МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Сервис для планирования кабельных сетей

Студенты гр. 0381 Ибатов Н. Э.

Печеркин А. С.

Степанова Е. М.

Преподаватель Заславский М. М.

Санкт-Петербург
2023

ЗАДАНИЕ

Студенты

Ибатов Н. Э.

Печеркин А. С.

Степанова Е. М.

Группа 0381

Тема проекта: Сервис для планирования кабельных сетей

Исходные данные:

Необходимо реализовать приложение для планирования кабельных сетей используя СУБД Neo4j.

Содержание пояснительной записки: “Содержание”, “Введение”, “Сценарий использования”, “Модель данных”, “Разработанное приложение”, “Выводы”, “Приложения”, “Литература”.

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 20.09.2023

Дата сдачи реферата: 23.12.2023

Дата защиты реферата: 23.12.2023

Студент гр. 0381		Ибатов Н. Э.
Студент гр. 0381		Печеркин А. С.
Студент гр. 0381		Степанова Е. М.
Преподаватель		Заславский М. М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать приложение для прокладки кабельных сетей используя СУБД Neo4j и рассмотреть насколько хорошо была выбрана данная СУБД. Для создания веб-приложения был выбран Nestjs и React. Исходный код и дополнительная информация по ссылке: https://github.com/moevm/nosql2h23-cable

SUMMARY

As part of this course, it was supposed to develop an application for laying cable networks using the Neo4j DBMS and consider how well this DBMS was chosen. Nestjs and React were chosen to create the web application. The source code and additional information are available at the link: https://github.com/moevm/nosql2h23-cable

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Актуальность решаемой проблемы

Проблема планирования кабельных сетей является актуальной для многих организаций и индивидуальных пользователей. Кабельные сети играют важную роль в передаче данных, интернет-соединении, телекоммуникациях и других сферах, поэтому эффективное планирование и управление этими сетями является критически важным.

1.2. Постановка задачи

Требуется реализовать сервис планирования кабельных сетей, в котором пользователь сможет для собственного проекта проложить кабельные сети, а также оценить нужное количество компонент для данной реализации.

1.3. Предлагаемое решение

Предлагаем реализовать веб-приложение с возможностью создания новых или импортирование ранее созданных проектов, редактирование проекта, а также добавления к нему комментариев и просмотр статистики по проекту.

1.4 Качественные требования к решению

- Возможность создавать и редактировать проекты
- Возможность добавлять комментарии к проекту
- Наличие импорта и экспорта данных в формате .json

2. СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ

2.1. Макет UI.

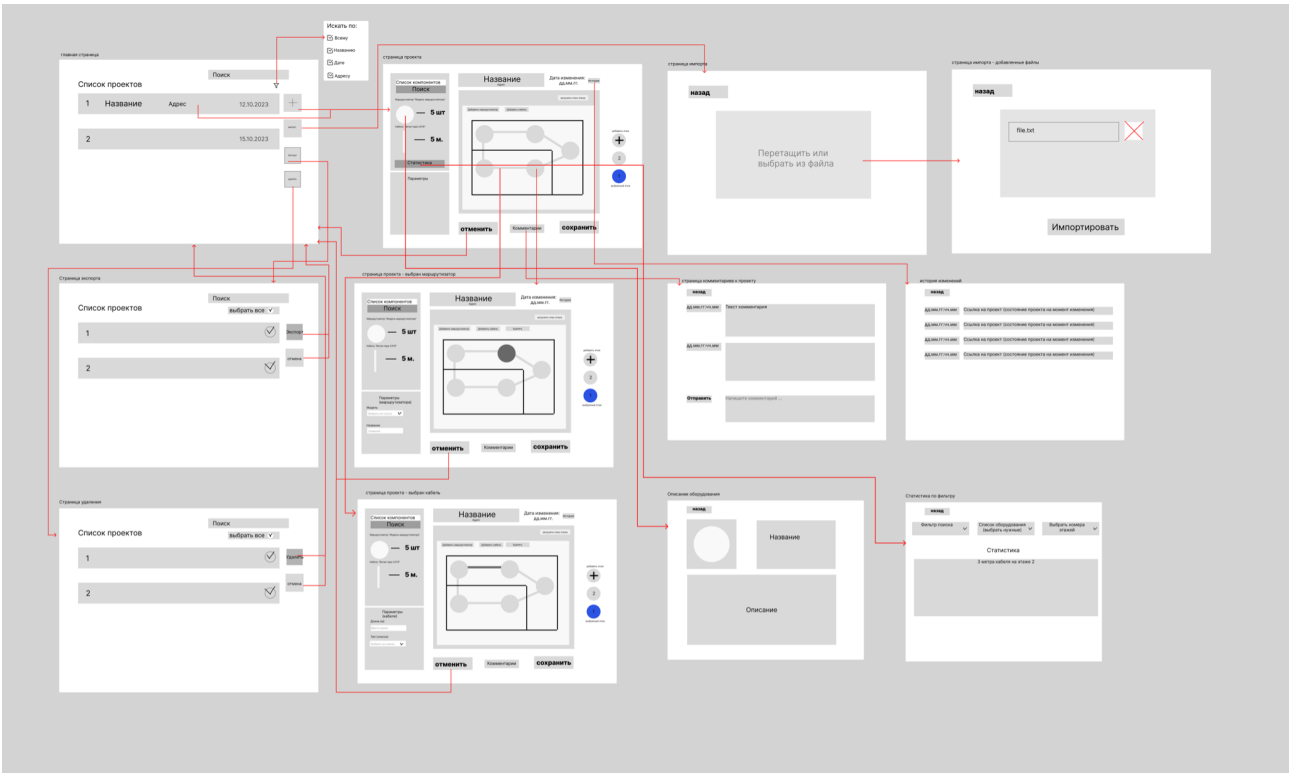

Рис. 1. Макет UI.

2.2. Сценарии использования для задачи

1. "Изменение названия маршрутизатора"

Действующее лицо: Пользователь Пользователь находится на главной странице(страница со списком проектов) Основной сценарий:

1. Пользователь выбирает проект из списка и нажимает на него
2. Происходит переход на страницу с информацией о проекте и редактором сети
3. Пользователь нажимает на иконку маршрутизатора, который он хочет изменить
4. Появляется раздел с параметрами выбранного маршрутизатора
5. Пользователь вводит в поле "Название" желаемое название
6. Пользователь нажимает на кнопку "Сохранить"
7. Изменения в проекте сохранены

2. "Прокладка кабеля"

Действующее лицо: Пользователь Пользователь находится на главной странице(страница со списком проектов) Основной сценарий:

1. Пользователь выбирает проект из списка и нажимает на него
2. Происходит переход на страницу с информацией о проекте и редактором сети
3. Пользователь нажимает на кнопку "Добавить кабель"
4. Редактор переходит в режим прокладки кабеля
5. Пользователь кликает на маршрутизатор, от которого он хочет начать прокладку кабеля и перемещает мышь
6. Кабель в виде линии следует за мышью
7. Пользователь кликает по плану этажа(не на маршрутизатор)
8. Добавляется промежуточная точка на кабеле(изгиб)
9. Пользователь кликает по маршрутизатору. к которому он хочет провести кабель
10. Кабель присоединяется и редактор выходит из режима прокладки кабеля
11. Пользователь нажимает на кнопку "Сохранить"
12. Изменения в проекте сохранены Альтернативный сценарий: Пользователь начал прокладку кабеля, но передумал
13. Пользователь нажимает правую кнопку мыши
14. Последняя промежуточная точка на кабеле отменяется или, если таких нет, прокладка кабеля отменяется

3. "Добавление этажа"

Действующее лицо: Пользователь Пользователь находится на главной странице(страница со списком проектов) Основной сценарий:

1. Пользователь выбирает проект из списка и нажимает на него
2. Происходит переход на страницу с информацией о проекте и редактором сети

3. Пользователь нажимает на кнопку "(+)"
4. Открывается пустой план
5. Пользователь нажимает на кнопку "Загрузить план"
6. Открывается окно файлового менеджера, где пользователь может выбрать картинку
7. Пользователь загружает картинку
8. Картинка отображается в качестве фона в редакторе
9. Пользователь нажимает на кнопку "Сохранить"
10. Изменения в проекте сохранены

4. "Удаление элемента сети"

Действующее лицо: Пользователь Пользователь находится на странице проекта Основной сценарий:

1. Пользователь нажимает на нужный элемент или выделяет несколько с помощью мыши
2. Выбранные элементы подсвечиваются некоторым цветом
3. Пользователь нажимает на кнопку удалить
4. Выбранные элементы исчезают
5. Пользователь нажимает на кнопку "Сохранить"
6. Изменения в проекте сохранены

5. "Создание проекта"

Действующее лицо: Пользователь Пользователь находится на главной странице(страница со списком проектов) Основной сценарий:

1. Пользователь нажимает на кнопку +
2. Происходит переход на страницу с информацией о проекте и редактором сети с одним пустым этажом

3. Пользователь заполняет проект
4. Пользователь нажимает на кнопку "Сохранить"
5. Изменения в проекте сохранены

6. "Изменение существующего кабеля"

Действующее лицо: Пользователь Пользователь находится на странице проекта Основной сценарий:

1. Пользователь нажимает на нужный кабель
2. Выбранные кабель подсвечивается некоторым цветом и появляются промежуточные точки на кабеле
3. Пользователь нажимает на одну из промежуточных точек и тянет её мышью
4. Промежуточная точка передвигается
5. Пользователь нажимает на кнопку "Сохранить"
6. Изменения в проекте сохранены Альтернативный сценарий:
7. Пользователь нажимает на нужный кабель
8. Выбранные кабель подсвечивается некоторым цветом и появляются промежуточные точки на кабеле
9. Пользователь дважды кликает на кабель(не на промежуточную точку)
10. На кабеле добавляется новая промежуточная точка
11. Пользователь нажимает на кнопку "Сохранить"
12. Изменения в проекте сохранены

7. “Найти проект”

Действующее лицо: Пользователь Пользователь находится на главной странице(страница со списком проектов) Основной сценарий:

1. Пользователь нажимает на поисковую строку
2. Вводит название/адрес/дату(дд.мм.гг).
3. Выводится список проектов, которые подходят по поиску.

8. “Удалить проект”

Действующее лицо: Пользователь Пользователь находится на главной странице(страница со списком проектов) Основной сценарий:

1. Пользователь нажимает на кнопку удалить
2. Происходит переход на страницу удаления
3. Пользователь может выбрать один или несколько проектов (можно выбрать всё)
4. При нажатие на кнопку "Удалить" выходит предупреждение, после чего удаляются выбранные проекты.
5. При нажатии на кнопку "Отмена" выходит предупреждение, после чего пользователь возвращается на страницу с проектами.

9. “Импортировать проект(ы)”

Действующее лицо: Пользователь Пользователь находится на главной странице(страница со списком проектов) Основной сценарий:

1. Пользователь нажимает кнопку "Импортировать", после чего его перекидывает на страницу импортирования проектов
2. Пользователь может перетащить в выделенную область файлы проектов либо нажать на данную область (откроется проводник)
3. Добавленные проекты будут отображаться в виде списка, после чего можно Импортировать данные файлы либо удалить лишние

10. “Экспортировать проект(ы)”

Действующее лицо: Пользователь Пользователь находится на главной странице(страница со списком проектов) Основной сценарий:

1. Пользователь нажимает кнопку "Экспортировать", после чего его перекидывает на страницу экспортирования проектов
2. Пользователь может выбрать один или несколько проектов (можно выбрать всё)

3. При нажатие на кнопку "Экспортировать" выходит предупреждение, после чего экспортируются выбранные проекты.
4. При нажатии на кнопку "Отмена" выходит предупреждение, после чего пользователь возвращается на страницу с проектами.

2.3. Вывод о том, какие операции (чтение или запись) будут преобладать для вашего решения.

В решении операции записи будут преобладать над операциями чтения. Исходя из того, что в основе нашего приложение лежит добавлении и изменении проектов по прокладке кабельных сетей. Добавление каждого элемента сети в проекте, такой как, например, кабель или маршрутизатор, требует отдельной операции. Таким образом, операция записи будет использоваться чаще.

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель данных (для вашей СУБД)

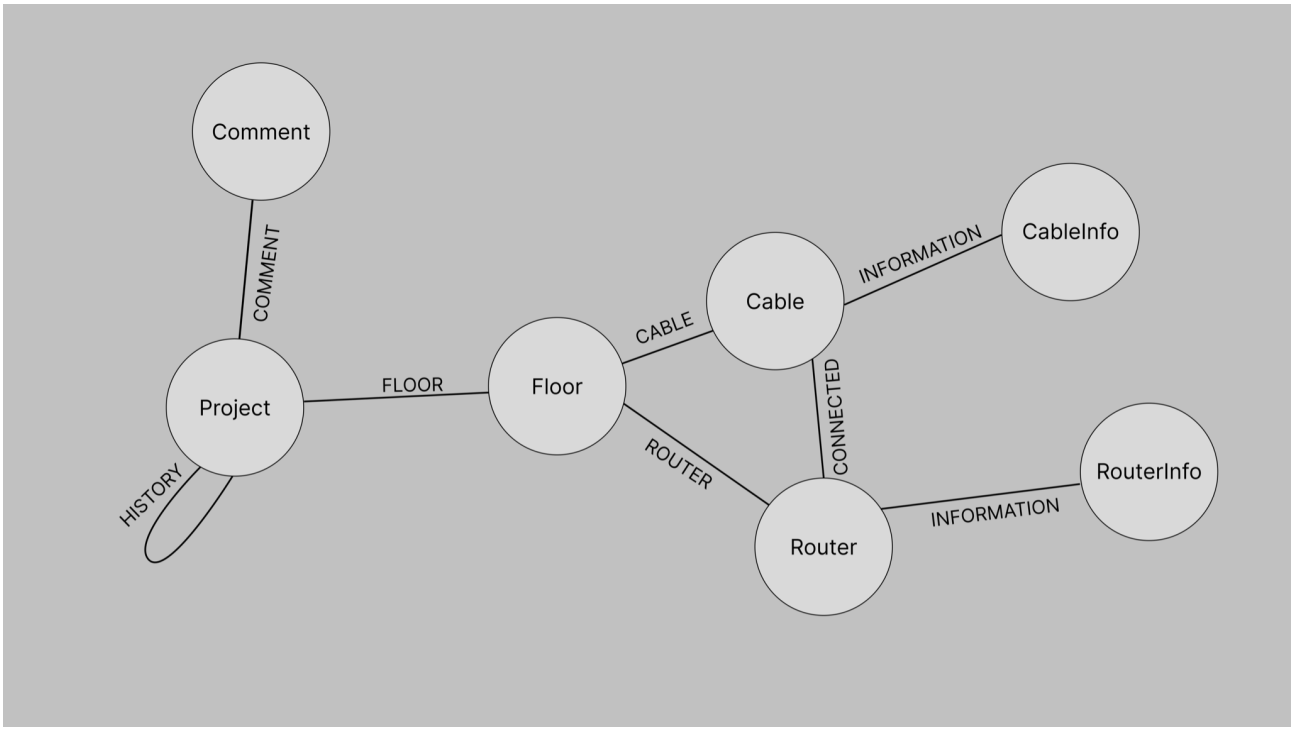

Рис. 2. Нереляционная модель данных.

Описание

- Проект:

  id - идентификатор проекта. Тип int. V = 4b.

  address - адрес проекта. Тип string. V = 100b.

  name - название проекта. Тип string. V = 100b.

  DateOfChange - дата создания/изменения. Тип Date. V = 3b.

- Этаж:

  number - номер этажа. Тип int. V = 4b

  Plan - ссылка на схему. Тип string. V = 100b

- Роутер:

  router_id - уникальный идентификатор роутера. Тип int. V = 4b

  name - название роутера. Тип string. V = 100b

- Кабель:

  cable_id - уникальный идентификатор кабеля. Тип int. V = 4b

  length - длина кабеля. Тип Float. V = 4b.
- RouterInfo:

  routerinfo_id - уникальный идентификатор роутера. Тип int. V = 4b.

  photo - ссылка на фотку. Тип string. V = 100b.

  description - описание компоненты. Тип string. V = 100b.

  model - модель роутера. Тип string. V = 100b.

  port_count - кол-во портов. Тип int. V = 4b.

  DateOfCreation - дата создания/изменения. Тип Date. V = 3b
- CableInfo:

  cableinfo_id - уникальный идентификатор кабеля. Тип int. V = 4b.

  photo - ссылка на фотку. Тип string. V = 100b.

  description - описание компоненты. Тип string. V = 100b

  type - тип кабеля. Тип string. V = 100b.

  DateOfCreation - дата создания/изменения. Тип Date. V = 3b
- Комментарий:

  comment_id - идентификатор комментария. Тип int. V = 4b.

  comment_text - текст комментария. Тип string. V = 256b

  comment_date - дата создания комментария. Тип Date. V = 3b
- Связь - ребро, соединяющее 2 узла.

  id - уникальный идентификатор узла. Тип Int. V = 4b

От каждого узла могут выходить рёбра, имеющие следующие метки:

- FLOOR - узел является этажом относительно узла, в которого входит ребро.
- COMMENT - узел является комментарием относительно узла, в которого входит ребро.

- CABLE - узел является кабелем относительно узла, в которого входит ребро.
- ROUTER - узел является роутером относительно узла, в которого входит ребро.
- CONNECTED - показывает, что кабель соединен с маршрутизатором.
- INFORMATION - узел является информацией компоненты относительно узла, в которого входит ребро.

Объем модели:

- Vкаб кабеля ~ 8 b
- Vрт роутера ~ 104 b
- Vэ инф. этажа ~ 104 b
- Vир инфо роутера ~ 311 b
- Vик инфо кабеля ~ 307 b
- Vк комментария ~ 263 b
- Vсв связи ~ 4 b
- Vпр проекта ~ 207 b

Объем данных для хранения

V(N) = (Vпр + (Vэ + Vсв) * Nf + (Vоп + Vкаб + Vрт + 2 * Vсв) * Nоб + Vk * Nk) * N

Nf(кол-во этажей) = 10

Nоб(кол-во оборудования) = 10

Nk(кол-во комментариев) = 2, тогда

V(N) = 6,4 * N kB, где N - кол-во проектов.

Запросы

1. Изменение названия маршрутизатора

```
MATCH(r:Router {router_id: ${id}}) SET r.name = ${name}
RETURN p
```

2. Прокладка кабеля

```
MATCH(r1:Router {router_id: ${id1}}) -[:ROUTER]-
(f:Floor)
MATCH(r2:Router {router_id: ${id2}})
MATCH(d:CableInfo {cableinfo_id: ${desc_id}})
CREATE (c:Cable {length: ${len}})
CREATE (c)-[:CABLE]->(f)
CREATE (c)<-[:INFORMATION]-(d)
CREATE (r1)-[:CONNECTED]->(c)
CREATE (r2)-[:CONNECTED]->(c)
RETURN c
```

3. Добавление этажа

```
MATCH(p:Project {id: ${proj_id}})
CREATE (p) <-[:FLOOR]- (f:Floor {number: ${floor_num},
plan: ${plan_link}})
RETURN f
```

4. Удаление элемента сети

```
MATCH(r:Router {router_id: ${id}})
DETACH DELETE r
MATCH(c:Cable {cable_id: ${id}})
DETACH DELETE c
```

5. Создание проекта

```
CREATE (p:Project {id: ${id},address: ${address},name:
${name},DateOfChange: ${date}}) <-[:FLOOR]- (f:Floor
{number: 1, plan: null})
RETURN p
```

6. Найти проект

```
MATCH(p:Project {address: ${address},name:
${name},DateOfChange: ${date}})
RETURN p
```

7. Удалить проект

```
MATCH(p:Project {id: ${id})
DETACH DELETE c
```

8. Прокладка кабеля между этажами

```
MATCH(r1:Router {router_id: ${id1}}) -[:ROUTER]-
(f1:Floor {number: ${floor1}})
MATCH(r2:Router {router_id: ${id2}}) -[:ROUTER]-
(f2:Floor {number: ${floor2}})
MATCH(d:CableInfo {cableinfo_id: ${desc_id}})
CREATE (c:Cable {length: ${len}})
CREATE (c)-[:CABLE]->(f1)
CREATE (c)-[:CABLE]->(f2)
CREATE (c)<-[:INFORMATION]-(d)
CREATE (r1)-[:CONNECTED]->(c)
CREATE (r2)-[:CONNECTED]->(c)
RETURN c
```

3.2. Аналог модели данных для SQL СУБД

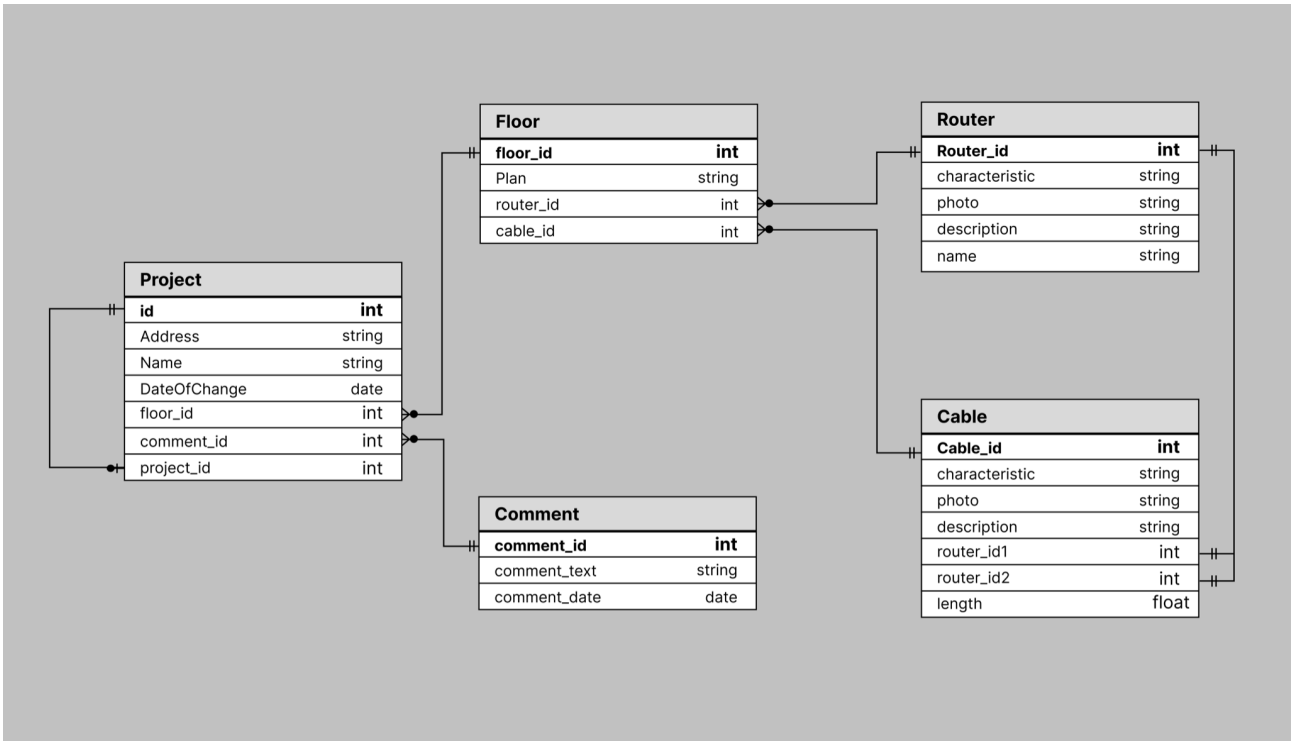

Рис. 3. Реляционная модель данных.

Описание:

- Проект

  ключ: id - уникальный идентификатор проекта. Тип int. V = 4b

  address - адрес проекта. Тип varchar(100). V = 200b

  name - название проекта. Тип varchar(100). V = 200b

  DateOfChange - дата создания. Тип Date. V = 3b

  внешний ключ: floor_id - id этажа. Тип int. V = 4b

  внешний ключ: comment_id - id комментария. Тип int. V = 4b

  внешний ключ: project_id - ссылка на проект. Тип int. V = 4b

- Этаж:

  ключ: floor_id - id этажа. Тип int. V = 4b

  Plan - ссылка на схему. Тип varchar(100). V = 200b

  внешний ключ: router_id - id роутера. Тип int. V = 4b

внешний ключ: cable_id - id кабеля. Тип int. V = 4b

- Роутер:

  ключ: router_id - уникальный идентификатор роутера. Тип int. V = 4b

  name - название роутера. Тип varchar(100). V = 200b

  characteristic - модель роутера. Тип varchar(100). V = 200b

  photo - ссылка на фото. Тип varchar(100). V = 200b

  description - описание роутера. Тип varchar(100). V = 200b

- Кабель:

  ключ: cable_id - уникальный идентификатор кабеля. Тип int. V = 4b

  characteristic - имя кабеля. Тип varchar(100). V = 200b

  photo - ссылка на фото. Тип varchar(100). V = 200b

  description - описание кабеля. Тип varchar(100). V = 200b

  router_id1 - первый роутер, который соединяет кабель. Тип Int. V = 4b

  router_id2 - второй роутер, который соединяет кабель. Тип Int. V = 4b

  length - длина кабеля. Тип int. V = 4b

- Комментарий:

  ключ: comment_id - уникальный идентификатор комментария. Тип int. V = 4b

  comment_text - текст комментария. Тип varchar(256). V = 512b

  comment_date - дата создания комментария. Тип Date. V = 3b

Объем:

- Vкаб кабеля ~ 616b
- Vрт роутера ~ 804 b
- Vэ этаж ~ 212 b
- Vк комментария ~ 519 b

- Vпр проекта ~ 419 b

V(N) = (Vпр + (Vэ * Nf) + (Vкаб + Vрт) * Nоб + (Vk * Nk)) * N

Пусть Nf(кол-во этажей) = 10

Nоб(кол-во оборудования) = 10

Nk(кол-во комментариев) = 2

V(N) = 17,7 * N kB, где N - кол-во проектов.

Запросы

1. Запрос на изменение названия маршрутизатора

```
UPDATE Router SET characteristic = “NewName“ WHERE id =
id1;
```

2. Запрос на добавление Кабеля:

```
INSERT Cable(cable_id, characteristic, photo ,
description, router_id1, router_id2)
VALUES (cable_id1, characteristic1, photo1 ,
description1, router_id11, router_id21);
```

3. Запрос на добавление Этажа:

```
INSERT Floor(floor_id, Plan, router_id , cable_id)
VALUES (floor_id1, Plan1, router_id1 , cable_id1);
```

4. Удаление элемента сети

- Кабель:

```
DELETE FROM Cable WHERE cable_id = 'cable_id1';
```

- Роутер:

```
DELETE FROM Router WHERE router_id = 'router_id1';
```

5. Запрос на добавление проекта:

```
INSERT Project(id, address, name, dateOfChange,
floor_id,comment_id, project_id)
VALUES (id1, address1, name1, dateOfCreation1,
floor_id1, comment_id1, project_id1);
```

6. Изменение существующего кабеля

```
UPDATE Cabel SET router_id1 = 'new_router_id1'
UPDATE Cabel SET router_id1 = 'new_router_id2'
```

7. Найти проект

- По названию “find_name”:

```
SELECT * FROM Project WHERE name = “find_name”;
```

- По дате “find_date”:

```
SELECT * FROM Project WHERE DateOfChange = “find_date”;
```

- По адресу “find_address”:

```
SELECT * FROM Project WHERE address = “find_address”;
```

8. Удаление проектов

- Удаление одного проекта по id = “id_del”

```
DELETE FROM Project WHERE id = 'id_del';
```

- Удалить все проекты

```
DELETE FROM Project
```

Сравнение моделей

NoSQL требует заметно меньше памяти, сравнивая объем данных в модели NoSQL и объем данных в модели SQL. Как указано в примере для Nf количества этажей, Noб количества оборудования и Nk количество комментариев в SQL данные занимают 17,7 * N, а в NoSQL 6,1 * N, при N = 10.

Вывод

NoSQL подходит для данной задачи лучше чем SQL, так как замечен очевидный выигрыш по памяти.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Описание

Приложение состоит из трех контейнеров: backend, frontend и db. Контейнер backend развертывается из докер образа на основе Node.js и содержит серверную часть приложения. Здесь осуществляется обработка запросов от клиента и взаимодействие с графовой базой данных Neo4j. Для работы с базой данных использовался модуль nest-neo4j из npm. При первом запуске сервера добавляется тестовый набор проектов.

Контейнер frontend основан на образе Nginx и содержит клиентскую часть приложения. Клиентская часть написана с использованием фреймворка React. Для навигации между страницами приложения используется модуль React Router. Интерфейс приложения позволяет пользователю переходить на различные страницы как через элементы интерфейса, так и изменяя адресную строку браузера.

Приложение позволяет просматривать список проектов с выводом информации, такой как название проекта, адрес, дата создания, количество этажей и количество комментариев. Также реализован функционал поиска по этим параметрам. При поиске отправляется запрос на сервер, который возвращает подходящие проекты. На странице списка проектов можно также удалять, импортировать и экспортировать проекты. Пользователь может отмечать галочками проекты, которые хочет удалить или экспортировать. При нажатии кнопки экспорта на сервере формируется JSON файл и отправляется клиенту для скачивания. При нажатии кнопки импорта пользователю предлагается загрузить JSON файл с проектами.

Также со страницы списка проектов можно создать новый проект или просмотреть один из существующих. Переходы реализованы через элементы модуля React Router. На странице проекта пользователь может изменять название проекта и адрес, а также перейти на страницу с комментариями и историей изменений. Главным компонентом на этой странице является редактор сети, реализованный с помощью элемента canvas. Он позволяет перемещать

маршрутизаторы при помощи мыши, добавлять и удалять компоненты, а также выполнять групповое удаление элементов. Также реализована возможность перемещения виртуальной камеры и изменения масштаба. При выборе элемента на плане этажа, появляется форма для изменения его параметров. Изменения сохраняются по нажатию кнопки. Также пользователю доступен поиск по всем компонентам на этаже. Он может выбрать компонент из списка и увидеть его положение на плане. Также пользователь может добавлять и удалять этажи.

На странице с комментариями пользователю доступны комментарии к проекту, он может добавлять комментарии и выполнять поиск по ним. Поиск реализован с использованием запроса на сервер, доступна фильтрация по содержимому и дате.

4.2. Использованные технологии

1. Docker и docker-compose: Docker контейнеры использовались для развертывания серверной и клиентской частей приложения.
2. Node.js и NestJs: Серверная часть приложения была реализована с использованием Node.js, а фреймворк NestJs обеспечивает структуру и удобство разработки, а также интеграцию с базой данных Neo4j.
3. React: Клиентская часть приложения была разработана с использованием React.
4. Redux: Для управления состоянием клиентского приложения была использована библиотека Redux. Она позволяет сохранять состояние проекта при переходе между страницами
5. Neo4j: Для хранения данных на сервере была использована графовая база данных Neo4j.

4.3. Снимки экрана приложения

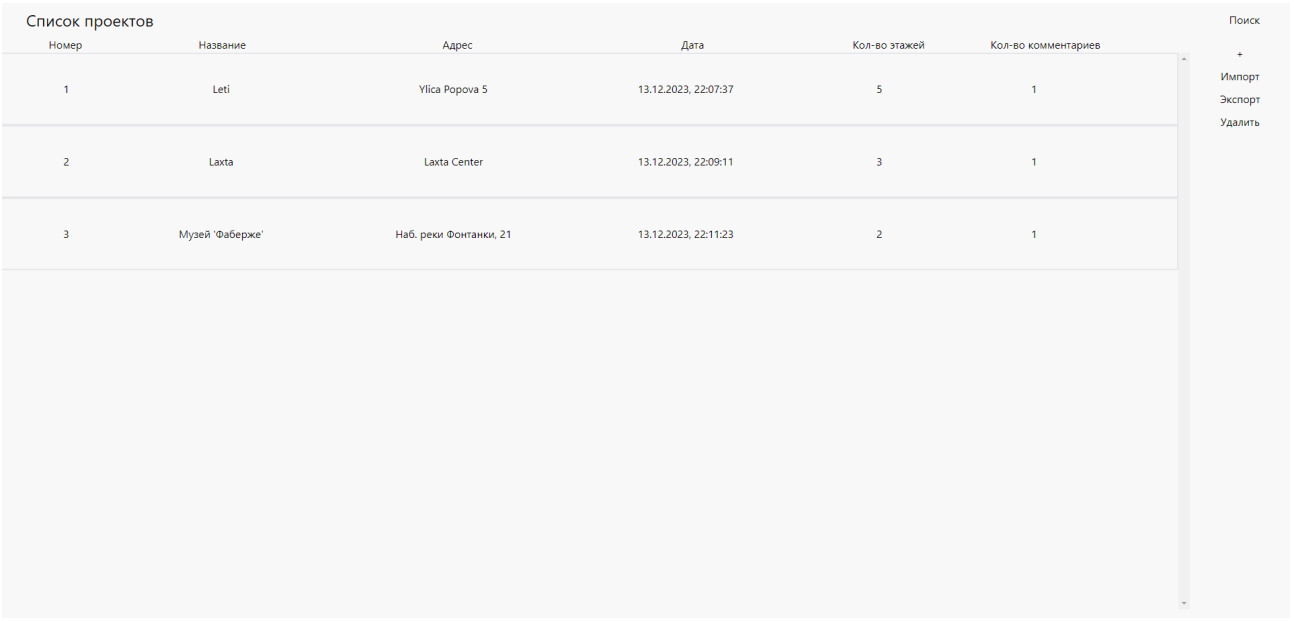

Рис. 4. Страница со списком проектов.

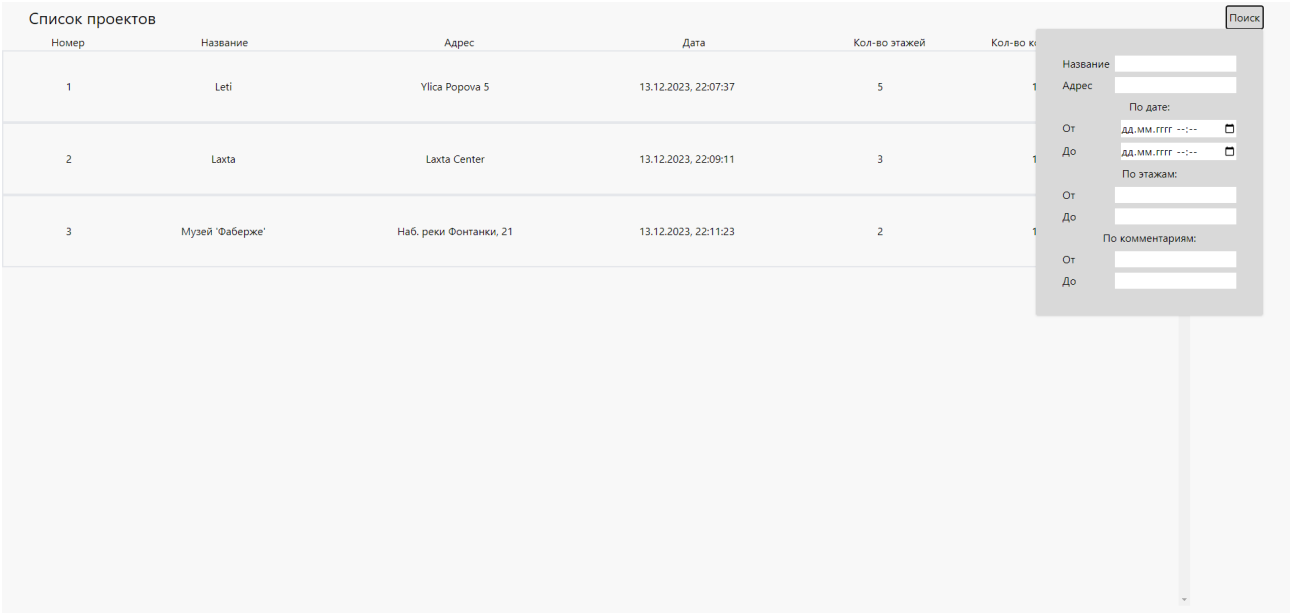

Рис. 5. Меню фильтра на странице со списком проектов.

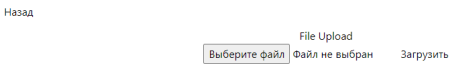

Рис. 6. Страница импорта.

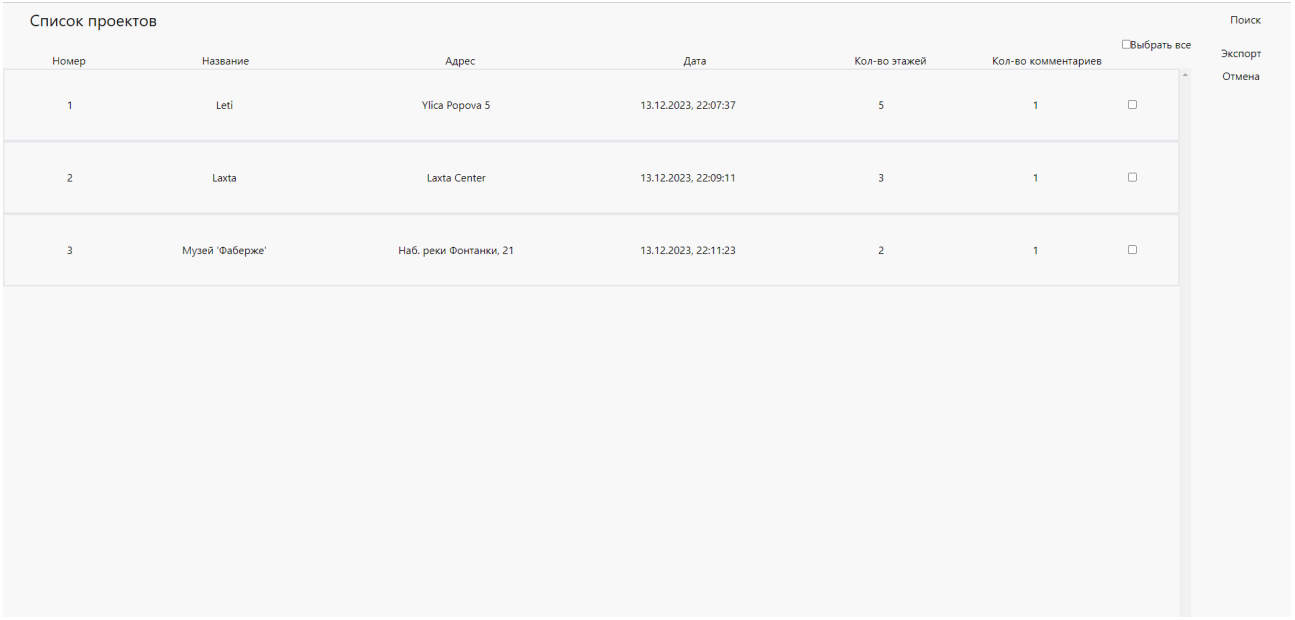

Рис. 7. Меню экспорта.

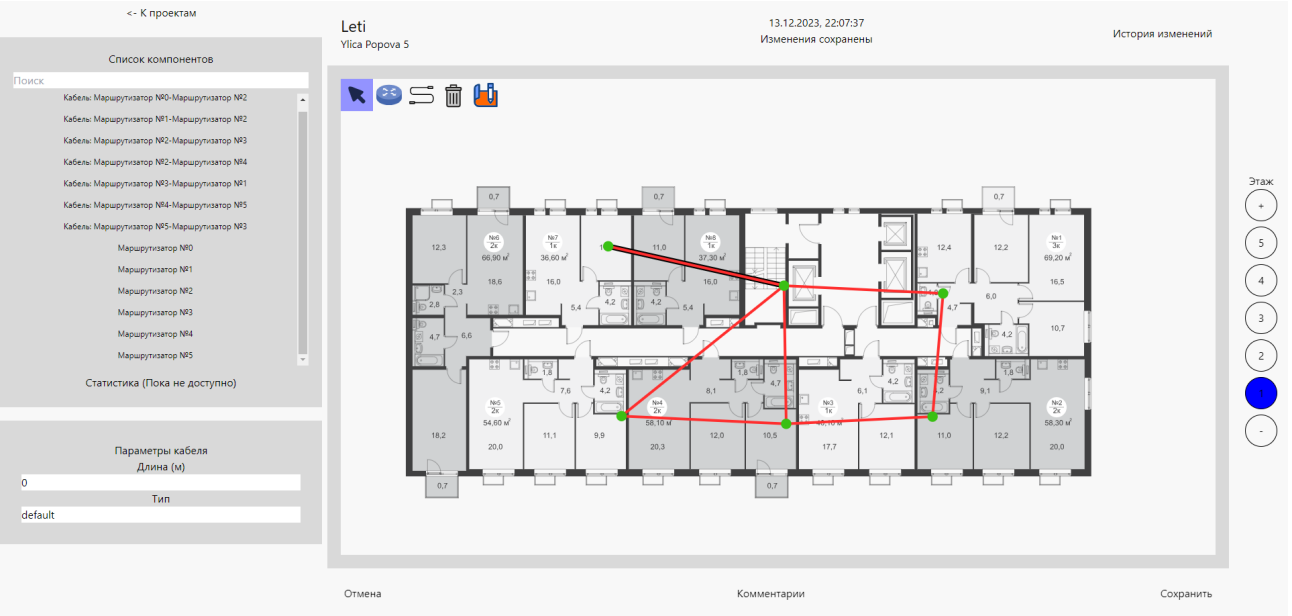

Рис. 8. Страница проекта.

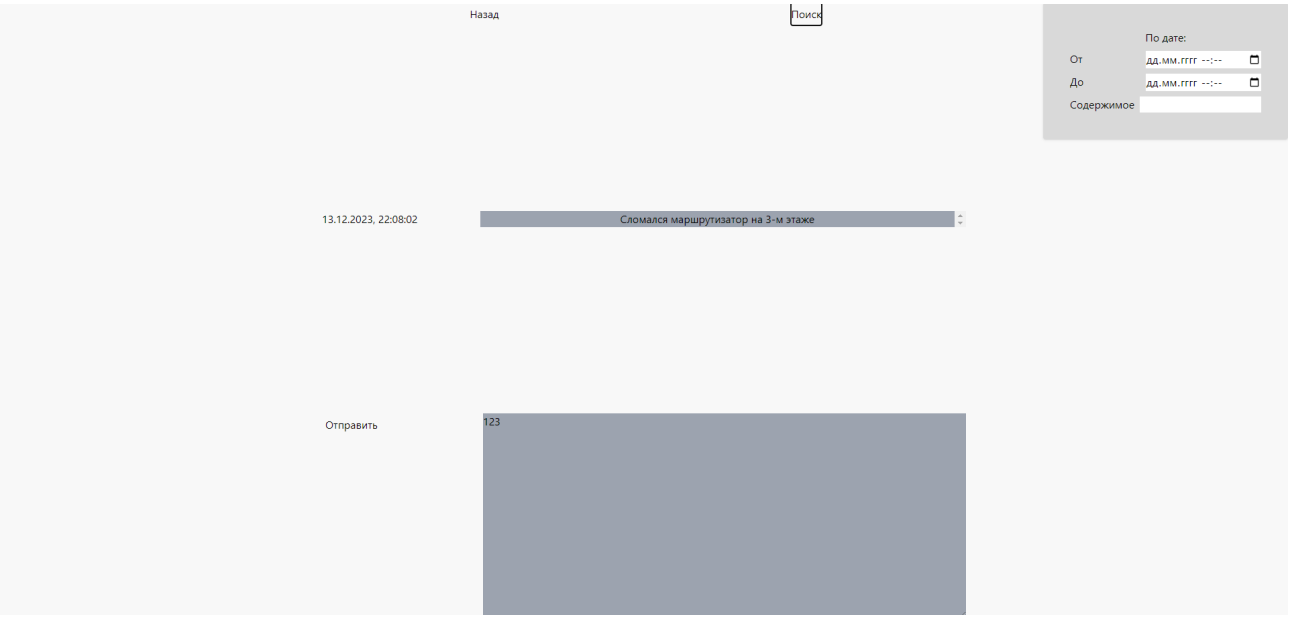

Рис. 9. Страница комментариев к проекту.

5. ВЫВОДЫ

5.1. Достигнутые результаты

В результате разработки было создано веб-приложение, предоставляющее пользователю возможность управлять проектами кабельных сетей. Приложение позволяет создавать, изменять, импортировать, экспортировать и удалять проекты.

Были реализованы функции добавления комментариев, просмотра предыдущих версий проекта и редактирования сетей с использованием графического редактора в браузере.

В приложении имеется возможность выполнять поиск проектов по различным параметрам, таким как название, адрес и дата создания. Также реализован поиск компонентов проекта по их названию.

Для разворачивания приложения используется docker-compose, что обеспечивает простоту и надежность процесса установки и настройки. При первом запуске приложение также инициализирует базу данных несколькими примерами проектов.

5.2. Недостатки и пути для улучшения полученного решения

В ходе разработки были выявлены некоторые недостатки и ошибки, связанные с ограниченным временем и ресурсами. Ниже описаны проблемы, которые нужно исправить:

1. Отсутствие функции загрузки плана этажа. Эта функция является важной для наглядной визуализации проектов. Необходимо разработать и интегрировать функциональность, позволяющую загружать план этажа в приложение.

2. Отсутствие статистики по проектам. Это может ограничить возможность анализа и мониторинга проектов. Необходимо добавить функции, позволяющие просматривать статистику, такую как общая длина кабелей и др.

3. Отсутствие выбора моделей роутеров и типов кабелей. Это ограничение может привести к ограниченным возможностям подбора кабелей для проектов.

Необходимо добавить функционал выбора моделей роутеров и типов кабелей, а также возможность просмотра описания и добавления новых.

4. Отсутствие расчета длины кабеля на основе схемы. Эта функция может значительно упростить процесс планирования и оптимизации проектов. Необходимо добавить возможность автоматического расчета длины кабеля на основе схемы.

5. Незаконченный внешний вид приложения. Улучшение интерфейса может улучшить пользовательский опыт и сделать приложение более привлекательным для использования.

6. Отсутствие поддержки нескольких пользователей. Добавление поддержки нескольких пользователей позволит совместно работать над проектами и обеспечить безопасность данных. Рекомендуется внедрить систему аутентификации и авторизации для поддержки нескольких пользователей.

7. Низкая эффективность использования памяти при сохранении версий проектов. Необходимо пересмотреть текущую систему сохранения версий проектов и изменить ее, чтобы использовать не создавать каждый раз копию проекта.

8. Неэффективное использование графовой базы данных. Поскольку преимущества графовой базы данных не оправдались в контексте задачи, необходимо пересмотреть использование данного типа базы данных и выбрать более подходящую альтернативу.

5.2. Будущее развитие решения

Дальнейшее развитие приложения может включать следующие направления:

1. Доработка редактора для возможности создания или загрузки планировок в различных форматах. Это позволит пользователям удобно работать с существующими планировками и вносить необходимые изменения.

2. Интеграция с системами мониторинга сетей. Это поможет пользователям получать актуальную информацию о состоянии сети, обнаруживать проблемы и быстро реагировать на них.

3. Создание сервиса для бригад обслуживания. Предоставление инструментов и информации для бригад обслуживания позволит им эффективно выполнять задачи по обслуживанию и ремонту сети.

4. Исправление ошибок и недостатков, выявленных в текущей версии приложения. Внимательный анализ обратной связи пользователей и устранение обнаруженных проблем позволят повысить качество и удобство использования приложения.

5. Улучшение производительности и оптимизация работы приложения. Проанализировать и устранить узкие места в работе системы, улучшить алгоритмы и процессы для повышения эффективности приложения.

6. Добавление новых функций и возможностей, которые могут быть полезными пользователям. Например, функция автоматического расчета оптимальных маршрутов кабелей или интеграция с базой данных компонентов и материалов для удобного подбора необходимых элементов.

7. Повышение безопасности и защиты данных пользователей. Внедрение мер безопасности, таких как шифрование данных и аутентификация, поможет защитить информацию и предотвратить несанкционированный доступ.

Продолжение развития приложения в указанных направлениях поможет улучшить функциональность, эффективность и удобство использования, что будет способствовать удовлетворенности пользователей и успеху на рынке.

6. ПРИЛОЖЕНИЯ

6.1. Документация по сборке и развертыванию приложения

1. Установить docker
2. Клонировать репозиторий в папку
3. В корневой папке запустить:
   ```
   docker compose up
   ```
4. Перейти в браузере по адресу http:/localhost:8080

6.2. Инструкция для пользователя

Первая страница – страница с проектами. На ней можно выбрать проект с помощью фильтра проектов (по названию, дате, адресу, кол-ву комментариев и кол-ву этажей) или просто щелкнуть по проекту.

Далее на странице самого проекта можно

- изменить имя проекта
- добавить этаж
- посмотреть планировку каждого этаже
- изменить планировку этажа
- добавить маршрутизатор
- добавить кабель
- посмотреть статистику
- посмотреть комментарий
- вернуться к списку проектов

На странице с комментариями можно:

- добавить комментарий
- найти определенный комментарий с помощью фильтра по дате и содержимому.

На странице со статистикой можно посмотреть статистику по проекту.

ЛИТЕРАТУРА

1. Репозиторий проекта // nosql2h23-cable URL : https://github.com/moevm/nosql2h23-cable
2. Документация Neo4j // URL: https://neo4j.com
3. Документация NestJs // URL: https://docs.nestjs.com
4. Документация React // URL: https://react.dev/learn