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 поддержки ЖКХ»

Студент гр. 0303 _____ Афанасьев Д.В.

Студент гр. 0303 _____ Морозов А.Ю.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2023

ЗАДАНИЕ

Студенты

Афанасьев Д.В.

Морозов А.Ю

Группа 0303

Тема проекта: Сервис поддержки ЖКХ

Исходные данные:

Сделать сервис фиксации проблем с ЖКХ инфраструктурой, планирования регулярных и внеплановых работ, формирования ответов на обращения собственников.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарий использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 20.09.2023

Дата сдачи реферата: 25.12.2023

Дата защиты реферата: 25.12.2023

Студент гр. 0303		Афанасьев Д.В.
Студент гр. 0303		Морозов А.Ю.
Преподаватель		Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема сервис поддержки ЖКХ с СУБД Neo4j. Во внимание будут приниматься такие аспекты как производительность и удобство разработки. Найти исходный код и всю дополнительную информацию можно по ссылке: https://github.com/moevm/nosql2h23-zkh

ANNOTATION

As part of this course, it was supposed to develop an application in a team on one of the set topics. The topic was chosen housing and communal services support with the Neo4j DBMS. Aspects such as performance and ease of development will be taken into account. You can find the source code and all additional information at the link: https://github.com/moevm/nosql2h23-zkh

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

Цель работы – создать высокопроизводительное и удобное решение для сервиса поддержки ЖКХ.

Было решено разработать веб-приложение, которое позволит фиксировать проблемы с ЖКХ инфраструктурой, планировать регулярные и внеплановые работ, формировать ответы на обращения собственников.

2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение с использованием СУБД Neo4j, а также с возможностью локального развертывания приложения с помощью docker-compose.

3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

3.1. Макеты UI

Рисунок 1 – Макет UI для пользователя менеджера

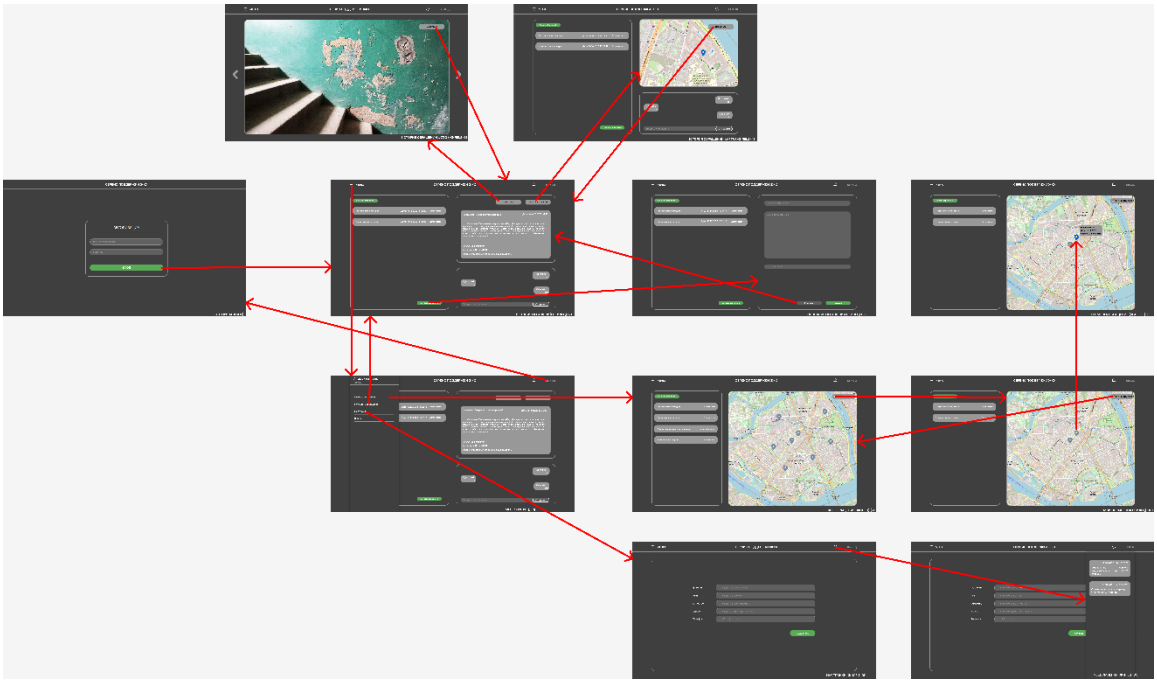


Рисунок 2 – Макет UI для пользователя жильца

Рисунок 3 – Макет UI для пользователя сотрудника

3.2. Описание сценариев использования

3.2.1. Сценарий использования - «Создания обращения»:

Действующее лицо: Жилец

Предусловие: Пользователь находится на странице авторизации

Основной сценарий:

1. Пользователь вводит логин и пароль;
2. Нажимает на кнопку авторизация;
3. После успешной авторизации попадает на страницу “История обращений”;
4. Пользователь нажимает на кнопку “Создать обращение”;
5. В правой части экрана, появилась форма для создания обращения;
6. После заполнения полей формы, при нажатии на кнопку “Создать” будет создано новое обращение.

Альтернативный сценарий:

1. Пользователь может отменить заполнение формы нажатием на кнопку “Отменить”

3.2.2. Сценарий использования - «Просмотр обращений пользователя»:

Действующее лицо: Жилец

Предусловие: Пользователь авторизован, находиться на странице “История обращений”

Основной сценарий:

1. Пользователь может выбрать обращение в списке, которое хочет посмотреть;
2. После нажатия на обращение, в правой части откроется информация об обращение с чатом;
3. Пользователь может открыть фотографии прикрепленные к обращению нажав на кнопку “Показать фото”;
4. Пользователь может просматривать все фото, при помощи стрелок “лево” и “право”;

Альтернативный сценарий:

1. Пользователь может просмотреть на карте место обращения для этого ему необходимо нажать на кнопку “Показать на карте”
2. После этого на месте обращения появится карта с геометкой;
3. Для возврата описания обращения необходимо нажать кнопку “Вернуться”.

3.2.3. Сценарий использования - «Изменение настроек пользователя»:

Действующее лицо: Жилец, Менеджер, Работник

Предусловие: Пользователь авторизован, находиться на странице “История обращений”

Основной сценарий:

1. Пользователь нажимает на иконку “Меню”;
2. В открывшемся списке, нажимает на поле “Настройки”;
3. Пользователь находиться на странице настроек, где может изменить сведения о себе;
4. После изменения сведений, может их сохранить нажав на кнопку “Сохранить”.

3.2.4. Сценарий использования - «Просмотр обращений»:

Действующее лицо: Работник

Предусловие: Пользователь находится на странице авторизации

Основной сценарий:

1. Пользователь вводит логин и пароль;
2. Нажимает на кнопку авторизация;
3. После успешной авторизации попадает на страницу “Мои задачи”;
4. Перед ним представлен список обращений, которые он должен выполнить;
5. Пользователь может выбрать обращение в списке, которое хочет посмотреть;
6. После нажатия на обращение, в правой части откроется информация об обращение;
7. Пользователь может открыть фотографии прикрепленные к обращению нажав на кнопку “Показать фото”;
8. Пользователь может просматривать все фото, при помощи стрелок “лево” и “право”;

Альтернативный сценарий:

1. Пользователь может просмотреть на карте место обращения для этого ему необходимо нажать на кнопку “Показать на карте”
2. После этого на месте обращения появится карта с геометкой;
3. Для возврата описания обращения необходимо нажать кнопку “Вернуться”.

3.2.5. Сценарий использования - «Просмотр обращений компании»:

Действующее лицо: Менеджер

Предусловие: Пользователь находится на странице авторизации

Основной сценарий:

1. Пользователь вводит логин и пароль;
2. Нажимает на кнопку авторизация;
3. После успешной авторизации попадает на страницу “Все обращения”;
4. Перед ним представлен список обращений, которые направлены в ЖКХ;
5. Пользователь может выбрать обращение в списке, которое хочет посмотреть;
6. После нажатия на обращение, в правой части откроется информация об обращение;
7. Пользователь может открыть фотографии, прикрепленные к обращению нажав на кнопку “Показать фото”;
8. Пользователь может просматривать все фото, при помощи стрелок “лево” и “право”;

Альтернативный сценарий:

1. Пользователь может просмотреть на карте место обращения для этого ему необходимо нажать на кнопку “Показать на карте”.
2. После этого на месте обращения появится карта с геометкой;
3. Для возврата описания обращения необходимо нажать кнопку “Вернуться”.

3.2.6. Сценарий использования - «Импорт данных»:

Действующее лицо: Менеджер

Предусловие: Пользователь авторизован, находиться на странице “Все обращения”

Основной сценарий:

1. Пользователь нажимает на иконку “Меню”;
2. В открывшемся списке, нажимает на поле “Настройки”;
3. Пользователь выбирает файл для импорта;
4. После нажимает на кнопку импортировать.

3.2.7. Сценарий использования - «Экспорт данных»:

Действующее лицо: Менеджер

Предусловие: Пользователь авторизован, находиться на странице “Все обращения”

Основной сценарий:

1. Пользователь нажимает на иконку “Меню”;
2. В открывшемся списке, нажимает на поле “Настройки”;
3. После нажимает на кнопку “Экспортировать”;
4. Происходит загрузка файла и данными из бд.

4. МОДЕЛЬ ДАННЫХ

4.1. Нереляционные модели данных

4.1.1. Описание назначений коллекций, типов данных и сущностей

Вершина: Менеджер/Manager

Атрибуты:

1. ФИО/name: String

2. Номер телефона/phone_number: String

3. Логин/login: String

4. Пароль/password: String

Вершина: Работник/Worker

Атрибуты:

1. ФИО/name: String

2. Номер телефона/phone_number: String

3. Логин/login: String

4. Пароль/password: String

Вершина: Жилец/Tenant

Атрибуты:

1. ФИО/name: String

2. Адрес/address: String

3. Номер телефона/phone_number: String

4. Логин/login: String

5. Пароль/password: String

Вершина: Обращение/Appeal

Атрибуты:

1. Название/name: String

2. Описание/description: String

3. Статус/status: String - [“Новое обращение”, “В обработке”, “В работе”, “Выполнено”]

4. Отзыв/feedback: String

5. Дата и время создания/created_at: LocalDateTime

6. Адрес/address: String

7. Долгота/longitude: Float

8. Широта/latitude: Float

9. Тип работ/type: String - [“Сантехника”, “Электрика”, “Ремонт”, “Садовые”, “Уборка”, “Прочее”]

Вершина: Работы проводимые ЖКХ/Activity

Атрибуты:

1. Название/name: String

2. Описание/description: String

3. Статус/status: String - [“Новая работа”, “В работе”, “Выполнено”]

4. Дата и время начала работ/date_start: LocalDateTime

5. Дата и время окончания работ/date_end: LocalDateTime

6. Адрес/address: String

7. Долгота/longitude: Float

8. Широта/latitude: Float

9. Тип работ/type: String - [“Сантехника”, “Электрика”, “Ремонт”, “Садовые”, “Уборка”, “Прочее”]

Вершина: Сообщение/Message

Атрибуты:

1. Сообщение/msg: String

2. Дата и время отправки/date: LocalDateTime

Вершина: История назанчений /Assigment

Атрибуты:

1. Дата и время наназначения на работу или обращение/date_start: LocalDateTime

2. Дата и время снятия с работы или обращения/date_end: LocalDateTime

Отношение: Управляет/Controls_the

• Показывает, какими заявками управляет менеджер

• Дата назначения/date_assignment

Отношение: Создал/Creates_a

• Показывает, какую заявку создал жидец и кукую работу завел менеджер

• Данные отсутствуют

Отношение: Отправил/Sent

• Показывает, какое сообщение отправил менеджер или жилец

• Данные отсутствуют

Отношение: Принадлежит/Belongs_to

• Показывает, какому оброщению принадлежит сообщение

• Данные отсутствуют

Отношение: Работает на/Works_on

• Показывает, над какими заявками работает работник

• Дата назначения/date_assignment

Отношение: Был назначен/Was_assigned

• Показывает, в каких сроках был назначен менеджер/работник

• Данные отсутствуют

Отношение: Назначен на/Assigned_on

• Показывает, на какие заявки/работы был назначен менеджер/работник

• Данные отсутствуют

4.1.2. Оценка удельного объема информации, хранимой в модели

В среднем на одно обращение приходиться около 1,5 жильцов, 0,5 менеджеров, 1 работника, 0,2 смены менеджера и 0,3 смены работника, 0,5 плановых работ с 0,1 сменой менеджера и 0,2 смены работника, также на одно обращение приходиться 20 сообщений, по одному ребру типа controls_the и около 60 обычных ребер, следовательно суммарный объем: * V(N) = (Vappeal + 1,5 * Vtenant + 0,5 * Vmanager + Vworker + 0,5 * Vactivity + 0,8 * Vassigment + 20 * Vmessage + 2 * Vcontrols_the + 60 * Vedge) * N

Суммарный объем системы для 500 обращений: * V(500) = (3618B + 1,5 * 500B + 0,5 * 200B + 200B + 0,5 * 2678B + 0,8 * 124B + 20 * 1064B + 2 * 64B + 240B) * 500 = 13,2MB

4.1.3. Примеры запросов к модели

• Создание обращения (use case №1) > CREATE (appeal:Appeal { > name: “Название обращения”, > description: “Описание обращения”, > status: “Статус обращения”, > created_at: datetime(), > address: “Адрес по которому было создано обращение”, > longitude: 59.907034, > latitude: 30.512711, > }) > WITH appeal > MATCH (tenant:Tenant) WHERE id(tenant) = {id_tenant} > CREATE (tenant)-[:Creates_a]->(appeal) > RETURN appeal

• Просмотр обращений пользователя (use case №2) > MATCH (tenant:Tenant)-[:Creates_a]->(nodes: Appeal) > WHERE id(tenant) = {id_tenant} > RETURN nodes

• Изменение настроек пользователя (use case №3) > MATCH (tenant:Tenant) > WHERE id(tenant) = {id_tenant} > WITH tenant > SET tenant.name = “new_name”, > tenant.phone_number = “new_number” > RETURN tenant

• Просмотр активных обращений (use case №5) > MATCH (nodes: Appeal) > WHERE n.status IN [‘Новое’, ‘В работе’] > RETURN nodes

• Просмотр уведомлений (use case №6) > MATCH (nodes: Activity) > WHERE n.status IN [‘Новое’, ‘В работе’] > RETURN nodes

• Просмотр обращений работника (use case №7) > MATCH (nodes_activity: Activity)<-[:Works_on]-(worker:Worker)-[:Works_on]->(nodes_appeal: Appeal) > WHERE id(worker) = {id_worker} > RETURN nodes_activity, nodes_appeal

• Изменение настроек работника (use case №8) > MATCH (worker:Worker) > WHERE id(worker) = {id_worker} > WITH worker > SET worker.name = “new_name”, > worker.phone_number = “new_number” > RETURN worker

• Просмотр обращений компании (use case №15) > MATCH (managers:Manager)-[:Controls_the]->(nodes: Activity)<-[:Created_a]-(tenants:Tenant) > RETURN nodes, managers, tenants

• Взятие обращения (use case №16) > MATCH (manager:Manager), (appeal: Appeal) > WHERE id(manager) = {id_manager} AND id(appeal) = {id_appeal} > CREATE (manager)-[:Controls_the]->(id_appeal)

• Просмотр плановых работ (use case №17) > MATCH (managers:Manager)-[:Created_a]->(nodes: Activity) > RETURN nodes, managers

• Создание плановых работ (use case №18) > CREATE (activity:Activity { > name: “Название работы”, > description: “Описание работы”, > status: “Статус работы”, > date_start: datetime("“), > date_end: datetime(”“), > address:”Адрес по которому будут идти работы", > longitude: 59.907034, > latitude: 30.512711, > }) > WITH activity > MATCH (manager:Manager) WHERE
• id(manager) = {id_manager} > CREATE (manager)-[:Creates_a]->(activity) > RETURN activity

• Изменение настроек менеджера (use case №19) > MATCH (manager:Manager) > WHERE id(manager) = {id_manager} > WITH manager > SET manager.name = “new_name”, > manager.phone_number = “new_number” > RETURN manager

• Просмотр обращений менеджера (use case №20) > MATCH (managers:Manager)-[:Controls_the]->(nodes: Activity) > WHERE id(manager) = {id_manager} > RETURN nodes

Таким образом для каждого юзкейса нужен один запрос.

4.1.4. Графическое представление модели

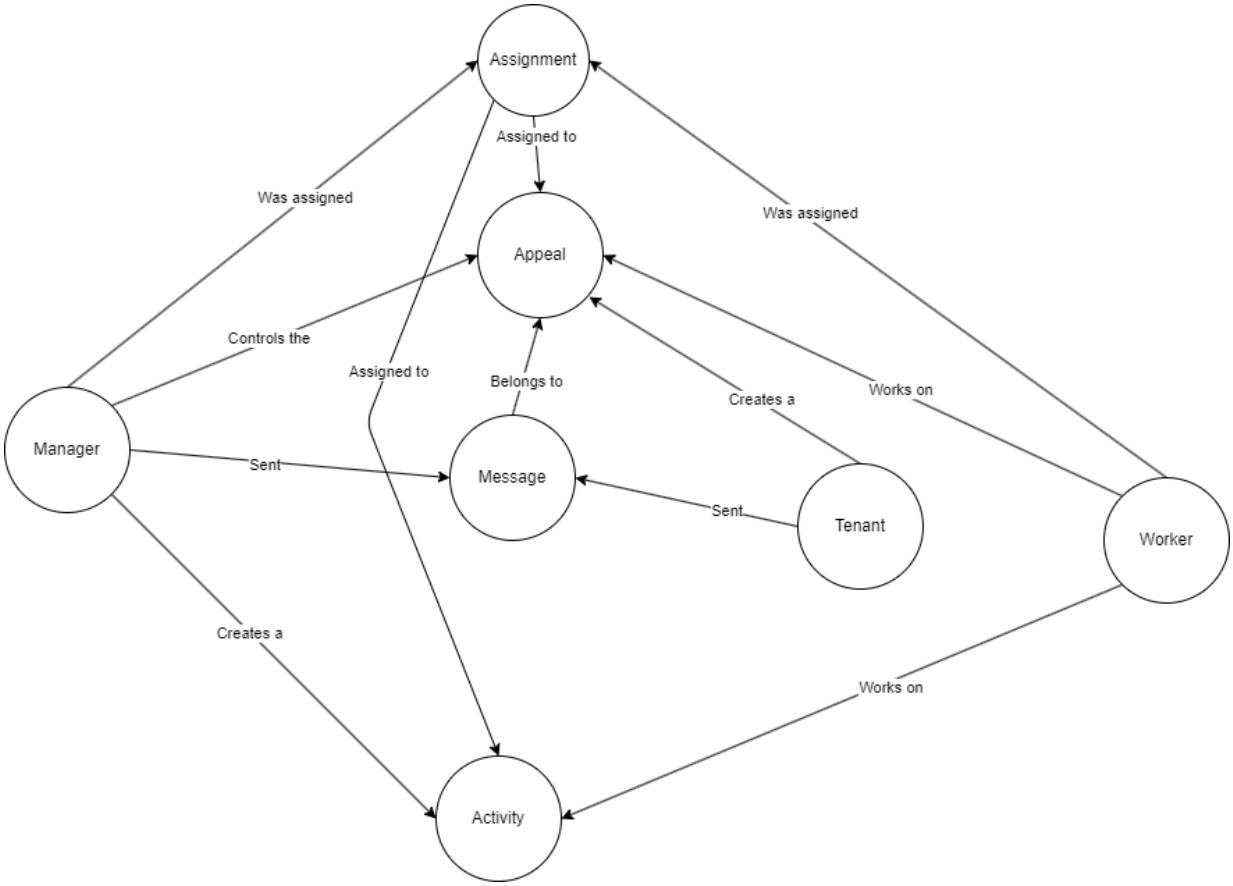


Рисунок 4 - Модель нереляционной БД

4.2. Реляционные модели данных

4.2.1. Описание назначений коллекций, типов данных и сущностей

Таблица Manager:

- id_manager - serial => Vid_m = 4b
- name - varchar(70) => Vn = 70B
- phone_number - varchar(12) => Vpn = 12B
- login - varchar(20)=> Vl = 20B
- password - varchar(8) => Vp = 8B

Vmanager = Vid_m + Vn + Vpn + Vl + Vp = 114B

Таблица Worker:

- id_worker - serial => Vid_w = 4b
- name - varchar(70) => Vn = 70B
- phone_number - varchar(12) => Vpn = 12B
- login - varchar(20)=> Vl = 20B

- password - varchar(8) => Vp = 8B Vworker = Vid_w + Vn + Vpn + Vl + Vp = 114B

Таблица Tenant

- id_tenant - serial Vid_t = 4b
- name - varchar(70) => Vn = 70B
- address - varchar(150) => Va = 150B
- phone_number - varchar(12) => Vpn = 12B
- login - varchar(20)=> Vl = 20B
- password - varchar(8) => Vp = 8B

Vtenant = Vid_t + Vn + Va + Vpn + Vl + Vp = 264B

Таблица Appeal

- id_appeal - serial => Vid_a = 4b
- name - varchar(100) => Vn = 100B
- description - varchar(1000) => Vd = 1000B
- feedback - varchar(500) => Vf = 500B
- status - varchar(15) => Vs = 15B
- type - varchar(10) => Vt = 10B
- created_at - timestamp => Vc_a = 8B
- address - varchar(150) => Va = 150B
- longitude - float => Vlon = 4B
- latitude - float => Vlat = 4B
- date_assignment - timestamp => Vd_a = 8B Vappeal = Vid_a + Vn + Vd + Vf + Vs + Vt + Vc_a + Va + Vlat + Vlon + Vd_a= 1803B

Таблица Activity

- id_activity - serial => Vid_a = 4b
- name - varchar(100) => Vn = 100B
- description - varchar(1000) => Vd = 1000B
- status - varchar(15) => Vs = 15B
- type - varchar(10) => Vt = 10B
- date_start - timestamp => Vd_s = 8B
- date_end - timestamp => Vd_e = 8B
- address - varchar(150) => Va = 150B
- longitude - float => Vlon = 4B
- latitude - float => Vlat = 4B
- date_assignment - timestamp => Vd_a = 8B Vactivity = Vid_a + Vn + Vd + Vs + Vt + Vd_s + Vd_e + Va + Vlat + Vlon + Vd_a = 1311B

Таблица Message_Tenant

- id_appeal- integer => Vid_a = 4b

- id_tenant - integer => Vid_t = 4b
- msg - varchar(500) => Vm = 500B
- date - timestamp => Vd_s = 8B Vmessage_tenant = Vid_a + Vid_t + Vm + Vd_s = 516B

Таблица Message_Manager

- id_appeal- integer => Vid_a = 4b
- id_manager - integer => Vid_m = 4b
- msg - varchar(500) => Vm = 500B
- date - timestamp => Vd_s = 8B Vmessage_manager = Vid_a + Vid_m + Vm + Vd_s = 516B

Таблица Appeal_Workers

- id_appeal- integer => Vid_a = 4b
- id_worker - integer => Vid_w = 4b
- date_assignment - timestamp => Vd_a = 8B Vappeal_workers = Vid_a + Vid_w + Vd_a= 16B

Таблица Activity_Workers

- id_activity - integer => Vid_a = 4b
- id_worker - integer => Vid_w = 4b
- date_assignment - timestamp => Vd_a = 8B Vactivity_workers = Vid_a + Vid_w + Vd_a = 16B

Таблица Assignment_activity_manager

- id_assignment - serial => Vid_a = 4b
- id_manager - integer => Vid_m = 4b
- id_activity - integer => Vid_a = 4b
- date_start - timestamp => Vd_s = 8B
- date_end - timestamp => Vd_e = 8B Vassignment_activity_manager = 28B

Таблица Assignment_appeal_manager

- id_assignment - serial => Vid_a = 4b
- id_manager - integer => Vid_m = 4b
- id_appeal - integer => Vid_a = 4b
- date_start - timestamp => Vd_s = 8B
- date_end - timestamp => Vd_e = 8B Vassignment_appeal_manager = 28B

Таблица Assignment_activity_worker

- id_assignment - serial => Vid_a = 4b
- id_worker - integer => Vid_w = 4b
- id_activity - integer => Vid_a = 4b
- date_start - timestamp => Vd_s = 8B

- date_end - timestamp => Vd_e = 8B Vassignment_activity_worker = 28B

Таблица Assignment_appeal_worker

- id_assignment - serial => Vid_a = 4b
- id_worker - integer => Vid_w = 4b
- id_appeal - integer => Vid_a = 4b
- date_start - timestamp => Vd_s = 8B
- date_end - timestamp => Vd_e = 8B Vassignment_appeal_worker = 28B

4.2.2. Оценка удельного объема информации, хранимой в модели

В среднем на одно обращение приходиться около 1,5 жильцов, 0,5 менеджеров, 1 работника, 0,2 смены менеджера и 0,3 смены работника, 0,5 плановых работ с 0,1 сменой менеджера и 0,2 смены работника, также на одно обращение приходиться 20 сообщений(Message_Tenant и Message_Manager) : * V(N) = (Vappeal + 1,5 * Vtenant + 0,5 * Vmanager + Vworker + Vappeal_workers + 0,5 * Vactivity + 0,5 * Vactivity_workers + 0,8 * Vassignment_appeal_worker + 20 * Vmessage_Tenant) * N.

Замечание:

- Vassignment_appeal_worker = Vassignment_activity_worker =Vassignment_appeal_manager =Vassignment_activity_manager - информация о смене менеджера или работника
- Vmessage_tenant = Vmessage_manager - сообщения менеджера или жильца Суммарный объем системы для 500 обращений:
- V(500) = (1803B + 1,5 * 264B + 0,5 * 114B + 114B + 16B + 0,5 * 1311B + 0,5 * 16B + 0,8 * 28B + 20 * 516B) * 500 = 6,3MB.

4.2.3. Пример запросов

- Создание обращения (use case №1) > INSERT INTO “Appeal” (“created_at”, “name”, “description”, “status”, “address”, “longitude”, “latitude”, “id_tenant”) VALUES (CURRENT_TIMESTAMP, ‘Название обращения’, ‘Описание обращения’, ‘Статус обращения’, ‘Адрес по которому было создано обращение’, ‘59.907034’, ‘30.512711’, :id_tenant);

- Просмотр обращений пользователя (use case №2) > SELECT “Appeal”.*, “Manager”.id, “Manager”.name, > “Tenant”.id, “Tenant”.name > FROM “Appeal” > JOIN “Manager” ON “Appeal”.id_manager = “Manager”.id_manager > JOIN “Tenant” ON “Appeal”.id_tenant = “Tenant”.id_tenant > WHERE “Appeal”.id_tenant = :id_tenant;

- Изменение настроек пользователя (use case №3) > UPDATE “Tenant” SET “name” = ‘new_name’, “phone_number” = ‘new_number’ WHERE “id_tenant” = :id_tenant;
- Просмотр активных обращений (use case №5) > SELECT “Appeal”.*, “Manager”.id, “Manager”.name, > “Tenant”.id, “Tenant”.name > FROM “Appeal” > JOIN “Manager” ON “Appeal”.id_manager = “Manager”.id_manager > JOIN “Tenant” ON “Appeal”.id_tenant = “Tenant”.id_tenant > WHERE “status” IN (‘Новое’, ‘В работе’);;
- Просмотр уведомлений (use case №6) > SELECT “Activity”.* , “Manager”.id, “Manager”.name > FROM “Activity” > JOIN “Manager” ON “Activity”.id_manager = “Manager”.id_manager
- Изменение настроек работника (use case №8) > UPDATE “Worker” SET “name” = ‘new_name’, “phone_number” = ‘new_number’ WHERE “id_worker” = :id_worker;
- Просмотр обращений компании (use case №15) > SELECT “Appeal”.*, “Manager”.id, “Manager”.name, > “Tenant”.id, “Tenant”.name > FROM “Appeal” > JOIN “Manager” ON “Appeal”.id_manager = “Manager”.id_manager > JOIN “Tenant” ON “Appeal”.id_tenant = “Tenant”.id_tenant
- Взятие обращения (use case №16) > UPDATE “Appeal” SET “id_manager” = :id_manager, WHERE “id_appeal” = :id_appeal;
- Просмотр плановых работ (use case №17) > SELECT “Activity”.*, “Manager”.id, “Manager”.name > FROM “Activity” > JOIN “Manager” ON “Activity”.id_manager = “Manager”.id_manager
- Создание плановых работ (use case №18) > INSERT INTO “Activity” (“name”, “description”, “status”, “address”, date_start, date_end, “longitude”, “latitude”, “id_manager”) VALUES (CURRENT_TIMESTAMP, ‘Название обращения’, ‘Описание обращения’, ‘Статус обращения’, ‘Адрес по которому было создано обращение’, ““,”“, ‘59.907034’, ‘30.512711’, :id_manager);
- Изменение настроек менеджера (use case №19) > UPDATE “Manager” SET “name” = ‘new_name’, “phone_number” = ‘new_number’ WHERE “id_manager” = :id_manager;
- Просмотр обращений менеджера (use case №20) > SELECT “Appeal”.*, “Manager”.id, “Manager”.name, > “Tenant”.id, “Tenant”.name > FROM “Appeal” > JOIN “Manager” ON “Appeal”.id_manager = “Manager”.id_manager > JOIN “Tenant” ON “Appeal”.id_tenant = “Tenant”.id_tenant > WHERE “Appeal”.id_manager = :id_manager;

4.2.4. Графическое представление модели

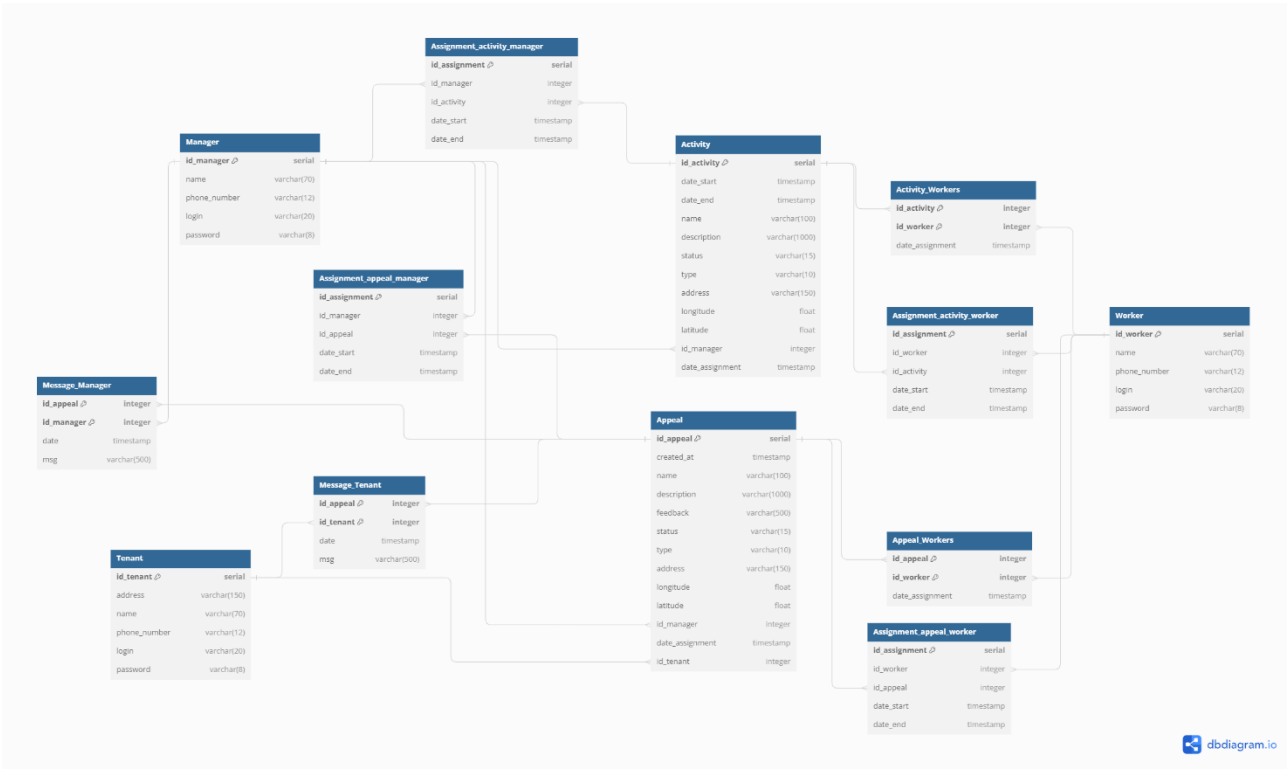


Рисунок 5 - Модель реляционной БД

4.3. Сравнение моделей

Удельный объем информации

В SQL реализации модели данных было создано много дополнительных таблиц для связывания сущностей, что увеличивает количество хранимых данных. Следовательно при большом количестве связей данных SQL хранит больше данных, чем NoSQL.

Запросы по отдельным юзкейсам

Для каждого Use case для обеих реализаций было создано по одному запросу (SQL было сложнее написать), но при усложнении Use case в SQL потребуется несколько запросов, в отличие от NoSQL.

4.4. Вывод

Исходя из всего вышеперечисленного можно сделать вывод, что NoSQL(Neo4j) больше подходит для данной системы из-за большого количества связей между данными.

5. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

5.1. Краткое описание

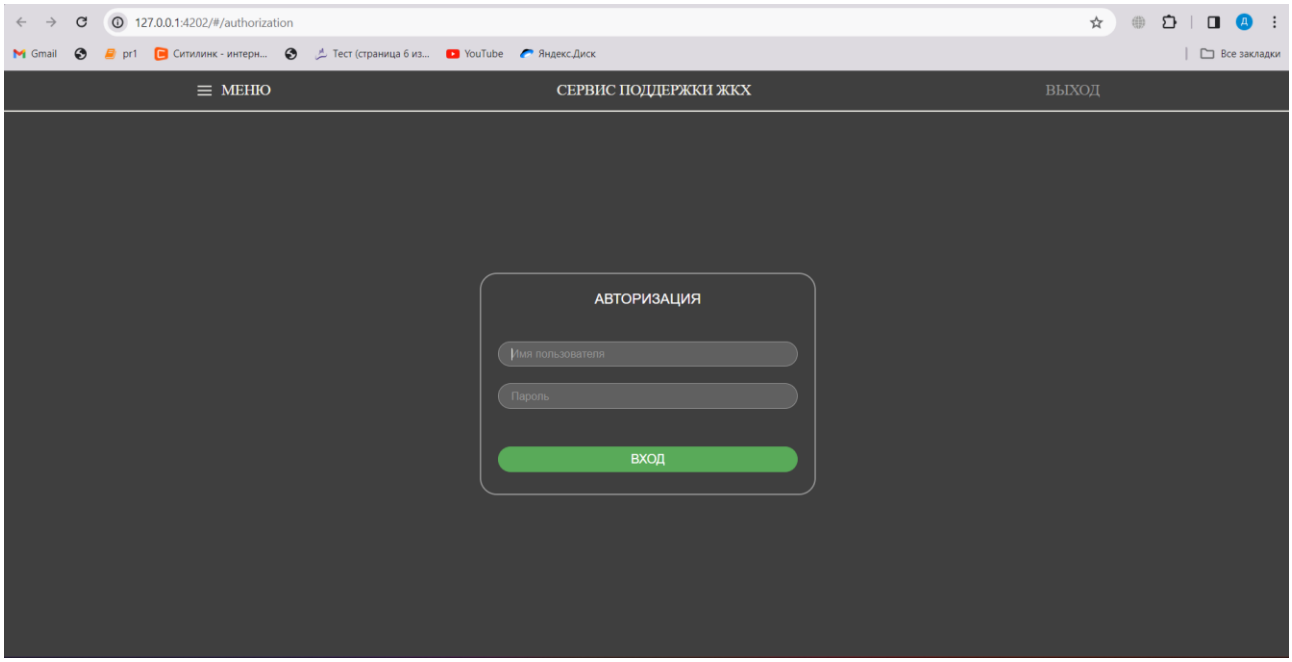
Back-end представляет из себя сервер, разработанн на языке программирования Java и в фреймворке Spring.

За основную СУБД взята Neo4j, взаимодействие происходит на основе Spring Data.

Front-end реализован на языке TypeScript и фреймворке Angular.

5.2. Снимки экрана приложения

Рисунок



6 – Страница Авторизации

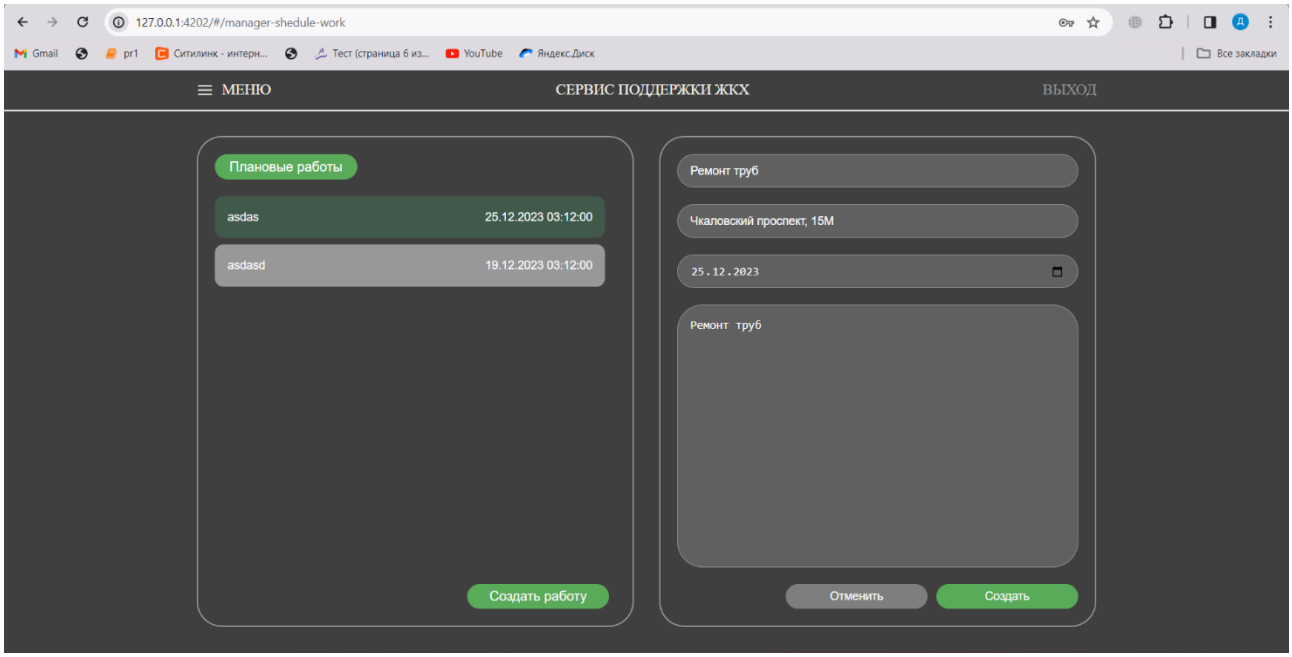


Рисунок 7 – Создание плановых работ

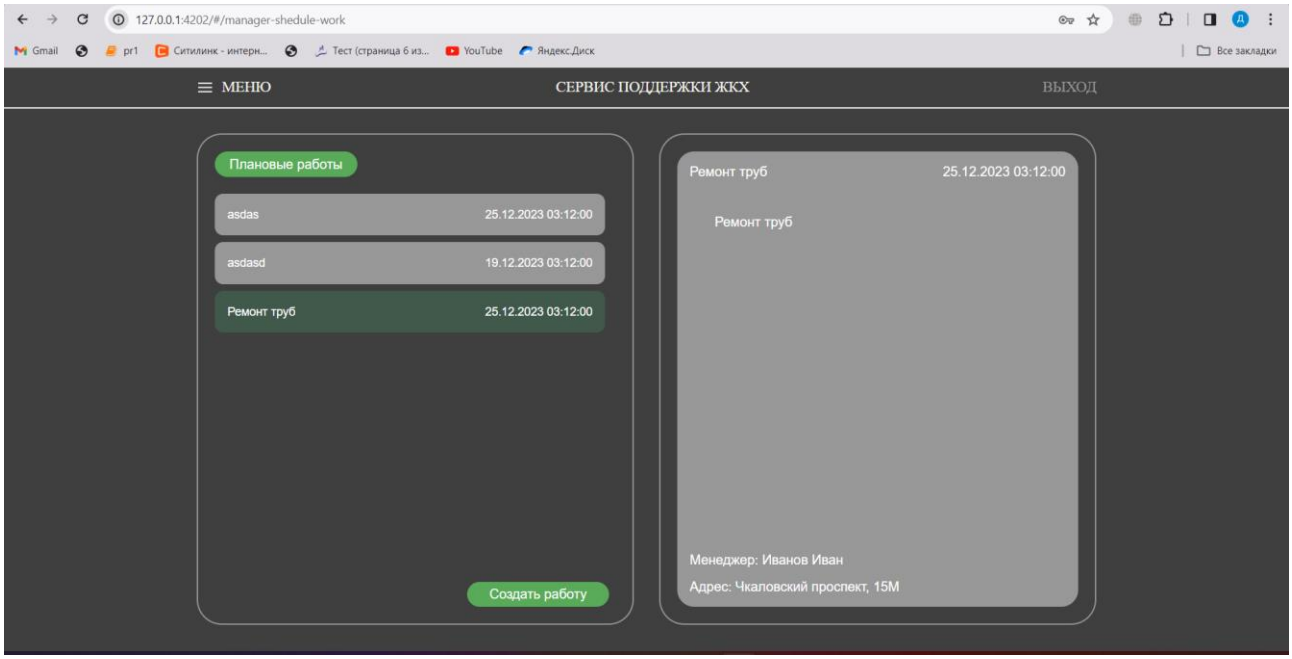


Рисунок 8 – Страница плановых работ

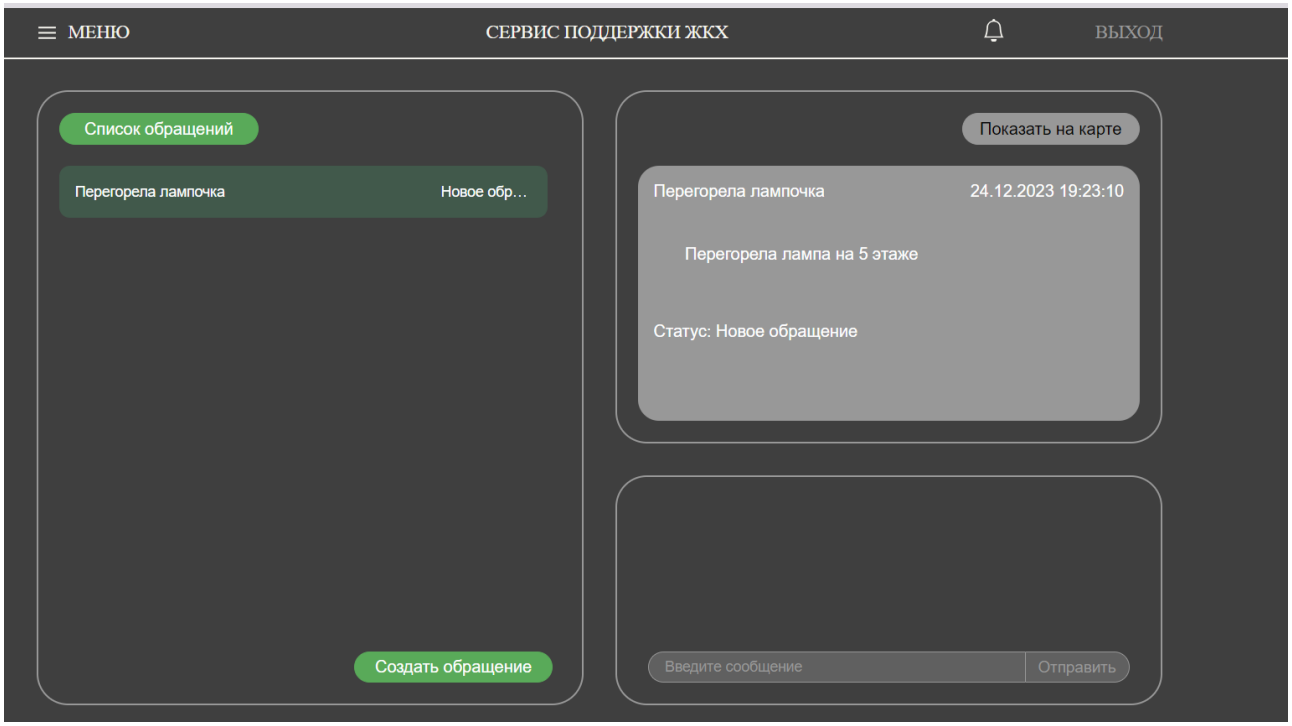


Рисунок 9 – Страница списка обращений пользователя

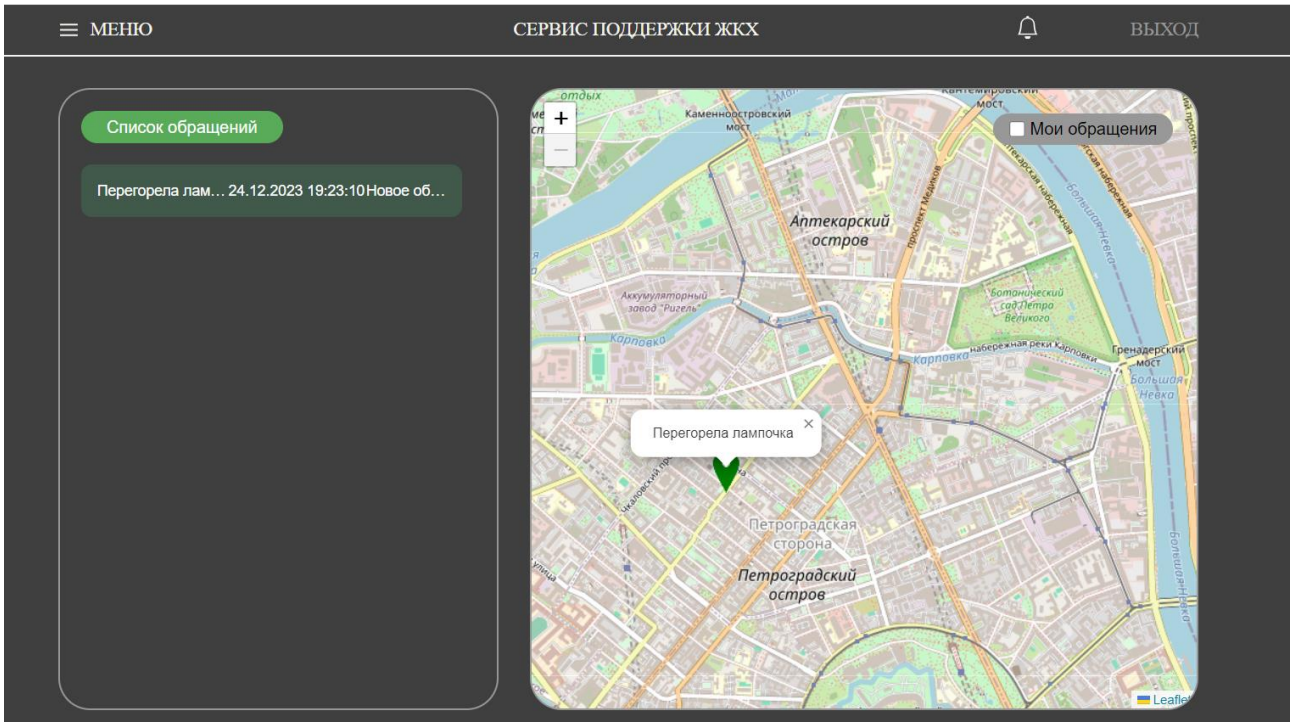


Рисунок 10 – Страница просмотра всех обращений

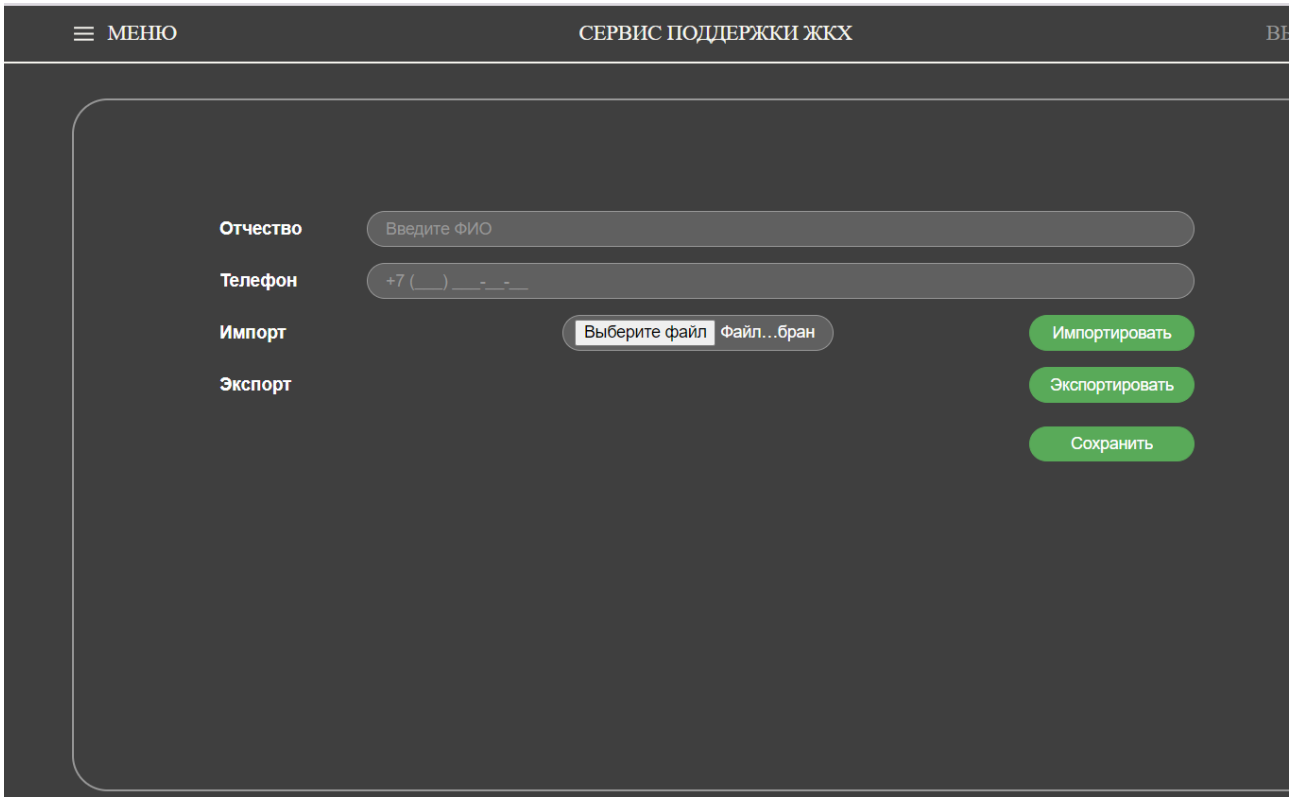


Рисунок 11 – Страница настроек менеджера с возможностью импорта и экспорта

6. ВЫВОД

6.1. Достигнутые результаты

В ходе работы было разработан сервис поддержки ЖКХ с использованием графовой базы данных Neo4j. Была реализована вся заявленная функциональность: корректная работа с Neo4j, добавление и работа с обращениями, развертывание приложения с помощью docker-compose.

6.2. Недостатки и пути для улучшения полученного решения

Нет возможности добавления пользователей через клиентскую часть. Отсутствие передача токенов в запросе к бэкенду, без него присутствует возможность использования не доступных запросов определенным ролям пользователей. Решением является реализация данных возможностей.

6.3. Будущее развитие решения

Планируется создание админской части и возможности добавления фотографий к обращениям.

7. ПРИЛОЖЕНИЯ

Документация по сборке и развертыванию приложения

1. Склонировать репозиторий (указан в списке литературы)
2. Развернуть приложения через docker-compose с помощью следующих команд:
   - sudo docker-compose build --no-cache
   - sudo docker-compose up –d
3. Открыть приложение в браузере по адресу: http://127.0.0.1:4202/

8. ЛИТЕРАТУРА

1. Ссылка на github проекта: https://github.com/moevm/nosql2h23-zkh/
2. Документация Neo4j: https://neo4j.com/docs
3. Документация Spring: https://spring.io/