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: ИНФОРМАЦИОННАЯ СИСТЕМА ПЕСЧАНОГО КАРЬЕРА

Студенты гр. 9383 _____ Гладких А.А.

_____ Камзолов Н.А.

_____ Крейсманн К.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2022

ЗАДАНИЕ

Студенты: Гладких А.А., Камзолов Н.А., Крейсманн К.В.

Группа 9383

Тема проекта: Разработка информационной системы песчаного карьера

Исходные данные:
Необходимо реализовать веб-приложения для управления работой песчаного карьера, с использованием СУБД Neo4j.

Содержание пояснительной записки:
«Содержание», «Введение», «Качественные требования к решению», «Сценарий использования», «Модель данных», «Разработанное приложение», «Выводы», «Приложения», «Литература»

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания: 01.09.2022

Дата сдачи реферата: 22.12.2022

Дата защиты реферата: 22.12.2022

Студенты гр. 9383 _____ Гладких А.А.

_____ Камзолов Н.А.

_____ Крейсманн К.В.

Преподаватель _____ Заславский М.М.

АННОТАЦИЯ

При прохождении данной дисциплины необходимо было разработать приложение в команде на одну из поставленных тем с использованием определенной СУБД. Была выбрана тема «ИС Песчаного карьера», в которой необходимо разработать приложение для управления (работа с сотрудниками, их сменами, с учетом добычи и так далее) песчаным карьером. При разработке использовался следующий стек технологий: JavaScript + Vue.js + Node.js на фронтенде; Kotlin + Spring + Neo4j на бэкенде. Найти исходный код и всю дополнительную информацию можно по ссылке: https://github.com/moevm/nosql2h22-sand-mine

SUMMARY

When passing this discipline, it was necessary to develop the application into one of the topics set using a DBMS. The topic "IS Sand Mine" was chosen, in which it is necessary to develop an application for managing (working with selection, their shifts, taking into account production, and so on) a sand pit. The following technologies are used in development: JavaScript + Vue.js + Node.js on the frontend; Kotlin + Spring + Neo4j on the backend. You can find the source code and all additional information at the link: https://github.com/moevm/nosql2h22-sand-mine

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность решаемой проблемы.

В настоящее время в качестве информационных систем песчаных карьеров используют информационные системы общего назначения, поэтому веб-сервисы визуально перегружены и трудны в изучении.

Постановка задачи.

Разработать информационную систему песчаного карьера. Разработанный веб-сервис должен иметь следующий функционал:

- учет добычи
- добавление/удаление/изменение информации о работнике
- проверка доступа работника в зону работ

Предлагаемое решение.

В рамках данной работы предлагается решение, которое построено с использованием графовой базы данных Neo4j, инструмента разработки Spring и языка Kotlin для реализации серверной части и инструмента разработки Vue.js и языка JavaScript для реализации клиентской части.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение с использованием СУБД Neo4j.

2. СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ

2.1. **Макет UI.** (рисунок 1)

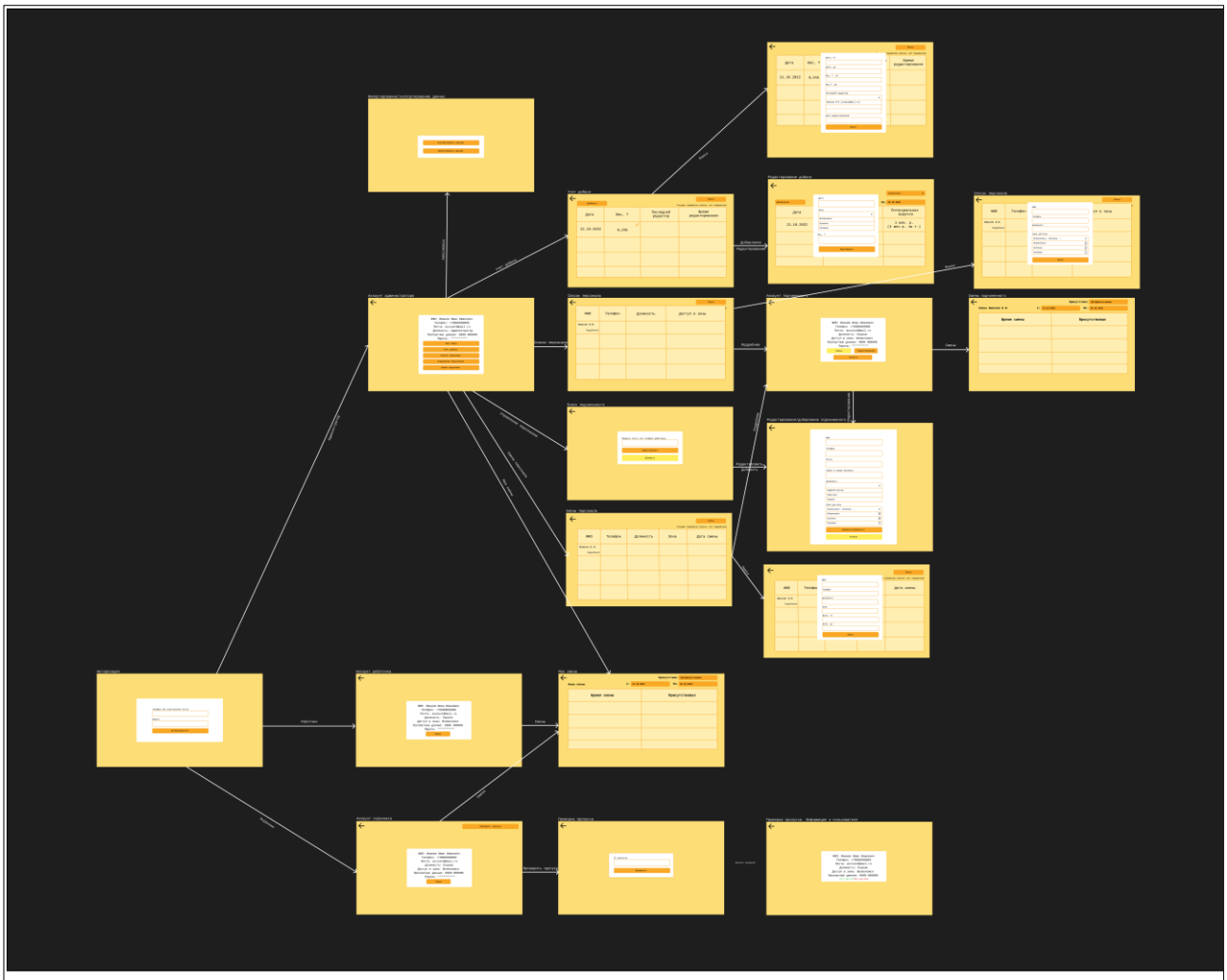

Рисунок 1 - Макет UI

Ссылка на макет доступна в разделе «Список использованных источников».

2.2. Сценарии использования для задачи импорта данных.

Сценарий использования - «Массовый импорт»

Действующее лицо: главный администратор

Основной сценарий:

1. Сценарий «Авторизация».
2. Пользователь нажимает кнопку «Данные».
3. Пользователь попадает на страницу массового импорта и экспорта.
4. Пользователь выбирает файл, из которого он хочет осуществить импорт, с помощью кнопки «Выберите файл».
5. Пользователь нажимает кнопку «Импортировать данные».
6. Пользователь подтверждает своё намерение в диалоговом окне об удалении всех имеющихся данных в базе с помощью кнопки «Подтвердить».
7. Данные импортируются.

Альтернативный сценарий:

- Пользователь не выбирает/выбирает несуществующий файл
- Пользователю высвечивается уведомление о том, что файл не выбран/выбран несуществующий файл
- Пользователю предлагается заново выбрать файл

Сценарий использования - «Добавление данных о добыче»

Действующее лицо: Администратор

Основной сценарий:

1. Администратор заходит на свою страницу
2. Администратор нажимает кнопку «Учет добычи»
3. Открывается новая страница с учетом добычи
4. Администратор жмет кнопку «Добавление»
5. Открывается модальное окно, где администратор может ввести информацию о добыче (дата, зона добычи, вес)
6. Администратор жмет кнопку «Подтвердить»
7. Данные записываются

Альтернативный сценарий:

- Администратор жмет кнопку «Отмена», модальное окно закрывается.
- Пользователь вводит данные с ошибкой – пользователь нажимает на иконку «карандаша» и редактирует информацию о добыче.

Сценарий использования – «Добавление нового работника»

Действующее лицо: Администратор

Основной сценарий:

1. Администратор заходит на свою страницу
2. Администратор жмет кнопку «Управление персоналом»
3. Открывается страница, с кнопкой «Добавить»
4. Администратор жмет кнопку «Добавить»
5. Открывается форма, которую нужно заполнить (ФИО, телефон, почта, паспорт, должность, зона доступа, пароль)
6. Администратор заполняет данные
7. Администратор жмет кнопку «Добавить»
8. Информация о пользователе сохраняется, администратор попадает на предыдущую страницу.

Альтернативный сценарий 1:

- Администратор жмет кнопку «Отмена»
- Администратор попадает на предыдущую страницу

Альтернативный сценарий 2:

- Администратор вводит существующего пользователя
- Выводится сообщение об этом

2.3. Сценарии использования для задачи представления данных.

Сценарий использования - «Просмотр информации о добыче за период в конкретной зоне»

Действующее лицо: Администратор

Основной сценарий:

1. Администратор заходит на свою страницу
2. Администратор жмет кнопку «Учет добычи»
3. Открывается страница с учетом добычи
4. Администратор жмет кнопку «Поиск»

5. Администратор меняет поля, соответствующие периоду «с» и «по»
6. Администратор выбирает зону из выпадающего списка
7. Соответствующая информация отображается

Сценарий использования - «Просмотр нужного персонала»

Действующее лицо: Администратор

Основной сценарий:

1. Администратор заходит на свою страницу
2. Администратор жмет кнопку «Список персонала»
3. Отображается страница со списком персонала
4. Администратор жмет кнопку «Поиск»
5. Администратор выбирает из выпадающего списка по каким должностям отобрать персонал.
6. Администратор выбирает из выпадающего списка из какой зоны выбирать персонал.
7. Соответствующая информация отображается

Сценарий использования - «Просмотр подробной информации о работнике»

Действующее лицо: Администратор

Основной сценарий:

1. Администратор заходит на свою страницу
2. Администратор жмет кнопку «Список персонала»
3. Открывается страница со списком персонала
4. У нужного работника администратор жмет кнопку «Подробнее»
5. Открывается страница с подробной информацией о работнике

Сценарий использования - «Просмотр смен работника»

Действующее лицо: Администратор

Основной сценарий:

1. Сценарий «Просмотр подробной информации о работнике»

2. Администратор жмет кнопку «Смены»

3. Открывается страница со сменами работника (время смены, присутствовал или нет).

4. Администратор выбирает период, за который показывать информацию.

5. Соответствующая информация отображается

Сценарий использования – «Просмотр всех смен персонала»

Действующее лицо: Администратор

Основной сценарий:

1. Администратор заходит на свою страницу

2. Администратор жмет кнопку «Смены персонала»

3. Открывается страница со сменами персонала (ФИО, должность, зона, дата).

4. Администратор жмет кнопку «Поиск»

5. Администратор выбирает должность, зону, период смен

6. Отображается обновленная информация

Сценарий использования – «Просмотр своих смен»

Действующее лицо: Администратор/Работник/Охранник

Основной сценарий:

1. Пользователь заходит на свою страницу

2. Пользователь жмет кнопку «Мои смены»/«Смены»

3. Открывается страница со сменами (время, присутствовал или нет)

4. Пользователь выбирает период, за который надо отобразить информацию

5. Информация обновляется

2.4. Сценарии использования для задачи анализа данных.

Сценарий использования – «Проверка пропуска у работника»

Действующее лицо: Охранник

Основной сценарий:

1. Охранник заходит на свою страницу
2. Охранник жмет кнопку «Проверить пропуск»
3. Отображается модальное окно с полем для ID пропуска
4. Охранник вводит ID пропуска и жмет кнопку «Проверить»
5. Выводится информация о владельце пропуска и о том, есть ли доступ к зоне у пропуска с таким ID или нет

2.5. Сценарии использования для задачи экспорта данных.

Сценарий использования - «Массовый экспорт»

Действующее лицо: главный администратор

Основной сценарий:

1. Сценарий «Авторизация».
2. Пользователь нажимает кнопку «Данные».
3. Пользователь попадает на страницу массового импорта и экспорта.
4. Пользователь нажимает кнопку «Экспортировать данные».
5. Данные экспортируются в файл типа XML и добавляются в загрузки в браузере пользователя.

2.6. Дополнительные сценарии использования.

Сценарий использования - «Авторизация»

Действующее лицо: Работник/Охранник/Администратор

Основной сценарий:

1. Пользователь заходит в приложение, попадает на страницу входа
2. Пользователь заполняет данные для авторизации
3. Пользователь попадет на страницу администратора или страницу работника/охранника

Альтернативный сценарий:

- Пользователь вводит неверные данные

- Пользователю предлагается ввести данные заново

Сценарий использования - «Увольнение работника»

Действующее лицо: Администратор

Основной сценарий:

1. Сценарий «Просмотр подробной информации о работнике»
2. Администратор жмет кнопку «Уволить»
3. Работник удаляется

Сценарий использования – «Редактирование информации о работнике»

Действующее лицо: Администратор

Основной сценарий:

1. Сценарий «Просмотр подробной информации о работнике»
2. Администратор жмет кнопку «Редактирование»
3. Отображается страница с полями работника (ФИО, телефон, почта, паспорт, должность, зоны доступа)
4. Администратор изменяет нужны поля
5. Администратор жмет кнопку «Изменить»
6. Данные сохраняются, администратор попадает на предыдущую страницу.

Альтернативный сценарий:

- Администратор жмет кнопку «Отмена»
- Администратор попадает на предыдущую страницу

Сценарий использования – «Редактирование информации о работнике 2»

Действующее лицо: Администратор

Основной сценарий:

1. Администратор заходит на свою страницу
2. Администратор жмет кнопку «Управление персоналом»

3. Открывается новая страница, где нужно ввести почту или телефон работника, чтобы его найти

4. Администратор вводит данные, жмет кнопку «Редактировать»

5. Отображается страница с полями работника (ФИО, телефон, почта, паспорт, должность, зоны доступа)

6. Администратор изменяет нужны поля

7. Администратор жмет кнопку «Изменить»

8. Данные сохраняются, администратор попадает на предыдущую страницу.

Альтернативный сценарий:

- Работника не удалось найти
- Выводится сообщение о том, что пользователя не удалось найти

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель данных (Neo4j).

3.1.1. Графическое представление (рисунок 2).

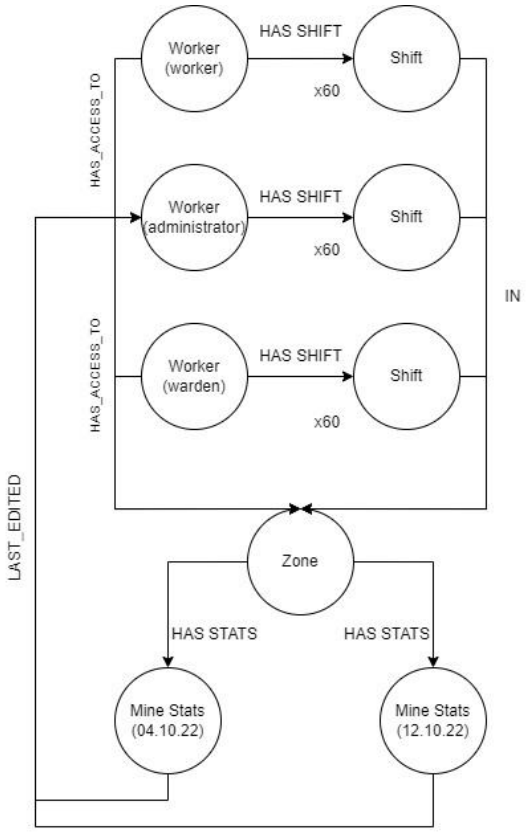

Рисунок 2 - Графическое представление модели данных Neo4j

3.1.2. Описание назначений коллекций, типов данных и сущностей

- Узел Worker - представляет сущность работника карьера
- Узел Shift - представляет сущность смены
- Узел Zone - представляет сущность зоны (карьера)
- Узел Mine Stats - представляет сущность статистики добычи у конкретной зоны

3.1.3. Оценка удельного объема информации, хранимой в модели

Узел Worker

Данный узел имеет следующие атрибуты:

- *id* - V_{id} = **4 байта**
- surname - string - $V_{surname} = 2b * N_s$, где $N_{surname}$ будет 50. Тогда $V_{surname} = 100b$
- name - string – $V_{name} = 2b * N_{name}$, где N_{name} будет 50. Тогда $V_{name} = 100b$
- patronymic - string $- V_{patronymic} = 2b * N_{patronymic}$, где $N\_patronymic$ будет 50. Тогда $V\_patronymic = 100b$
- email - string - $V_{email} = 2b * N_{email}$, где N_{email} будет 50. Тогда $V_{email} = 100b$
- phone_number - string - $V_{phonenumber} = 15b$
- passport - string - $V_{passport} = 2b * N_{passport}$, где $N_{passport}$ будет 50. Тогда $V_{passport} = 100b$
- role - string - $V_{role} = 2b * N_{role}$, где N_{role} будет 64. Тогда V_{role}=$128b$
- pass_id - $V_{passid} = 4b$
- password - string - $V_{password} = 128b$

Также у узла есть связь(и):

- HAS_ACCESS_TO с узлом Zone

$V_{hasaccessto} = 4b$,

- HAS_SHIFT с узлами смен (максимум на 30 дней назад и вперед от текущей даты)

$V_{hasshift} = 60 * 4b = 240b$

Суммарный объем памяти, занимаемый одним узлом Worker:

$V_{worker} = V_{id} + V_{surname} + V_{name} + V_{patronymic} + V_{email} +$

$V_{phonenumber} + V_{passport} + V_{role} + V_{passid} + V_{password} + V_{hasaccessto} +$

$V_{hasshift} = 1038b$

Узел Shift

Данный узел имеет следующие атрибуты:

- *id* – $\boldsymbol{V_{\text{id}}}$ **= 4 байта**
- date - DateTime - $\boldsymbol{V_{date}}$ **8 байт**
- attended - Boolean - $\boldsymbol{V_{attended}}$ **1 бит**

Также у узла есть связь(и):

- IN с зоной, в которой проходит текущая смена.

$\boldsymbol{V_{in} = 4b}$,

Суммарный объем памяти, занимаемый одним узлом Shift:

$$\boldsymbol{V_{shift} = V_{id} + V_{date} + V_{attended} + V_{in} = 12b\ 1bit}$$

Узел Zone

Данный узел имеет следующие атрибуты:

- *id* - $\mathbf{V_{id}}$ = **4 байта**
- name - string - $\boldsymbol{V_{name} = 2b * N_{name}}$, где $\boldsymbol{N_{name}}$ будет 50. Тогда $\boldsymbol{V_{name}}$=$\boldsymbol{100b}$

Также у узла есть связь(и):

- HAS_STATS с узлом Mine Stats (максимум хранится статистика за последние 365 дней)

$\boldsymbol{V_{hasstats}}$=$\boldsymbol{365 * 4b = 1460b}$,

Суммарный объем памяти, занимаемый одним узлом Mine Stats:

$$\boldsymbol{V_{zone} = V_{id} + V_{name} + V_{hasstats} = 1564b}$$

Узел Mine Stats

Данный узел имеет следующие атрибуты:

- *id* - $\boldsymbol{V_{\text{id}}}$ **4 байта**

- date - Date – $\boldsymbol{V\_date = 3b}$
- weight - float - $\boldsymbol{V_{weight} = 4b}$
- last_edit_time - DateTime - $\boldsymbol{V_{lastedittime} = 8b}$

Также у узла есть связь(и):

- LAST_EDITED с узлом Worker, который последний редактировал данный узел. $\boldsymbol{V_{lastedited} = 4b}$

Суммарный объем памяти, занимаемый одним узлом Mine Stats:

$$\boldsymbol{V_{minestats} = V_{id} + V_{date} + V_{weight} + V_{lastedittime} + V_{lastedited} = 23b}$$

3.1.4. Избыточность модели

Пусть:

- **W** - число работников;
- **Z** - число зон(128 - берем по максимуму);
- $\boldsymbol{V_i^W}$ - объем данных, занимаемый полем i узла Worker;
- $\boldsymbol{V_i^S}$ - объем данных, занимаемый полем i узла Shift;
- $\boldsymbol{V_i^Z}$ - объем данных, занимаемый полем i узла Zone;
- $\boldsymbol{V_i^M}$ - объем данных, занимаемый полем i узла Mine Stats;
- Количество доступов у каждого работника берем по максимуму;

"Чистый" объем данных:

$$\boldsymbol{V_{\text{чист}} = W * ((V_{id}^W + V_{name}^W + V_{surname}^W + V_{patronymic}^W + V_{email}^W + V_{phonenumber}^W + V_{passport}^W + V_{passid} + V_{password}^W + V_{role}^W + 128 * V_{id}^Z) + 60 * (V_{date}^S + V_{attended}^S + V_{id}^Z)) + Z * ((V_{name}^Z + V_{id}^Z) + 365 * (V_{date}^M + V_{weight}^M + V_{lastedittime}^M + V_{id}^W)) = W * ((4 + 100 + 100 + 100 + 100 + 15 + 100 + +4 + 128 + 128 + 128 * 4) + 60 * (8 + 1/8 + 4)) + 128 * ((100 + 4) + 365 * (3 + 4 + 8 + 4)) = W * 2018 + 900992b}$$

При W = 100, количество данных, занимаемой "чистыми" данными = 100 * 2018 + 900992b = **2918992**

Фактический объем данных для модели Neo4j:

$$\boldsymbol{V}_{\text{факт}} = \boldsymbol{W} * ((\boldsymbol{V}_{\boldsymbol{id}}^{\boldsymbol{W}} + \boldsymbol{V}_{\boldsymbol{name}}^{\boldsymbol{W}} + \boldsymbol{V}_{\boldsymbol{surname}}^{\boldsymbol{W}} + \boldsymbol{V}_{\boldsymbol{patronymic}}^{\boldsymbol{W}} + \boldsymbol{V}_{\boldsymbol{email}}^{\boldsymbol{W}} + \boldsymbol{V}_{\boldsymbol{phonenumber}}^{\boldsymbol{W}} + \boldsymbol{V}_{\boldsymbol{passport}}^{\boldsymbol{W}} + \boldsymbol{V}_{\boldsymbol{passid}} + \boldsymbol{V}_{\boldsymbol{password}}^{\boldsymbol{W}} + \boldsymbol{V}_{\boldsymbol{role}}^{\boldsymbol{W}} + \boldsymbol{Z} * \boldsymbol{V}_{\boldsymbol{hasaccessto}} + \mathbf{60} * \boldsymbol{V}_{\boldsymbol{hasshift}}) + \mathbf{60} * (\boldsymbol{V}_{\boldsymbol{id}} + \boldsymbol{V}_{\boldsymbol{date}} + \boldsymbol{V}_{\boldsymbol{attended}} + \boldsymbol{V}_{\boldsymbol{in}})) + \boldsymbol{Z} * ((\boldsymbol{V}_{\boldsymbol{id}}^{\boldsymbol{Z}} + \boldsymbol{V}_{\boldsymbol{name}}^{\boldsymbol{Z}} + \mathbf{365} * \boldsymbol{V}_{\boldsymbol{hasstats}}^{\boldsymbol{Z}}) + \mathbf{365} * (\boldsymbol{V}_{\boldsymbol{id}}^{\boldsymbol{M}} + \boldsymbol{V}_{\boldsymbol{date}}^{\boldsymbol{M}} + \boldsymbol{V}_{\boldsymbol{weight}}^{\boldsymbol{M}} + \boldsymbol{V}_{\boldsymbol{lasteditime}} + \boldsymbol{V}_{\boldsymbol{lastedited}})) = \boldsymbol{W} * ((\mathbf{4} + \mathbf{100} + \mathbf{100} + \mathbf{100} + \mathbf{100} + \mathbf{15} + \mathbf{100} + \mathbf{4} + \mathbf{128} + \mathbf{128} + \mathbf{128} * \mathbf{4} + \mathbf{60} * \mathbf{4}) + \mathbf{60} * (\mathbf{4} + \mathbf{8} + \mathbf{1/8} + \mathbf{4})) + \mathbf{128} * ((\mathbf{4} + \mathbf{100} + \mathbf{365} * \mathbf{4}) + \mathbf{365} * (\mathbf{4} + \mathbf{3} + \mathbf{4} + \mathbf{8} + \mathbf{4})) = \boldsymbol{W} * \mathbf{2498} + \mathbf{1274752}\boldsymbol{b}$$

При W = 100, количество данных, занимаемой Neo4j моделью = 100 * 2498 + 1274752 = **3772752**

$$\frac{\boldsymbol{V}_{\text{факт}}}{\boldsymbol{V}_{\text{чист}}} = \mathbf{1.29}$$

3.1.5. Запросы к модели, с помощью которых реализуются сценарии использования.

Ниже приведены некоторые примеры запросы к модели, все запросы находятся в приложении 3 – «Запросы Neo4j» .

1. Добавление работника

```
«create(
    worker: WORKER {
        name: "ivan",
        surname: "ivanov",
        pass_id: "12312312"
        role: "worker",
        patronymic: "ivanovich",
        email: "ivanov@mail.ru",
        phone_number: "89123456789",
        passport: "0000000000",
        password: "password"
    }
)»
```

2. Редактирование информации о работнике

```
«match(
    worker: WORKER {
        email: "ivanov@gmail.com"
    }
)
set worker.surname = "kuzubov",
    worker.name = "mihail"»
```

3. Добавление смены работника

```
«match(
    worker: WORKER {
        email:"ivan1@mail.com"
    }
)
create(
    worker
)-[:HAS_SHIFT]->(
    shift: SHIFT {
        date: date("2022-06-01"), attended: TRUE
    }
)»
```

3.2. Реляционная модель данных (SQL)

3.2.1. Графическое представление (рисунок 3).

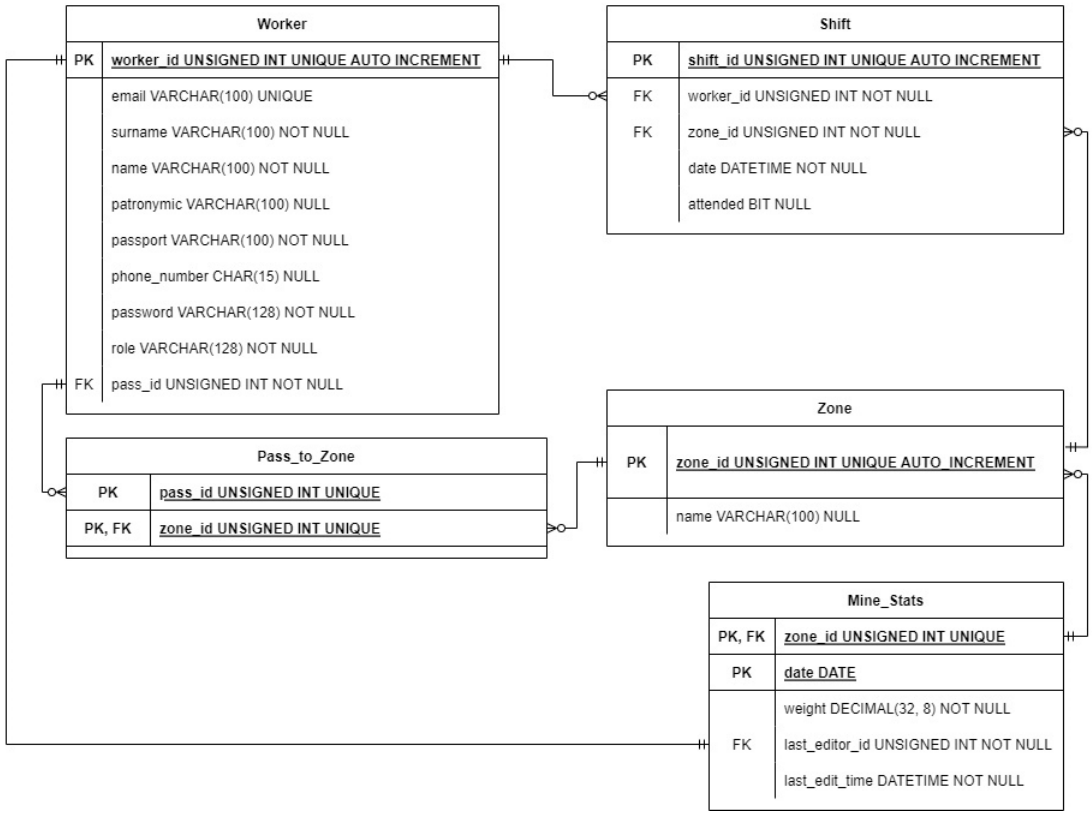

Рисунок 3 - Графическое представление модели данных SQL

3.2.2. Описание назначений коллекций, типов данных и сущностей

- Таблица Worker - таблица работников
- Таблица Pass_to_Zone - таблица, содержащая id пропуска и id зоны, в которую он ведет
- Таблица Shift - таблица смен работника
- Таблица Zone - таблица зон
- Таблица Mine_Stats - таблица со статистикой о добыче в каждой зоне

3.2.3. Оценка удельного объема информации, хранимой в модели

Таблица Worker

Данная таблица имеет следующие поля:

- worker_id - UNSIGNED INT - $\boldsymbol{V_{workerid} = 4b}$
- email - VARCHAR(100) - $\boldsymbol{V_{email} = 100b}$
- surname - VARCHAR(100) - $\boldsymbol{V_{surname} = 100b}$
- name - VARCHAR(100) - $\boldsymbol{V_{name} = 100b}$
- patronymic - VARCHAR(100) - $\boldsymbol{V_{patronymic} = 100b}$
- passport - VARCHAR(100) - $\boldsymbol{V_{passport} = 100b}$
- phone_number - CHAR(15) - $\boldsymbol{V_{phonenumber} = 15b}$
- password - VARCHAR(128) - $\boldsymbol{V_{password} = 128b}$
- role - VARCHAR(128) - $\boldsymbol{V_{role} = 128b}$
- pass_id - UNSIGNED INT - $\boldsymbol{V_{passid} = 4b}$

Суммарный объем памяти, занимаемый одной строкой в таблице Worker:

$$\boldsymbol{V_{worker} = V_{workerid} + V_{email} + V_{surname} + V_{name} + V_{patronymic} + V_{passport} + V_{phonenumber} + V_{password} + V_{role} + V_{passid} = 779b}$$

Таблица Pass_to_Zone

Данная таблица имеет следующие поля:

- pass_id - UNSIGNED INT- $\boldsymbol{V_{passid} = 4b}$
- zone_id - UNSIGNED INT- $\boldsymbol{V_{zoneid} = 4b}$

Ограничения на количество зон такие же. Тогда суммарный объем памяти, занимаемый одним пропуском в таблице Pass_to_Zone:

$$\boldsymbol{V_{passtozone} = 128 * (V_{passid} + V_{zoneid}) = 1024b}$$

Таблица Shift

Данная таблица имеет следующие поля:

- worker_id - VARCHAR(100) - $\boldsymbol{V_{workerid} = 4b}$
- shift_id - UNSIGNED INT- $\boldsymbol{V_{shiftid} = 4b}$

- zone_id - UNSIGNED INT- $\boldsymbol{V_{zoneid} = 4b}$
- date - DATETIME - $\boldsymbol{V_{date} = 8b}$
- attended - BIT - $\boldsymbol{V_{attended} = 1bit}$

Ограничения на количество смен работника такие же. Тогда суммарный объем памяти, занимаемый одним работником в таблице Shift:

$$\boldsymbol{V_{shift} = 60 * (V_{workerid} + V_{shiftid} + V_{zoneid} + V_{date} + V_{attended}) = 1208b}$$

Таблица Zone

Данная таблица имеет следующие поля:

- zone_id - UNSIGNED INT- $\boldsymbol{V_{zoneid} = 4b}$
- name - VARCHAR(100) – $\boldsymbol{V_{name} = 100b}$

Ограничения на количество зон такие же. Тогда суммарный объем памяти, занимаемый одной строкой в таблице Pass_to_Zone:

$$\boldsymbol{V_{zone} = V_{zoneid} + V_{name} = 104b}$$

Таблица Mine_Stats

Данная таблица имеет следующие поля:

- zone_id - UNSIGNED INT- $\boldsymbol{V_{zoneid} = 4b}$
- date - DATE - $\boldsymbol{V_{date} = 3b}$
- weight - DECIMAL(32, 8) - $\boldsymbol{V_{weight} = 4b}$
- last_editor_id - UNSIGNED INT - $\boldsymbol{V_{lasteditorid} = 4b}$
- last_edit_time - DATETIME - $\boldsymbol{V_{lastedittime} = 8b}$

Ограничения на количество статистик добычи такие же. Тогда суммарный объем памяти, занимаемый одной зоной в таблице Mine_Stats:

$$V_{minestats} = 365 * (V_{zoneid} + V_{date} + V_{weight} + V_{lasteditorid} + V_{lasteditortime}) = 8395b$$

3.2.4. Избыточность модели

Пусть:

- **W** - число работников;
- **Z** - число зон(128 - берем по максимуму);
- V_i^W - объем данных, занимаемый полем i таблицы Worker;
- V_i^S - объем данных, занимаемый полем i таблицы Shift;
- V_i^Z - объем данных, занимаемый полем i таблицы Zone;
- V_i^M - объем данных, занимаемый полем i таблицы Mine Stats;
- V_i^{PZ} - объем данных, занимаемый полем i таблицы Pass to Zone;
- Количество доступов у каждого работника берем по максимуму;

"Чистый" объем данных.

Чистый объем данных посчитали ранее. $V_{\text{чист}} = W * 1598 + 900992b$

При W = 100, количество данных, занимаемой "чистыми" данными ***= 100 * 2018 + 900992b = 2918992b***

Фактический объем данных для модели SQL:

$$V_{\text{факт}} = W * ((V_{workerid}^W + V_{email}^W + V_{surname}^W + V_{name}^W + V_{patronymic}^W + V_{phonenumber}^W + V_{passport}^W + V_{passid}^W + V_{password}^W + V_{role}^W) + 60 * (V_{workerid}^S + V_{shiftid}^S + V_{zoneid}^S + V_{date}^S + V_{attended}^S) + Z * (V_{passid}^{PZ} + V_{zoneid}^{PZ})) + Z * ((V_{zoneid}^Z + V_{name}^Z) + 365 * (V_{zoneid}^M + V_{date}^M + V_{weight}^M + V_{lasteditorid}^M + V_{lastedittime}^M)) = W * ((4 + 100 + 100 + 100 + 100 + 15 + 100 + 4 + 128 + 128) + 60 * (4 + 4 + 4 + 8 + 1/8) + 128 * (4 + 4)) + 128 * ((4 + 100) + 365 * (4 + 3 + 4 + 4 + 8)) = W * 3010b + Z * 1274752b$$

При W = 100, количество данных, занимаемой SQL моделью = 100 * 3010 + 1274752 = **4284752**

$$\frac{V_{\text{факт}}}{V_{\text{чист}}} = \mathbf{1.46}$$

3.2.5. Запросы к модели, с помощью которых реализуются сценарии использования.

Ниже приведены некоторые примеры запросы к модели, все запросы находятся в приложении 4 – «Запросы SQL» .

1. Получение смен работника за период

```
«SELECT SHIFT.date, SHIFT.attended FROM WORKER
INNER JOIN SHIFT ON WORKER.worker_id == SHIFT.worker_id WHERE WORKER.worker_id ==1
AND
            SHIFT.date BETWEEN DATETIME("2021-01-02") AND
                               DATETIME("2022-02-02")»
```

2. Получение всех администраторов и охранников

```
«SELECT * FROM WORKER
WHERE WORKER.role == 'admin' OR
      WORKER.role == 'warden'»
```

3. Получение статистики о добыче за определенный период в зоне

```
«SELECT * FROM MINE_STATS
WHERE MINE_STATS.date < "2023-01-01" AND
      MINE_STATS.date > "2022-01-01" AND
      MINE_STATS.zone_id == 1»
```

3.3. Сравнение моделей

- Отношение фактических данных к "чистым" дали следующие результаты: для SQL - 1.46, для NoSQL - 1.29. Таким образом, получили, что NoSQL модель более экономична по памяти, нежели SQL.
- Neo4j имеет преимущество в виде графовой структуры. Данная структура позволяет избежать создание дополнительных таблиц для связей путем создания связей между узлами данных.

- Количество запросов, необходимых для выполнения сценариев использования в SQL модели больше.

Вывод: для данной задачи определенно лучше использовать NoSQL модель. Она занимает меньше места, и с ней проще работать.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание.

Разработанное приложение является информационной системой для учета добычи и сотрудников песчаного карьера.

Backend часть приложения написана на Kotlin и является посредником для извлечения данных из БД, а также обработчиком данных перед передачей их на Frontend для отображения.

Frontend часть приложения является Web-приложением, которое делает API запросы на Backend и отображает полученные данные.

4.2. Схема экранов приложения. (рисунок 4)

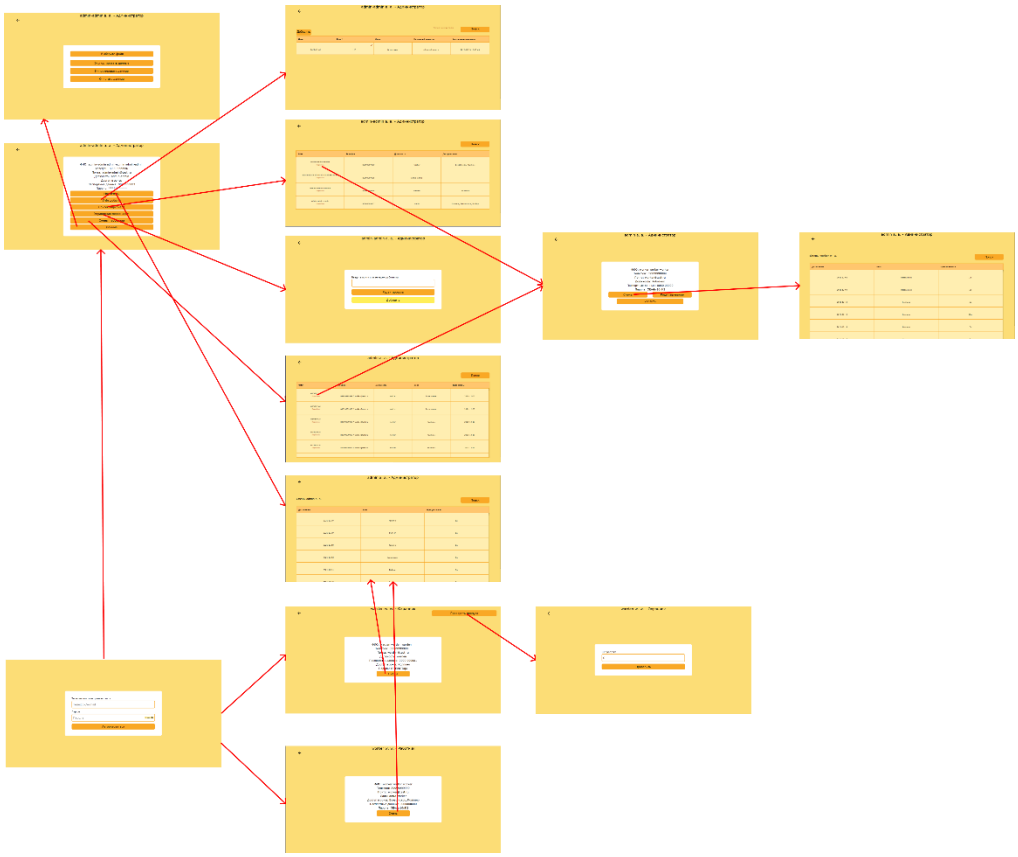

Рисунок 4 – схема экранов приложения.

Ссылка на схему доступна в разделе «Список использованных источников».

4.3. Использованные технологии.

БД: Neo4j

Backend: Kotlin, Spring

Frontend: JavaScript, CSS, VueJS

4.4. Ссылки на приложение.

1. Ссылка на GitHub – https://github.com/moevm/nosql2h22-sand-mine

5. ВЫВОДЫ

Достигнутые результаты.

В ходе работы была разработана информационная система для сотрудников песчаного карьера. Разработанное приложение позволяет легко добавлять, удалять и изменять работников, добычи, смены, а также допуски на объекты. Помимо этого, информационная система поддерживает импорт и экспорт данных в формате XML.

Недостатки и пути улучшения полученного решения.

На данный момент при запросе получения данных для заполнения таблицы Backend достает из БД и присылает все данные – для оптимизации можно реализовать пагинацию (присылать данные только тогда, когда они нужны. Например, при скролле таблицы)

Еще одной проблемой является то, что информация о текущем авторизованном пользователе хранится на стороне Frontend’а, что не безопасно. Хорошим решением было бы реализовать сокетное взаимодействие между Frontend’ом и Backend’ом, чтобы данная информация хранилась на сервере.

Будущее развитие решения.

Планируется добавить разделение ролей обычных работников, разработка мобильного приложения, выполняющего те же функции, что и web.

6. ПРИЛОЖЕНИЯ

6.1. Документация по сборке и развертыванию приложения

1. Скачать проект из репозитория (указан в ссылках на приложение)
2. В корневой папке проекта выполнить команду docker-compose up –build

6.2. Инструкция для пользователя.

Чтобы запустить приложение необходимо перейти по ссылке http://localhost:8080.

6.3. Запросы Neo4j

1. Добавление работника

```
create(
   worker: WORKER {
      name: "ivan",
      surname: "ivanov",
      pass_id: "12312312"
      role: "worker",
      patronymic: "ivanovich",
      email: "ivanov@mail.ru",
      phone_number: "89123456789",
      passport: "0000000000",
      password: "password"
   }
)
```

2. Редактирование информации о работнике

```
match(
   worker: WORKER {
      email: "ivanov@gmail.com"
   }
)
set worker.surname = "kuzubov",
   worker.name = "mihail"
```

3. Добавление смены работника

```
match(
   worker: WORKER {
      email:"ivan1@mail.com"
```

```
    }
)
create(
    worker
)-[:HAS_SHIFT]->(
    shift: SHIFT {
        date: date("2022-06-01"), attended: TRUE
    }
)
```

4. Получение смен работника

```
match (
    worker: WORKER {
        email: "ivanov@gmail.com"
    }
)-[:HAS_SHIFT]->(
    shift: SHIFT
) return shift
```

5. Получение смен работника за период

```
match(
    worker: WORKER {
        email: "ivanov@gmail.com"
    }
)-[:HAS_SHIFT]->(
    shift: SHIFT
)
where shift.date > date("2022-01-01") and
      shift.date < date("2023-01-01")
return shift
```

6. Получение всех смен

```
match(
    shift: SHIFT
) return shift
```

7. Добавление зоны к смене

```
match(
    worker: WORKER {
        email: "ivanov@mail.ru2"
    }
```

```
)-[:HAS_SHIFT]->(
    shift: SHIFT
), (
    zone: ZONE {
        name: "ZONE1"
    }
)
create(
    shift
)-[:IN]->(
    zone
)
```

8. Получение работника по email

```
match(
    n: WORKER
)
where n.email = "ivanov@mail.ru"
return n
```

9. Получение всех работников

```
match(
    worker: WORKER
)
return worker
```

10. Получение всех администраторов и охранников

```
match(
    worker: WORKER
)
where worker.role = "admin" or
      worker.role = "warden"
return worker
```

11. Удаление работника из базы данных по email

```
match(
    worker: WORKER {
        email: "ivanov@mail.ru"
    }
)-[r1:HAS_SHIFT]->(
    shift: SHIFT
```

```
)-[r2]->(n)
delete r1, shift

match(
    worker: WORKER {
        email: "ivanov@mail.ru"
    }
)-[r1:HAS_ACCESS_TO]->(
    zone: ZONE
)
delete r1, worker
```

Сначала удаляется связь работника со сменой, смена. Потом удаляется связь работника с зоной и сам работник.

12. Добавление доступа работника в зону

```
match(
    worker: WORKER {
        email: "alex3@mail.ru"
    }
), (zone: ZONE {
    name: "ZONE1"
    }
)
create(
    worker
)-[r1:HAS_ACCESS_TO]->(
    zone
)
```

13. Удаление доступа работника в зону

```
match(
    worker: WORKER {
        email: "alex3@mail.ru"
    }
)-[r1:HAS_ACCESS_TO]->(
    zone: ZONE {
        name: "ZONE1"
    }
)
delete r1
```

14. Добавление зоны

```
create(
    zone: ZONE {
        name: "ZONE1"
    }
)
```

15. Удаление зоны

```
match(
    worker: WORKER
)-[r1:HAS_ACCESS_TO]->(
    zone: ZONE {
        name: "ZONE1"
    }
)-[r2:HAS_STATS]->(
    mine_stats: MINE_STATS
)
delete r1, r2, mine_stats, zone
```

- При удалении зоны, удаляются связи и статистика

16. Добавление статистики добычи

```
match(
    worker: WORKER {
        email: "ivan1@mail.com"
    }
)
create(
    mine_stats: MINE_STATS {
        date: date("2022-01-01"),
        last_edit_time: datetime("2022-01-01T11:00:00"),
        weight: 1.234
    }
)<-[:LAST_EDITED]-(
    worker
)
```

17. Добавление статистики добычи к зоне

```
match(
    zone: ZONE {
        name: "ZONE1"
    }
```

```
), (
    worker: WORKER {
        email: "ivan1@mail.com"
    }
)
create(
    worker
)-[:LAST_EDITED]->(
    mine_stats: MINE_STATS {
        date: date("2022-01-01"),
        last_edit_time: datetime("2022-01-01T11:00:00"),
        weight: 1.234
    }
), (
    zone
)-[:HAS_STATS]->(
    mine_stats
)
```

18. Редактирование статистики о добыче в зоне

```
match(
    zone: ZONE {
        name: "ZONE1"
    }
)-[has_stats:HAS_STATS]->(
    mine_stats: MINE_STATS
)-[last_edited:LAST_EDITED]->(
    worker: WORKER
)
set mine_stats.last_edit_time = datetime("2022-11-11T11:10:11"),
    mine_stats.weight = 3.123
delete last_edited
create(
    worker
)-[:LAST_EDITED]->(
    mine_stats
)
```

19. Получение всей статистики о добыче

```
match(
    mine_stats: MINE_STATS
)
return mine_stats
```

20. Получение статистики о добыче в определенной зоне

```
match(
   zone: ZONE {
      name: "ZONE1"
   }
)-[:HAS_STATS]->(
   mine_stats
)
return mine_stats
```

21. Получение статистики о добыче за определенный период в зоне

```
match(
   zone: ZONE {
      name: "ZONE1"
   }
)-[:HAS_STATS]->(
   mine_stats
)
where mine_stats.date > date("2022-11-10") and
    mine_stats.date < date("2023-01-01")
return mine_stats
```

22. Получение работника по пропуску

```
match(
   worker: WORKER {
      pass_id: "12312312"
   }
)
return worker
```

23. Получение зоны на который проходит смена работника

```
match(
   worker: WORKER {
      mail: "ivanov@mail.ru"
   }
)-[r1:HAS_SHIFT]->(
   shift: SHIFT {
      date: date("2022-10-01")
   }
)-[r2:IN]->(
   zone: ZONE
```

```
)
return zone
```

24. Получить зоны, в которые есть пропуск у работника

```
match(
    worker: WORKER {
        pass_id: "12312312"
    }
)-[:HAS_ACCESS_TO]->(
    zone: ZONE
)
return zone
```

6.4. Запросы SQL

1. Создание таблицы WORKER

```
CREATE TABLE WORKER (
    worker_id INTEGER PRIMARY KEY AUTOINCREMENT,
    email VARCHAR(100) NOT NULL UNIQUE,
    name VARCHAR(100) NOT NULL,
    surname VARCHAR(100) NOT NULL,
    surname VARCHAR(100) NOT NULL,
    patronymic VARCHAR(100),
    passport VARCHAR(100) NOT NULL UNIQUE,
    phone_number CHAR(15) UNIQUE,
    password VARCHAR(64) NOT NULL,
    role VARCHAR(64) NOT NULL,
    pass_id UNSIGNED INT NOT NULL UNIQUE,
    FOREIGN KEY (pass_id) REFERENCES PASS_TO_ZONE (pass_id)
);
```

2. Создание таблицы ZONE

```
CREATE TABLE ZONE (
    zone_id INTEGER PRIMARY KEY AUTOINCREMENT,
    name VARCHAR(100)
);
```

3. Создание таблицы PASS_TO_ZONE

```
CREATE TABLE PASS_TO_ZONE (
    pass_id INTEGER,
```

```
    zone_id INTEGER,
    PRIMARY KEY (pass_id, zone_id),
    FOREIGN KEY (zone_id) REFERENCES zone(zone_id)
);
```

4. Создание таблицы Shift

```
CREATE TABLE Shift (
    worker_id INTEGER,
    shift_id INTEGER PRIMARY KEY AUTOINCREMENT,
    zone_id UNSIGNED INT NOT NULL,
    date DATETIME NOT NULL,
    attended BIT,
    FOREIGN key (worker_id) REFERENCES worker (worker_id)
);
```

5. Создание таблицы Mine_Stats

```
CREATE TABLE MINE_STATS (
    zone_id UNSIGNED INT,
    date DATE,
    weight DECIMAL(32,8) NOT NULL,
    last_editor_id UNSIGNED INT NOT NULL,
    last_edit_time DATETIME NOT NULL,
    FOREIGN KEY (zone_id) REFERENCES ZONE (zone_id),
    FOREIGN KEY (last_editor_id) REFERENCES WORKER (worker_id),
    PRIMARY KEY (zone_id,date)
);
```

6. Добавление работника

```
INSERT INTO WORKER (
    email,
    name,
    patronymic,
    passport,
    phone_number,
    password,
    role,
    pass_id,
    surname
) VALUES (
  'ivanov@mail.ru',
```

```
  'ivan',
  'ivanovich',
  '000000000',
  '89141395555',
  'password',
  'worker',
  '123123123',
  'ivanov'
);
```

7. Редактирование информации о работнике

```
UPDATE WORKER
SET role = 'admin',
    name = 'piter'
WHERE email = 'ivanov1@mail.ru'
```

8. Получение работника по email

```
SELECT * FROM WORKER where email = 'ivanov@mail.ru'
```

9. Добавление смены работника

```
INSERT INTO SHIFT (
    worker_id,
    zone_id,
    date,
    attended
) VALUES (
  1,
  1,
  "2022-01-01",
  TRUE
);
```

10. Получение смен работника

```
SELECT SHIFT.date, SHIFT.attended FROM WORKER
INNER JOIN SHIFT ON WORKER.worker_id == SHIFT.worker_id WHERE WORKER.worker_id == 1
```

11. Получение смен работника за период

```
SELECT SHIFT.date, SHIFT.attended FROM WORKER
INNER JOIN SHIFT ON WORKER.worker_id == SHIFT.worker_id WHERE WORKER.worker_id ==1 AND
```

```
        SHIFT.date BETWEEN DATETIME("2021-01-02") AND
                DATETIME("2022-02-02")
```

12. Получение всех смен

```
SELECT * FROM SHIFT
```

13. Добавление зоны к смене у работника

```
UPDATE SHIFT
SET zone_id = 2
WHERE SHIFT.shift_id in (
   SELECT shift_id FROM SHIFT
   INNER JOIN WORKER ON (SHIFT.worker_id==WORKER.worker_id)
   WHERE WORKER.worker_id == 1 AND
       SHIFT.date="2022-01-02"
)
```

14. Получение всех работников

```
SELECT * FROM WORKER
```

15. Получение всех администраторов и охранников

```
SELECT * FROM WORKER
WHERE WORKER.role == 'admin' OR
    WORKER.role == 'warden'
```

16. Удаление работника из базы данных по email

```
DELETE FROM WORKER
WHERE WORKER.email == 'ivanov@mail.ru'
```

17. Добавление зоны

```
INSERT INTO ZONE(
   name
) VALUES (
   "zone3"
)
```

18. Удаление зоны

```
DELETE FROM ZONE
WHERE ZONE.name == "zone3"
```

19. Добавление статистики добычи

```
INSERT INTO MINE_STATS (
    zone_id,
    date,
    weight,
    last_editor_id,
    last_edit_time
) VALUES(
    1,
    DATE("2022-01-01"),
    32.11,
    1,
    DATETIME("2022-01-01 11:00:00")
)
```

20. Редактирование статистики о добыче в зоне

```
UPDATE MINE_STATS
SET weight = 11.11,
    last_editor_id = 1,
    last_edit_time = "2022-03-03 11:00:12"
WHERE zone_id IN (
    SELECT zone_id FROM ZONE
    WHERE ZONE.name == "zone1"
)
```

21. Получение всей статистики о добыче

```
SELECT * FROM MINE_STATS
```

22. Получение статистики о добыче за определенный период в зоне

```
SELECT * FROM MINE_STATS
WHERE MINE_STATS.date < "2023-01-01" AND
      MINE_STATS.date > "2022-01-01" AND
      MINE_STATS.zone_id == 1
```

23. Получение работника по пропуску

```
SELECT * FROM WORKER
WHERE WORKER.pass_id == 123123123
```

24. Получение зоны на который проходит смена работника

```
SELECT ZONE.zone_id, ZONE.name FROM WORKER
INNER JOIN SHIFT ON WORKER.worker_id == SHIFT.worker_id
INNER JOIN ZONE ON SHIFT.zone_id == ZONE.zone_id
WHERE WORKER.worker_id == 1
```

25. Добавление зоны к пропуску работника

```
INSERT INTO PASS_TO_ZONE (
    pass_id,
    zone_id
) VALUES (
  '123123123',
  '1'
)
```

26. Получение зоны, в которые есть пропуск у работника

```
SELECT ZONE.zone_id, ZONE.name FROM WORKER
INNER JOIN PASS_TO_ZONE ON WORKER.pass_id == PASS_TO_ZONE.pass_id
INNER JOIN ZONE ON PASS_TO_ZONE.zone_id == ZONE.zone_id
WHERE WORKER.pass_id == 123123123
```

7. ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА

1. Документация Neo4j: https://neo4j.com/docs/

2. Документация Spring: https://docs.spring.io/spring-framework/docs/current/reference/html/

3. Документация Spring-Boot: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

4. Документация Kotlin: https://kotlinlang.org/docs

5. Документация Vue.js: https://vuejs.org/guide

6. Документация Node.js: https://nodejs.org/en/docs/

7.Figma – макет UI : https://www.figma.com/file/xmcBCymtGpEbRmavSCRpX0/Untitled?node-id=71%3A105