Комитет по образованию Правительства Санкт-Петербурга

**САНКТ-ПЕТЕРБУРГСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОНЫХ ТЕХНОЛОГИЙ**

**Отчет по практической работе № 3**

**МДК 01.03 Разработка мобильных приложений**

**Тема: Разработка интерактивного графического приложения**

Выполнил

студент Группы 493

сидоров антон дмитриевич

Проверила Фомин А. В.

Оценка \_\_\_\_\_\_\_\_

Санкт-Петербург 2022

**Содержание**

[1. Цели работы 3](#_Toc118059753)

[2. Диаграмма базы данных 3](#_Toc118059754)

[3. Макеты экранов приложения и их описание 3](#_Toc118059755)

[4. Программный код 7](#_Toc118059756)

[5. Демонстрация работы приложения 7](#_Toc118059757)

[6. Вывод 13](#_Toc118059758)

[Приложение 14](#_Toc118059759)

# Цели работы

Разработать приложение для обработки изображения несколькими потоками.

# Диаграмма базы данных

Диаграмма базы данных представлена на рисунке 1.

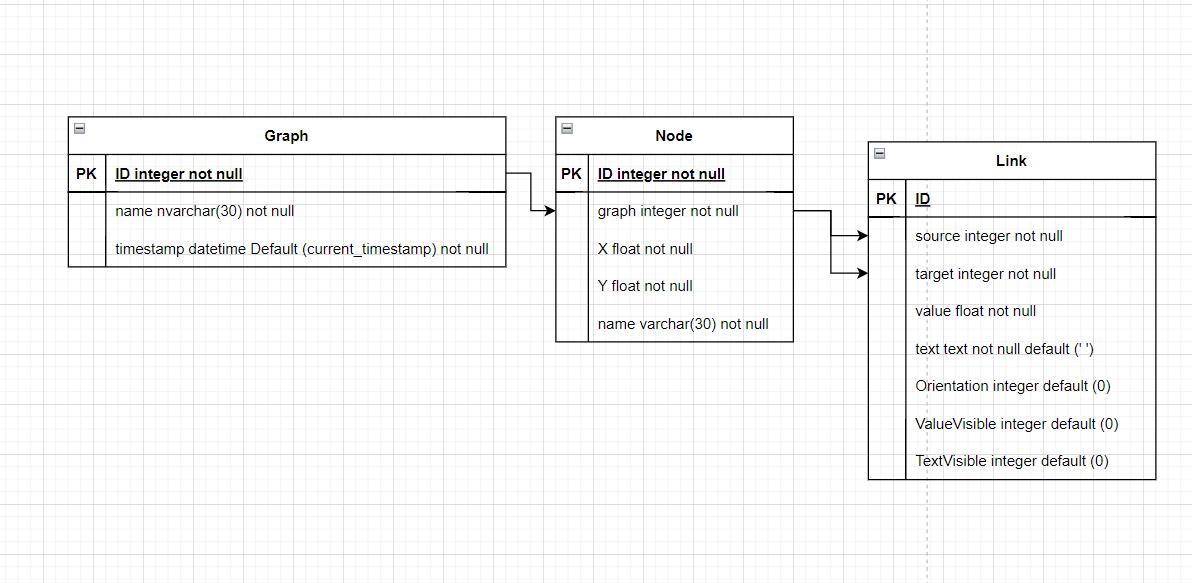


Рисунок 1 – Диаграмма базы данных

# Макеты экранов приложения и их описание

* 1. Примечания по макетам

Окна в работающем приложения могут незначительно отличаться от их макетов.

* 1. Макеты окон

Начальное окно представлено на рисунке 2.

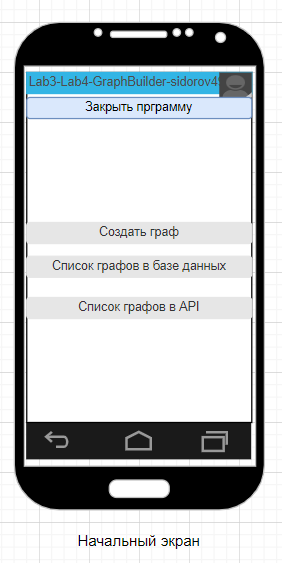


Рисунок 2 – Начальный экран

Окно редактирования графа показано на рисунке 3.

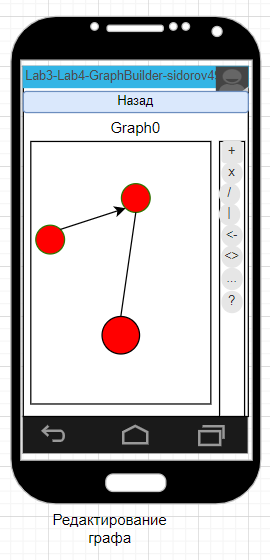


Рисунок 3 – Окно редактирования графа

Окно свойств узла графа показано на рисунке 4.

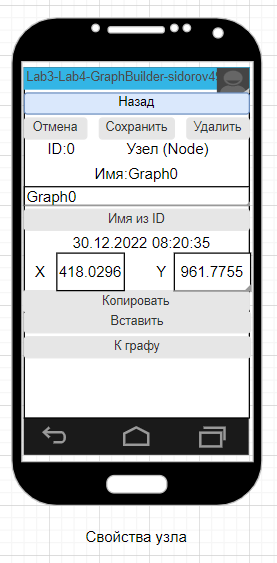


Рисунок 4 – Окно свойств узла графа

Окно свойств графа показано на рисунке 5.

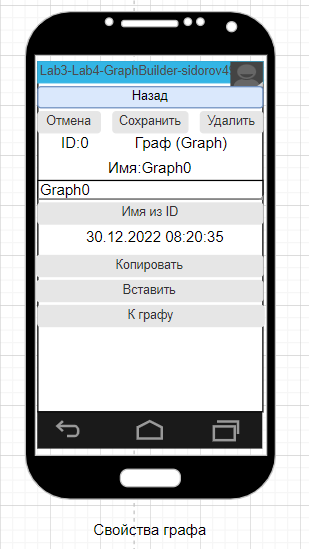


Рисунок 5 – Окно свойств графа

Окно свойств ребра графа показано на рисунке 6.

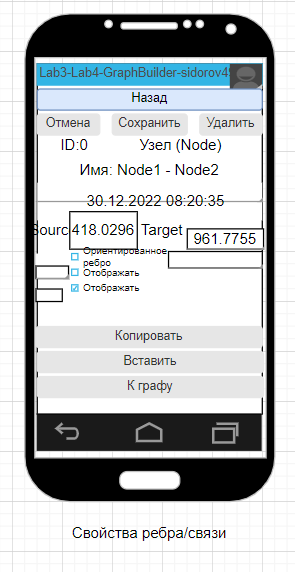


Рисунок 6 – Окно свойств ребра графа

Окно свойств ребра графа показано на рисунке 7. На этом окне может располагаться список графов, а при запуске из редактора графов – список узлов, или список рёбер.

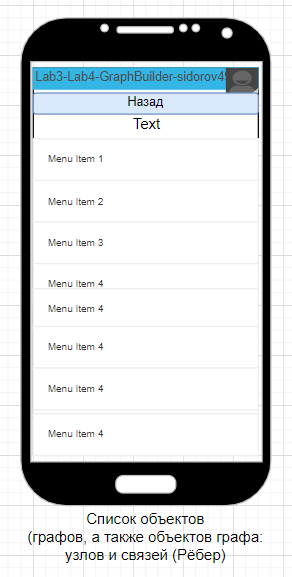


Рисунок 7 – Окно списка объектов графа

# Программный код

Программный код условно можно разбить на 4 части:

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

Далее всё буде рассмотрено подробнее.

* 1. База данных SQLite

Код работы с базой данных представлен в приложении 1.

* 1. Код работы окон приложения

Программный код начального окна представлен в приложении 2.

Программный код окна редактора графов представлен в приложении 3.

Программный код окна свойств объектов графа (графа, узла, ребра) представлен в приложении 4.

Программный код окна списка объектов графа (графа, узла, ребра) в приложении 5.

* 1. Классы объектов графа

Один из этих классов – список объектов графа, представленный в приложении 6

Следующие 3 класса объектов графа, которые используются в визуальном отображении, наследуются от одного базового абстрактного класса *GraphElement*, представленного на рисунке 7. Это классы:

* *Собственно, Граф*, представленный в приложении 8
* *Узел графа*, представленный в приложении 9.
* *Ребро графа (Или, связь графа)*, представленный в приложении 10.

Ещё, вспомогательный класс для отображения узлов и рёбер – цвет элементов графа, представленный в приложении 11.

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 12.

* 1. Код компонентов интерфейса

Рисовальная панель для графа представлена на рисунке 13.

Текст с надписью представлен в приложении 14.

Текст с флажком представлен на рисунке 15

# Демонстрация работы приложения

Данное приложение будет тестироваться на телефоне *BQ-5731L\_08* с операционной системой *Android 9*.

Откроем приложение, как показано на рисунке 8.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Запущенное приложение

Откроем редактор графов, как показано на рисунке 9.

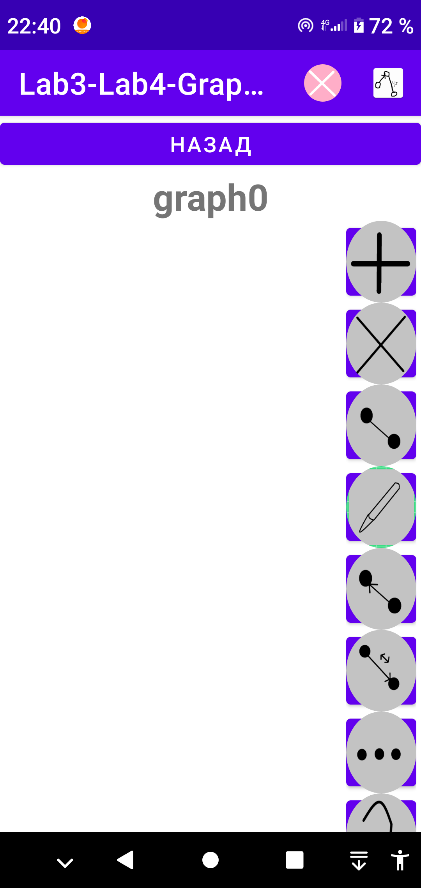


Рисунок 9 – Запущенное приложение

Добавим узел, как показано на рисунке 10.

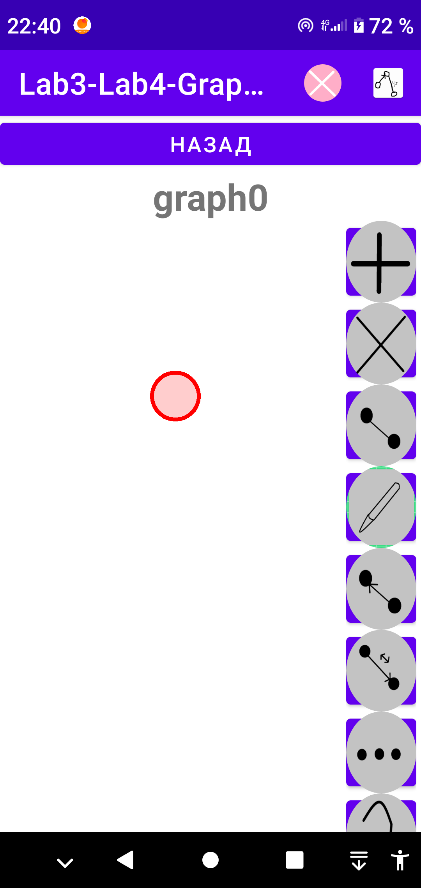


Рисунок 10 – Запущенное приложение

Добавим ещё один узел и выберем его, как показано на рисунке 11.

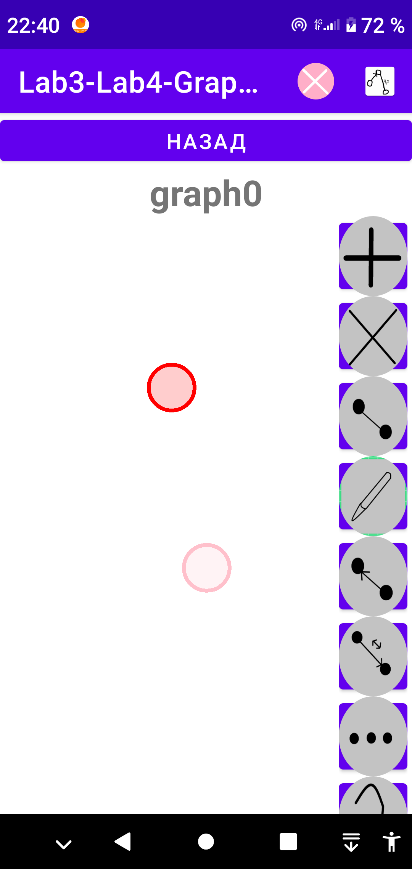


Рисунок 11 – Запущенное приложение

Выберем ещё один узел, как показано на рисунке 12.

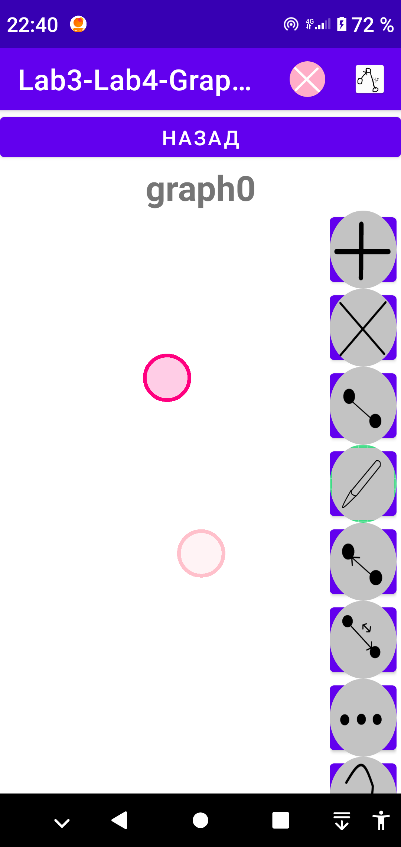


Рисунок 12 – Запущенное приложение

Соединим эти 2 узла неориентированным ребром, как показано на рисунке 13.

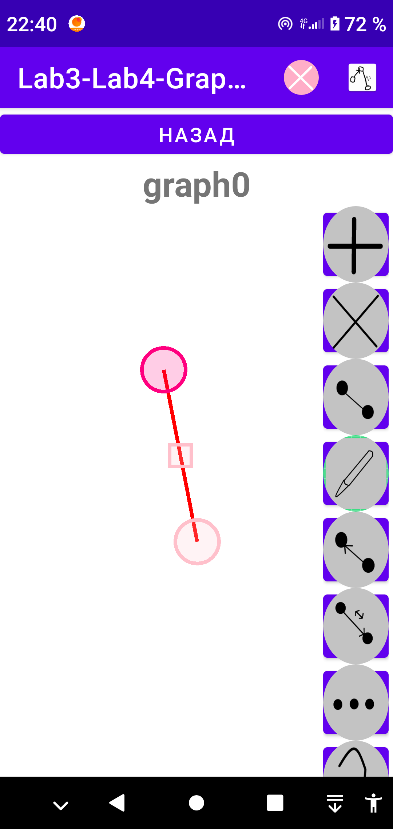


Рисунок 13 – Запущенное приложение

Выберем ребро, как показано на рисунке 14.

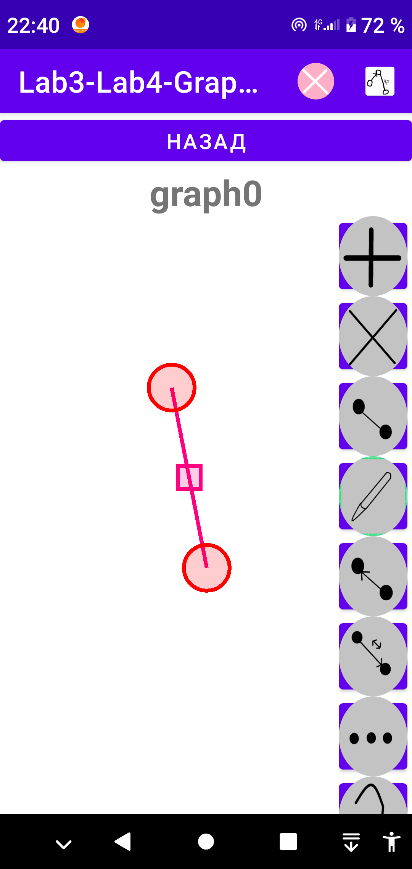


Рисунок 14 – Запущенное приложение

Откроем окно свойств ребра, как показано на рисунке 15.

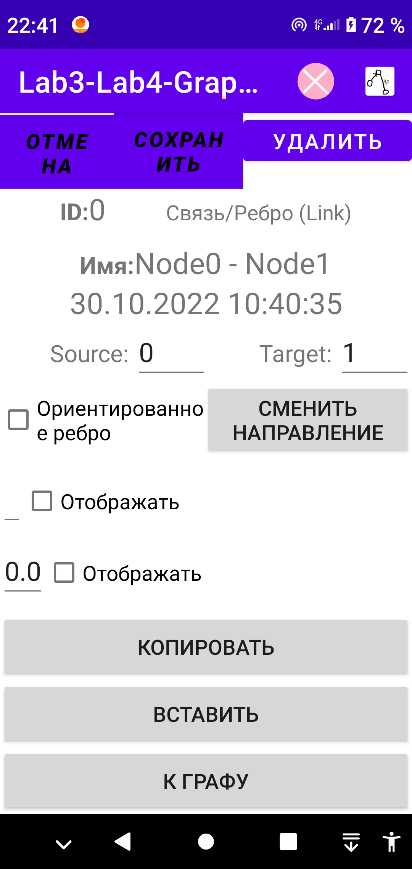


Рисунок 15 – Запущенное приложение

Сделаем ребро ориентированным с добавлением текстовой надписи и числового значения, как показано на рисунке 16.

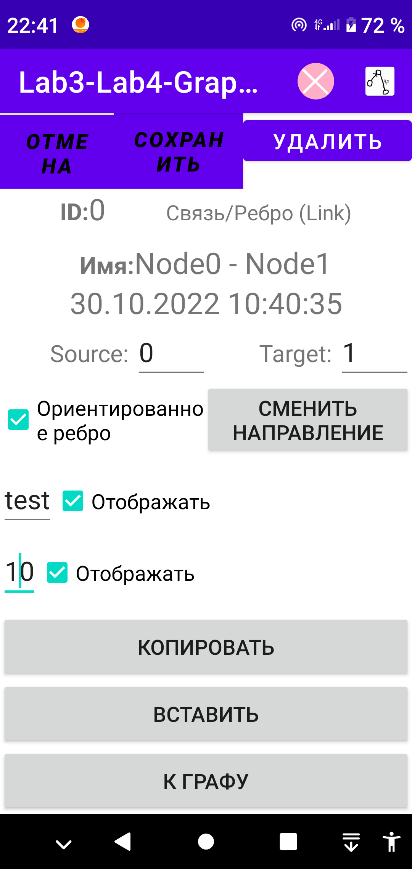


Рисунок 16 – Запущенное приложение

Изменение сохраним, как показано на рисунке 17.

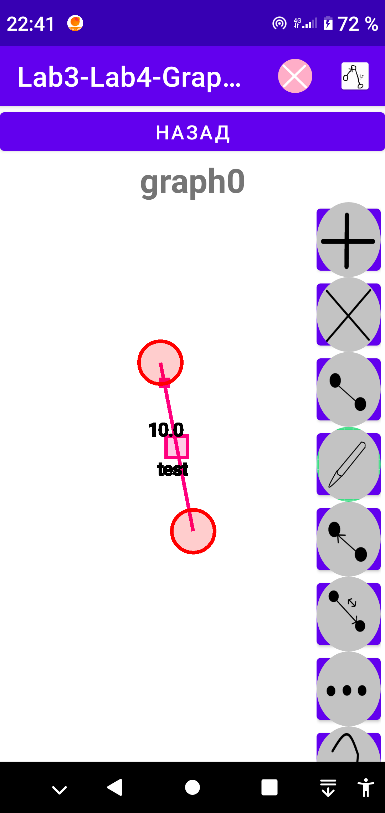


Рисунок 17 – Запущенное приложение

Добавим ещё один узел и выберем 2 узла, как показано на рисунке 18.

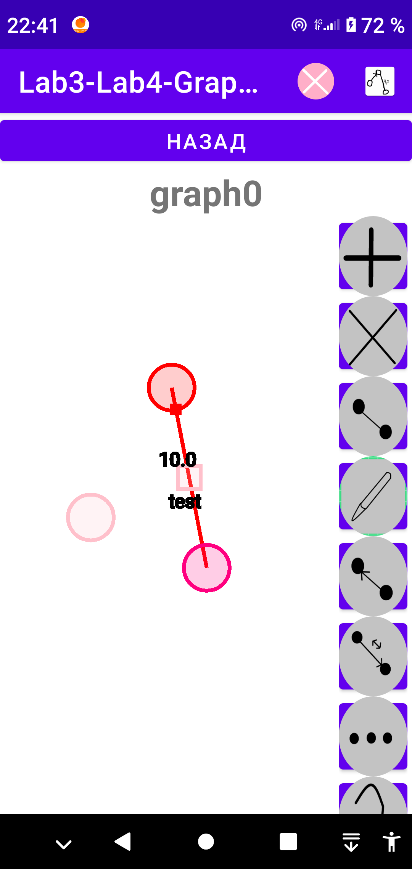


Рисунок 18 – Запущенное приложение

Соединим эти 2 узла ориентированным ребром, как показано на рисунке 19.

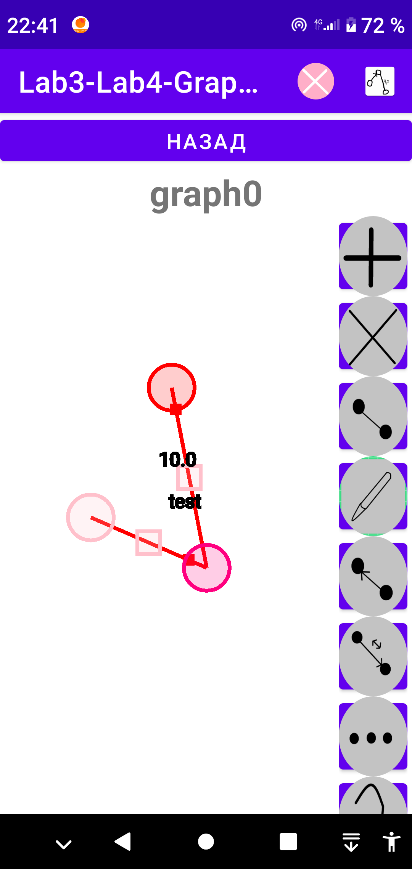


Рисунок 19 – Запущенное приложение

Откроем окно свойств графа, как показано на рисунке 20.

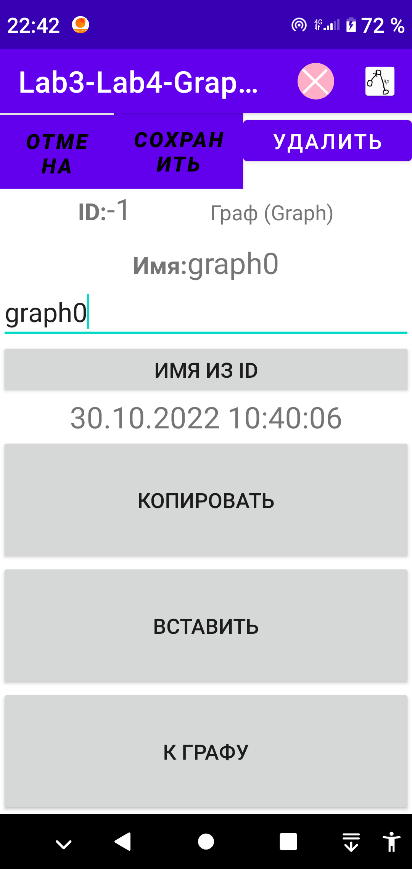


Рисунок 20 – Запущенное приложение

Изменим имя графа, как показано на рисунке 21.



Рисунок 21 – Запущенное приложение

Изменения сохраним.

Добавим ещё один узел, как показано на рисунке 22.

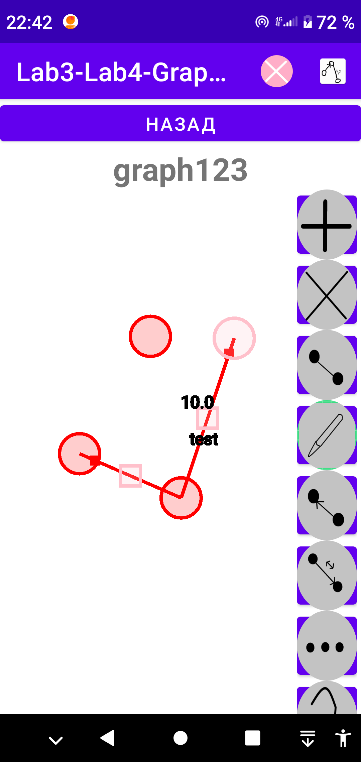


Рисунок 22 – Запущенное приложение

После одного сохранения изменений, все последующие изменения сохраняются автоматически.

Откроем список графов, как показано на рисунке 24.

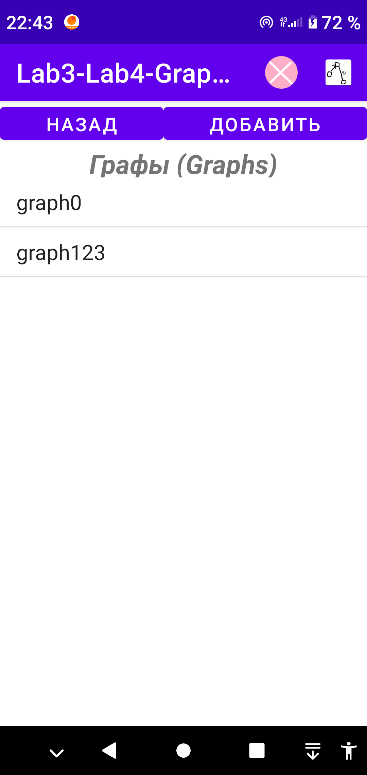


Рисунок 23 – Запущенное приложение

Выберем последний граф в списке (с которым, только что работали), как показано на рисунке 24.

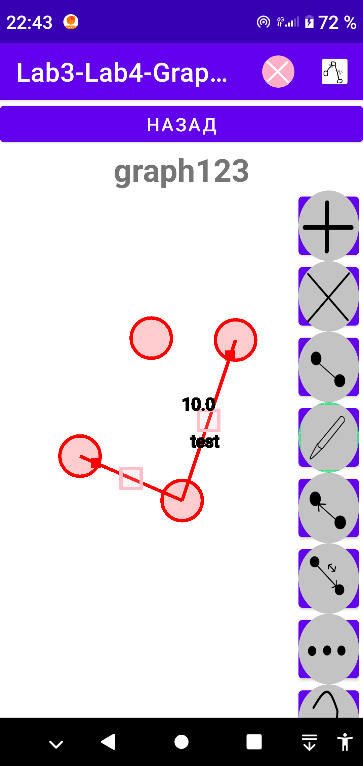


Рисунок 24 – Запущенное приложение

Выберем узел, который без рёбер, как показано на рисунке 25.

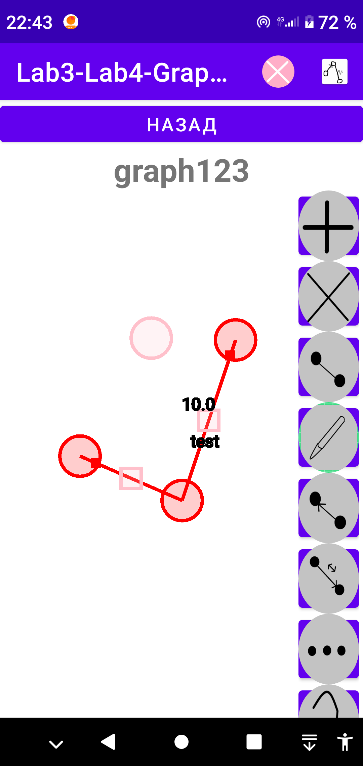


Рисунок 25 – Запущенное приложение

Откроем окно свойств узла, как показано на рисунке 26.

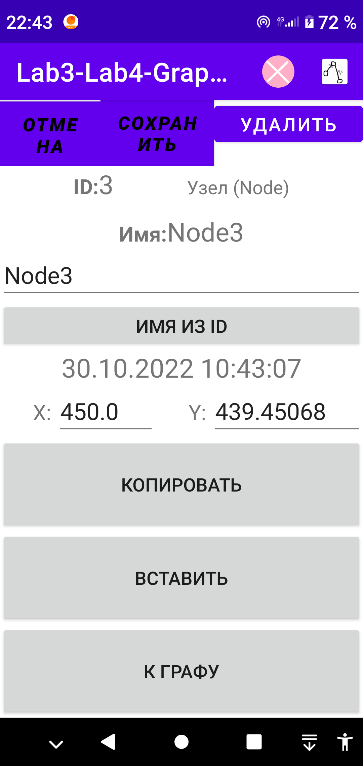


Рисунок 26 – Запущенное приложение

Изменим координаты узла, как показано на рисунке 27.

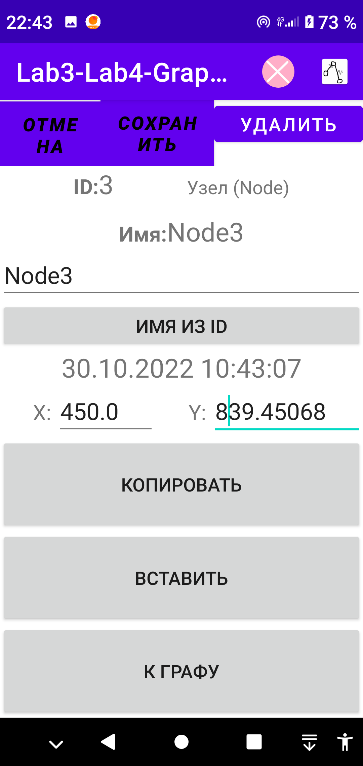


Рисунок 27 – Запущенное приложение

Сохраним изменения, как показано на рисунке 28.

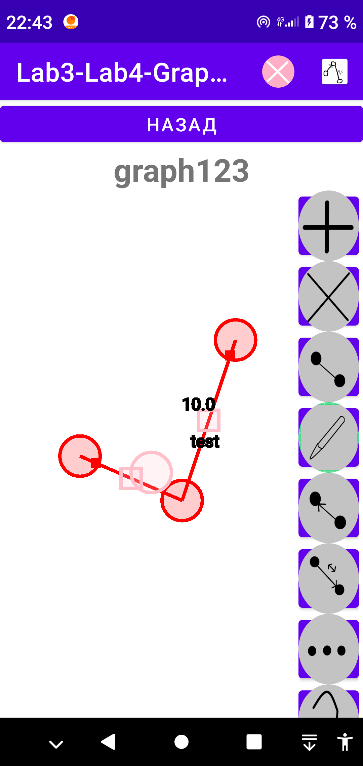


Рисунок 28 – Запущенное приложение

Перейдём к списку узлов и рёбер, как показано на рисунке 29.

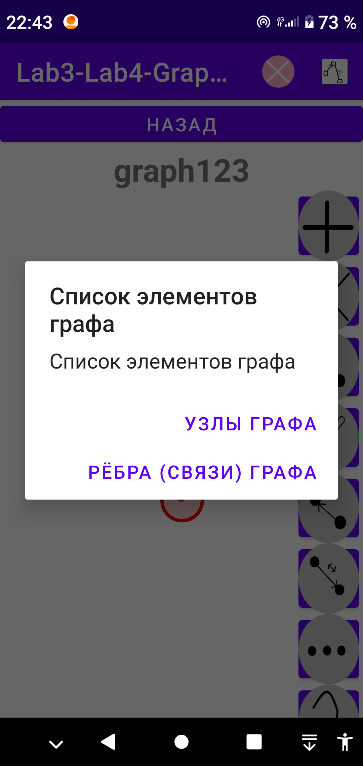


Рисунок 29 – Запущенное приложение

Список узлов показан на рисунке 30.

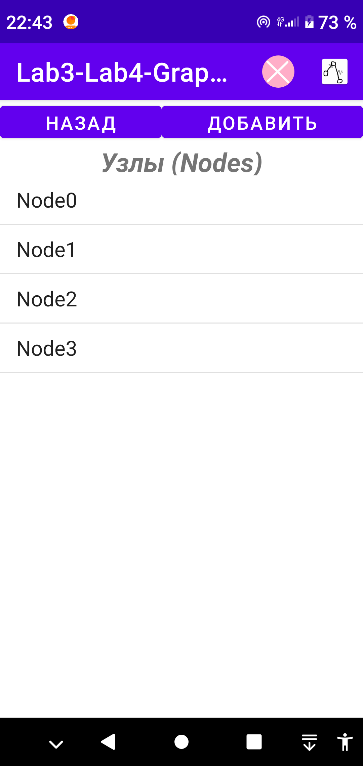


Рисунок 30 – Запущенное приложение

Список рёбер показан на рисунке 31.

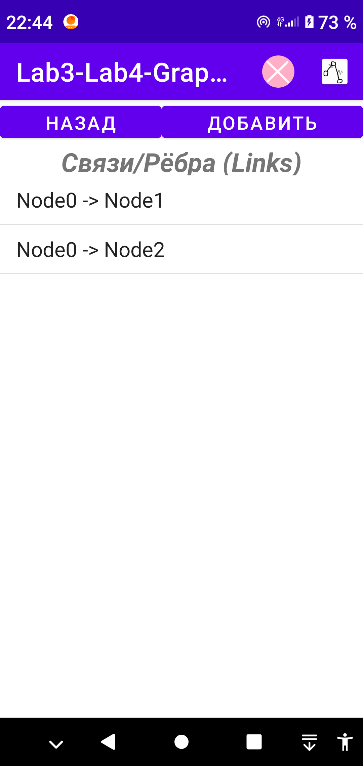


Рисунок 31 – Запущенное приложение

Откроем свойства одного из узлов, как показано на рисунке 32.

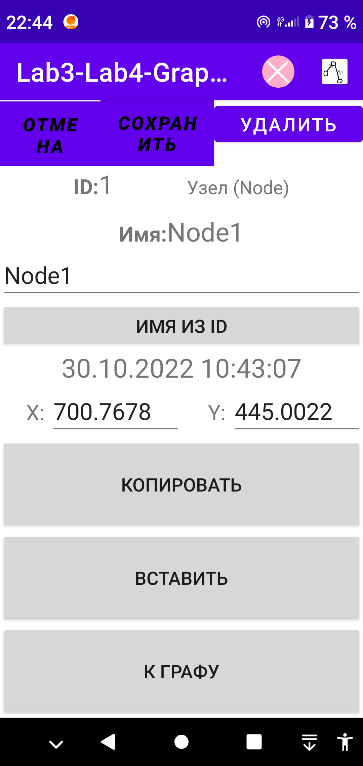


Рисунок 32 – Запущенное приложение

Изменим имя узла, как показано на рисунке 33.

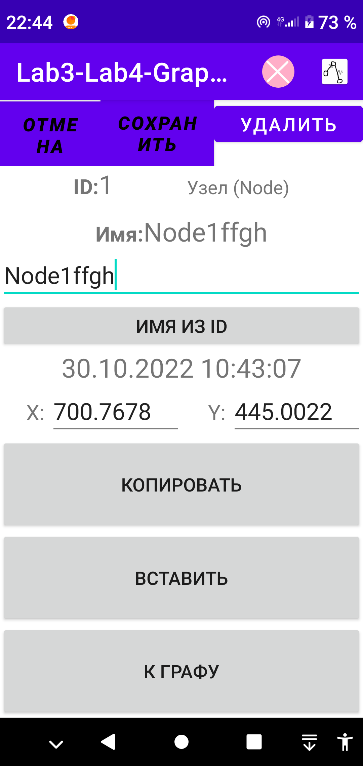


Рисунок 33 – Запущенное приложение

Откроем свойства одного уз узлов, как показано на рисунке 34.

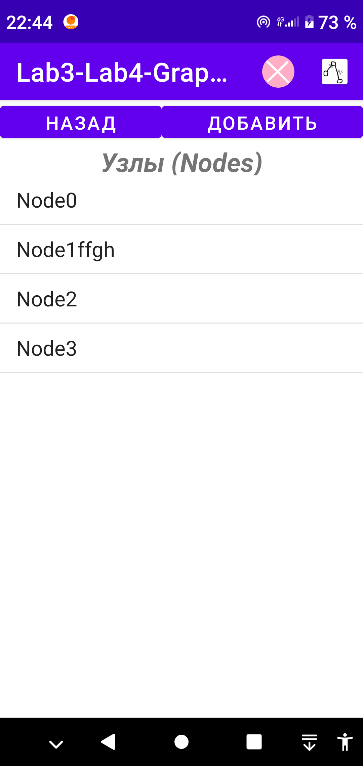


Рисунок 34 – Запущенное приложение

# Вывод

Освоено рисование графов в java.

# Приложение

**Приложение 1. Программный код SQLite**

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 начального окна

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

import androidx.annotation.NonNull;

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;

public class MainActivity extends AppCompatActivity {

Button exit;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

exit = findViewById(R.id.ExitButton);

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());

}

@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)

{

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

startActivity(i);

}

public void GraphList(View v)

{

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

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

startActivityForResult(i, 100);

}

}

Приложение 3. 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.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 {

Button exit;

GraphView graphs;

LinearLayout panelGraphs;

TextView GraphNameView;

@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() {

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

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

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

}

};

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;

}

}

@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 EditNode(View v)

{

if(!graphs.Selection())

return;

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

GrapsParams.GraphElement = graphs.GetSelected();

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);}

@Override

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

GrapsParams.Run\_Graph = false;

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

{

graphs.invalidate();

}

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());

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

{

if(data != null)

{

int id = GrapsParams.elementID();

if(GrapsParams.GraphElement.IsNode())

{

graphs.DeleteNode(id);

}

else if(GrapsParams.GraphElement.IsLink())

{

graphs.DeleteLink(id);

}

else

{

Intent intent = getIntent();

setResult(556, intent);

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)

{

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

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

@Override

public void onClick(DialogInterface dialogInterface, int i) {

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

StartList(v);

}

});

bld.setCancelable(true);

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);

}

}

Приложение 4. Свойства объектов графа

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.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);

}

});

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);

}

});

}

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());

}

}

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 5. Список объектов графа

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.AppCompatActivity;

import android.content.Context;

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;

public Graph Graph()

{

return graphs.GetGraph();

}

ArrayAdapter<GraphElement> list;

void update\_graphs()

{

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

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

graphs.clear();

graphs.addListGraphs(GrapsParams.graphList);

update\_list();

}

void update\_list()

{

if(!graphs.IsGraph())

graphs.SetGraph(graph);

list.notifyDataSetChanged();

}

public Context GetContext()

{

return this;

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

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

graphs = GrapsParams.graphList;

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);

typeText.setText(graphs.GetName());

elementList = findViewById(R.id.listElements1);

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) {

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

GrapsParams.GraphElement = graphs.get(i);

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();

}

@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)

{

Intent intent = getIntent();

setResult(554, intent);

finish();

}

@Override

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

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

{

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(requestCode==556|| resultCode == 556)

{

try {

if (data != null) {

int id = GrapsParams.elementID();

if (GrapsParams.GraphElement.IsNode()) {

graph.DeleteNode(id);

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

graph.DeleteLink(id);

}

else if(GrapsParams.GraphElement.IsGraph())

{

Graph graph = GrapsParams.GraphElement.Graph();

GrapsParams.DB.delete\_graph(graph);

update\_graphs();

return;

}

}

}

catch(Exception ex)

{

}

}

if(GrapsParams.graphList.IsGraph())

update\_graphs();

else

update\_list();

super.onActivityResult(requestCode,resultCode,data);

}

public void AddElements(View v)

{

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();

GrapsParams.DB.GetListNodes(graph);

GrapsParams.DB.GetListLinks(graph);

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

startActivityForResult(intent1, 100);

return;

}

startActivityForResult(i, 100);

}

public void PropertyGraph(View v)

{

}

}

Приложение 6. 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 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 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();

addListGraphs(list);

}

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

return super.addAll(graphs);

}

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 7. 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 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;

}

}

# Приложение 8. Собственно, Граф

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

import java.util.ArrayList;

public class Graph extends GraphElement {

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 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());

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());

}

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 9. 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, 450.0f, graph);

}

public Node (Node node, Graph graph)

{

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

}

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());

}

}

Приложение 10. 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 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;

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 Node Target()

{

SetNodes();

return target;

}

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 11. 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 12. Буфер обмена для объектов графа

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

public class GrapsParams {

public static int GraphID = 0;

public static Graph Graph;

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;

}

Приложение 13. 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()

{

return selectedNowLink;

}

public Node GetSelectedNode()

{

return graph.GetNode(selectedNowNode);

}

public Link GetSelectedLink()

{

return graph.GetLink(selectedNowLink);

}

public void SetNode(int index, Node node)

{

graph.SetNode(index, node);

}

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 = GrapsParams.GraphElement.Link();

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;

}

catch (Exception ex)

{

}

}

}

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

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);

SetGraph(graph);

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();

float x = n.X + dX;

float y = n.Y + dy;

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

SetGraph(graph);

return node;

}

public void DeleteNode(int index)

{

int delete = index;

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);

}

public void DeleteLink (int id)

{

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

{

graph.DeleteLink(id);

}

}

public void ChangeOrientationLink(int id)

{

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

{

graph.GetLink(id).ChangeOrientationLink();

SetGraph(graph);

}

}

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 sx = source.X - dX;

float sy = source.Y - dY;

float tx = target.X - dX;

float ty = target.Y - dY;

p.setStyle(Paint.Style.FILL);

p.setColor(color.GetBorderColor());

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

if (l.Orientation) {

float xRad, yRad;

float URad = (rad\*2f)/(sx + tx);

URad = 1f - URad;

xRad = (tx - sx)\*URad;

yRad = (ty - sy) \* URad;

xRad += sx;

yRad += sy;

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;

}

@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;

if(dY == 0)

sY = y;

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;

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;

SetGraph(graph);

return true;

}

}

selectedNowLink = -1;

SetGraph(graph);

toach = true;

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;

dY = y - sY;

}

SetGraph(graph);

return true;

}

//break;

case MotionEvent.ACTION\_UP:

{

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;

}

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 14. 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 15. 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);

}

}