

КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА

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

ОТЧЕТ
по практической работе
«Разработка интерактивного графического приложения с использованием
REST API»
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.....	8
ОПИСАНИЕ ФУНКЦИЙ API	11
ДЕМОНСТРАЦИЯ ФУНКЦИЙ	21
ИТОГ РАБОТЫ	46

ОПИСАНИЕ ИНТЕРФЕЙСА

Макет экрана авторизации отображен на рисунке 1.

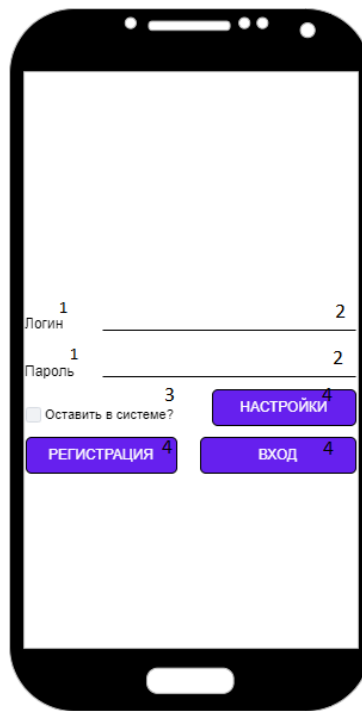


Рисунок 1 – Макет экрана авторизации

На экране авторизации расположены элементы под следующими номерами:

1 – TextView

2 – EditText

3 – CheckBox

4 – Button

Макет экрана регистрации отображен на рисунке 2.

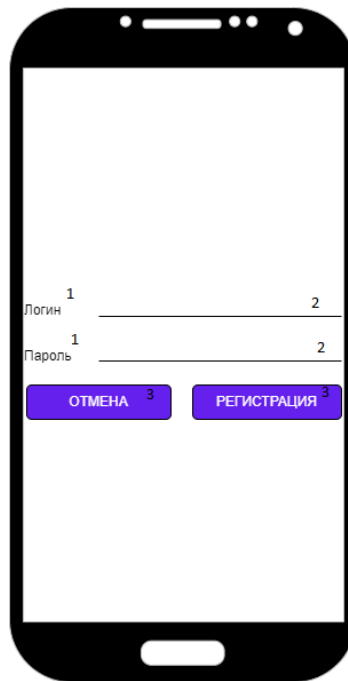


Рисунок 2 – Макет экрана регистрации

На экране регистрации расположены элементы под следующими номерами:

- 1 – TextView
- 2 – EditText
- 3 – Button

Макет экрана управления графами отображен на рисунке 3.

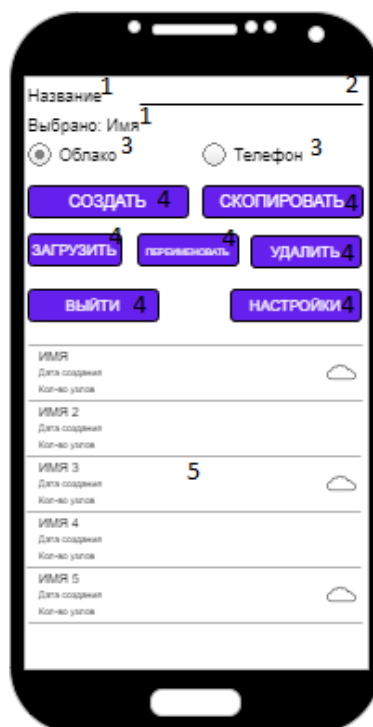


Рисунок 3 – Макет экрана управления графами

На экране управления графами расположены элементы под следующими номерами:

- 1 – TextView
- 2 – EditText
- 3 – RadioButton
- 4 – Button
- 5 – ListView

Макет основного экрана для интерактивного взаимодействия с графом отображен на рисунке 4.

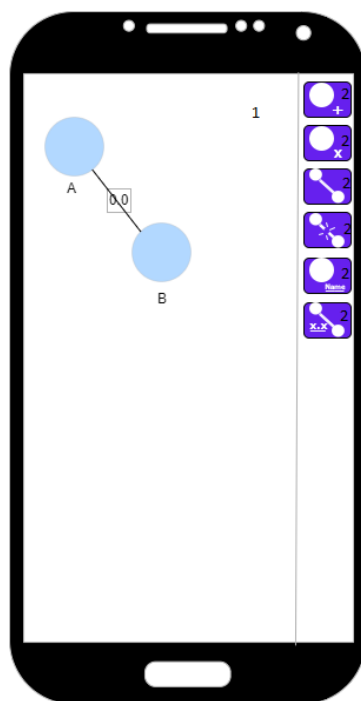


Рисунок 4 – Макет основного экрана

На основном экране расположены элементы под следующими номерами:

- 1 – SurfaceView
- 2 – Button

Макет экрана настроек аккаунта отображен на рисунке 5.

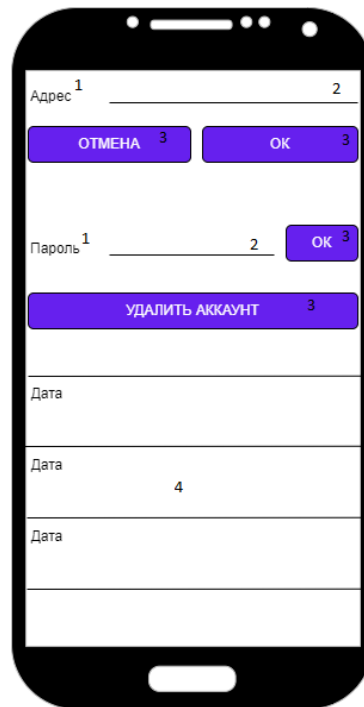


Рисунок 5 – Макет экрана настроек аккаунта

1 – TextView

2 – EditText

3 – Button

4 - ListView

Макет окна, который всплывает при вводе названия узла или веса ребра показан на рисунке 6.

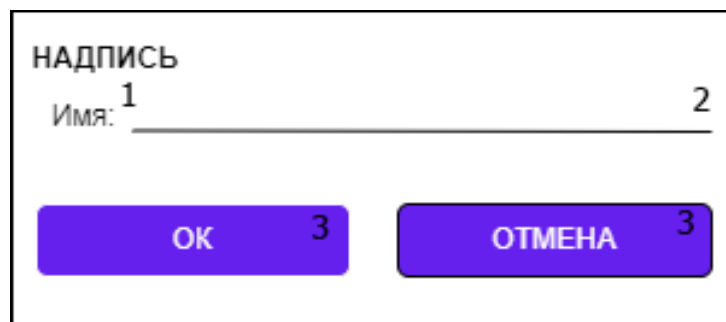


Рисунок 6 – Макет всплывающего окна для ввода значений на графе

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 экране авторизации кнопки имеют следующие действия: переход на форму с настройками, переход на форму с регистрацией нового аккаунта, а также авторизация с введенными данными. На экране регистрации расположены кнопки для возврата на форму авторизации и регистрация с введенными данными. На экране управления графами кнопки используются для создания, копирования, загрузки, переименования, удаления графа и перехода на форму настроек аккаунта, а также выход из текущего аккаунта. 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 настроек кнопки имеют подтверждающую функцию: сохранение нового адреса или возвращение к значению по умолчанию, сохранение нового пароля аккаунта, удаления аккаунта.

TextView используется как подпись для определения пользователем поля ввода данных, также отображает выбранный граф.

EditText является полем для ввода данных пользователем. Например, ввод названия графа, названия узла, значения веса ребра, логина и пароля аккаунта, адреса сервера.

RadioButton – кнопка, которая принимает два значения (нажатое и не нажатое), при этом из нескольких RadioButton можно активировать только один переключатель. Служат для выбора типа сохранения графа (сохранение на сервер или в локальную базу данных телефона).

CheckBox – флажок который принимает два значения (нажатое и не нажатое). Служит для подтверждения сохранения данных для входа в аккаунт

ListView – компонент, отображающий список. Каждый элемент можно выбрать. В данном приложении используется пользовательский вид адаптера – компонента для отображения списка в определенном виде. Используется для выбора графа и сессий входов в систему.

СТРУКТУРА БАЗЫ ДАННЫХ

ER-диаграмма отображена на рисунке 7.

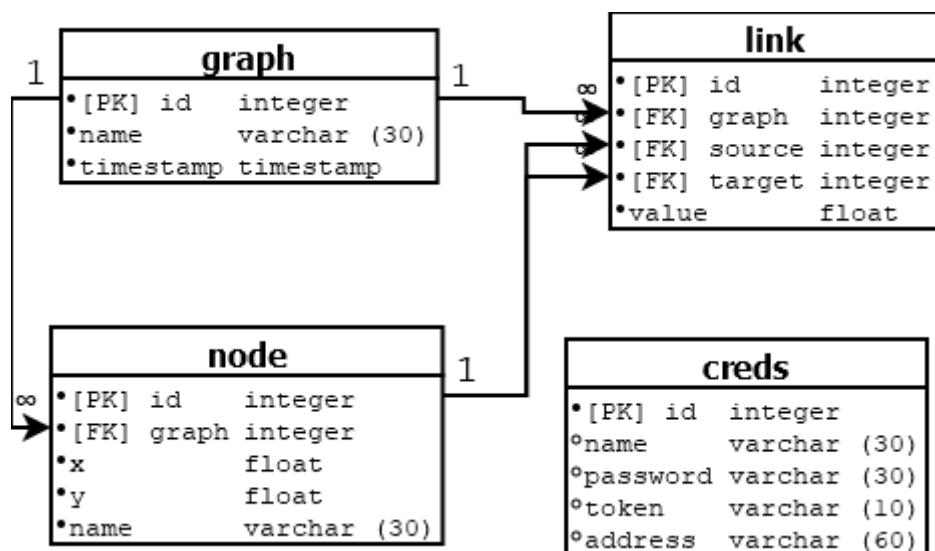


Рисунок 7 – ER-диаграмма

Таблица graph состоит из следующих столбцов:

id – идентификатор графа

name – название графа

timestamp – время создания графа в формате unix timestamp

Имя	Тип	Размер	Уникальное	Пустое	Ключ
id	integer	-	да	нет	pk
name	varchar	30	нет	нет	-
timestamp	timestamp	-	нет	нет	-

Примеры входных данных:

1|Имя|1668412371

2|Имя 2|1668412391

Таблица node состоит из следующих столбцов:

id – идентификатор узла

graph – номер графа, к которому относится узел

x – координата по x

y – координата по y

name – название узла

Имя	Тип	Размер	Уникальное	Пустое	Ключ
id	integer	-	да	нет	pk
graph	integer	-	нет	нет	fk
x	float	-	нет	нет	-
y	float	-	нет	нет	-
name	varchar	30	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

source – номер точки source

target – номер точки target

value – значение узла

Имя	Тип	Размер	Уникальное	Пустое	Ключ
id	integer	-	да	нет	pk
graph	integer	-	нет	нет	fk
source	integer	-	нет	нет	fk
target	integer	-	нет	нет	fk
value	float	-	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

Таблица creds состоит из следующих столбцов:

id – идентификатор

name – логин для входа в систему

password – пароль для входа в систему

token – токен активной сессии входа

address – адрес сервера

Имя	Тип	Размер	Уникальное	Пустое	Ключ
id	integer	-	да	нет	pk
name	varchar	30	нет	да	-
password	varchar	30	нет	да	-
token	varchar	10	нет	да	-
address	varchar	60	нет	да	-

Примеры входных данных:

1|kashitsin|pwd|6xvq4kq1s3|http://nodegraph.spbcoit.ru:5000

1| | |http://nodegraph.spbcoit.ru:5000

1|kashitsin|pwd||http://nodegraph.spbcoit.ru:5000

1| | |

ОПИСАНИЕ ФУНКЦИЙ API

Приложение взаимодействует с сервером по протоколу HTTP, получая ответы в виде JSON формата. Сервер работает с СУБД sqlite3.

Набор входных данных:

Name – название аккаунта, графа, узла

Secret – пароль аккаунта

Token – токен сессии (сгенерированные случайно 10 символов для идентификации регистрационных данных)

Id – идентификатор графа, точки, узла, соединения

X – местоположение узла по x

Y – местоположение узла по y

Source – точка от которой идет соединение

Target – точка куда идет соединение

Value – значение соединения

Существует следующий набор запросов:

Функция account/create

Тип запроса: PUT

Создание нового аккаунта

Входные параметры: name аккаунта, secret

Пример вызова отображен на рисунке 8

The screenshot shows a REST client interface with a PUT request to the endpoint `{{site}}/account/create?name=kashitsin493&secret=password`. The 'Params' tab is selected, showing a table of query parameters:

	KEY	VALUE
<input checked="" type="checkbox"/>	name	kashitsin493
<input checked="" type="checkbox"/>	secret	password

Рисунок 8 – Пример вызова функции account/create

Типы ответа: код 200, 500

Функция account/delete

Тип запроса: DELETE

Удаление аккаунта по токену сессии

Входные параметры: token

Пример вызова отображен на рисунке 9

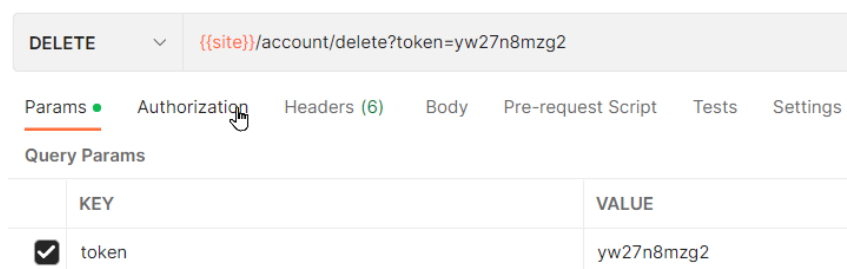


Рисунок 9 – Пример вызова функции account/delete

Типы ответов: код 200, код 500

Функция account/update

Тип запроса: POST

Изменение пароля аккаунта, в данный момент находящегося в системе

Входные параметры: token, secret

Пример вызова отображен на рисунке 10

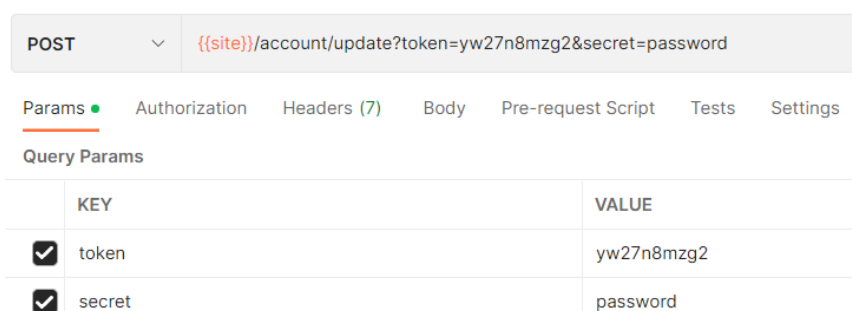


Рисунок 10 – Пример вызова функции account/update

Типы ответа: код 200, 500

Функция session/close

Тип запроса: DELETE

Закрытие сессии входа в аккаунт

Входные параметры: token

Пример вызова отображен на рисунке 11

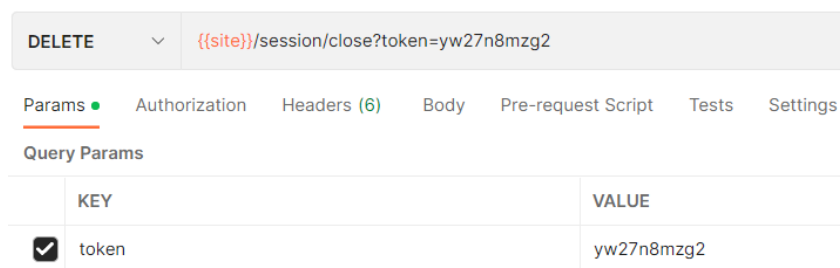


Рисунок 11 – Пример вызова функции session/close

Примеры ответов: код 200, код 500

Функция session/list

Тип запроса: GET

Получение списка сессий

Входные параметры: token

Пример вызова отображен на рисунке 12

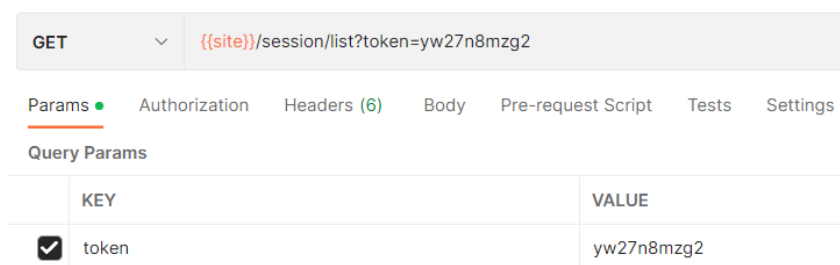


Рисунок 12 – Пример вызова функции session/list

Примеры ответов: массив из значений id сессии, token, timestamp сессии

Функция session/open

Тип запроса: PUT

Создание новой сессии входа в систему

Входные параметры: name аккаунта, secret

Пример вызова отображен на рисунке 13

PUT

▼

{{site}}/session/open?name=kashitsin493&secret=password

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	name	kashitsin493
<input checked="" type="checkbox"/>	secret	password

Рисунок 13 – Пример вызова функции session/open

Примеры ответов: {token: “yw27n8mzg2”}, код 500

Функция graph/create

Тип запроса: PUT

Создание нового пустого графа

Входные параметры: token, name графа

Пример вызова отображен на рисунке 14

PUT

▼

{{site}}/graph/create?token=yw27n8mzg2&name=Имя

Params

Authorization

Headers (7)

Body

Pre-request Script

Test

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	token	yw27n8mzg2
<input checked="" type="checkbox"/>	name	Имя

Рисунок 14 – Пример вызова функции graph/create

Примеры ответов: {id:2} (id созданного графа), код 500

Функция graph/list

Тип запроса: GET

Получения списка всех графов пользователя

Входные параметры: token

Пример вызова отображен на рисунке 15

GET	▼	{{site}}/graph/list?token=yw27n8mzg2
Params	Authorization	Headers (6)
Body	Pre-request Script	Tests
Settings		
Query Params		
	KEY	VALUE
<input checked="" type="checkbox"/>	token	yw27n8mzg2

Рисунок 15 – Пример вызова функции graph/list

Примеры ответов: массив из значений id графа, timestamp графа, названия, количестве узлов в графе

Функция graph/delete

Тип запроса: DELETE

Удаление выбранного графа у аккаунта

Входные параметры: token, id графа

Пример вызова отображен на рисунке 16

DELETE	▼	{{site}}/graph/delete?token=yw27n8mzg2&id=2
Params	Authorization	Headers (6)
Body	Pre-request Script	Tests
Settings		
Query Params		
	KEY	VALUE
<input checked="" type="checkbox"/>	token	yw27n8mzg2
<input checked="" type="checkbox"/>	id	2

Рисунок 16 – Пример вызова функции graph/delete

Примеры ответов: код 200, код 500

Функция graph/update

Тип запроса: POST

Переименование выбранного графа

Входные параметры: token, id графа, name графа

Пример вызова отображен на рисунке 17

POST ▼ {{site}}/graph/update?token=yw27n8mzg2&id=2&name=Переименовано

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	token	yw27n8mzg2
<input checked="" type="checkbox"/>	id	2
<input checked="" type="checkbox"/>	name	Переименовано

Рисунок 17 – Пример вызова функции graph/update

Примеры ответов: код 200, код 500

Функция node/create

Тип запроса: PUT

Добавление нового узла

Входные параметры: token, id графа, x, y, name узла

Пример вызова отображен на рисунке 18

PUT ▼ {{site}}/node/create?token=yw27n8mzg2&id=2&x=100.00&y=100.00&name=A

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	token	yw27n8mzg2
<input checked="" type="checkbox"/>	id	2
<input checked="" type="checkbox"/>	x	100.00
<input checked="" type="checkbox"/>	y	100.00
<input checked="" type="checkbox"/>	name	A

Рисунок 18 – Пример вызова функции node/create

Примеры ответов: {"id":1} (id созданной точки) или код 500

Функция node/delete

Тип запроса: DELETE

Удаление выбранной точки

Входные параметры: token, id точки

Пример вызова отображен на рисунке 19

DELETE	▼	{{site}}/node/delete?token=yw27n8mzg2&id=3
Params ●	Authorization	Headers (6)
Body	Pre-request Script	Tests
Settings		
Query Params		
	KEY	VALUE
<input checked="" type="checkbox"/>	token	yw27n8mzg2
<input checked="" type="checkbox"/>	id	3

Рисунок 19 – Пример вызова функции node/delete

Примеры ответов: код 200, код 500

Функция node/list

Тип запроса: GET

Получение всех точек выбранного графа

Входные параметры: token, id графа

Пример вызова отображен на рисунке 20

GET	▼	{{site}}/node/list?token=yw27n8mzg2&id=2
Params ●	Authorization	Headers (6)
Body	Pre-request Script	Tests
Settings		
Query Params		
	KEY	VALUE
<input checked="" type="checkbox"/>	token	yw27n8mzg2
<input checked="" type="checkbox"/>	id	2

Рисунок 20 – Пример вызова функции node/list

Примеры ответов: массив из значений id,x,y,name точек, или код 500

Функция node/update

Тип запроса: POST

Обновление местоположения и имени точки

Входные параметры: token, id точки, x, y, name точки

Пример вызова отображен на рисунке 21

POST

▼

{{site}}/node/update?token=yw27n8mzg2&id=1&x=0&y=0&name=A (2)

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	token	yw27n8mzg2
<input checked="" type="checkbox"/>	id	1
<input checked="" type="checkbox"/>	x	0
<input checked="" type="checkbox"/>	y	0
<input checked="" type="checkbox"/>	name	A (2)

Рисунок 21 – Пример вызова функции node/list

Примеры ответов: код 200, код 500

Функция link/create

Тип запроса: PUT

Создание соединения между двумя точками

Входные параметры: token, source, target, value

Пример вызова отображен на рисунке 22

PUT

▼

{{site}}/link/create?token=yw27n8mzg2&source=1&target=2&value=1.0

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	token	yw27n8mzg2
<input checked="" type="checkbox"/>	source	1
<input checked="" type="checkbox"/>	target	2
<input checked="" type="checkbox"/>	value	1.0

Рисунок 22 – Пример вызова функции link/create

Примеры ответов: { "id": 1 } (id созданного соединения), код 500

Функция link/delete

Тип запроса: DELETE

Удаление выбранного соединения

Входные параметры: token, id соединения

Пример вызова отображен на рисунке 23

DELETE

▼

{{site}}/link/delete?token=yw27n8mzg2&id=1

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	token	yw27n8mzg2
<input checked="" type="checkbox"/>	id	1

Рисунок 23 – Пример вызова функции link/delete

Примеры ответов: код 200, код 500

Функция link/list

Тип запроса: GET

Получение списка соединений у выбранного графа

Входные параметры: token, id графа

Пример вызова отображен на рисунке 24

GET

▼

{{site}}/link/list?token=x45yjdkx7z&id=30

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	token	x45yjdkx7z
<input checked="" type="checkbox"/>	id	30

Рисунок 24 – Пример вызова функции link/list

Примеры ответов: массив из значений id соединения, source, target, value, или код 500

Функция link/update

Тип запроса: POST

Изменение значения соединения

Входные параметры: token, id соединения, value

Пример вызова отображен на рисунке 25

POST	▼	{{site}}/link/update?token=yw27n8mzg2&id=1&value=2.0
Params	Authorization	Headers (7)
Body	Pre-request Script	Tests
Settings		
Query Params		
	KEY	VALUE
<input checked="" type="checkbox"/>	token	yw27n8mzg2
<input checked="" type="checkbox"/>	id	1
<input checked="" type="checkbox"/>	value	2.0

Рисунок 25 – Пример вызова функции link/update

Примеры ответов: код 200, код 500

ДЕМОНСТРАЦИЯ ФУНКЦИЙ

Иконка приложения отображена на рисунке 26.



Рисунок 26 – Иконка приложения

После запуска приложения, пользователь видит экран авторизации на рисунке 27.

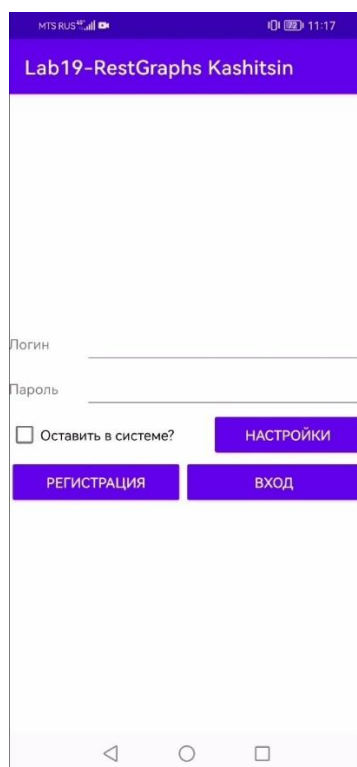


Рисунок 27 – Вид приложения после запуска

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

В локальной базе хранится логин, пароль, сессия успешного входа и адрес конечной точки API.

Класс Request имеет поля base – адрес конечной точки API, thread – поток, который занимается отправкой запроса. Адрес точки загружается с локальной базы данных, таблицы настроек аккаунта. В адрес добавляется передача параметров. Создается соединение, выбирается метод, время на ожидание, считывается поток байтов и записывается в строку, отключается соединение. Если ошибок во время этого процесса не возникло, и сервер возвратил положительный код, то срабатывает метод onSuccess, иначе onFail. При создании экземпляра этого класса, эти методы переопределяются.

Новый пользователь должен пройти регистрацию по кнопке «Регистрация», ввести логин и пароль. Если регистрация прошла успешно, то пользователь увидит экран на рисунке 28.



Рисунок 28 – Регистрация пользователя

Если регистрация прошла неудачно, например, попытка создание аккаунта с одинаковым логином, появится следующее сообщение на рисунке 29.

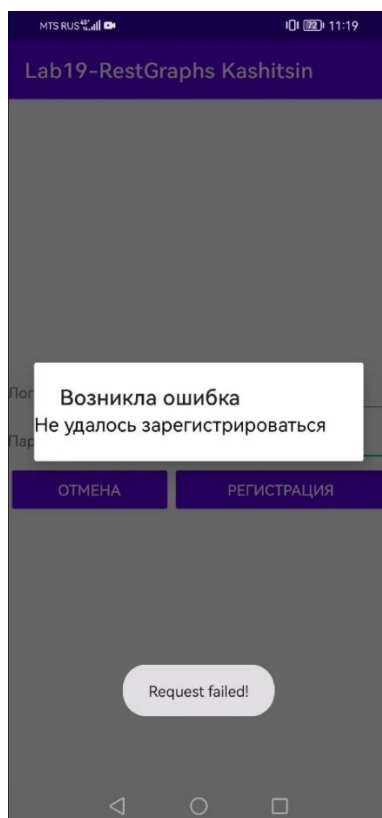


Рисунок 29 – Ошибка при регистрации

Всплывающее окно является Activity, Layout которой с TextView показывается на AlertDialog, он отображается в методе onFail, оттуда посылается текст исключения и типизируется ошибка.

Запись на сервере о создании аккаунта отображена на рисунке 30.

Query +/- table definition

```
SELECT *  
FROM "account" WHERE name = 'Kashitsin'
```

Execute Export JSON Export CSV SQL Help ▾

Results (1) Permalink

id	name	secret
18	Kashitsin	pwd

Рисунок 30 – Создание аккаунта (на сервере)

После регистрации аккаунта, пользователь сможет авторизоваться, введя логин и пароль, а также включить опцию запоминания в системе. Это отображено на рисунке 31.

Рисунок 31 – Авторизация пользователя

Если авторизация прошла успешно, и опция запоминания была включена, то логин, пароль, и активная сессия будут сохранены в локальной базе. Так, после выхода из приложения, данные будут заполняться, а если токен сессии будет существовать на сервере, то используя его, пользователь пропустит этап авторизации. Это отображено на рисунке 32.

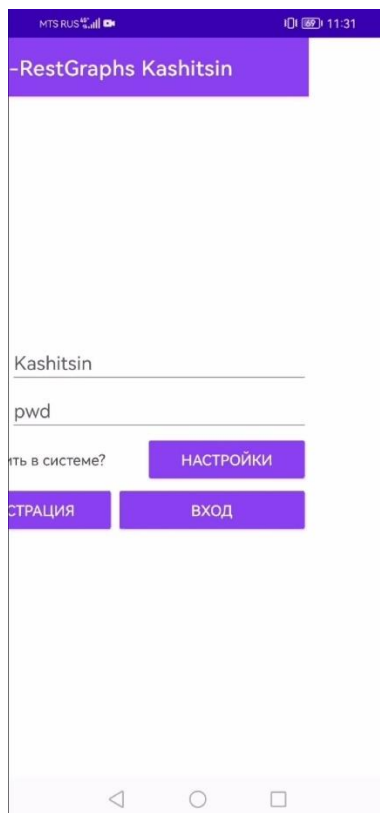


Рисунок 32 – Сохранение сессии авторизации

Если сессия будет закрытой, поля останутся заполненными, но придется снова пройти авторизацию.

Токен сессии – это сгенерированный набор 10 символов, который идентифицирует определенный аккаунт среди активных пользователей, таким образом действия с одним аккаунтом может происходить с нескольких устройств. Класс Session содержит значение токена, времени создания. Время от сервера поступает в формате timestamp в секундах, что потом переводится в дату и время.

После успешной авторизации пользователь увидит форму работы с графами. Обновление списка графов происходит с сервера и с локальной базы.

Список графов отправляется на устройство в виде массива JSON из значений id, названия, timestamp создания графов, количества узлов в графах.

Для визуализации списка создавался пользовательский адаптер ListView – GraphAdapter. Компоненты созданного view: TextView для названия и для даты с количеством точек. Если граф получен с сервера, то добавляется компонент ImageView с изображением облака.

В поле ввода вводится название графа, по кнопке «Создать» производится запрос на создание графа с введенным именем. Создадим следующие 2 графа на рисунке 33.

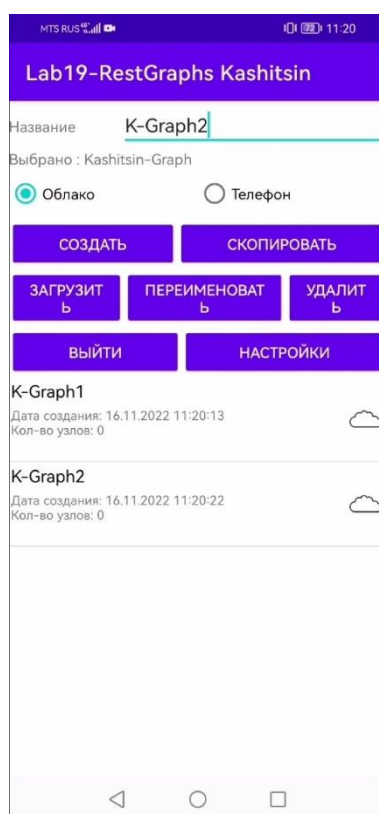


Рисунок 33 – Создание графов

Запись на сервере о создании графов отображена на рисунке 34.

Query

+/- table definition

SELECT *
FROM "graph" WHERE account = 18

Execute

Export JSON

Export CSV

SQL Help ▾

Results (2)

Permalink

id	account	name	timestamp
44	18	K-Graph1	2022-11-16 08:20:13
45	18	K-Graph2	2022-11-16 08:20:22

Рисунок 34 - Создание графов (на сервере)

Загрузка происходит получением массивов JSON узлов и соединений. Для узлов параметрами является id, название, координаты. Для узлов: id, id двух точек, значение. Все данные структурируются в list переменные с сохранением идентификаторов.

После добавления пользователем точки, отправляется запрос на сервер, если сервер успешно вернул id созданного Node, то он добавляется в список, происходит отрисовка.

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.

Пользователь может передвинуть точку. На этот раз, старые координаты запоминаются для плавного перемещения по экрану, при этом выбор точки не спадает. Как только пользователь отпускает палец после передвижения узла, отправляется запрос на обновление координат у нужной точки по ее индексу. Если ответ сервера отрицательный, то точка возвращается в исходное положение.

Если добавить ещё одну точку, и выбрав её, то если уже существует выбранная точка, новая отдельно окрашивается в другой цвет.

Если выбраны две точки, то пользователь может их соединить. Отправляется запрос на создание соединения с идентификаторами выбранных точек. Если сервер вернул идентификатор созданного соединения, то создается Link, с выбранными Nodes и добавляется в список, происходит отрисовка.

Нажав на нарисованный квадрат, можно выбрать соединение, и дать значение ребра. Выбор осуществляется также, как и с узлами. Отправляется запрос на обновление значения соединения. Если сервер вернул положительный ответ, то значение в списке обновляется, и появляется значение на нарисованном

Пользователь также может давать название узлам. Ситуация аналогична с обновлением значений у соединения.

Загрузим граф K-Graph1. Создадим 3 точки, соединим последовательно. Дадим двум точкам названия А и В, значение одного соединения – 2.89. Граф показан на рисунке 35.

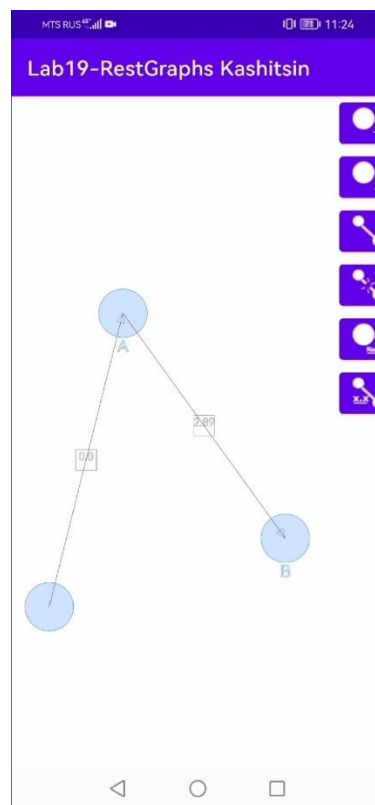


Рисунок 35 – Нарисованный граф K-Graph1

Запись на сервере о создании трех точек отображена на рисунке 36.

Query					+/- table definition
<pre>SELECT * FROM "node" WHERE graph = 44</pre>					
<div>ExecuteExport JSONExport CSVSQL Help ▾</div>					
Results (3)					Permalink
id	graph	x	y	name	
72	44	792.35895	1277.4907	B	
73	44	323.814	627.25073	A	
74	44	110.98976	1475.4052		

Рисунок 36 – Создание точек графа (на сервере)

Запись на сервере о соединении точек А-В показана на рисунке 37.

Query				+/- table definition
<pre>SELECT * FROM "link" WHERE source = 73</pre>				
<div>ExecuteExport JSONExport CSVSQL Help ▾</div>				
Results (1)				Permalink
id	source	target	value	
25	73	72	2.89	

Рисунок 37 – Соединение точек А-В (на сервере)

Запись на сервере о соединении точки А с точкой на координатах (110, 1475) отображена на рисунке 38.

Query +/- table definition

```
SELECT *  
FROM "link" WHERE source = 74
```

Execute Export JSON Export CSV SQL Help ▾

Results (1) Permalink

id	source	target	value
26	74	73	0.0

Рисунок 38 – Соединение точки с А (на сервере)

Удалив точку А, получив связанные с ней соединения (все соединения на графе), отправляется запрос на удаление соединений и выбранных точек по идентификатору. Если запросы были успешными, то граф перерисовывается, что показано на рисунке 39.

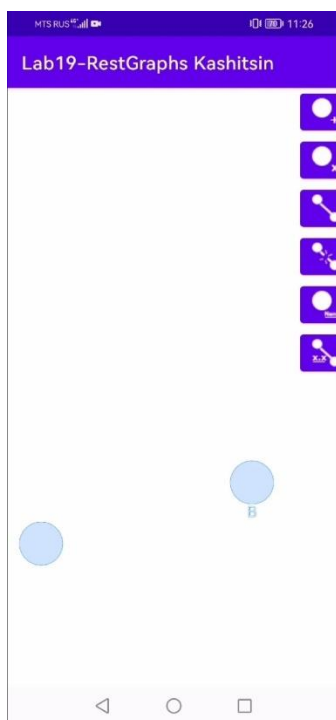


Рисунок 39 – Отображение графа после удаления точки А

Отсутствие записей на сервере о соединениях показано на рисунке 40.

Query +/- table definition

```
SELECT *  
FROM "link" WHERE source = 74 OR source = 73
```

Execute Export JSON Export CSV SQL Help ▾

Empty result set.

Рисунок 40 – Отсутствие соединений (на сервере)

Запись о текущих точках на сервере показана на рисунке 41.

Query +/- table definition

```
SELECT *  
FROM "node" WHERE graph = 44
```

Execute Export JSON Export CSV SQL Help ▾

Results (2) Permalink

id	graph	x	y	name
72	44	792.35895	1277.4907	B
74	44	110.98976	1475.4052	

Рисунок 41 – Узлы графа K-Graph1 (на сервере)

Если не удалось отправить на сервер координаты перемещенной точки, то точка возвращается на исходное положение после отпускания пальцем.

Перемещение точки в исходное положение после неуспешного обновления координат показано на рисунке 42.

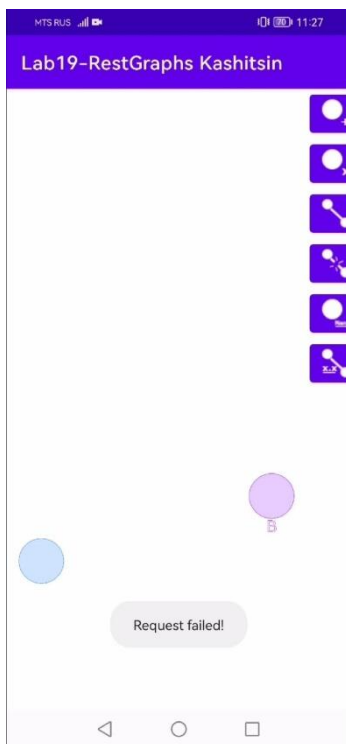


Рисунок 42 – Отмена перемещения

Переименование графа происходит после запроса на обновление выбранного графа, параметром запроса будет новое название. Переименование графа K-Graph2 в Kashitsin-Graph показано на рисунке 43.

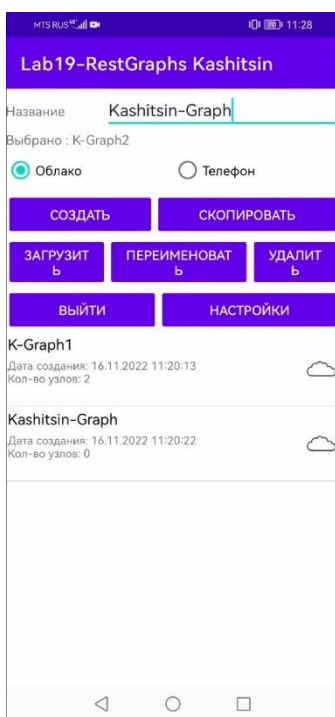


Рисунок 43 – Переименование графа

Запись о переименовании графа на сервере показана на рисунке 44.

Structure

Content

Query

Drop

Import

Query

+/- table definition

```
SELECT *
FROM "graph" WHERE account = 18
```

Execute

Export JSON

Export CSV

SQL Help ▾

Results (2)

Permalink

id	account	name	timestamp
44	18	K-Graph1	2022-11-16 08:20:13
45	18	Kashitsin-Graph	2022-11-16 08:20:22

Рисунок 44 – Переименование графа (на сервере)

Нарисуем следующий граф Kashitsin-Graph на рисунке 45.

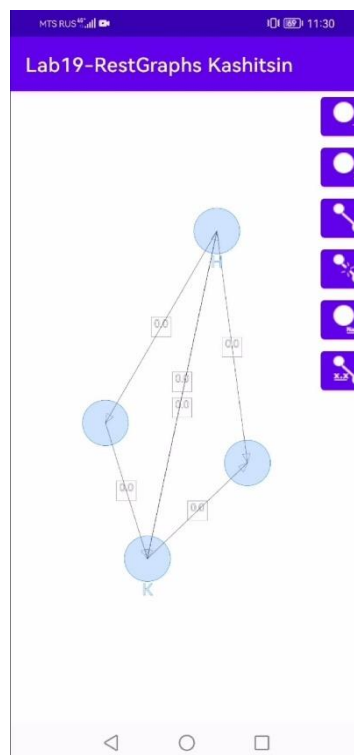
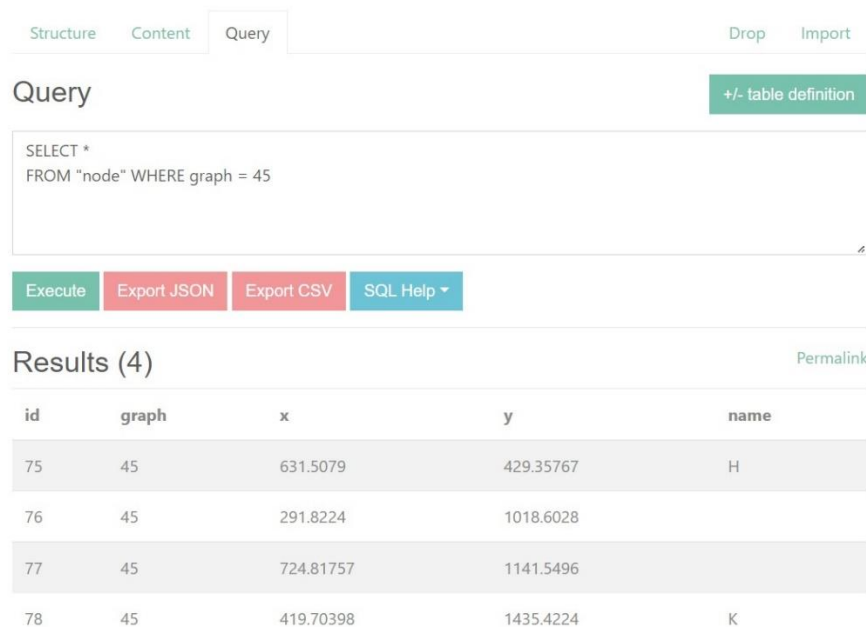


Рисунок 45 – Нарисованный граф Kashitsin-Graph

Запись о точках графа Kashitsin-Graph на сервере показана на рисунке 46.



The screenshot shows a web interface with tabs for Structure, Content, and Query. The Query tab is active, displaying a SQL query: `SELECT * FROM "node" WHERE graph = 45`. Below the query are buttons for Execute, Export JSON, Export CSV, and SQL Help. The Results section shows 4 rows of data in a table with columns: id, graph, x, y, and name.

id	graph	x	y	name
75	45	631.5079	429.35767	H
76	45	291.8224	1018.6028	
77	45	724.81757	1141.5496	
78	45	419.70398	1435.4224	K

Рисунок 46 – Нарисованный граф Kashitsin-Graph (на сервере)

Пользователь может скопировать граф. Посылается запрос на создание графа с названием выбранного графа. Если запрос был обработан, то id созданного графа используется для создания точек. Посылается запрос на просмотр точек выбранного графа для копирования. Вся информация о списке точек сохраняется в двумерный массив. Во второй части массива будут точки скопированного графа. При перечислении массива точек выбранного графа, точки создаются уже по id скопированного графа. Созданные точки дополняют двумерный массив до тех пор, пока он полностью не заполнится. Затем просматривается список соединений выбранного графа, точки скопированного массива используются для создания соединений скопированного графа. На этом

заканчивается копирование. Копирование графа Kashitsin-Graph показано на рисунке 47.

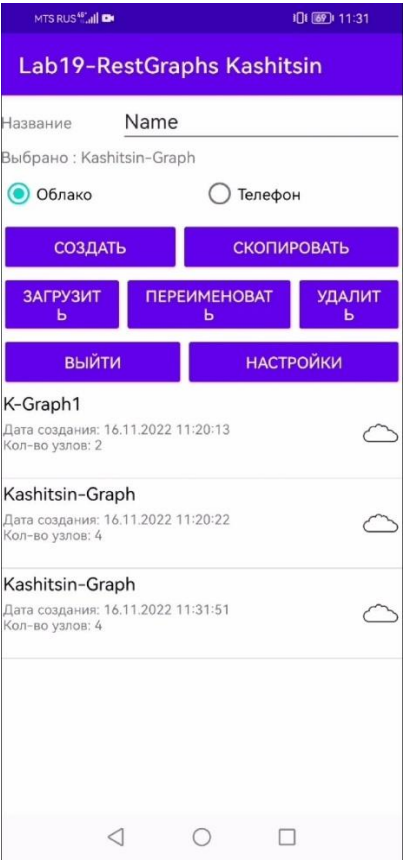


Рисунок 47 – Список графов после копирования

Запись о списке графов после копирования на сервере показана на рисунке 48.

Query +/- table definition

```
SELECT *
FROM "graph" WHERE account = 18
```

Execute Export JSON Export CSV SQL Help ▾

Results (3) Permalink

id	account	name	timestamp
44	18	K-Graph1	2022-11-16 08:20:13
45	18	Kashitsin-Graph	2022-11-16 08:20:22
46	18	Kashitsin-Graph	2022-11-16 08:31:51

Рисунок 48 – Список графов после копирования (на сервере)

Запись о точках скопированного на сервере показана на рисунке 49.

Query +/- table definition

```
SELECT *  
FROM "node" WHERE graph = 46
```

Execute Export JSON Export CSV SQL Help ▾

Results (4) Permalink

id	graph	x	y	name
79	46	631.5079	429.35767	H
80	46	291.8224	1018.6028	
81	46	724.81757	1141.5496	
82	46	419.70398	1435.4224	K

Рисунок 49 – Скопированный граф (на сервере)

Удаление графа K-Graph1 происходит отправкой запроса на удаление выбранного графа происходит на рисунке 50.

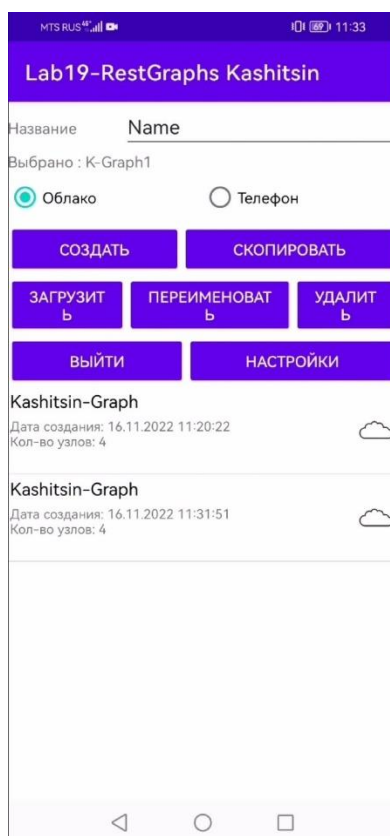


Рисунок 50 – Удаление графа K-Graph1

Запись об отсутствии графа K-Graph1 на сервере показана на рисунке 51.

Query +/- table definition

```
SELECT *  
FROM "graph" WHERE account = 18
```

Execute Export JSON Export CSV SQL Help ▾

Results (2) Permalink

id	account	name	timestamp
45	18	Kashitsin-Graph	2022-11-16 08:20:22
46	18	Kashitsin-Graph	2022-11-16 08:31:51

Рисунок 51 – Отсутствие графа K-Graph1 (на сервере)

Создадим граф на локальной базе данных, выбрав опцию «Телефон» при создании графа. При этом знак облака уже не отображается. При работе с графом, сохраненным на телефоне, запросы на сервер не отправляются. Вместо этого используются запросы sql. Структура локальной базы данных похожа на структуру базы на сервере, за исключением таблиц аккаунтов и сессий. Работа с локальной базой данных показана в предыдущей практической работе, но на этот раз все происходит в реальном времени. Создание графа на локальной базе данных показано на рисунке 52.

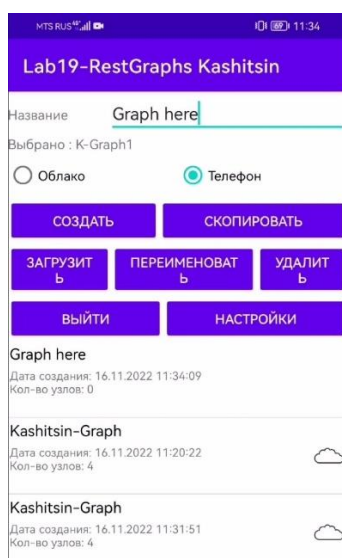


Рисунок 52 – Создание локального графа

Рисование графа Graph here показано на рисунке 53.

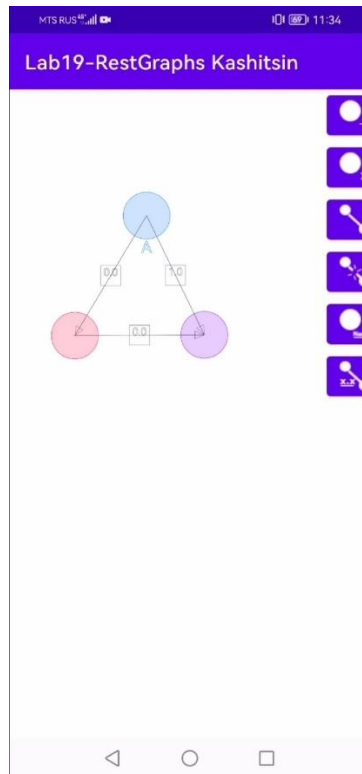


Рисунок 53 – Рисование графа Graph here

Переименование Graph here в Name происходит на рисунке 54.

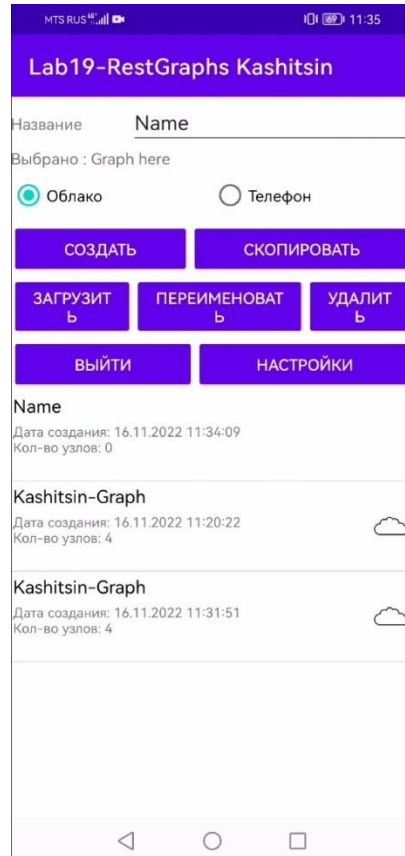


Рисунок 54 – Переименование локального графа

Копирование графа Name показано на рисунке 55.

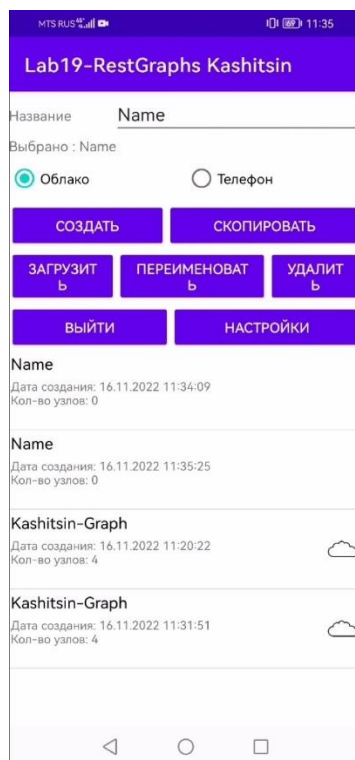


Рисунок 55 – Копирование локального графа
Скопированный локальный граф показан на рисунке 56.

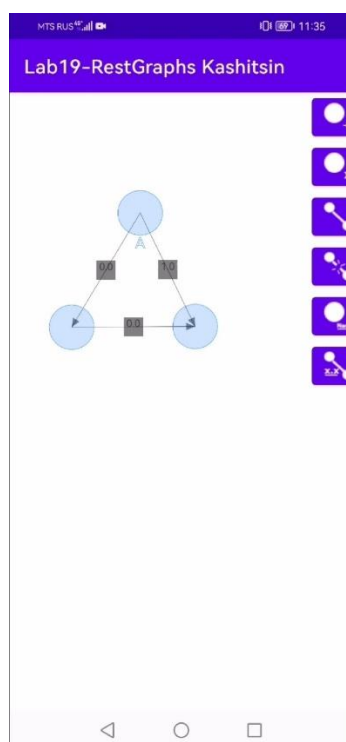


Рисунок 56 – Скопированный локальный граф

Удаление локального графа показано на рисунке 57.

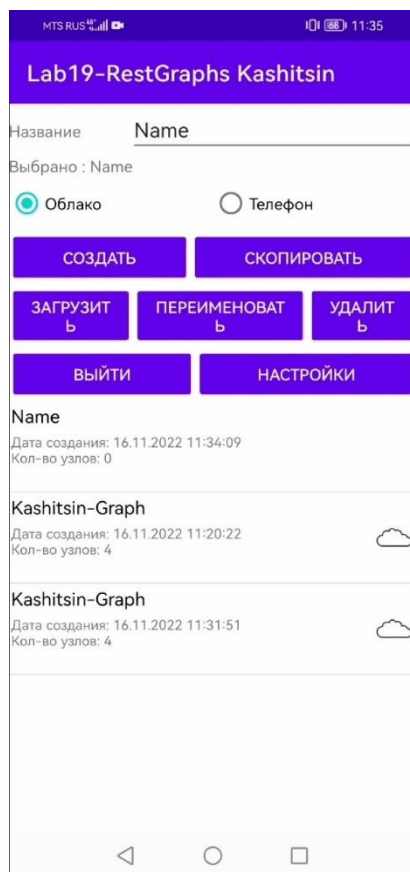


Рисунок 57 – Удаление локального графа

Пользователь может поменять пароль на форме настроек. Там же есть возможность удалить аккаунт, посмотреть список входов. Смена пароля показана на рисунке 58.

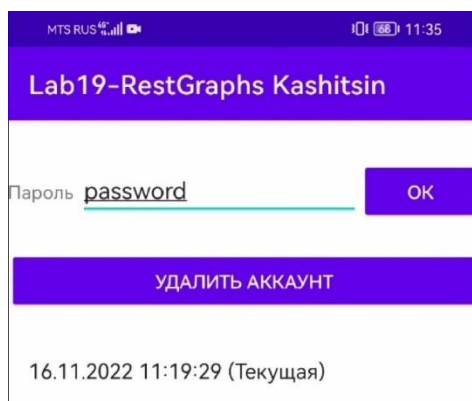


Рисунок 58 – Смена пароля

Пароль изменяется при указании токена сессии необходимого аккаунта.

Запись о смене пароля на сервере показана на рисунке 59.

Query +/- table definition

```
SELECT *  
FROM "account" WHERE name = 'Kashitsin'
```

Execute Export JSON Export CSV SQL Help ▾

Results (1) Permalink

id	name	secret
18	Kashitsin	password

Рисунок 59 – Смена пароля (на сервере)

Теперь перезайдем в аккаунт несколько раз, создав несколько сессий.

Список входов (сессий) показан на рисунке 60.

MTS RUS 11:37

Lab19-RestGraphs Kashitsin

Пароль ОК

УДАЛИТЬ АККАУНТ

16.11.2022 11:36:42
16.11.2022 11:37:31
16.11.2022 11:37:33 (Текущая)

Рисунок 60 – Создание сессий

Запись о списке сессий на сервере показана на рисунке 61.

Query +/- table definition

```
SELECT *  
FROM "session" WHERE account = 18
```

Execute Export JSON Export CSV SQL Help ▾

Results (3) Permalink

id	account	token	timestamp
315	18	f001gsotvk	2022-11-16 08:36:42
316	18	kprmxz6f6p	2022-11-16 08:37:31
317	18	e2f9y0r4gi	2022-11-16 08:37:33

Рисунок 61 – Список сессий (на сервере)

Удаление сессии входа происходит по нажатию на определенную дату на рисунке 62.

MTS RUS 11:37

Lab19-RestGraphs Kashitsin

Пароль

16.11.2022 11:37:31

16.11.2022 11:37:33 (Текущая)

Рисунок 62 – Удаление сессии

Запись о удалении сессии на сервере показана на рисунке 63.

Query +/- table definition

```
SELECT *  
FROM "session" WHERE account = 18
```

Execute Export JSON Export CSV SQL Help ▾

Results (2) Permalink

id	account	token	timestamp
316	18	kprmzx6f6p	2022-11-16 08:37:31
317	18	e2f9y0r4gi	2022-11-16 08:37:33

Рисунок 63 – Список сессии после удаления токена (на сервер)

Удаление текущей сессии приведет к попаданию на форму авторизации, так как последующие действия требуют активного токена. Также на этой форме есть переход к настройкам, где можно ввести сайт, на который будут отправляться запросы. Укажем неправильный адрес на рисунке 64.

MTS RUS 4G+ 11:39

Lab19-RestGraphs Kashitsin

Адрес http://nodegraph.spbcoit.ru

ОТМЕНА ОК

Рисунок 64 – Ввод конечной точки API

Попытка входа в аккаунт показана на рисунке 65.

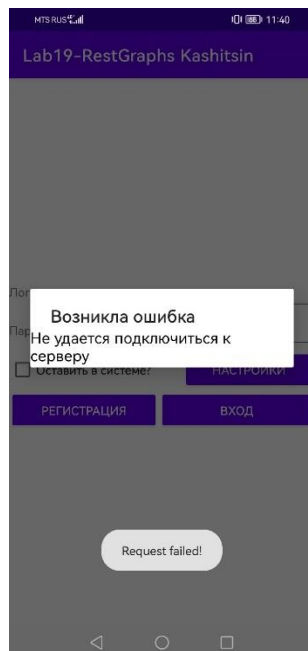


Рисунок 65 – Недоступность API

Не только отсутствие активной сессии не позволяет работать с графом, но и отсутствие подключения. Так как граф рисуется только после успешного ответа от сервера, невозможно создать точку без соединения. Это продемонстрировано на рисунке 66.

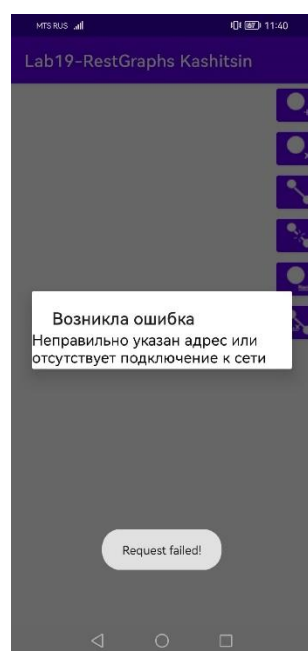


Рисунок 66 – Отсутствие соединения при добавлении точки

Пользователь может удалить аккаунт на форме настроек по соответствующей кнопке. Отсутствие аккаунта на сервере показано на рисунке 67.

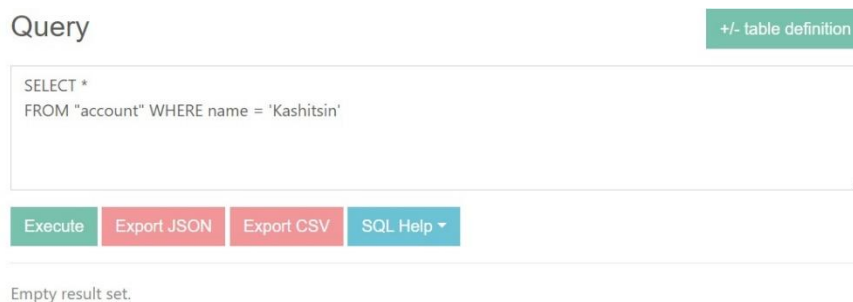


Рисунок 67 – Удаление аккаунта (на сервере)

Теперь невозможно войти в аккаунт. Удаление аккаунта приводит к удалению всех сессий, графов, соответствующих узлов и соединений. Попытка входа после удаления аккаунта показано на рисунке 68.

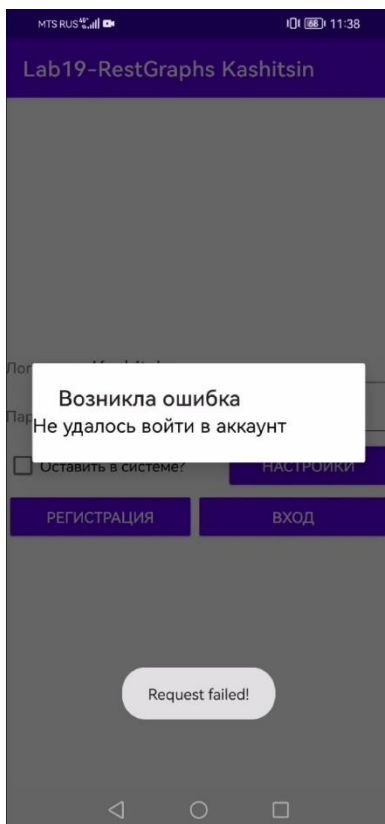


Рисунок 68 – Неуспешный вход после удаления

ИТОГ РАБОТЫ

Ссылка на репозиторий с готовым проектом:
<https://github.com/aza1rat/LabGraphsAPI>