 1);

//nameLayout.addView(nameEditLayout);

nameEditLayout.setLayoutParams(params);

//params = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, LinearLayout.LayoutParams.MATCH\_PARENT);

nameEdit = new androidx.appcompat.widget.AppCompatEditText(this);

nameEdit.setLayoutParams(params);

nameEdit.setText(nameLabel.getText().toString());

nameEdit.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

String name = nameEdit.getText().toString();

nameLabel.setText(name);

}

@Override

public void afterTextChanged(Editable editable) {

}

});

nameEditLayout.addView(nameEdit);

buttonNameID = new Button(this);

buttonNameID.setLayoutParams(params);

buttonNameID.setText("Имя из ID");

buttonNameID.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

nameEdit.setText(graphElement.GetNameFromID());

}

});

nameEditLayout.addView(buttonNameID);

if(graphElement.IsNode() || graphElement.IsGraph()) {

nameLayout.addView(nameEditLayout);

}

attributesPanel = findViewById(R.id.AttributesPanel);

xyPanel = new LinearLayout(this);

stPanel = new LinearLayout(this);

xyPanel.setLayoutParams(params);

stPanel.setLayoutParams(params);

xPole = new LayoutPoleInput(this);

xPole.setLayoutParams(params1);

xyPanel.addView(xPole);

yPole = new LayoutPoleInput(this);

yPole.setLayoutParams(params1);

xyPanel.addView(yPole);

OrientationPanel = new LinearLayout(this);

OrientationPanel.setLayoutParams(params);

OrientationGraph = new CheckBox(this);

OrientationGraph.setLayoutParams(params1);

OrientationGraph.setText("Ориентированное ребро");

OrientationPanel.addView(OrientationGraph);

ChangeOrientation = new Button(this);

ChangeOrientation.setLayoutParams(params1);

ChangeOrientation.setText("Сменить направление");

ChangeOrientation.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

String source = xPole.InputPole().getText().toString();

String target = yPole.InputPole().getText().toString();

String change = source;

source = target;

target = change;

xPole.InputPole().setText(source);

yPole.InputPole().setText(target);

}

});

OrientationPanel.addView(ChangeOrientation);

LinkText = new TextVisibleView(this);

LinkValue = new TextVisibleView(this);

if(graphElement.IsNode())

{

attributesPanel.addView(xyPanel);

Node n = graphElement.Node();

xPole.SignaturePole().setText("X: ");

xPole.InputPole().setText(String.valueOf(n.X));

yPole.SignaturePole().setText("Y: ");

yPole.InputPole().setText(String.valueOf(n.Y));

}

else if(graphElement.IsLink()){

attributesPanel.addView(xyPanel);

Link n = graphElement.Link();

xPole.SignaturePole().setText("Source: ");

try {

nameEdit.setText(nameLabel.getText().toString());

xPole.InputPole().setText(String.valueOf(n.sourceID));

}

catch (Exception ex) {

}

yPole.SignaturePole().setText("Target: ");

try{

yPole.InputPole().setText(String.valueOf(n.targetID));

}

catch (Exception ex) {

}

mainPanel.addView(OrientationPanel);

OrientationGraph.setChecked(n.Orientation);

try{

LinkText.SetText(n.GetText());

LinkText.SetTextVisible(n.TextVisible);

}

catch (Exception ex) {

}

mainPanel.addView(LinkText);

try{

LinkValue.SetText(n.GetTextValue());

LinkValue.SetTextVisible(n.ValueVisible);

}

catch (Exception ex) {

}

mainPanel.addView(LinkValue);

}

else if(graphElement.IsGraph())

{

}

//UpdateElement();

dateTime = findViewById(R.id.DateTimeText);

dateTime.setText(graphElement.TimeStamp);

copy = new Button(this);

past = new Button(this);

copy.setText("Копировать");

past.setText("Вставить");

copy.setLayoutParams(params);

past.setLayoutParams(params);

copy.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

CopyElement(view);

}

});

past.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

PastElement(view);

}

});

Boolean node = GrapsParams.API && GrapsParams.GraphElement.IsNode();

Boolean link = GrapsParams.API && GrapsParams.GraphElement.IsLink();

Boolean past1 = node || link;

past1 = !past1;

if(past1)

{

mainPanel.addView(copy);

mainPanel.addView(past);

}

toGraph = new Button(this);

toGraph.setText("К графу");

toGraph.setLayoutParams(params);

//mainPanel.addView(toGraph);

toGraph.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

ToGraph(view);

}

});

if(!GrapsParams.GraphElement.IsGraph())

return;

GrapsParams.NowGraph = GrapsParams.GraphElement.Graph();

Button toAPI = new Button(this);

mainPanel.addView(toAPI);

toAPI.setLayoutParams(params);

if(!GrapsParams.API) {

toAPI.setText("Сохранить в API");

toAPI.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

GrapsParams.NowGraph.SetName(nameEdit.getText().toString());

Intent i = new Intent(GetContent(), AutorizationActivity.class);

startActivityForResult(i, 100);

}

});

}

else

{

toAPI.setText("Сохранить на устройство");

toAPI.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

GrapsParams.DB.upload\_graph(GrapsParams.GraphElement.Graph());

AlertDialog.Builder builder = new AlertDialog.Builder(GetContent());

AlertDialog dialog = builder.create();

dialog.setMessage("Граф успешно сохранён");

dialog.show();

GrapsParams.GraphElement.Set\_API\_ID(-1);

}

});

}

}

public Context GetContent()

{

return this;

}

public void ToGraph (View v)

{

if(GrapsParams.Run\_Graph)

finish();

else {

Graph graph = new Graph();

if (GrapsParams.GraphElement.IsGraph()) {

graph = GrapsParams.GraphElement.Graph();

}

else

{

graph = GrapsParams.GraphElement.GetGraph();

}

GrapsParams.NowGraph = graph;

Intent i = new Intent(this, GraphEdit2.class);

GrapsParams.Run\_Graph = true;

startActivity(i);

}

}

public void UpdateElement()

{

try{

nameLabel.setText(graphElement.GetName());

}

catch (Exception ex) {

}

try {

if (graphElement.IsNode()) {

nameEdit.setText(nameLabel.getText().toString());

//attributesPanel.addView(xyPanel);

Node n = graphElement.Node();

xPole.SignaturePole().setText("X: ");

xPole.InputPole().setText(String.valueOf(n.X));

yPole.SignaturePole().setText("Y: ");

yPole.InputPole().setText(String.valueOf(n.Y));

} else if (graphElement.IsLink()) {

//attributesPanel.addView(xyPanel);

Link n = graphElement.Link();

xPole.SignaturePole().setText("Source: ");

try {

xPole.InputPole().setText(String.valueOf(n.sourceID));

} catch (Exception ex) {

}

yPole.SignaturePole().setText("Target: ");

try {

yPole.InputPole().setText(String.valueOf(n.targetID));

} catch (Exception ex) {

}

//mainPanel.addView(OrientationPanel);

OrientationGraph.setChecked(n.Orientation);

try {

LinkText.SetText(n.GetText());

LinkText.SetTextVisible(n.TextVisible);

} catch (Exception ex) {

}

//mainPanel.addView(LinkText);

try {

LinkValue.SetText(n.GetTextValue());

LinkValue.SetTextVisible(n.ValueVisible);

} catch (Exception ex) {

}

//mainPanel.addView(LinkValue);

} else if (graphElement.IsGraph()) {

nameEdit.setText(nameLabel.getText().toString());

if(GrapsParams.API)

{

CreateGraph.PastGraph(this, graphElement.Graph());

}

}

}

catch(Exception ex)

{

}

}

public void Save(View v)

{

String name = nameEdit.getText().toString();

nameLabel.setText(name);

if(graphElement.IsNode()) {

Node node = graphElement.Node();

node.SetName(name);

try {

node.X = Float.valueOf(xPole.InputPole().getText().toString());

node.Y = Float.valueOf(yPole.InputPole().getText().toString());

}

catch (Exception ex)

{

return;

}

}

else if (graphElement.IsLink())

{

Link node = graphElement.Link();

try {

int source = Integer.valueOf(xPole.InputPole().getText().toString());

int target = Integer.valueOf(yPole.InputPole().getText().toString());

node.SetNodes(source, target);

node.Orientation = OrientationGraph.isChecked();

node.TextVisible = LinkText.IsTextVisible();

node.SetText(LinkText.GetText());

node.ValueVisible = LinkValue.IsTextVisible();

node.SetValue(LinkValue.GetText());

nameEdit.setText(nameLabel.getText().toString());

}

catch (Exception ex)

{

return;

}

}

else if(graphElement.IsGraph())

{

Graph node = graphElement.Graph();

node.SetName(name);

Graph graph = node;

//if(node.Get\_API\_ID() > -1)

//GrapsParams.DB.upload\_graph(graph);

}

GrapsParams.GraphElement = graphElement;

Intent intent = getIntent();

setResult(555, intent);

finish();

}

public void Delete(View v)

{

String name = nameEdit.getText().toString();

nameLabel.setText(name);

if(graphElement.IsNode()) {

Node node = graphElement.Node();

node.SetName(name);

}

GrapsParams.GraphElement = graphElement;

Intent intent = getIntent();

setResult(556, intent);

finish();

}

@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)

{

finish();

}

public void PastElement(View v)

{

try {

int id = graphElement.Get\_API\_ID();

if (graphElement.EqualsTypes(GrapsParams.GraphCopy)) {

graphElement = GrapsParams.GraphCopy.CopyElement();

graphElement.Set\_API\_ID(id);

UpdateElement();

}

}

catch (Exception ex)

{

}

}

public void CopyElement(View v)

{

GrapsParams.GraphCopy = graphElement.CopyElement();

}

}

Приложение 7. Список объектов графа и список сессий

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import androidx.activity.result.contract.ActivityResultContracts;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import android.app.Activity;

import android.content.Context;

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.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.ListView;

import android.widget.TextView;

import java.util.ArrayList;

public class GraphElementsListActivity extends AppCompatActivity {

Button exit;

Graph graph;

TextView typeText;

GraphElement\_List graphs;

ListView elementList;

Boolean run1 = true, run2 = true;

Boolean add;

ArrayList<Session> Sessions;

ArrayAdapter<Session> SessionsAdapter;

Button AddElement;

public Graph Graph()

{

return graphs.GetGraph();

}

Thread thread;

void update\_API()

{

//for(int i = 0; i < 20; i++)

while (run1)

{

final Boolean[] run = {true};

while (run[0]) {

try {

try {

if(!run1)

break;

}

catch(Exception ex)

{

}

if(run2) {

GetActivity().runOnUiThread(() ->

{

if(run2) {

try {

update\_graphs();

} catch (Exception ex) {

run[0] = false;

}

}

});

try {

Thread.sleep(4000);

} catch (InterruptedException e) {

e.printStackTrace();

}

run[0] = run1;

if(add && run2) {

add = false;

}

}

} catch (Exception ex) {

break;

}

}

}

try {

GetActivity().runOnUiThread(() ->

{

try {

if (!GrapsParams.HaveSession(GetActivity())) {

Intent intent = getIntent();

setResult(500, intent);

finish();

return;

}

}

catch(Exception ex)

{

}

});

}

catch (Exception ex)

{

}

}

Activity GetActivity()

{

return this;

}

ArrayAdapter<GraphElement> list;

Boolean node, graph1, link;

void update\_graphs() {

if(!GrapsParams.SessionsList) {

GraphElement\_List graphs1 = GrapsParams.graphList;

GrapsParams.graphs.clear();

GrapsParams.graphList.clear();

if (!GrapsParams.API) {

if (graphs.IsGraph() || graph1)

GrapsParams.graphs = new GraphElement\_List(GrapsParams.DB.GetListGraphs());

} else {

if (graph1)

GrapsParams.graphs = GraphListHelper.GetGraphs(this);

else {

if (node) {

GrapsParams.graphs = ListNodesHelper.GetNodes(this, GrapsParams.GraphAPI);

} else if (link) {

GrapsParams.graphs = ListLinksHelper.GetLinks(this, GrapsParams.GraphAPI);

}

}

}

GrapsParams.graphList = new GraphElement\_List(GrapsParams.graphs);

graphs.clear();

if (graph1 || !GrapsParams.API)

graphs.addListGraphs(GrapsParams.graphList);

else {

graphs.AddElements(GrapsParams.graphList);

}

}

else

{

Sessions.clear();

Sessions.addAll(ListSessionsHelper.GetSessions(this));

}

update\_list();

}

void update\_list()

{

if(!GrapsParams.SessionsList) {

if (!graphs.IsGraph() && !graph1) {

if (!GrapsParams.API)

graphs.SetGraph(graph);

else {

}

}

list.notifyDataSetChanged();

}

else

{

SessionsAdapter.notifyDataSetChanged();

}

}

public Context GetContext()

{

return this;

}

@Override

protected void onCreate(Bundle savedInstanceState) {super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_graph\_elements\_list);

Boolean graphYes = GrapsParams.graphs.IsGraph();

AddElement = findViewById(R.id.AddElement);

add = false;

if(GrapsParams.API && !graphYes)

{

// GrapsParams.GraphAPI = GrapsParams.GraphElement.GetGraph();

}

if(GrapsParams.API)

{

if(!GrapsParams.HaveSession(this))

{

Intent intent = getIntent();

setResult(500, intent);

finish();

return;

}

}

if(GrapsParams.SessionsList)

{

AddElement.setVisibility(View.GONE);

}

Runnable run = new Runnable() {

@Override

public void run() {

update\_API();

}

};

thread = new Thread(run);

if(!GrapsParams.SessionsList) {

graphs = GrapsParams.graphList;

try {

node = GrapsParams.ElementName == GraphElementName.Node;

link = GrapsParams.ElementName == GraphElementName.Link;

graph1 = GrapsParams.ElementName == GraphElementName.Graph;

if (node == graph1 && link == graph1) {

throw new Exception();

}

if (node == graph1 && node == true) {

throw new Exception();

}

if (link == graph1 && link == true) {

throw new Exception();

}

if (node == link) {

throw new Exception();

}

} catch (Exception ex) {

node = false;

link = false;

graph1 = true;

}

graph = graphs.GetGraph();

}

exit = findViewById(R.id.CloseEditorElements);

exit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Exit\_Click(view);

}

});

typeText = findViewById(R.id.TypeText);

if(!GrapsParams.SessionsList) {

typeText.setText(graphs.GetName());

}

else

{

typeText.setText("Сессии (Текущая сессия - "+GrapsParams.Session+")");

}

elementList = findViewById(R.id.listElements1);

if(!GrapsParams.SessionsList) {

list = new ArrayAdapter<GraphElement>(this, android.R.layout.simple\_list\_item\_1, graphs);

elementList.setAdapter(list);

elementList.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

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

run2 = false;

Intent intent = new Intent(GetContext(), GraphElementEdit.class);

GrapsParams.GraphElement = graphs.get(i);

if (GrapsParams.API) {

if (!GrapsParams.HaveSession(GetActivity())) {

Intent intent1 = getIntent();

setResult(500, intent1);

finish();

return;

}

//thread.stop();

if (GrapsParams.GraphElement.IsGraph()) {

Graph graph = GrapsParams.GraphElement.Graph();

GrapsParams.CreateGraph(GetActivity(), graph);

GrapsParams.NowGraph = graph;

GrapsParams.Run\_Graph = true;

Intent intent1 = new Intent(GetContext(), GraphEdit2.class);

startActivityForResult(intent1, 100);

return;

}

if (GrapsParams.GraphElement.IsLink()) {

Link link = GrapsParams.GraphElement.Link();

source = link.IDsourceAPI();

target = link.IDtargetAPI();

}

startActivityForResult(intent, 100);

return;

} else if (GrapsParams.GraphElement.IsGraph()) {

Graph graph = GrapsParams.GraphElement.Graph();

GrapsParams.NowGraph = graph;

GrapsParams.Run\_Graph = true;

graph.ClearNodes();

GrapsParams.DB.GetListNodes(graph);

GrapsParams.DB.GetListLinks(graph);

Intent intent1 = new Intent(GetContext(), GraphEdit2.class);

startActivityForResult(intent1, 100);

return;

}

startActivityForResult(intent, 100);

}

});

if (GrapsParams.graphList.IsGraph())

update\_graphs();

else

update\_list();

}

else

{

Sessions = new ArrayList<>();

SessionsAdapter = new ArrayAdapter<Session>(this, android.R.layout.simple\_list\_item\_1, Sessions);

elementList.setAdapter(SessionsAdapter);

elementList.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

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

if (!GrapsParams.HaveSession(GetActivity())) {

Intent intent1 = getIntent();

setResult(500, intent1);

finish();

return;

}

Session session = Sessions.get(i);

SessionView(session);

}

});

}

if(GrapsParams.API)

thread.start();

}

public void SessionView(Session session)

{

run2 = false;

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

builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

run2 = true;

}

});

builder.setNegativeButton("Закрыть", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

if(!GrapsParams.HaveSession(GetActivity()))

{

Intent intent = getIntent();

setResult(500, intent);

finish();

return;

}

CloseSession close = new CloseSession(GetActivity(), session.Key);

close.SendStop();

if(!close.Ready)

{

Intent intent = getIntent();

setResult(500, intent);

finish();

return;

}

else

{

SessionCloseView(session);

}

}

});

AlertDialog dialog = builder.create();

dialog.setMessage(" Сессия - "+session.Key+"\n Дата открытия - "+session.GetDatetime());

dialog.setCancelable(false);

dialog.show();

}

void SessionCloseView(Session session)

{

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

builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

if(!GrapsParams.HaveSession(GetActivity()))

{

Intent intent = getIntent();

setResult(500, intent);

finish();

return;

}

else {

run2 = true;

}

}

});

AlertDialog dialog = builder.create();

dialog.setMessage(" Сессия - "+session.Key+" успешно закрыта");

dialog.setCancelable(false);

dialog.show();

}

int source, target;

@Override

public void finish() {

try {

NoRun();

}

catch (Exception ex)

{

}

super.finish();

}

@Override

public void startActivityForResult(Intent intent, int requestCode) {

run2 = false;

super.startActivityForResult(intent, requestCode);

}

@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)

{

try {

thread.stop();

}

catch(Exception ex)

{

}

Intent intent = getIntent();

setResult(554, intent);

finish();

}

void NoRun()

{

if(!GrapsParams.SessionsList) {

GrapsParams.ElementName = GraphElementName.Graph;

}

run1 = false;

run2 = false;

GrapsParams.SessionsList = false;

}

@Override

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

if (GrapsParams.API && !GrapsParams.HaveSession(GetActivity())) {

Intent intent1 = getIntent();

setResult(500, intent1);

finish();

return;

}

if (requestCode==555 || resultCode == 555 || requestCode==550 || resultCode == 550) // Проверяем код результата (2-ая Activity была запущена с кодом 555)

{

if (data != null && !GrapsParams.API) // Вернула ли значение вторая Activity нам Intent с данными, или, просто, закрылась

{

int id = GrapsParams.elementID();

if(GrapsParams.graphList.IsNode())

{

if(id < 0) {

id = graph.AddNode().id();

}

graph.SetNode(id, GrapsParams.GraphElement.Node());

}

else if (GrapsParams.graphList.IsLink())

{

if(id < 0) {

try {

Link link1 = GrapsParams.GraphElement.Link();

Link link = graph.AddLink(link1);

//link.Orientation = link1.Orientation;

//link.Value = link1.Value;

//link.Text = link1.Text;

//link.ValueVisible = link1.ValueVisible;

//link.TextVisible = link1.TextVisible;

}

catch(Exception ex)

{

}

}

else

{

try {

Link link1 = GrapsParams.GraphElement.Link();

Link link = graph.SetLink(id, link1);

//link.Orientation = link1.Orientation;

//link.Value = link1.Value;

//link.Text = link1.Text;

//link.ValueVisible = link1.ValueVisible;

//link.TextVisible = link1.TextVisible;

}

catch(Exception ex)

{

}

}

}

else if(GrapsParams.graphList.IsGraph())

{

Graph graph = GrapsParams.GraphElement.Graph();

GrapsParams.DB.upload\_graph(graph);

update\_graphs();

return;

}

}

else if(GrapsParams.API)

{

if(GrapsParams.GraphElement.IsNode()) {

Node n = GrapsParams.GraphElement.Node();

if(add)

{

CreateNode.NodeCreate(this, GrapsParams.GraphAPI, n);

run2 = true;

return;

}

String url = GrapsParams.GetUrl(this) + "node/update";

ApiHelper helper = new ApiHelper(this) {

@Override

public void send(String req, String payload) {

Method = "POST";

super.send(req, payload);

}

};

helper.SendStop(url, "token=" + GrapsParams.Session + "&id=" + n.IDinAPI + "&x=" + n.X + "&y=" + n.Y + "&name=" + n.GetName());

}

else if (GrapsParams.GraphElement.IsLink()) {

String url = GrapsParams.GetUrl(this) + "link/update";

Link n = GrapsParams.GraphElement.Link();

if(add)

{

CreateLink.LinkCreate(this, GrapsParams.GraphAPI, n);

run2 = true;

return;

}

ApiHelper helper = new ApiHelper(this) {

@Override

public void send(String req, String payload) {

//Method = "POST";

super.send(req, payload);

}

};

helper.Method = "POST";

try {

if(n.IDtargetAPI() == target && n.IDsourceAPI() == source)

throw new Exception();

url = GrapsParams.GetUrl(this)+"link/delete";

helper.Method = "DELETE";

helper.SendStop(url,"token=" + GrapsParams.Session + "&id=" + n.IDinAPI);

//"token=" + GrapsParams.Session + "&id=" + n.IDinAPI

//"token=" + GrapsParams.Session + "&id=" + n.IDinAPI

if(helper.Ready)

CreateLink.LinkCreate(GetActivity(), graph, n);

} catch (Exception ex) {

helper.SendStop(url, "token=" + GrapsParams.Session + "&id=" + n.IDinAPI + "&value=" + n.Value);

}

}

run2 = true;

}

}

else if(requestCode==556|| resultCode == 556)

{

try {

if (data != null) {

int id = GrapsParams.elementID();

if (GrapsParams.GraphElement.IsNode()) {

if(!GrapsParams.API)

graph.DeleteNode(id);

else

{

Node n = GrapsParams.GraphElement.Node();

String url = GrapsParams.GetUrl(this)+"node/delete";

ApiHelper helper = new ApiHelper(this)

{

@Override

public void send(String req, String payload) {

Method = "DELETE";

super.send(req, payload);

}

};

helper.SendStop(url,"token="+GrapsParams.Session+"&id="+n.IDinAPI);

run2 = true;

}

} else if(GrapsParams.GraphElement.IsLink()) {

if(!GrapsParams.API)

graph.DeleteLink(id);

else

{

Link n = GrapsParams.GraphElement.Link();

String url = GrapsParams.GetUrl(this)+"link/delete";

ApiHelper helper = new ApiHelper(this)

{

@Override

public void send(String req, String payload) {

Method = "DELETE";

super.send(req, payload);

}

};

helper.SendStop(url,"token="+GrapsParams.Session+"&id="+n.IDinAPI);

run2 = true;

}

}

else if(GrapsParams.GraphElement.IsGraph())

{

Graph graph = GrapsParams.GraphElement.Graph();

if(!GrapsParams.API) {

GrapsParams.DB.delete\_graph(graph);

update\_graphs();

}

else

{

String url = GrapsParams.GetUrl(this)+"graph/delete";

ApiHelper helper = new ApiHelper(this)

{

@Override

public void send(String req, String payload) {

Method = "DELETE";

super.send(req, payload);

}

};

helper.send(url,"token="+GrapsParams.Session+"&graph="+graph.IDinAPI);

try {

helper.th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

if(!helper.Ready)

{

helper.send(url,"token="+GrapsParams.Session+"&id="+graph.IDinAPI);

try {

helper.th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

run2 = true;

}

return;

}

}

}

catch(Exception ex)

{

}

}

if(GrapsParams.graphList.IsGraph() || graph1) {

if(!GrapsParams.API) {

update\_graphs();

}

else {

run2 = true;

}

}

else

update\_list();

if(GrapsParams.API)

run2 = true;

super.onActivityResult(requestCode,resultCode,data);

}

public void AddElements(View v)

{

run2 = false;

add = true;

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

GrapsParams.GraphElement = graphs.add();

if(GrapsParams.GraphElement.IsGraph())

{

Graph graph = GrapsParams.GraphElement.Graph();

GrapsParams.NowGraph = graph;

GrapsParams.Run\_Graph = true;

graph.ClearNodes();

if(!GrapsParams.API) {

GrapsParams.DB.GetListNodes(graph);

GrapsParams.DB.GetListLinks(graph);

}

else

{

CreateGraph.GraphCreate(this, graph);

}

Intent intent1 =new Intent(GetContext(), GraphEdit2.class);

startActivityForResult(intent1, 100);

return;

}

startActivityForResult(i, 100);

}

public void PropertyGraph(View v)

{

}

}

Приложение 8. Программный код окна редактирования URL-ссылки

package com.example.lab3\_lab4\_graphbuilder\_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.Button;

import android.widget.EditText;

public class UrlActivity extends AppCompatActivity {

Button exit;

EditText domain, port;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_url);

exit = findViewById(R.id.ExitButton);

domain = findViewById(R.id.DomainNameText);

port = findViewById(R.id.PortText);

domain.setText(GrapsParams.DomainUrl);

port.setText(GrapsParams.PortUrl);

}

public void DomainClear\_Click(View v)

{

domain.setText("");

}

public void DomainStart\_Click(View v)

{

domain.setText(GrapsParams.DomainUrl);

}

public void PortStart\_Click(View v)

{

port.setText(GrapsParams.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;

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)

{

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

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

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

Intent i = getIntent();

finish();

}

}

Приложение 9. Программный код окна авторизации, регистрации и смены пароля

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;

import android.text.method.HideReturnsTransformationMethod;

import android.text.method.PasswordTransformationMethod;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.CheckBox;

import android.widget.CompoundButton;

import android.widget.EditText;

import android.widget.LinearLayout;

public class AutorizationActivity extends AppCompatActivity {

Button exit;

CheckBox save, show;

EditText password, logIn;

Button LogIn, Registration;

LinearLayout LoginLayout, LoginInputLayout;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_autorization);

exit = findViewById(R.id.closeAutorization);

LogIn = findViewById(R.id.LogIn);

Registration = findViewById(R.id.Registration);

LoginLayout = findViewById(R.id.LoginLayout);

LoginInputLayout = findViewById(R.id.LoginImputLayout);

if(GrapsParams.ChangePassword)

{

LogIn.setText("Установить пароль");

LoginInputLayout.setVisibility(View.GONE);

Registration.setVisibility(View.GONE);

if(!GrapsParams.HaveSession(this))

{

Intent intent = getIntent();

setResult(500, intent);

finish();

return;

}

}

show = findViewById(R.id.ShowPasswordRegistrate);

password = findViewById(R.id.PasswordRegistrate);

password.setTransformationMethod(PasswordTransformationMethod.getInstance());

logIn = findViewById(R.id.LogInRegistrate);

show.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

@Override

public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked) {

if(!isChecked)

password.setTransformationMethod(PasswordTransformationMethod.getInstance());

else

password.setTransformationMethod(HideReturnsTransformationMethod.getInstance());

}

});

}

Context GetContext()

{

return this;

}

public void SessionOpen(ApiHelper helper)

{

if(!GrapsParams.API)

{

CreateGraph createGraph = new CreateGraph(helper, GrapsParams.NowGraph)

{

@Override

public void Run()

{

GraphOutput(this);

}

};

//GraphOutput(helper);

createGraph.Send();

}

}

public void GraphOutput(ApiHelper helper)

{

CloseSession session = new CloseSession(helper)

{

@Override

public void MessageReadyOutput(String message) {

runOnUiThread(() -> {

finish();

});

}

};

session.Send();

//finish();

}

public void Registrate\_Click(View v)

{

String login = logIn.getText().toString();

String password = this.password.getText().toString();

if(!CheckNullLogin(login))

return;

CreateAccount session = new CreateAccount(this, login, password){

@Override

public void MessageOutput(String message) {

AlertDialog.Builder dialog = new AlertDialog.Builder(GetContext());

AlertDialog dlg = dialog.create();

dlg.setMessage(message);

dlg.show();

}

};

session.Send();

}

Boolean CheckNullLogin(String login)

{

Boolean check = true;

if(login == "" || login.equals("") || login.isEmpty())

{

check = false;

AlertDialog.Builder builder = new AlertDialog.Builder(GetContext());

builder.setPositiveButton("ОК", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

}

});

builder.setTitle("Ошибка");

AlertDialog dialog = builder.create();

dialog.setMessage("Введите, хотя бы, логин \n" +

"Желательно, также и пароль");

dialog.setCancelable(false);

dialog.show();

}

return check;

}

@Override

public void finish() {

GrapsParams.ChangePassword = false;

super.finish();

}

public void save\_onClick(View v)

{

String password = this.password.getText().toString();

if(GrapsParams.ChangePassword)

{

if(!GrapsParams.HaveSession(this)) {

Intent intent = getIntent();

setResult(500, intent);

finish();

return;

}

ApiHelper helper = new ApiHelper(this);

helper.Method = "POST";

helper.SendStop(GrapsParams.GetUrl(this)+"account/update", "token="+GrapsParams.Session+"&secret="+password);

if(!helper.Ready)

{

Intent intent = getIntent();

setResult(500, intent);

finish();

return;

}

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

builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

finish();

}

});

AlertDialog dialog = builder.create();

dialog.setMessage("Пароль успешно изменён");

dialog.setCancelable(false);

dialog.show();

return;

}

String login = logIn.getText().toString();

if(!CheckNullLogin(login))

return;

GetSession session = new GetSession(this, login, password){

@Override

public void MessageReadyOutput(String message) {

if(!GrapsParams.Registration) {

SessionOpen(this);

}

else

{

GrapsParams.OpenSession(session);

finish();

}

}

@Override

public void MessageOutput(String message) {

AlertDialog.Builder dialog = new AlertDialog.Builder(GetContext());

AlertDialog dlg = dialog.create();

dlg.setMessage(message);

dlg.show();

}

};

session.Send();

}

@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);

}

Приложение 10. Программный код Начального окна API

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;

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.view.ViewGroup;

import android.widget.Button;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.TextView;

public class ApiMainActivity extends AppCompatActivity {

Button exit;

TextView session;

EditText sessionEdit;

public String GetSession()

{

return "Сессия: "+GrapsParams.GetSession(this);

}

public void GraphListOpen\_Click(View v)

{

GrapsParams.SessionsList = false;

try {

GraphListHelper helper = new GraphListHelper(this, GrapsParams.Session);

helper.Send();

try {

helper.th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

helper.on\_ready();

GrapsParams.graphList = new GraphElement\_List(helper.graphs);

}

catch(Exception ex)

{

}

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

startActivityForResult(i, 100);

}

@Override

protected void onCreate(Bundle savedInstanceState) {

GrapsParams.ChangePassword = false;

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_api\_main);

exit = findViewById(R.id.ApiClose);

GrapsParams.API = true;

GrapsParams.Registration = true;

session = findViewById(R.id.sessionText);

DB\_Sessions.GetSession(this);

session.setText(GetSession());

if(!GrapsParams.HaveSession(this))

NoSession();

if(GrapsParams.HaveSession(this))

{

}

}

@Override

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

GrapsParams.ChangePassword = false;

session.setText(GetSession());

DB\_Sessions.SetSession(this);

if(resultCode == 500)

{

NoSession();

}

super.onActivityResult(requestCode, resultCode, data);

}

void NoSession()

{

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

builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

}

});

AlertDialog dialog = builder.create();

dialog.setMessage("Войдите или зарегистрируйтесь в системе");

dialog.setCancelable(false);

dialog.show();

}

public void Autorization\_Click(View v)

{

Intent i = new Intent(GetContent(), AutorizationActivity.class);

startActivityForResult(i, 100);

}

public Context GetContent()

{

return this;

}

@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)

{

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

finish();

}

public void CloseSession\_Click(View v)

{

try {

GrapsParams.CloseSession(this, true);

session.setText(GetSession());

}

catch (Exception ex)

{

}

}

public void DeleteAccount\_Click(View v)

{

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

builder.setPositiveButton("ОК", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

}

});

AlertDialog dialog = builder.create();

dialog.setCancelable(false);

try {

ApiHelper helper = new ApiHelper(this);

helper.Method = "DELETE";

helper.SendStop(GrapsParams.GetUrl(this)+"/account/delete", "token="+GrapsParams.Session);

if(!helper.Ready)

throw new Exception();

else

{

dialog.setMessage("Аккаунт успено удалён");

dialog.show();

}

session.setText(GetSession());

}

catch (Exception ex)

{

try {

session.setText(GetSession());

}

catch (Exception ex1)

{

}

dialog.setMessage("Сессия не была открыта");

dialog.setTitle("Ошибка");

dialog.show();

}

}

public void ChangePassword(View v)

{

GrapsParams.ChangePassword = true;

Intent i = new Intent(GetContent(), AutorizationActivity.class);

startActivityForResult(i, 100);

}

public void ListSessions\_Click(View v)

{

GrapsParams.SessionsList = true;

Intent i = new Intent(GetContent(), GraphElementsListActivity.class);

startActivityForResult(i, 100);

}

}

Приложение 11. Список объектов графа

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import androidx.annotation.NonNull;

import java.util.ArrayList;

public class GraphElement\_List extends ArrayList<GraphElement> {

Graph graph;

public Graph GetGraph()

{

return graph;

}

public int IdElementFromAPI(int id\_api)

{

try

{

int id = id\_api;

for(int i = 0; i < size(); i++)

{

try

{

GraphElement node = get(i);

int api = node.IDinAPI;

if(api == id)

return i;

}

catch (Exception ex)

{

}

}

return -1;

}

catch (Exception ex)

{

return -1;

}

}

public GraphElement ElementFromAPI(int id\_api) {

try {

int id = IdElementFromAPI(id\_api);

GraphElement node = get(id);

return node;

} catch (Exception ex) {

return null;

}

}

public GraphElementName elementName;

public void SetGraph(Graph graph, GraphElementName elementName)

{

this.graph = graph;

clear();

this.elementName = elementName;

if(elementName == GraphElementName.Node)

addListNodes(graph.nodes);

else if(elementName == GraphElementName.Link)

addListLinks(graph.links);

}

public void AddElements(GraphElement\_List elements)

{

if(elements.IsGraph())

addListGraphs(elements);

else if(elements.IsNode())

addListNodes(elements);

else if(elements.IsLink())

addListLinks(elements);

}

public void SetGraph()

{

SetGraph(GetGraph());

}

public void SetGraph(Graph graph)

{

SetGraph(graph, elementName);

}

public GraphElement\_List()

{

super();

}

public GraphElement\_List(Graph graph, GraphElementName elementName)

{

this();

SetGraph(graph, elementName);

}

public GraphElement\_List(ArrayList<Graph> graphs)

{

this();

elementName = GraphElementName.Graph;

addListGraphs(graphs);

}

public void addListGraphs(GraphElement\_List list)

{

if(!list.IsGraph())

return;

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

{

add(list.get(i));

}

elementName = GraphElementName.Graph;

}

public GraphElement\_List(GraphElement\_List list)

{

this();

if(list.IsGraph())

addListGraphs(list);

else if(list.IsNode())

addListNodes(list);

else if(list.IsLink())

addListLinks(list);

}

public boolean addListNodes(@NonNull ArrayList<Node> graphs) {

return super.addAll(graphs);

}

public boolean addListNodes(@NonNull GraphElement\_List graphs) {

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

{

GraphElement graphElement = graphs.get(i);

if(graphElement.IsNode())

add(graphElement);

}

return size() > 0;

}

public boolean addListLinks(@NonNull GraphElement\_List graphs) {

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

{

GraphElement graphElement = graphs.get(i);

if(graphElement.IsLink())

add(graphElement);

}

return size() > 0;

}

public boolean addListLinks(@NonNull ArrayList<Link> graphs) {

return super.addAll(graphs);

}

public boolean addListGraphs(@NonNull ArrayList<Graph> graphs) {

return super.addAll(graphs);

}

public boolean addListGraphElements(@NonNull ArrayList<GraphElement> graphs) {

return super.addAll(graphs);

}

public boolean addListNodes(int index, @NonNull ArrayList<Link> graphs) {

return super.addAll(index, graphs);

}

public boolean addListLinks(int index, @NonNull ArrayList<Link> graphs) {

return super.addAll(index, graphs);

}

public boolean addListGraphs(int index, @NonNull ArrayList<Graph> graphs) {

return super.addAll(index, graphs);

}

public boolean addListGraphElements(int index, @NonNull ArrayList<GraphElement> graphs) {

return super.addAll(index, graphs);

}

public boolean IsGraph()

{

try {

if(size() == 0)

throw new Exception();

boolean graph = true;

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

graph = graph && get(i).IsGraph();

}

return graph;

}

catch(Exception ex)

{

return elementName == GraphElementName.Graph;

}

}

public boolean IsNode()

{

try {

if(size() == 0)

throw new Exception();

boolean graph = true;

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

graph = graph && get(i).IsNode();

}

return graph;

}

catch(Exception ex)

{

return elementName == GraphElementName.Node;

}

}

public boolean IsLink()

{

try {

if(size() == 0)

throw new Exception();

boolean graph = true;

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

graph = graph && get(i).IsLink();

}

return graph;

}

catch(Exception ex)

{

return elementName == GraphElementName.Link;

}

}

public String GetName()

{

if(IsNode())

{

return "Узлы (Nodes)";

}

else if(IsLink())

{

return "Связи/Рёбра (Links)";

}

else if(IsGraph())

{

return "Графы (Graphs)";

}

else

{

return "";

}

}

public GraphElement add()

{

if(IsLink())

return new Link(graph);

else if (IsNode())

return new Node(graph);

else if (IsGraph())

return new Graph();

else

return null;

}

}

Приложение 12. Базовый класс для всех компонентов

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import java.text.SimpleDateFormat;

import java.util.Date;

public abstract class GraphElement {

protected String NameElement;

public String GetName()

{

return NameElement;

}

public GraphElement(String name)

{

NameElement = name;

SetDatetimeNow();

}

public void SetDatetimeNow()

{

SetTimeStamp(new Date());

}

public abstract Graph GetGraph();

public String TimeStamp;

public void SetdateTime(String date)

{

SetTimeStamp(new Date(date));

}

public void SetTimeStamp(Date date)

{

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

}

public void SetTimeStamp(long date)

{

SetTimeStamp(new Date(date));

}

public String GetDatetime()

{

return TimeStamp;

}

public Date GetTimeStamp()

{

return new Date(TimeStamp);

}

public abstract String TypeText();

public boolean IsNode()

{

return this instanceof Node;

}

public boolean IsLink()

{

return this instanceof Link;

}

public boolean IsGraph()

{

return this instanceof Graph;

}

public Graph ToGraph() {return (Graph) this;}

public Node ToNode()

{

return (Node)this;

}

public Node Node()

{

return ToNode();

}

public Link Link() {

return ToLink();

}

public Link ToLink() {

return (Link) this;

}

public Graph Graph() {

return ToGraph();

}

private int API\_ID = -1;

public int Get\_API\_ID()

{

return API\_ID;

}

public void Set\_API\_ID(int id)

{

API\_ID = id;

}

private boolean have\_api = false;

public boolean GetHaveAPI()

{

return have\_api;

}

public void SetHaveAPI(boolean have)

{

have\_api = have;

}

@Override

public String toString() {

return GetName();

}

public abstract int ID();

public int id()

{

return ID();

}

public void SetNameFromID()

{

NameElement = GetNameFromID();

}

public abstract String GetNameFromID();

public abstract GraphElement CopyElement();

public abstract GraphElement CopyElement(Graph graph);

public boolean EqualsTypes(GraphElement element)

{

if(this.IsGraph() && element.IsGraph())

return true;

else if (this.IsNode() && element.IsNode())

return true;

else if (this.IsLink() && element.IsLink())

return true;

else

return false;

}

public int IDinAPI = -1;

public Boolean HaveAPI()

{

return IDinAPI>-1;

}

public Boolean RunAPI = false;

public Boolean ApiReady = false;

}

Приложение 13. Собственно, Граф

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import java.util.ArrayList;

public class Graph extends GraphElement {

public int nodesAPI = 0;

@Override

public Boolean HaveAPI() {

Boolean have = super.HaveAPI();

if(have) {

for (int i = 0; i < NodeCount(); i++) {

have = have && GetNode(i).HaveAPI();

if (!have)

break;

}

}

return have;

}

public GraphElement\_List GetNodes()

{

return new GraphElement\_List(this, GraphElementName.Node);

}

public GraphElement\_List GetLinks()

{

return new GraphElement\_List(this, GraphElementName.Link);

}

public void SetName(String name)

{

NameElement = name;

}

public ArrayList<Node> nodes = new ArrayList<>();

public Node GetNode(int id)

{

return nodes.get(id);

}

public Node AddNode(float x, float y, String name)

{

Node node = new Node(x, y, name, this);

//nodes.add(node);

AddNode(node);

return node;

}

public Node NodeFromAPI(int id\_api)

{

int id = id\_api;

for(int i = 0; i < NodeCount(); i++)

{

try

{

Node node = GetNode(i);

int api = node.IDinAPI;

if(api == id)

return node;

}

catch (Exception ex)

{

}

}

return null;

}

public int IdNodeFromAPI(int id\_api)

{

try

{

return NodeFromAPI(id\_api).ID();

}

catch (Exception ex)

{

return -1;

}

}

public int IdLinkFromAPI(int id\_api)

{

try

{

return LinkFromAPI(id\_api).ID();

}

catch (Exception ex)

{

return -1;

}

}

public Link LinkFromAPI(int id\_api)

{

int id = id\_api;

for(int i = 0; i < LinkCount(); i++)

{

try

{

Link node = GetLink(i);

int api = node.IDinAPI;

if(api == id)

return node;

}

catch (Exception ex)

{

}

}

return null;

}

public Node AddNode()

{

return AddNode(new Node(this));

}

public Node AddNode(float x, float y)

{

Node node = new Node(x, y, this);

//nodes.add(node);

AddNode(node);

node.SetNameFromID();

return node;

}

public Node AddNode(Node node1)

{

Node node = node1;

node1.SetGraph(this);

nodes.add(node);

return node;

}

public Node InsertNode(int index, float x, float y, String name)

{

Node node = new Node(x, y, name, this);

InsertNode(index, node);

return node;

}

public Node InsertNode(int index, float x, float y)

{

Node node = new Node(x, y, this);

//nodes.add(index, node);

InsertNode(index, node);

return node;

}

public Node InsertNode(int index, Node node1)

{

Node node = node1;

node1.SetGraph(this);

nodes.add(index, node);

return node;

}

public void DeleteNode(int id)

{

Node node = GetNode(id);

int linkCount = LinkCount();

for (int i = 0; i < linkCount; i++) {

Link link = GetLink(i);

if (link.ContainsNode(id)) {

DeleteLink(i);

linkCount = LinkCount();

i--;

}

else

{

link.DecrimentAfterID(id);

}

}

nodes.remove(id);

}

public void ClearNodes()

{

nodes.clear();

ClearLinks();

}

public void ClearLinks()

{

links.clear();

}

public Boolean ContainsNode(Node node)

{

return nodes.contains(node);

}

public int IndexNode(Node node)

{

int index = nodes.indexOf(node);

return index;

}

public int NodeCount()

{

return nodes.size();

}

public ArrayList<Link> links = new ArrayList<>();

public Link GetLink(int id)

{

return links.get(id);

}

public Link AddLink(int source, int target, float value)

{

Link node = new Link(this, source, target, value);

if(ContainsLink(node, true))

return null;

links.add(node);

return node;

}

public Node SetNode(int index, float x, float y, String name)

{

Node node = GetNode(index);

node.SetNode(x, y, name);

return node;

}

public Node SetNode(int index, Node node1)

{

Node node = GetNode(index);

node.SetNode(node1);

return node;

}

public Graph GetGraph()

{

return this;

}

public Link SetLink(int index, Link link)

{

Graph graph = GetGraph();

Link l = graph.GetLink(index);

l.Orientation = link.Orientation;

l.sourceID = link.sourceID;

l.targetID = link.targetID;

l.Text = link.Text;

l.Value = link.Value;

l.TextVisible = link.TextVisible;

l.ValueVisible = link.ValueVisible;

l.Set\_API\_ID(link.Get\_API\_ID());

l.IDinAPI = link.IDinAPI;

for(int i = 0; i < graph.LinkCount(); i++)

{

Link l1 = graph.GetLink(i);

if(i != index)

if(!l1.Orientation)

{

if(l1.ContainsNodes(l.sourceID, l.targetID))

{

DeleteLink(index);

return null;

}

else

{

}

}

else

{

if(l1.sourceID == l.sourceID && l1.targetID == l.targetID)

{

DeleteLink(index);

return null;

}

}

}

return l;

}

public int IdNodeFromApi(int api)

{

for(int i = 0; i< NodeCount(); i++)

{

if(GetNode(i).Get\_API\_ID() == api)

return i;

}

return -1;

}

public Link AddLink(int source, int target, float value, Boolean orientation)

{

Link node = new Link(this, source, target, value);

node.Orientation = orientation;

if(ContainsLink(node, true))

return null;

links.add(node);

return node;

}

public Link AddLink(int source, int target)

{

Link node = new Link(this, source, target);

if(ContainsLink(node, true))

return null;

links.add(node);

return node;

}

public Link AddLink(int source, int target, boolean orientation)

{

Link node = new Link(this, source, target);

node.Orientation = orientation;

if(ContainsLink(node, true))

return null;

links.add(node);

return node;

}

public Link InsertLink(int index, int source, int target, float value)

{

Link node = new Link(this, source, target, value);

links.add(index, node);

return node;

}

public void DeleteLink(int id)

{

links.remove(id);

}

public Boolean ContainsLink(Link node)

{

return ContainsLink(node, false);

}

public Boolean ContainsLink(Link node, boolean add)

{

if(!add)

return links.contains(node);

else

{

return ContainsLink(node.sourceID, node.targetID, node.Orientation);

}

}

public Boolean ContainsLink(int source, int target, Boolean orientation)

{

for(int i = 0; i < LinkCount(); i++)

{

Link link = GetLink(i);

if(!orientation)

{

if(link.ContainsNodes(GetNode(source), GetNode(target)))

return true;

}

else

{

if(link.sourceID == source && link.targetID == target)

return true;

}

}

return false;

}

public int IndexLink(Link node)

{

return links.indexOf(node);

}

public void DeleteLink(Node node)

{

links.remove(node);

}

public int LinkCount()

{

return links.size();

}

public Graph(String name)

{

super(name);

}

public Graph(String name, int id)

{

this(name+String.valueOf(id));

}

public Graph(int id)

{

this("graph", id);

}

public Graph()

{

this(GrapsParams.GraphID);

}

@Override

public String TypeText() {

return "Граф (Graph)";

}

@Override

public int ID() {

return Get\_API\_ID();

}

@Override

public String GetNameFromID() {

return "graph"+String.valueOf(id());

}

@Override

public GraphElement CopyElement() {

Graph graph = new Graph(GetName());

for(int i = 0; i < NodeCount(); i++)

{

graph.AddNode(GetNode(i).CopyElement().Node());

}

for(int i = 0; i < LinkCount(); i++)

{

graph.AddLink(GetLink(i).CopyElement().Link());

}

graph.IDinAPI = IDinAPI;

return graph;

}

public boolean HaveNode(int index)

{

return index >-1 && index < NodeCount();

}

public GraphElement\_List GetList(GraphElementName name)

{

return new GraphElement\_List(this, name);

}

public Link AddLink(Link link)

{

Link result = AddLink(link.sourceID, link.targetID, link.Orientation);

result = SetLink(result.id(), link);

return result;

}

@Override

public GraphElement CopyElement(Graph graph) {

graph.ClearNodes();

for(int i = 0; i < NodeCount(); i++)

{

graph.AddNode(GetNode(i).CopyElement().Node());

}

for(int i = 0; i < LinkCount(); i++)

{

graph.AddLink(GetLink(i).CopyElement().Link());

}

return graph;

}

}

Приложение 14. Узел графа

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

public class Node extends GraphElement {

public float X = 100, Y =100;

private Graph graph;

public void SetName(String name)

{

NameElement = name;

}

@Override

public String GetNameFromID() {

return "Node" + String.valueOf(id());

}

@Override

public GraphElement CopyElement() {

return new Node(this, GetGraph());

}

@Override

public GraphElement CopyElement(Graph graph) {

Node node = CopyElement().Node();

node.SetGraph(graph);

return node;

}

public int ID()

{

try {

int index = graph.IndexNode(this);

return index;

}

catch (Exception ex)

{

return -1;

}

}

public Graph GetGraph()

{

return graph;

}

public void SetGraph(Graph graph)

{

try

{

this.graph.DeleteNode(id());

}

catch (Exception ex)

{

}

this.graph = graph;

}

public Node(float x, float y, String name, Graph graph)

{

super(name);

X = x;

Y = y;

SetGraph(graph);

}

public Node(float x, float y, Graph graph)

{

this(x, y, "", graph);

SetNameFromID();

}

public Node(Graph graph)

{

this(450.0f, 350.0f, graph);

}

public Node (Node node, Graph graph)

{

this(node.X, node.Y, node.GetName(), graph);

IDinAPI = node.IDinAPI;

}

public float rad = 0.0f;

@Override

public String TypeText() {

return "Узел (Node)";

}

public void SetNode(float x, float y, String name)

{

X = x;

Y = y;

NameElement = name;

}

public void SetNode(Node node)

{

SetNode(node.X, node.Y, node.GetName());

IDinAPI = node.IDinAPI;

}

}

Приложение 15. Связь или ребро графа

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import java.util.ArrayList;

public class Link extends GraphElement {

Graph graph;

public float Value = 0.0f;

public float ValueWithCheck()

{

try {

if (ValueVisible)

return Value;

else

return 0;

}

catch (Exception ex)

{

return 0;

}

}

public float GetValue()

{

return Value;

}

public String GetTextValue()

{

return String.valueOf(GetValue());

}

public void SetValue(float value)

{

Value = value;

}

public void SetValue(String value)

{

SetValue(Float.valueOf(value));

}

public boolean Orientation = false;

public String Text = "";

public void SetText(String text)

{

Text = text;

}

public String GetText()

{

return Text;

}

public boolean TextVisible = false;

public boolean ValueVisible = false;

ArrayList<Node> nodes = new ArrayList<>();

@Override

public String GetNameFromID() {

return GetName();

}

@Override

public GraphElement CopyElement() {

Link link = new Link(GetGraph());

link.sourceID = this.sourceID;

link.targetID = this.targetID;

link.Orientation = this.Orientation;

link.Text = Text;

link.Value = Value;

link.TextVisible = TextVisible;

link.ValueVisible = ValueVisible;

link.IDinAPI = IDinAPI;

return link;

}

@Override

public GraphElement CopyElement(Graph graph) {

Link node = CopyElement().Link();

node.SetGraph(graph);

return node;

}

public void DecrimentAfterID(int id)

{

//id++;

if(sourceID >= id)

sourceID--;

if(targetID >= id)

targetID--;

SetNodes();

}

public void ChangeNode()

{

int change = sourceID;

sourceID = targetID;

targetID = change;

SetNodes();

}

public void ChangeOrientationLink()

{

ChangeNode();

}

public int ID()

{

try {

return graph.IndexLink(this);

}

catch (Exception ex){

return -1;

}

}

public int id()

{

return ID();

}

public Graph Graph()

{

return graph;

}

private Node source, target;

public Node Source()

{

SetNodes();

return source;

}

public int IDsourceAPI()

{

return Source().IDinAPI;

}

public Node Target()

{

SetNodes();

return target;

}

public int IDtargetAPI()

{

return Target().IDinAPI;

}

public int sourceID, targetID;

ArrayList<Integer> IDs = new ArrayList<>();

public void SetGraph(Graph graph)

{

try

{

this.graph.DeleteLink(id());

}

catch (Exception ex)

{

}

this.graph = graph;

}

public void SetNodes()

{

SetNodes(sourceID, targetID);

}

public void SetNodes(int source, int target)

{

if(target > graph.NodeCount())

target = graph.NodeCount()-1;

if(source == target)

source = target-1;

this.source = graph.GetNode(source);

this.target = graph.GetNode(target);

nodes.clear();

nodes.add(this.source);

nodes.add(this.target);

IDs.clear();

sourceID = source;

targetID = target;

IDs.add(source);

IDs.add(target);

NameElement = this.source.GetName() + " -> " + this.target.GetName();

}

public Boolean ContainsNode(Node node)

{

return nodes.contains(node);

}

public Boolean ContainsNode(int node)

{

boolean have = IDs.contains(node);

return have;

}

public Boolean ContainsNodes(Node node1, Node node2)

{

return ContainsNode(node1) && ContainsNode(node2);

}

public Boolean ContainsNodes(int node1, int node2)

{

boolean have = ContainsNode(node1) && ContainsNode(node2);

return have;

}

public Link(Graph graph, int source, int target)

{

this(graph);

SetNodes(source, target);

}

public Link(Graph graph)

{

super("");

SetGraph(graph);

}

public Link(Graph graph, int source, int target, float value)

{

this(graph, source, target);

Value = value;

ValueVisible = true;

}

@Override

public String TypeText() {

return "Связь/Ребро (Link)";

}

@Override

public String GetName() {

String line = "-";

if(Orientation)

line+=">";

String name = source.GetName() + " " + line + " " + target.GetName();

return name;

}

@Override

public Graph GetGraph() {

return graph;

}

}

Приложение 16. Цвет элементов графа

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.graphics.Color;

public class ColorNode {

int fill = Color.rgb(255, 255, 255);

public int GetBorderColor()

{

return fill;

}

public int GetFillColor()

{

int color = fill;

int red = Color.red(color);

int green = Color.green(color);

int blue = Color.blue(color);

return Color.argb(50,red, green, blue);

}

public int GetBorderRed()

{

return Color.red(GetBorderColor());

}

public int GetBorderGreen()

{

return Color.green(GetBorderColor());

}

public int GetBorderBlue()

{

return Color.green(GetBorderColor());

}

public int GetFillRed()

{

return Color.red(GetFillColor());

}

public int GetFillGreen()

{

return Color.green(GetFillColor());

}

public int GetFillBlue()

{

return Color.green(GetFillColor());

}

public void SetFillColor(int color)

{

int red = Color.red(color);

int green = Color.green(color);

int blue = Color.blue(color);

fill = Color.rgb(red, green, blue);

}

public void SetFillColor(int red, int green, int blue)

{

SetFillColor(Color.rgb(red, green, blue));

}

public ColorNode()

{

}

public ColorNode(int color)

{

this();

SetFillColor(color);

}

public ColorNode(int red, int green, int blue)

{

this();

SetFillColor(Color.rgb(red, green, blue));

}

}

Приложение 17. Буфер обмена для объектов графа

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import android.content.Context;

import androidx.appcompat.app.AlertDialog;

import org.jetbrains.annotations.Nullable;

public class GrapsParams {

public static int GraphID = 0;

public static Graph Graph;

public static String Session = "";

public static Boolean API = false;

public static GraphElement GraphElement;

public static int elementID()

{

return GraphElement.id();

}

public static GraphElement\_List graphList;

public static GraphElement GraphCopy;

public static DB\_Graphs DB;

public static GraphElement\_List graphs;

public static boolean Run\_Graph = false;

public static Graph NowGraph;

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

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

public static Boolean Registration = false;

public static Boolean Opened;

public static Graph GraphAPI;

public static void OpenSession(String session)

{

Session = session;

Opened = true;

Registration = true;

}

public static Boolean HaveSession()

{

try

{

if(!Opened)

{

}

}

catch (Exception ex)

{

Opened = false;

}

if(Session == "" || Session.equals("") || Session == null || !Opened)

{

Session = "";

Opened = false;

return false;

}

return true;

}

public static Boolean HaveSession(Activity ctx)

{

if(!HaveSession())

return false;

ListSessionsHelper helper = new ListSessionsHelper(ctx, Session, false);

helper.SendStop();

Opened = helper.Ready;

return HaveSession();

}

public static String GetSession()

{

HaveSession();

return Session;

}

public static String GetSession(Activity ctx)

{

HaveSession(ctx);

return Session;

}

public static void CloseSession(String session, Activity ctx)

{

CloseSession(session, ctx, false);

}

public static void CloseSession(Activity ctx)

{

CloseSession(ctx, false);

}

public static void CloseSession(String session, Activity ctx, Boolean message)

{

Boolean message1 = message;

CloseSession close = new CloseSession(ctx, session)

{

@Override

public void MessageOutput(String message) {

if(message1)

{

try {

AlertDialog.Builder builder = new AlertDialog.Builder(ctx);

AlertDialog dialog = builder.create();

dialog.setMessage(message);

dialog.show();

}

catch(Exception ex)

{

}

}

}

};

close.Send();

}

public static void CloseSession(Activity ctx, Boolean message)

{

CloseSession(Session, ctx, message);

Session = "";

Opened = false;

}

public static String GetUrl(Context context)

{

UrlStorege.GetDB(context).GetUrl();

return Url(context);

}

public static String Url(@Nullable Context context) {

String url = "";

int end = 0;

String end1 = "";

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

try {

if(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 void CreateGraph(Activity ctx, Graph graph)

{

ListNodesHelper.GetNodes(ctx, graph);

ListLinksHelper.GetLinks(ctx, graph);

}

public static GraphElementName ElementName;

public static Boolean ChangePassword = false;

public static Boolean SessionsList = false;

}

Приложение 18. Рисовальная панель для объектов графа

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.icu.number.Scale;

import android.view.MotionEvent;

import android.view.SurfaceView;

import android.widget.Toast;

public class GraphView extends SurfaceView

{

public boolean GetHaveAPI()

{

return graph.GetHaveAPI();

}

public void SetHaveAPI(boolean have)

{

graph.SetHaveAPI(have);

}

Graph graph;

Paint p;

ColorNode NoSelect, Select1, Select2;

int selected1 = -1, selected2 = -1, selectedNowLink = -1;

int selectedNowNode = -1;

public int SelectedNowLink()

{

if(selectedNowLink > graph.LinkCount())

selectedNowLink=-1;

return selectedNowLink;

}

public Node GetSelectedNode()

{

return graph.GetNode(selectedNowNode);

}

public Link GetSelectedLink()

{

return graph.GetLink(selectedNowLink);

}

public void SetNode(int index, Node node)

{

Node node1 = graph.GetNode(index);

BeforeEditNode(node1);

graph.SetNode(index, node);

node1 = graph.GetNode(index);

AfterEditNode(node1, "update");

}

public void SetNode(Node node)

{

int node1 = SelectedNowNode();

if(node1 > -1 && node1 < graph.NodeCount())

SetNode(SelectedNowNode(), node);

}

public void SetGraphElement(GraphElement element)

{

if(element.IsNode()) {

SetNode(element.Node());

}

else if(element.IsLink())

{

try {

Link link1 = element.Link();

int source = -1;

int target = -1;

BeforeEditNode(link1);

try {

Link link2 = graph.GetLink(selectedNowLink);

source = link2.IDsourceAPI();

target = link2.IDtargetAPI();

}

catch (Exception ex)

{

link1.GetText();

}

Link link = graph.SetLink(selectedNowLink, link1);

link.Orientation = link1.Orientation;

link.Value = link1.Value;

link.Text = link1.Text;

link.ValueVisible = link1.ValueVisible;

link.TextVisible = link1.TextVisible;

if(link.IDsourceAPI() == source && link.IDtargetAPI() == target)

{

AfterEditNode(link, "update");

}

else

{

AfterEditNode(link, "delete");

AfterEditNode(link, "insert");

}

}

catch (Exception ex)

{

AfterEditNode(null, "select");

}

}

}

public Boolean Selection()

{

return SelectedNowNode() > -1 || SelectedNowLink() > -1;

}

public GraphElement GetSelected()

{

int node = SelectedNowNode();

int link = SelectedNowLink();

if(node > -1 && node < graph.NodeCount())

{

return GetSelectedNode();

}

else if (link > -1 && link < graph.LinkCount())

{

return GetSelectedLink();

}

else return null;

}

public int SelectedNowNode()

{

if(sel1)

return selected1;

else if (sel2)

return selected2;

else {

if(selectedNowNode > graph.NodeCount())

selectedNowNode = -1;

return selectedNowNode;

}

}

float csx, csy;

float CSX()

{

return csx;

}

float CTX()

{

return CSX()+1800.f;

}

float CSY()

{

return csy;

}

float CTY()

{

return CSY()+3600.f;

}

public GraphView(Context context) {

super(context);

p = new Paint();

NoSelect = new ColorNode(255, 0, 0);

Select1 = new ColorNode(255, 192, 203);

Select2 = new ColorNode(255, 0, 127);

selected1 = selected2 = selectedNowLink = -1;

Graph graph = new Graph();

setWillNotDraw(false);

SetGraph(graph);

setWillNotDraw(false);

invalidate();

//canvas.scale(0.0f, 0.0f, 100.0f, 100.0f);

}

public GraphView(Graph graph, Context context)

{

this(context);

SetGraph(graph);

}

public Link SetLink(int source, int target, boolean orientation)

{

if(source<0 || source >= graph.NodeCount())

return null;

if(target<0 || target >= graph.NodeCount())

return null;

Link link = graph.AddLink(source, target, orientation);

BeforeEditNode(link);

SetGraph(graph);

AfterEditNode(link, "insert");

return link;

}

public Link SetLink(int source, int target)

{

return SetLink(source, target, true);

}

public Link SetLink()

{

return SetLink(selected1, selected2);

}

public Link SetLink(Link link)

{

return SetLink(link.sourceID, link.targetID, link.Orientation);

}

public Link SetLink(boolean orientation)

{

if(selectedNowLink < 0 || selectedNowLink >= graph.LinkCount())

return SetLink(selected1, selected2, orientation);

else

{

Link l = graph.GetLink(selectedNowLink);

l.Orientation = orientation;

for(int i = 0; i < graph.LinkCount(); i++)

{

Link l1 = graph.GetLink(i);

if(i != selectedNowLink)

if(l.Orientation)

{

if(l1.ContainsNodes(l.sourceID, l.targetID));

{

DeleteLink(selectedNowLink);

selectedNowLink = -1;

SetGraph(graph);

return null;

}

}

else

{

if(l1.sourceID == l.sourceID && l1.targetID == l.targetID)

{

DeleteLink(selectedNowLink);

selectedNowLink = -1;

SetGraph(graph);

return null;

}

}

}

SetGraph(graph);

return l;

}

}

public Graph GetGraph()

{

return graph;

}

public void SetGraph(Graph graph)

{

this.graph = graph;

invalidate();

}

public Node AddNode(float x, float y)

{

Node node = graph.AddNode(x, y);

SetGraph(graph);

return node;

}

public Node AddNode(Node node1)

{

node1 = new Node(node1, graph);

Node node = graph.AddNode(node1);

SetGraph(graph);

return node;

}

public Node AddNode(float x, float y, String name)

{

Node node = graph.AddNode(x, y, name);

SetGraph(graph);

return node;

}

public Node AddNode()

{

Graph graph1 = new Graph();

Node n = graph1.AddNode();

BeforeEditNode(n);

float x = n.X + dX;

float y = n.Y + dY;

Node node = graph.AddNode(x, y);

//Node node = graph.AddNode(sX, sY);

SetGraph(graph);

AfterEditNode(n, "insert");

return node;

}

public void DeleteNode()

{

DeleteNode(SelectedNowNode());

}

public void DeleteNode(int index)

{

if(index < 0 || index > graph.NodeCount())

return;

int delete = index;

Node node = graph.GetNode(index);

BeforeEditNode(node);

if(delete < graph.NodeCount() && delete > -1)

{

graph.DeleteNode(delete);

}

int selectedNowNode = delete;

if(selected1 == selectedNowNode)

{

selected1 = selected2;

selected2 = -1;

}

else if (selected2 == selectedNowNode)

{

selected2 = -1;

}

if(selected1 > selectedNowNode)

selected1--;

if(selected2 > selectedNowNode)

selected2 --;

SetGraph(graph);

AfterEditNode(node, "delete");

}

public void DeleteLink()

{

DeleteLink(SelectedNowLink());

}

public void DeleteLink (int id)

{

if(id > -1 && id < graph.LinkCount())

{

Link link = graph.GetLink(id);

BeforeEditNode(link);

graph.DeleteLink(id);

AfterEditNode(link, "delete");

}

}

public void ChangeOrientationLink(int id)

{

if(id > -1 && id < graph.LinkCount())

{

Link link = graph.GetLink(id);

BeforeEditNode(link);

graph.GetLink(id).ChangeOrientationLink();

SetGraph(graph);

AfterEditNode(link, "delete");

AfterEditNode(link, "insert");

}

}

public void ChangeOrientationLink()

{

ChangeOrientationLink(SelectedNowLink());

}

public void Delete()

{

if(selectedNowNode > -1) {

DeleteNode(selectedNowNode);

if (selected1 == selectedNowNode) {

selected1 = selected2;

selected2 = -1;

} else if (selected2 == selectedNowNode) {

selected2 = -1;

}

if (selected1 > selectedNowNode)

selected1--;

if (selected2 > selectedNowNode)

selected2--;

}

else if (selectedNowLink > -1)

{

DeleteLink(selectedNowLink);

}

SetGraph(graph);

}

public void paint()

{

invalidate();

}

float dX = 0.0f, dY = 0.0f;

float sX = 0.0f, sY = 0.0f;

float halfside = 40.0f;

float rad = 60.0f;

@Override

protected void onDraw(Canvas canvas) {

try {

halfside = rad/2f;

float csx = CSX(), csy = CSY(), ctx = CTX(), cty = CTY();

p.setStrokeWidth(10.0f);

//canvas.scale(1800.0f, 3600.0f);

canvas.drawColor(Color.rgb(255, 255, 255));

int links = graph.LinkCount();

p.setStyle(Paint.Style.FILL);

for (int i = 0; i < links; i++) {

try {

ColorNode color;

if(i == selectedNowLink) color = Select2;

else color = NoSelect;

Link l = graph.GetLink(i);

Node source = l.Source();

Node target = l.Target();

float sx1 = source.X;

float tx1 = target.X;

float sy1 = source.Y;

float ty1 = target.Y;

float sx = sx1 - dX;

float sy = sy1 - dY;

float tx = tx1 - dX;

float ty = ty1 - dY;

float sx2 = sx1, sy2 = sy1, tx2 = tx1, ty2 = ty1;

p.setStyle(Paint.Style.FILL);

p.setColor(color.GetBorderColor());

canvas.drawLine(sx, sy, tx, ty, p);

if (l.Orientation) {

float xRad, yRad;

//sx1 = Math.abs(sx1);

//tx1 = Math.abs(tx1);

//sy1 = Math.abs(sy1);

//ty1 = Math.abs(ty1);

//float URadX = (rad\*2f)/(sx1 + tx1);

//float URadY = (rad\*2f)/(sy1 + ty1);

float max = 0.2f;

float SX = Math.abs(tx - sx);

//float URadX = (rad\*2f)/SX;

float URadX = max;

while(URadX > max)

{

URadX/=2;

}

float SY = Math.abs(ty - sy);

//float URadY = (rad\*2f)/SY;

float URadY = max;

while(URadY > max)

{

URadY/=2;

}

//float URadY = URadX;

/\*URadX = 1f - URadX;

URadY = 1f - URadY;

xRad = (tx - sx)\*URadX;

yRad = (ty - sy) \* URadY;

xRad += sx;

yRad += sy;\*/

xRad = yRad = 0;

if(tx == sx)

{

xRad = sx;

}

else if(tx > sx)

{

xRad = tx - SX\*URadX;

}

else

{

xRad = tx + SX\*URadX;

}

if(ty == sy)

{

yRad = sy;

}

else if(ty > sy)

{

yRad = ty - SY\*URadY;

}

else

{

yRad = ty + SY\*URadY;

}

float d = halfside / 2f;

canvas.drawRect(xRad - d, yRad - d, xRad + d, yRad + d, p);

}

float cx = (sx + tx) \* 0.5f;

float cy = (sy + ty) \* 0.5f;

float x0 = cx - halfside;

float x1 = cx + halfside;

float y0 = cy - halfside;

float y1 = cy + halfside;

if(i == selectedNowLink) color = Select2;

else color = Select1;

p.setStyle(Paint.Style.FILL);

p.setColor(color.GetFillColor());

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

p.setStyle(Paint.Style.STROKE);

p.setColor(color.GetBorderColor());

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

float width = p.getStrokeWidth();

p.setStrokeWidth(width/2f);

p.setTextSize(p.getStrokeWidth()\*10f);

p.setColor(Color.BLACK);

float length = p.getTextSize()/2f;

float length1 = 1f;

if(l.TextVisible)

{

float yv = y1;

if(l.sourceID > l.targetID) {

length1 \*= (-1);

yv = y0;

}

canvas.drawText(l.Text, x0 - length, yv+(p.getStrokeWidth()\*10f\*length1), p);

}

if(l.ValueVisible)

{

if(l.sourceID > l.targetID)

length\*=(-1);

float yv = y0;

if(l.sourceID > l.targetID)

yv = y1;

canvas.drawText(String.valueOf(l.Value), x0 - (length\*2f), yv, p);

}

p.setStrokeWidth(width);

}

catch(Exception ex)

{

try {

graph.DeleteLink(i);

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

boolean sel1 = false, sel2 = false;

if (selected2 < 0 || selected2 >= graph.NodeCount()) {

selected2 = -1;

selectedNowNode = selected1;

} else if (selected1 < 0 || selected1 >= graph.NodeCount()) {

if (selected2 >= 0)

selected1 = selected2;

else

selected1 = -1;

selected2 = -1;

selectedNowNode = selected1;

} else {

selectedNowNode = selected2;

}

for (int i = 0; i < graph.NodeCount(); i++) {

try {

Node n = graph.GetNode(i);

n.rad = rad;

p.setStyle(Paint.Style.FILL);

ColorNode color;

if (i == selected1) {

color = Select1;

sel1 = true;

} else if (i == selected2) {

color = Select2;

sel2 = true;

} else

color = NoSelect;

float nx = n.X - dX, ny = n.Y - dY;

p.setColor(color.GetFillColor());

canvas.drawCircle(nx, ny, rad, p);

p.setStyle(Paint.Style.STROKE);

p.setColor(color.GetBorderColor());

canvas.drawCircle(nx, ny, rad, p);

}

catch(Exception ex)

{

}

}

NameView();

if(graph.Get\_API\_ID() > -1)

{

Save();

}

return;

}

catch(Exception ex)

{

ex.printStackTrace();

Toast.makeText(getContext(), ex.getMessage(), Toast.LENGTH\_LONG);

}

//super.onDraw(canvas);

}

boolean toach = false;

float dx, dy;

float selX, selY;

boolean sel1, sel2;

private float c (float a, float d) {

return a + d;

}

public void BeforeEditNode(GraphElement n)

{

}

public void AfterEditNode(GraphElement n, String method)

{

}

@Override

public boolean onTouchEvent(MotionEvent event) {

float x = event.getX();

float y = event.getY();

float xc = c(x, dX), yc = c(y, dY);

int action = event.getAction();

switch (action)

{

case MotionEvent.ACTION\_DOWN:

{

if(dX == 0)

sX = x;

else

sX=x+dX;

if(dY == 0)

sY = y;

else

sY = y+dY;

xc = c(x, dX);

yc = c(y, dY);

toach = false;

for(int i = 0; i< graph.NodeCount(); i++)

{

Node n = graph.GetNode(i);

float nx = n.X;

float ny = n.Y;

selX = nx;

selY = ny;

float nRad = n.rad;

nRad = nRad\*nRad;

float rad = (float)(Math.pow(xc - nx, 2)+Math.pow(yc - ny, 2));

if(rad <= nRad)

{

toach = true;

dx = nx - xc;

dy = ny - yc;

selectedNowLink = -1;

selectedNowNode = i;

BeforeEditNode(n);

if(selected1 > -1)

{

if(i == selected1)

{

sel1 = true;

//SetGraph(graph);

return true;

}

else

{

if(i == selected2)

{

sel2 = true;

}

else

{

selected2 = i;

}

}

}

else

{

selected1 = i;

}

SetGraph(graph);

return true;

}

}

selected1 = -1;

selected2 = -1;

selectedNowNode = -1;

for(int i = 0; i < graph.LinkCount(); i++)

{

Link l = graph.GetLink(i);

Node s = l.Source();

Node t = l.Target();

float sx = s.X;

float tx = t.X;

float sy = s.Y;

float ty = t.Y;

float cx = (sx + tx) \* 0.5f;

float cy = (sy + ty) \* 0.5f;

float x0 = cx - halfside;

float x1 = cx + halfside;

float y0 = cy - halfside;

float y1 = cy + halfside;

if(i == selectedNowLink)

{

selectedNowLink = -1;

SetGraph(graph);

return true;

}

else if (xc >= x0 && xc <= x1 && yc >= y0 && yc <= y1)

{

selectedNowLink = i;

Link link = graph.GetLink(i);

BeforeEditNode(link);

//Link link = graph.GetLink(i);

//BeforeEditNode(link);

SetGraph(graph);

return true;

}

}

selectedNowLink = -1;

SetGraph(graph);

toach = true;

AfterEditNode(null, "select");

return true;

}

//break;

case MotionEvent.ACTION\_MOVE:

{

if(selectedNowNode > -1 && toach)

{

Node n = graph.GetNode(SelectedNowNode());

n.X = xc + dx;

n.Y = yc + dy;

}

else

{

//dX = x - sX;

dX = sX - x;

//dY = y - sY;

dY = sY - y;

}

SetGraph(graph);

return true;

}

//break;

case MotionEvent.ACTION\_UP:

{

AfterEditNode(null, "select");

toach = false;

int i = SelectedNowNode();

if(i < 0 || i >= graph.NodeCount()) {

return true;

}

Node n = graph.GetNode(i);

sel2 = sel2 && (n.X == selX && n.Y == selY);

sel1 = sel1 && (n.X == selX && n.Y == selY) && !sel2;

if(sel2)

{

selected1 = i;

selected2 = -1;

}

if(sel1)

{

selected1 = -1;

selectedNowNode = -1;

}

AfterEditNode(n, "update");

SetGraph(graph);

sel1 = sel2 = false;

return true;

}

//break;

}

return super.onTouchEvent(event);

}

public String GetName()

{

return graph.GetName();

}

public void NameView()

{

}

public void Save()

{

}

}

Приложение 19. Текст с надписью

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.content.Context;

import android.view.View;

import android.view.ViewGroup;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.TextView;

public class LayoutPoleInput extends LinearLayout {

TextView signaturePole;

EditText inputPole;

public TextView SignaturePole()

{

return signaturePole;

}

public EditText InputPole()

{

return inputPole;

}

Context context;

public Context GetContext()

{

return context;

}

public LayoutPoleInput(Context context) {

super(context);

this.context = context;

signaturePole = new TextView(this.context);

signaturePole.setTextAlignment(View.TEXT\_ALIGNMENT\_TEXT\_END);

signaturePole.setTextSize(16f);

inputPole = new EditText(this.context);

LayoutParams params = new LayoutParams(ViewGroup.LayoutParams.WRAP\_CONTENT, ViewGroup.LayoutParams.WRAP\_CONTENT, 1);

signaturePole.setLayoutParams(params);

inputPole.setLayoutParams(params);

addView(signaturePole);

addView(inputPole);

}

}

Приложение 20. Текст с флажком

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.content.Context;

import android.view.ViewGroup;

import android.widget.CheckBox;

import android.widget.EditText;

import android.widget.LinearLayout;

public class TextVisibleView extends LinearLayout {

Context context;

public Context GetContext()

{

return context;

}

private EditText inputText;

public EditText InputText()

{

return inputText;

}

public String GetText()

{

return InputText().getText().toString();

}

public void SetText(String text)

{

InputText().setText(text);

}

private CheckBox checkBoxVisible;

public CheckBox CheckBoxVisible()

{

return checkBoxVisible;

}

public boolean IsTextVisible()

{

return CheckBoxVisible().isChecked();

}

public void SetTextVisible(Boolean visible)

{

CheckBoxVisible().setChecked(visible);

}

public TextVisibleView(Context context) {

super(context);

this.context = context;

setOrientation(HORIZONTAL);

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, LinearLayout.LayoutParams.WRAP\_CONTENT, 1);

LinearLayout.LayoutParams params1 = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.WRAP\_CONTENT, LinearLayout.LayoutParams.WRAP\_CONTENT, 1);

this.setLayoutParams(params1);

inputText = new EditText(GetContext());

inputText.setLayoutParams(params1);

this.addView(inputText);

checkBoxVisible = new CheckBox(GetContext());

checkBoxVisible.setLayoutParams(params1);

checkBoxVisible.setText("Отображать");

this.addView(checkBoxVisible);

}

}

Приложение 21. Базовый класс API

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

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;

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());

}

}

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;

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

{

Ready = false;

String url1 = req;

if(!JsonInput)

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(() -> {

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 Boolean Ready = false;

public void on\_ready()

{

on\_ready(res);

}

public void SendStop()

{

Ready = false;

Send();

try {

th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

public void SendStop(String req, String payload)

{

Ready = false;

send(req, payload);

try {

th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

public void Send()

{

}

}

Приложение 22. Открытие сессии

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import org.json.JSONException;

public class GetSession extends ApiHelper {

public String login, password;

public GetSession(Activity ctx, String login, String password) {

this(ctx);

this.login = login;

this.password = password;

}

public GetSession(Activity ctx) {

super(ctx);

Method = "PUT";

}

@Override

public void on\_ready(String res) {

String session;

try {

session = StringFromJsonObject(res, "token");

} catch (JSONException e) {

session = null;

}

this.session = session;

OutputSession(session);

super.on\_ready(res);

}

public void OutputSession(String session)

{

}

@Override

public String GetMessageFatal() {

return "Неверный логин или пароль";

}

@Override

public String GetMessageReady() {

return "Вы успешно вошли в систему";

}

public void Send()

{

send(GrapsParams.Url(ctx)+"session/open", "name="+login+"&secret="+password);

}

}

Приложение 23. Закрытие сессии

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import org.json.JSONException;

public class CloseSession extends ApiHelper{

public CloseSession(Activity ctx) {

super(ctx);

Method = "DELETE";

}

public CloseSession(Activity ctx, String session)

{

this(ctx);

this.session = session;

}

public CloseSession(ApiHelper helper)

{

this(helper.ctx, helper.session);

}

public void Send()

{

send(GrapsParams.Url(ctx)+"session/close", "token="+session);

}

@Override

public String GetMessageReady() {

return "Вы успешно закрыли сессию";

}

@Override

public String GetMessageFatal() {

return "Сессия не была открыта";

}

}

Приложение 24. Просмотр списка сессии

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import java.util.ArrayList;

public class ListSessionsHelper extends ApiHelper {

public Boolean GetList = true;

public ListSessionsHelper(Activity ctx) {

super(ctx);

Sessions = new ArrayList<>();

Method="GET";

}

public ArrayList<Session> Sessions;

public ListSessionsHelper(Activity ctx, Boolean getList) {

this(ctx);

GetList = getList;

}

public ListSessionsHelper(Activity ctx, String session) {

this(ctx);

this.session = session;

}

public ListSessionsHelper(Activity ctx, String session, Boolean getList) {

this(ctx, session);

GetList = getList;

}

public Boolean Did = false;

@Override

public void on\_ready(String res) {

if(GetList)

{

if(Did)

return;

try {

Did = true;

JSONArray arrayGraphs = new JSONArray(res);

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

{

JSONObject jsonObject = arrayGraphs.getJSONObject(i);

int id = jsonObject.getInt("id");

String session = jsonObject.getString("token");

Session Session = new Session(session);

Session.ID = id;

Session.SetTimeStamp(jsonObject.getLong("timestamp"));

Sessions.add(Session);

}

} catch (JSONException e) {

e.printStackTrace();

}

Did = true;

}

super.on\_ready(res);

}

@Override

public void Send()

{

send(GrapsParams.Url(ctx)+"session/list", "token="+session);

}

public static ArrayList<Session> GetSessions(Activity ctx)

{

ListSessionsHelper helper = new ListSessionsHelper(ctx,GrapsParams.Session);

helper.SendStop();

if(helper.Ready)

helper.on\_ready();

return helper.Sessions;

}

}

Приложение 25. Создание аккаунта

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

public class CreateAccount extends ApiHelper {

public String login, password;

public CreateAccount(Activity ctx, String login, String password) {

super(ctx);

this.login = login;

this.password = password;

Method = "PUT";

}

@Override

public String GetMessageFatal() {

return "Не удалось зарегистрироваться";

}

@Override

public String GetMessageReady() {

return "Вы успешно зарегистрировались";

}

public void Send() {

send(GrapsParams.Url(ctx) + "account/create", "name=" + login + "&secret=" + password);

}

}

Приложение 26. Базовый класс для графовAPI

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import android.se.omapi.Session;

public class GraphHelper extends ApiHelper{

public Graph graph;

public Boolean MakeGraphID = true;

public GraphIdView id = GraphIdView.graph;

public String GetGraphIdView()

{

return id.toString();

}

public Boolean IsGraph()

{

return id == GraphIdView.graph;

}

public void ChangeGraphIdView()

{

if(IsGraph())

id = GraphIdView.id;

else

id = GraphIdView.graph;

}

public void GraphChangeId()

{

if(IsGraph())

ChangeGraphIdView();

}

public GraphHelper(Activity ctx, String session, GraphIdView id) {

super(ctx);

this.graph = graph;

this.session = session;

Method = "PUT";

}

public GraphHelper(Activity ctx, String session, Graph graph, GraphIdView id) {

this(ctx, session, id);

this.graph = graph;

}

public GraphHelper(Activity ctx, String session) {

this(ctx, session, GraphIdView.graph);

}

public GraphHelper(ApiHelper helper, GraphIdView id)

{

this(helper.ctx, helper.session, id);

}

public GraphHelper(ApiHelper helper)

{

this(helper, GraphIdView.graph);

}

public void Send()

{

}

public Boolean Did = false;

@Override

public void on\_ready(String res) {

super.on\_ready(res);

Run1(true);

Run();

}

public GraphHelper GetGraph()

{

return this;

}

@Override

public void on\_fail() {

super.on\_fail();

Run1(false);

ctx.runOnUiThread(()->{

Run();

});

}

public void Run()

{

}

public void Run1(Boolean ready)

{

}

}

Приложение 27. Создание графовAPI

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import org.json.JSONException;

public class CreateGraph extends GraphHelper{

public CreateGraph(ApiHelper helper, Graph graph, GraphIdView id) {

super(helper, id);

this.graph = graph;

Method = "PUT";

MakeGraphID = false;

}

public CreateGraph(ApiHelper helper, Graph graph) {

this(helper, graph, GraphIdView.graph);

}

public CreateGraph(Activity ctx, String session, Graph graph, GraphIdView id) {

super(ctx, session, id);

this.graph = graph;

Method = "PUT";

MakeGraphID = false;

}

public CreateGraph(Activity ctx, String session, Graph graph) {

this(ctx, session, graph, GraphIdView.graph);

}

Boolean SetBool(Boolean[] bool1, Boolean[] bool2)

{

Boolean result = true;

for(int i = 0; i < bool1.length; i++)

{

result = result && bool1[i];

if(!result)

break;

}

if(result)

{

for(int i = 0; i < bool2.length; i++)

{

result = result && bool2[i];

if(!result)

break;

}

}

return result;

}

public void SetFalse(Boolean[] bool)

{

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

{

bool[i] = false;

}

}

@Override

public void on\_ready(String res) {

try {

graph.IDinAPI = IntFromJsonObject(res, "id");

} catch (JSONException e) {

graph.IDinAPI = Integer.getInteger("a");

}

for (int i = 0; i < graph.NodeCount(); i++) {

Node node = graph.GetNode(i);

int finalI = i;

CreateNode node1 = new CreateNode(this, node, GraphIdView.graph);

node1.Send();

try {

node1.th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

if(node1.Ready)

node1.on\_ready();

CreateNode node2 = new CreateNode(this, node, GraphIdView.id);

node2.Send();

try {

node2.th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

if(node2.Ready)

node2.on\_ready();

}

for(int i = 0; i < graph.LinkCount(); i++)

{

Link link = graph.GetLink(i);

int source = link.sourceID;

int target = link.targetID;

link.SetNodes(source, target);

CreateLink link1 = new CreateLink(this, link);

link1.Send();

try {

link1.th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

BaseReady(res);

}

public void BaseReady(String res)

{

super.on\_ready(res);

}

@Override

public String GetMessageFatal() {

return "Не удалось сохранить граф";

}

@Override

public String GetMessageReady() {

return "Граф успешно сохранён";

}

public void Send()

{

send(GrapsParams.Url(ctx)+"graph/create", "token="+session+"&name="+graph.GetName());

}

public static void GraphCreate(Activity ctx, Graph graph)

{

CreateGraph graph1 = new CreateGraph(ctx, GrapsParams.Session, graph);

graph1.SendStop();

}

public static void PastGraph(Activity ctx, Graph graph)

{

Graph graph1 = graph.CopyElement().Graph();

graph1.ClearNodes();

ListNodesHelper.GetNodes(ctx, graph1);

for(int i = 0; i < graph1.NodeCount(); i++) {

Node node = graph1.GetNode(i);

ApiHelper api = new ApiHelper(ctx);

api.Method = "DELETE";

String url = GrapsParams.GetUrl(ctx) + "node/delete";

api.SendStop(url, "token=" + GrapsParams.Session + "&id=" + node.IDinAPI);

}

GraphCreate(ctx, graph);

}

}

Приложение 28. Просмотр списка графовAPI

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

public class GraphListHelper extends GraphHelper {

public GraphListHelper(Activity ctx, String session) {

super(ctx, session);

Method = "GET";

}

public GraphListHelper(ApiHelper helper) {

this(helper.ctx, helper.session);

}

GraphElement\_List graphs = new GraphElement\_List();

public static GraphElement\_List GetGraphs(Activity ctx)

{

GraphListHelper helper = new GraphListHelper(ctx, GrapsParams.Session);

helper.Send();

try {

helper.th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

helper.on\_ready();

return new GraphElement\_List(helper.graphs);

}

@Override

public void on\_ready(String res) {

try {

JSONArray arrayGraphs = new JSONArray(res);

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

{

JSONObject jsonObject = arrayGraphs.getJSONObject(i);

Graph graph = new Graph();

graph.IDinAPI = jsonObject.getInt("id");

graph.nodesAPI = jsonObject.getInt("nodes");

graph.SetName(jsonObject.getString("name"));

graph.SetTimeStamp(jsonObject.getLong("timestamp"));

graphs.add(graph);

}

} catch (JSONException e) {

e.printStackTrace();

}

super.on\_ready(res);

}

@Override

public void Send() {

send(GrapsParams.Url(ctx)+"graph/list", "token="+session);

}

}

Приложение 29. Базовый класс для узлов графаAPI

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

public class NodeHelper extends GraphHelper{

public Node node;

public NodeHelper(Activity ctx, String session, Graph graph, GraphIdView id) {

super(ctx, session, graph, id);

this.graph = graph;

Method = "PUT";

}

public NodeHelper(Activity ctx, String session, Graph graph) {

this(ctx, session, graph, GraphIdView.graph);

}

public NodeHelper(Activity ctx, Graph graph, String session, Node node) {

this(ctx, session, graph, GraphIdView.graph);

this.node = node;

}

public NodeHelper(GraphHelper graphHelper)

{

this(graphHelper.ctx, graphHelper.session, graphHelper.graph, graphHelper.id);

}

public NodeHelper(GraphHelper graphHelper, Node node)

{

this(graphHelper.ctx, graphHelper.session, graphHelper.graph, graphHelper.id);

this.node = node;

}

}

Приложение 30. Создание узла графаAPI

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import org.json.JSONException;

public class CreateNode extends NodeHelper {

public CreateNode(GraphHelper graphHelper, Node node, GraphIdView id) {

super(graphHelper, node);

this.id = id;

}

public CreateNode(Activity ctx, Graph graph, String Session, Node node) {

super(ctx, graph, Session, node);

}

public CreateNode(GraphHelper graphHelper) {

super(graphHelper);

}

@Override

public void on\_ready(String res) {

try {

node.IDinAPI = IntFromJsonObject(res, "id");

} catch (JSONException e) {

node.IDinAPI = Integer.getInteger("a");

}

super.on\_ready(res);

}

@Override

public void Send() {

String params = "token="+session+"&"+GetGraphIdView()+"="+graph.IDinAPI;

params+="&x="+node.X+"&y="+node.Y+"&name="+node.GetName();

send(GrapsParams.Url(ctx)+"node/create", params);

}

public static void NodeCreate(Activity ctx, Graph graph, Node node)

{

CreateNode create = new CreateNode(ctx, graph, GrapsParams.Session, node);

create.SendStop();

if(!create.Ready)

{

create.ChangeGraphIdView();

create.SendStop();

}

}

}

Приложение 31. Просмотр списка узлов графа API

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

public class ListNodesHelper extends GraphHelper {

public ListNodesHelper(Activity ctx, String session, GraphIdView id) {

super(ctx, session, id);

Method = "GET";

}

public GraphElement\_List graphs = new GraphElement\_List();

public ListNodesHelper(Activity ctx, String session, Graph graph, GraphIdView id) {

super(ctx, session, graph, id);

Method = "GET";

}

public ListNodesHelper(Activity ctx, String session) {

super(ctx, session);

Method = "GET";

}

public ListNodesHelper(ApiHelper helper, GraphIdView id) {

super(helper, id);

Method = "GET";

}

public ListNodesHelper(ApiHelper helper) {

super(helper);

Method = "GET";

}

@Override

public void Send() {

String params = "token="+session+"&"+GetGraphIdView()+"="+graph.IDinAPI;

send(GrapsParams.Url(ctx)+"node/list", params);

}

@Override

public void on\_ready(String res) {

if(Did)

return;

graph.ClearNodes();

try {

Did = true;

JSONArray arrayGraphs = new JSONArray(res);

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

{

JSONObject jsonObject = arrayGraphs.getJSONObject(i);

Node graph = new Node(this.graph);

graph.IDinAPI = jsonObject.getInt("id");

graph.X = Float.valueOf(String.valueOf(jsonObject.getDouble("x")));

graph.Y = Float.valueOf(String.valueOf(jsonObject.getDouble("y")));

graph.SetName(jsonObject.getString("name"));

this.graph.AddNode(graph);

}

} catch (JSONException e) {

e.printStackTrace();

}

Did = true;

super.on\_ready(res);

}

public static GraphElement\_List GetNodes(Activity ctx, Graph graph)

{

ListNodesHelper helper = new ListNodesHelper(ctx, GrapsParams.Session, graph, GraphIdView.graph);

helper.SendStop();

if(helper.Ready)

helper.on\_ready();

else

{

helper.ChangeGraphIdView();

helper.SendStop();

if(helper.Ready)

helper.on\_ready();

}

GraphElement\_List nodes = new GraphElement\_List(graph, GraphElementName.Node);

return nodes;

}

}

Приложение 32. Базовый класс для связей графа API

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

public class NodeHelper extends GraphHelper{

public Node node;

public NodeHelper(Activity ctx, String session, Graph graph, GraphIdView id) {

super(ctx, session, graph, id);

this.graph = graph;

Method = "PUT";

}

public NodeHelper(Activity ctx, String session, Graph graph) {

this(ctx, session, graph, GraphIdView.graph);

}

public NodeHelper(Activity ctx, Graph graph, String session, Node node) {

this(ctx, session, graph, GraphIdView.graph);

this.node = node;

}

public NodeHelper(GraphHelper graphHelper)

{

this(graphHelper.ctx, graphHelper.session, graphHelper.graph, graphHelper.id);

}

public NodeHelper(GraphHelper graphHelper, Node node)

{

this(graphHelper.ctx, graphHelper.session, graphHelper.graph, graphHelper.id);

this.node = node;

}

}

Приложение 33. Создание связей графа API

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import org.json.JSONException;

public class CreateLink extends LinkHelper{

public CreateLink(GraphHelper helper, Link link) {

super(helper, link, GraphIdView.graph);

}

public CreateLink(Activity ctx, String session, Graph graph, Link link)

{

super(ctx, session, graph, link);

}

@Override

public void on\_ready(String res) {

try {

link.IDinAPI = IntFromJsonObject(res, "id");

} catch (JSONException e) {

link.IDinAPI = Integer.getInteger("a");

}

super.on\_ready(res);

}

public static void LinkCreate(Activity ctx, Graph graph, Link link)

{

CreateLink createLink = new CreateLink(ctx, GrapsParams.Session, graph, link);

createLink.SendStop();

if(createLink.Ready)

createLink.on\_ready();

}

@Override

public void Send() {

float value = 0;

try

{

if(link.ValueVisible)

value = link.Value;

}

catch (Exception ex)

{

}

Node source = link.Source();

int sourceID = source.IDinAPI;

Node target = link.Target();

int targetID = target.IDinAPI;

String params = "token="+session;

params+="&source="+link.IDsourceAPI()+"&target="+link.IDtargetAPI()+"&value="+value;

send(GrapsParams.Url(ctx)+"link/create", params);

}

}

Приложение 34. Просмотр списка связей графа API

package com.example.lab3\_lab4\_graphbuilder\_sidorov493;

import android.app.Activity;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

public class ListLinksHelper extends GraphHelper {

public ListLinksHelper(Activity ctx, String session, GraphIdView id) {

super(ctx, session, id);

Method = "GET";

}

public ListLinksHelper(Activity ctx, String session, Graph graph, GraphIdView id) {

super(ctx, session, graph, id);

Method = "GET";

}

public ListLinksHelper(Activity ctx, String session) {

super(ctx, session);

Method = "GET";

}

public ListLinksHelper(ApiHelper helper, GraphIdView id) {

super(helper, id);

Method = "GET";

}

public ListLinksHelper(ApiHelper helper) {

super(helper);

Method = "GET";

}

@Override

public void Send() {

String params = "token="+session+"&"+GetGraphIdView()+"="+graph.IDinAPI;

send(GrapsParams.Url(ctx)+"link/list", params);

}

@Override

public void on\_ready(String res) {

if(Did)

return;

graph.ClearLinks();

try {

JSONArray arrayGraphs = new JSONArray(res);

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

{

Did = true;

JSONObject jsonObject = arrayGraphs.getJSONObject(i);

Link graph = new Link(this.graph);

int id = jsonObject.getInt("id");

int source = jsonObject.getInt("source");

int target = jsonObject.getInt("target");

source = this.graph.IdNodeFromAPI(source);

target = this.graph.IdNodeFromAPI(target);

graph.SetNodes(source, target);

graph.Value = Float.valueOf(String.valueOf(jsonObject.getDouble("value")));

graph.ValueVisible = true;

graph.Orientation = true;

graph.IDinAPI = id;

this.graph.AddLink(graph);

}

} catch (Exception e) {

e.printStackTrace();

}

Did = true;

super.on\_ready(res);

}

public static GraphElement\_List GetLinks(Activity ctx, Graph graph)

{

ListLinksHelper helper = new ListLinksHelper(ctx, GrapsParams.Session, graph, GraphIdView.graph);

helper.SendStop();

if(helper.Ready)

helper.on\_ready();

else

{

helper.ChangeGraphIdView();

helper.SendStop();

if(helper.Ready)

helper.on\_ready();

}

graph = helper.graph;

GraphElement\_List links = new GraphElement\_List(graph, GraphElementName.Link);

return links;

}

}