КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА

Санкт-Петербургское государственное
бюджетное профессиональное образование учреждение
«Колледж информационных технологий»

ОТЧЕТ
по практической работе
«Разработка интерактивного графического приложения»
по МДК01.03.

Работу выполнил студент 493 гр.:
Кашицын Артем Андреевич
Преподаватель:
Фомин Александр Валерьевич

Санкт-Петербург 2022

ОГЛАВЛЕНИЕ

ОПИСАНИЕ ИНТЕРФЕЙСА 3
СТРУКТУРА БАЗЫ ДАННЫХ 6
ДЕМОНСТРАЦИЯ ФУНКЦИЙ 9
ИТОГ РАБОТЫ 19

ОПИСАНИЕ ИНТЕРФЕЙСА

Макет основного экрана приложения отображен на рисунке 1.

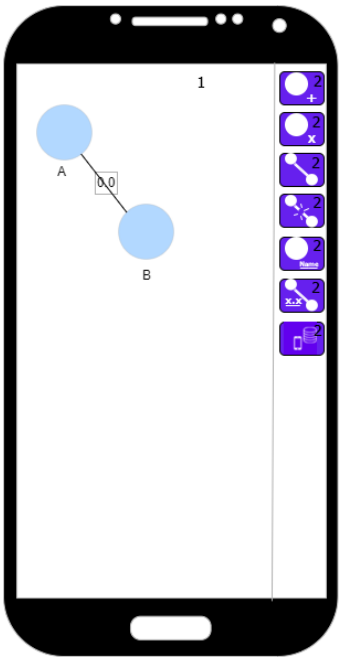

Рисунок 1 – Макет основного экрана

На основном экране расположены элементы под следующими номерами:

1 – SurfaceView

2 – Button

Макет экрана управления базой данных отображен на рисунке 2.

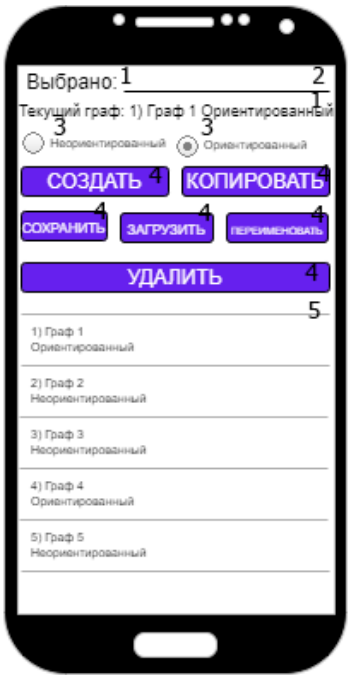

Рисунок 2 – Макет экрана управления базой данных

1 – TextView

2 – EditText

3 – Radio Button

4 – Button

5 – ListView

Макет окна, который всплывает при вводе названия узла или веса ребра показан на рисунке 3.

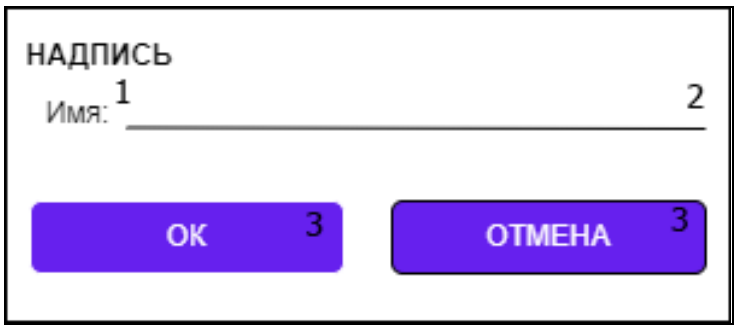

Рисунок 3 – Макет всплывающего окна для ввода значений на графе

1 – TextView

2 – EditText

3 – Button

SurfaceView отображает работу с графом. На этом компоненте рисуются узлы, ребра графа и их значение. Пользователь может выбирать составляющие графа и двигать узлы.

Button – это кнопка, по нажатию которой выполняется определенная функция. На основном экране каждая кнопка имеет иконку, обозначающее действие: добавить, удалить узел, соединить выбранные узлы, удалить соединение, дать название узлу, ввести вес ребра, перейти на форму управления графами в локальной базе данных. На экране управления графами кнопки используются для создания, копирования, сохранения, загрузки, переименования, удаления графа.

TextView используется как подпись для определения пользователем поля ввода данных, также отображает текущий граф, нарисованный на GraphView.

EditText является полем для ввода данных пользователем. Например, ввод названия графа, названия узла, значения веса ребра.

RadioButton – кнопка, которая принимает два значения (нажатое и не нажатое), при этом из нескольких RadioButton можно активировать только один переключатель. Служат для выбора типа графа (ориентированного или неориентированного).

ListView – компонент, отображающий список. Каждый элемент можно выбрать, используется для выбора графа сохраненного в локальной базе данных.

СТРУКТУРА БАЗЫ ДАННЫХ

ER-диаграмма отображена на рисунке 4.

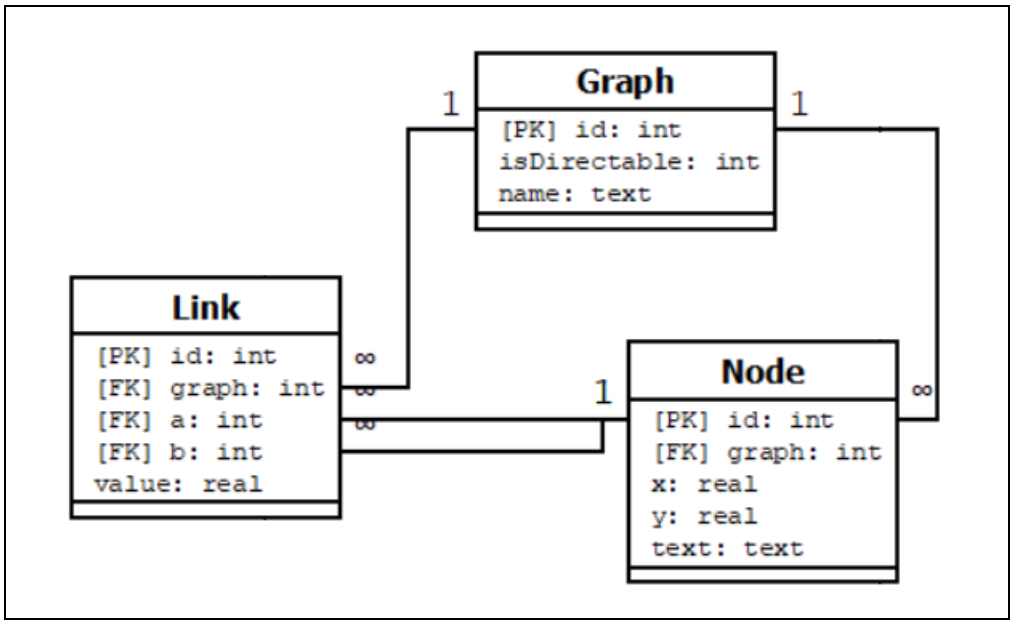

Рисунок 4 – ER-диаграмма

Таблица Graph состоит из следующих столбцов:

id – идентификатор графа

isdirectable – является ли граф ориентированным

name – название графа

Имя	Тип	Размер	Уникальное	Пустое	Ключ
id	integer	-	да	нет	pk
isdirectable	integer	1	нет	нет	-
name	text	10	нет	нет	-

Примеры входных данных:

1|1|Ориентированный

2|0|Неориентированный

Таблица Node состоит из следующих столбцов:

id – идентификатор узла

graph – номер графа, к которому относится узел

x – координата по x

y – координата по y

text – название узла

Имя	Тип	Размер	Уникальное	Пустое	Ключ
id	integer	-	да	нет	pk
graph	integer	-	нет	нет	fk
x	real	-	нет	нет	-
y	real	-	нет	нет	-
text	text	10	нет	да	-

Примеры входных данных:

1|1|424.69934|497.82797|

2|1|474.65314|1215.5176|

3|1|468.15912|882.16174|

4|1|472.65494|157.97491|

5|2|526.1054|767.2114|C

6|2|146.95654|776.7073|B

7|2|357.26178|347.39297|A

Таблица Link состоит из следующих столбцов:

id – идентификатор ребра

graph – номер графа, к которому относится ребро

a – номер точки a

b – номер точки b

value – значение узла

Имя	Тип	Размер	Уникальное	Пустое	Ключ
id	integer	-	да	нет	pk
graph	integer	-	нет	нет	fk
a	integer	-	нет	нет	fk
b	integer	-	нет	нет	fk

value	real	-	нет	нет	-

Примеры входных данных:

1|1|4|1|1.0

2|1|1|3|2.0

3|1|3|2|3.0

4|2|7|6|0.0

5|2|6|5|0.0

6|2|5|7|0.0

ДЕМОНСТРАЦИЯ ФУНКЦИЙ

Иконка приложения отображена на рисунке 5.

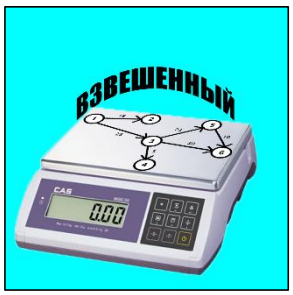

Рисунок 5 – Иконка приложения

После запуска приложения, пользователь видит пустое поле и панель инструментов справа. Вид приложения после запуска показан на рисунке 6.

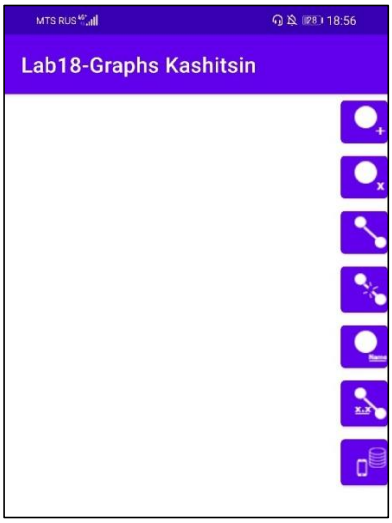

Рисунок 6 – Вид приложения после запуска

Модель данных для базы составляют три класса: Graph, Node, Link.

Graph представляет собой номер графа, хранящийся в базе данных, название графа, является ли граф ориентированным.

Node – координаты x, y узла и его название

Link – два соединенных экземпляра Node, вещественное значение веса ребра.

Компонент SurfaceView является пользовательским GraphView, который занимается логикой поведения графа, и отрисовкой графа.

Отрисовка представляет собой перебор элементов списка соединений и рисование линий по координатам двух точек, если соединение было выбрано, соединение меняет цвет, посередине линии рисуется квадрат, с помощью

которого можно выбрать соединение и задать значение, а если граф является ориентированным, то рисуется стрелка к направлению последней точки соединения. Затем происходит перебор узлов, рисуется круги, выделяются отдельным цветом, если они выбраны, и рисуется название узла.

Класс GraphHelper хранит текущие значение графа, список узлов и соединений, а также имеет функции по управлению этими списками.

Класс DBHelper работает с локальной базой данных приложения

После добавления пользователем точки, создается новый Node с координатами 100 100 и добавляется в список, происходит отрисовка. Добавление точки показано на рисунке 7.

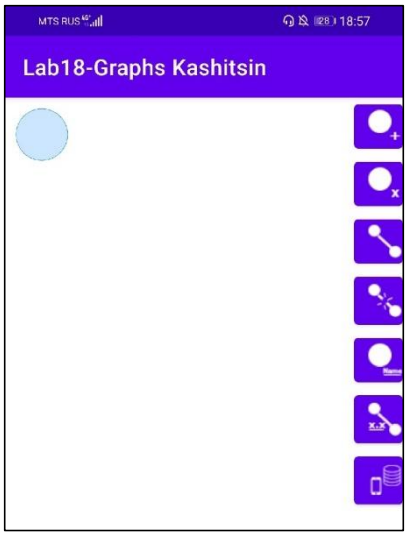

Рисунок 7 – Добавление точки

Пользователь может коснуться отрисованной точки. Срабатывает onTouchEvent на GraphView. Определяются координаты касания, определяется тип движения по экрану. Если пользователь просто нажал на экран, то проводится поиск по списку точек. То есть, если нажатие произошло внутри окружности, то возвращается ее номер, иначе все выбранные точки отменяются. Если точка была выбрана до этого, но выбирается еще одна, то это считается второй выбранной точкой. Выбор точки показан на рисунке 8.

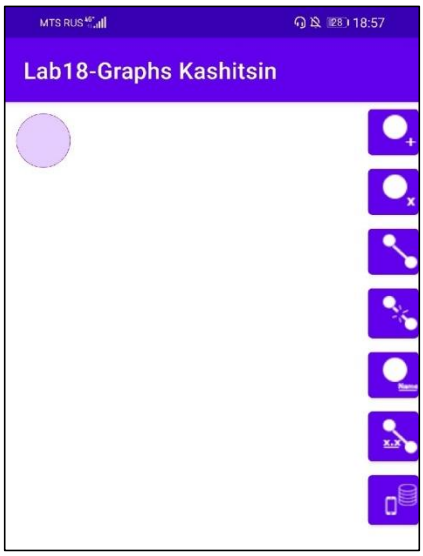

Рисунок 8 – Выбор точки

Пользователь может передвинуть точку. На этот раз, старые координаты запоминаются для плавного перемещения по экрану, при этом выбор точки не спадает. Передвижение показано на рисунке 9.

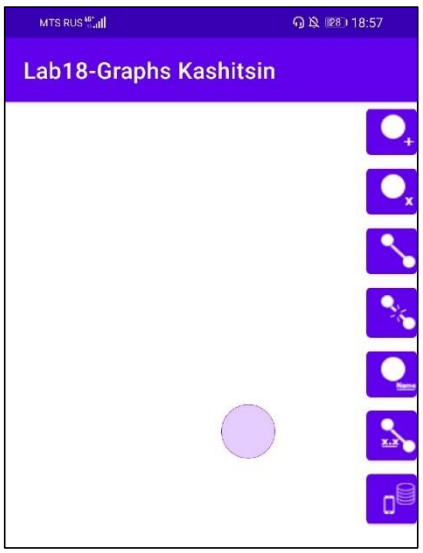

Рисунок 9 – Передвижение точки

Если добавить ещё одну точку, и выбрав её, то если уже существует выбранная точка, новая отдельно окрашивается в другой цвет. Выбор второй точки показан на рисунке 10.

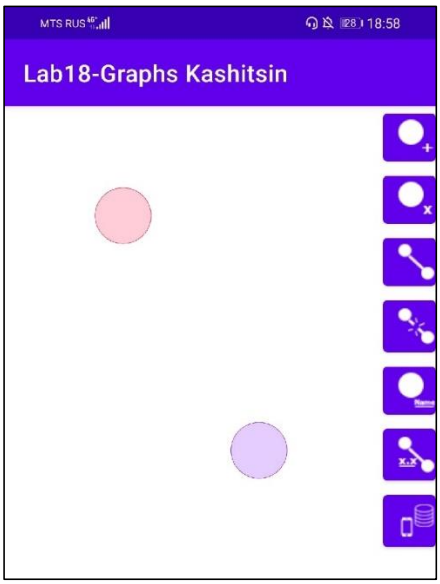

Рисунок 10 – Выбор второй точки

Если выбраны две точки, то пользователь может их соединить. Если граф неориентированный, то две одинаковые точки не могут быть соединены повторно. Если граф ориентированный, то такая ситуация допустима, но только если точки поменяли местами. Создается Link, с выбранными Nodes и добавляется в список, происходит отрисовка. Создание соединения показано на рисунке 11.

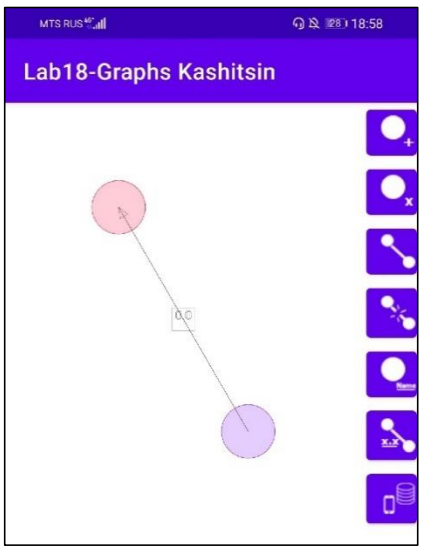

Рисунок 11 – Соединение точек

Пользователь может поменять выбранные точки местами, и создать вторую связь в другом направлении. Прорисовка обеспечивается методом Reverse (повернуть узлы соединения), и проверить есть ли такое соединение,

тогда квадраты со значением ребра будут немного оторваны от линии по разные стороны. Это отображено на рисунке 12.

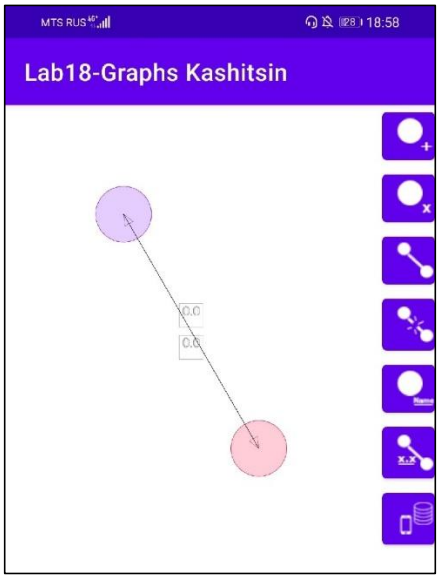

Рисунок 12 – Создание двух связей при ориентированном графе

Нажав на нарисованный квадрат, можно выбрать соединение, и дать значение ребра. Выбор осуществляется также, как и с узлами. Для ввода значения используется окно на рисунке 13.

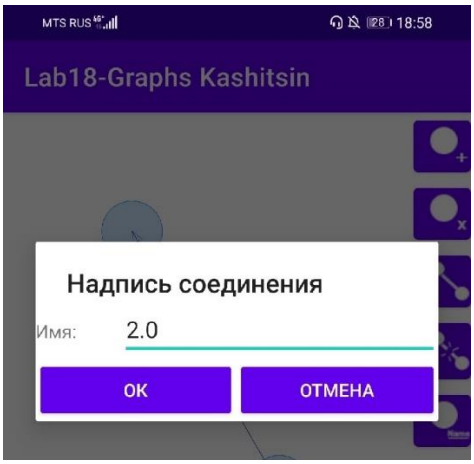

Рисунок 13 – Ввод значений на графе (ребро)

После сохранений значений, пользователь увидит изображение на рисунке 14.

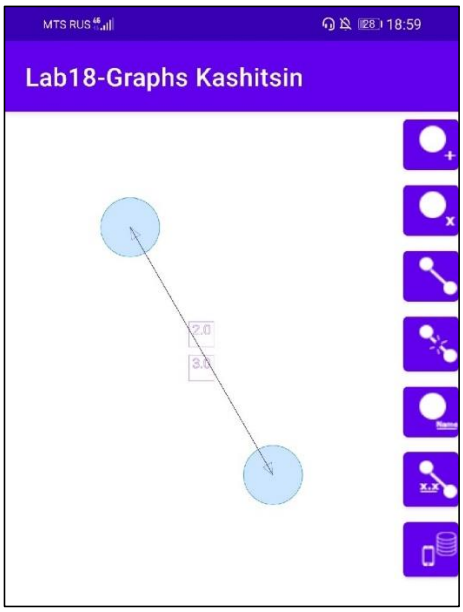

Рисунок 14 – Обеспечение ввода данных для двух соединений

Удалим соединение со значением 3.0. Удаление соединения показано на рисунке 15.

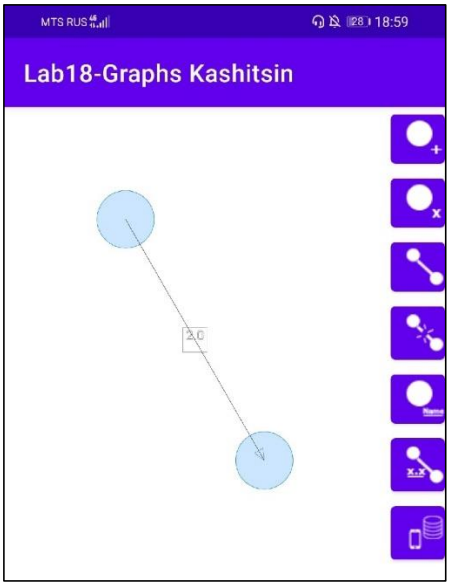

Рисунок 15 – Удаление соединения

Пользователь также может давать название узлам, используя то же окно для ввода значений. Завершенный граф показан на рисунке 16.

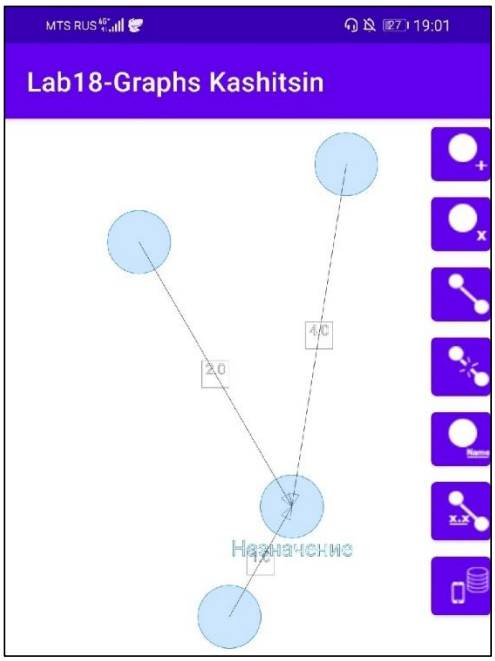

Рисунок 16 – Название для точки

После перехода на форму с управлением графами, введем название «Маршрут» и сохраним граф как ориентированный. Происходит перебор элементов списка с соединением, и индексы точек сохраняются в двумерный массив. Сначала добавляются точки с определенного индекса. Затем из двумерного массива с прибавлением начального индекса, значения добавляются в таблицу соединений. Если граф не был создан, но то что пользователь нарисовал необходимо сохранить, то этот граф создается, и составляющие получает значение индекса. Сохранение показано на рисунке 17.

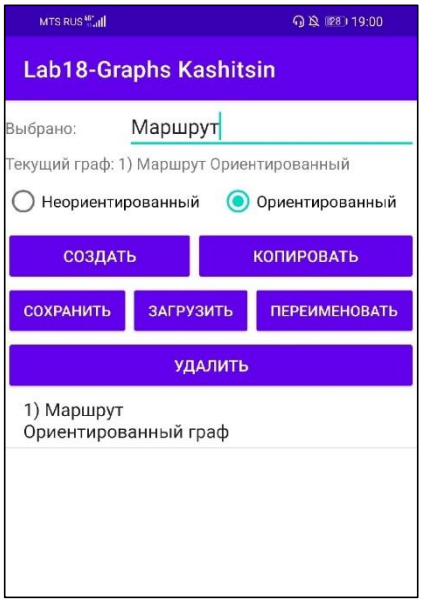

Рисунок 17 – Сохранение графа

Создадим новый неориентированный граф с названием «Треугольник» и выберем его нажатием, а затем загрузим. Загрузка происходит созданием списка идентификаторов точек, после добавления точек, при загрузке соединений, индексы после загрузки точек сравниваются с созданным списком. Если произошло соответствие, то создается соединение. Загрузки с базы данных переводятся в списки класса GraphHelper. Теперь видим, что текущий граф поменялся. Создание и загрузка графа показана на рисунке 18.

Рисунок 18 – Создание и загрузка графа

Создадим граф, показанный на рисунке 19.

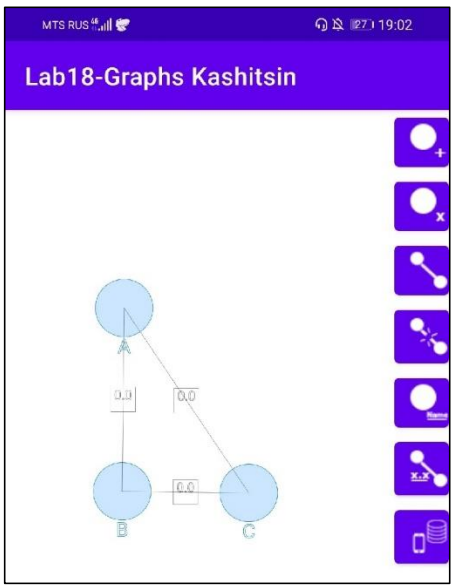

Рисунок 19 – Рисовка неориентированного графа

Теперь сохраним созданный граф, и скопируем его. Копирование представлено выборкой всех точек и соединений и загрузка их при изменении идентификаторов. Копирование показано на рисунке 20.

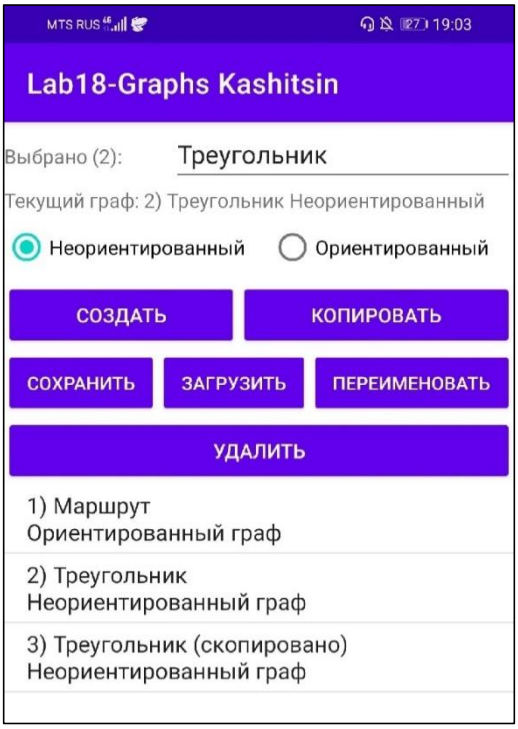

Рисунок 20 – Копирование графа

Переименуем скопированный граф в «Треугольник ABC». Переименование показано на рисунке 21.

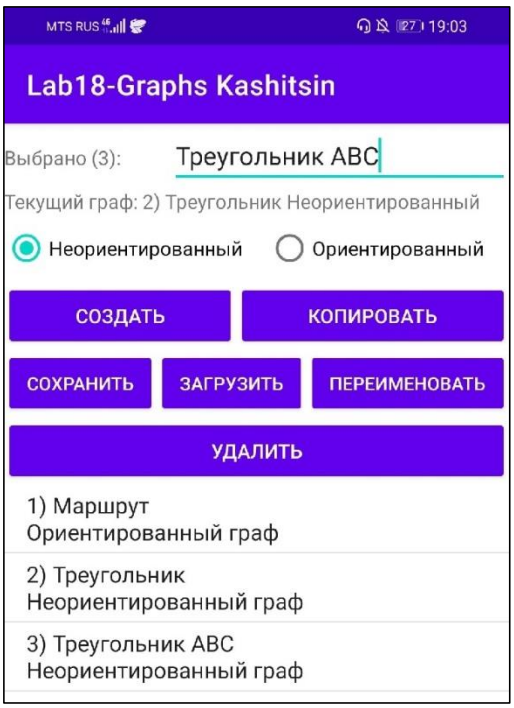

Рисунок 21 – Переименование графа

Удалим предыдущую версию треугольника. Удаление показано на рисунке 22.

Рисунок 22 – Удаление графа

Загрузим версию «Треугольник ABC». Граф показан на рисунке 23.

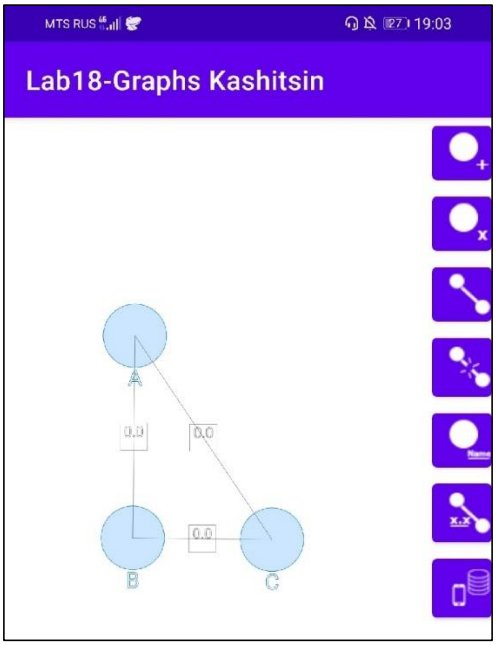

Рисунок 23 – Загрузка скопированного графа

ИТОГ РАБОТЫ

Ссылка на репозиторий с готовым проектом: https://github.com/aza1rat/LabGraphs