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

КУРСОВАЯ РАБОТА

по дисциплине «Введение в нереляционные базы данных»

Тема: Сервис портфолио для программистов

Студент гр. 9303 _____ Дюков В.А.

Студент гр. 9303 _____ Лойконен М.Р.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2022

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студенты

Дюков В.А.

Лойконен М.Р.

Группа 9303

Тема работы: сервис портфолио для программистов

Исходные данные:

Необходимо реализовать клиентскую и серверную части для сервиса портфолио для программистов с использованием СУБД MongoDb.

Содержание пояснительной записки:

«Содержание», «Введение», «Качественные требования к решению», «Сценарии использования», «Модель данных», «Разработанное приложение», «Выводы», «Список использованных источников», «Приложения»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 08.09.2022

Дата сдачи реферата: 26.12.2022

Дата защиты реферата: 26.12.2022

Студент _____ Дюков В.А.

Студент _____ Лойконен М.Р.

Преподаватель _____ Заславский М.М.

АННОТАЦИЯ

В данной работе была разработана система портфолио для программистов. При разработке использовались следующие технологии: основной язык программирования – TypeScript, клиентская часть – React, SCSS, серверная часть – NodeJS, Express, используемая СУБД – MongoDB. Для развертывания системы использовались Docker и Docker-compose. В результате работы было разработано приложение, позволяющее пользователям авторизовываться и просматривать профили других пользователей с помощью системы поиска с фильтрацией результатов.

SUMMARY

In this work, a portfolio system for programmers was developed. The following technologies were used in the development: the main programming language is TypeScript, the client part is React, SCSS, the server part is NodeJS, Express, the DBMS used is MongoDB. Docker and Docker-compose were used to deploy the system. As a result of the work, an application was developed that allows users to log in and view the profiles of other users using a search engine with filtered results.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время IT-специалистам очень важно при устройстве на работу иметь портфолио, в котором будет содержаться информация о навыках и проектах, в которых работал специалист. Поэтому было бы очень удобно, если специалист будет хранить всю эту информацию в едином месте.

Цель работы – разработка сервиса портфолио для программистов.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать клиент-серверное приложение, использующее для хранения данных СУБД MongoDB.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет UI

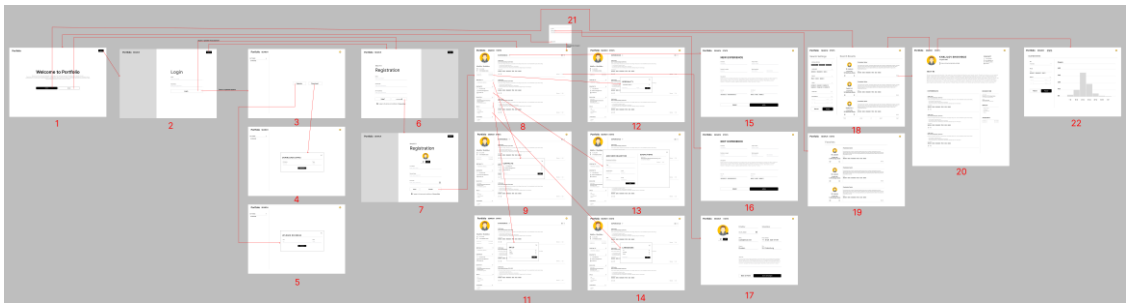

Рисунок 1 - Макет пользовательского интерфейса

2.2. Сценарии использования по задачам

I. Сценарий использования - “Авторизация”

Действующее лицо: Пользователь сервиса.

Начальные условия: Пользователь зашел на начальную страницу [1].

Основной сценарий:

1. Пользователь нажимает кнопку ‘Sign in’
2. Пользователь вводит данные, с которыми он регистрировался.
3. Пользователь нажимает кнопку ‘Login’.
4. Перевод пользователя на страницу [8].

Альтернативные сценарии:

Пункт 2.

- Уведомление о неправильном вводе данных, возврат к пункту 1.
- Пользователь авторизовался как администратор, перевод на страницу [3].

II. Сценарий использования - “Регистрация”

Действующее лицо: Пользователь сервиса.

Начальные условия: Пользователь зашел на начальную страницу [1].

Основной сценарий:

1. Пользователь нажимает кнопку ‘Sign Up’, происходит переход на страницу [6].

2. Пользователь вводит данные для регистрации.

3. Пользователь нажимает кнопку *‘Далее’*, происходит переход пользователя на страницу [7].

4. Пользователь вводит данные профиля (заполнение полей, кнопки с выбором).

5. Пользователь нажимает кнопку ‘Sign Up’, происходит переход пользователя на страницу [8], регистрация успешна.

Альтернативные сценарии:

Пункт 2.

- Уведомление о уже занятом email, возврат к пункту 1.
- Уведомление о неправильном вводе данных, возврат к пункту 1.

Пункт 5.

- Уведомление о неправильном вводе данных, возврат к пункту 1.

III. Сценарий использования - “Редактирование страницы профиля”

Действующее лицо: Пользователь сервиса.

Начальные условия: Пользователь находится на странице [8].

Основной сценарий:

1. Пользователь нажимает одну из кнопок ‘+’ в сегментах ‘CONTACS’, ‘EDUCATION’, ‘SKILLS’, ‘LANGUAGES’, ‘SPECIALITY’.

2. Открывается модульное окно (МО), соответственно [9], [11], [12], [13] или [14].

3. Пользователь вводит необходимые данные в МО.

4. Пользователь нажимает кнопку ‘ADD’ в МО.

5. Модульное окно закрывается, новые данные отображаются на странице.

Альтернативный сценарий:

Пункт 2.

1. Пользователь нажимает ‘X’ рядом с элементом списка в МО, элемент списка удаляется.

2. Пользователь нажимает ‘X’ сверху слева в МО - кнопка выхода.

3. Модульное окно закрывается, данные о пользователе изменены.

IV. Сценарий использования - “Редактирование основной информации о пользователе”

Действующее лицо: Пользователь сервиса.

Основной сценарий:

1. Пользователь нажимает иконку в правом верхнем углу экрана

2. В появившемся окне нажимает пункт ‘Edit profile’, происходит переход на страницу [17].

3. Пользователь изменяет данные профиля (заполнение полей, кнопки с выбором).

4. Пользователь нажимает кнопку ‘Save Changes’.

5. Перевод пользователя на окно [8], сохранение измененных данных.

Альтернативный сценарий:

Пункт 4:

1. Пользователь нажимает кнопку ‘Back to Home’.

2. Перевод пользователя на окно [8], измененные данные профиля не применены.

V. Сценарий использования - “Добавление опыта работы”

Действующее лицо: Пользователь сервиса.

Начальные условия: Пользователь находится на странице [8].

Основной сценарий:

1. Пользователь нажимает кнопку ‘+’ рядом с надписью Experience, происходит переход на страницу [15].

2. Пользователь заполняет данные об опыте работы (заполнение полей).

3. Пользователь нажимает кнопку ‘ADD’.

4. Перевод пользователя на окно [8], добавление нового опыта работы в профиле.

Альтернативный сценарий:

Пункт 3:

1. Пользователь нажимает кнопку ‘BACK’.

2. Перевод пользователя на окно [8], новый опыт работы не создан.

VI. Сценарий использования - “Редактирование опыта работы”

Действующее лицо: Пользователь сервиса.

Начальные условия: Пользователь находится на странице [8].

Основной сценарий:

1. Пользователь нажимает кнопку ‘edit’ у соответствующей записи в списке, происходит переход на страницу [16].

2. Пользователь редактирует данные об опыте работы (заполнение полей).

3. Пользователь нажимает кнопку ‘SAVE’.

4. Перевод пользователя на окно [8], изменение в существующем опыте работы сохранены.

Альтернативный сценарий:

Пункт 1:

- Пользователь нажимает кнопку ‘remove’, запись удаляется

Пункт 3:

1. Пользователь нажимает кнопку ‘BACK’.

2. Перевод пользователя на окно [8], изменения в опыте работы не применены.

VII. Сценарий использования - “Поиск”

Действующее лицо: Пользователь сервиса.

Начальные условия: Пользователь находится на странице [18].

Основной сценарий:

1. Пользователь устанавливает характеристики для поиска соответствующих им пользователей (выбор из списка, заполнение полей).

2. Пользователь исключает часть выбранных характеристик нажатием ‘X’ около них.

3. Пользователь нажимает кнопку ‘Apply’

4. Производится автоматическая фильтрация и вывод подходящих результатов в ‘Search Results’.

Альтернативный сценарий (только авторизованный пользователь):

Пункт 3.

- Пользователь нажимает кнопку ‘Reset’, данные поиска сбрасываются

Пункт 4.

- Пользователь нажимает кнопку добавления в закладки у одного из пользователей, пользователь добавляется в избранные. Возврат к пункту 3.

1. Пользователь нажимает кнопку ‘Portfolio’.
2. Перевод пользователя на страницу [8].

VIII. Сценарий использования - “Просмотр избранных”

Действующее лицо: Пользователь сервиса.

Начальные условия: Пользователь находится на странице [19].

Основной сценарий:

1. Пользователь просматривает избранных пользователей.

2. Пользователь нажимает кнопку ‘more’ у одного из пользователей.

3. Перевод пользователя на страницу [20] (просмотр профиля другого пользователя).

Альтернативный сценарий:

Пункт 1.

- Пользователь нажимает кнопку ‘закладки’ у одного из пользователей, выбранный пользователь удаляется из избранных. Возврат к пункту 1.

IX. Сценарий использования - “Просмотр профиля другого пользователя”

Действующее лицо: Пользователь сервиса.

Начальные условия: Пользователь находится на странице [18].

Основной сценарий:

1. Пользователь нажимает кнопку ‘more’ у заинтересовавшего его пользователя, происходит переход на страницу [20].
2. Пользователь просматривает профиль пользователя.

X. Сценарий использования - “Массовый импорт/экспорт данных”

Действующее лицо: Администратор сервиса.

Начальные условия: Администратор находится на странице [3].

Основной сценарий:

1. Администратор нажимает кнопку ‘Upload’.
2. Открывается модульное окно (МО).
3. (В МО) Администратор вводит путь к файлу с данными о пользователях и его тип.
4. (В МО) Администратор нажимает кнопку ‘UPLOAD’.
5. Модульное окно закрывается, данные из файла загружены в БД.

Альтернативный сценарий:

1. Администратор нажимает кнопку ‘Download’.
2. Открывается модульное окно (МО).
3. (В МО) Администратор вводит название файла с данными о пользователях и его тип.
4. (В МО) Администратор нажимает кнопку ‘DOWNLOAD’.
5. Модульное окно закрывается, данные из БД скачиваются в виде файла.

XI. Сценарий использования - “Просмотр статистики”

Действующее лицо: Пользователь сервиса.

Начальные условия: Пользователь находится на странице [22].

Основной сценарий:

1. Пользователь выбирает интересующие его параметры.
2. Пользователь нажимает кнопку ‘Apply’.
3. Выводится статистика в виде диаграммы.

Альтернативный сценарий:

Пункт 2.

- Пользователь нажимает кнопку ‘Reset’, данные для отбора статистики сбрасываются.

Для данного решения преобладать будут операции записи, так как основной функционал заключается в добавлении различной информации пользователем (добавление информации о проектах, изученных языков программирования, навыков, образования).

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель данных

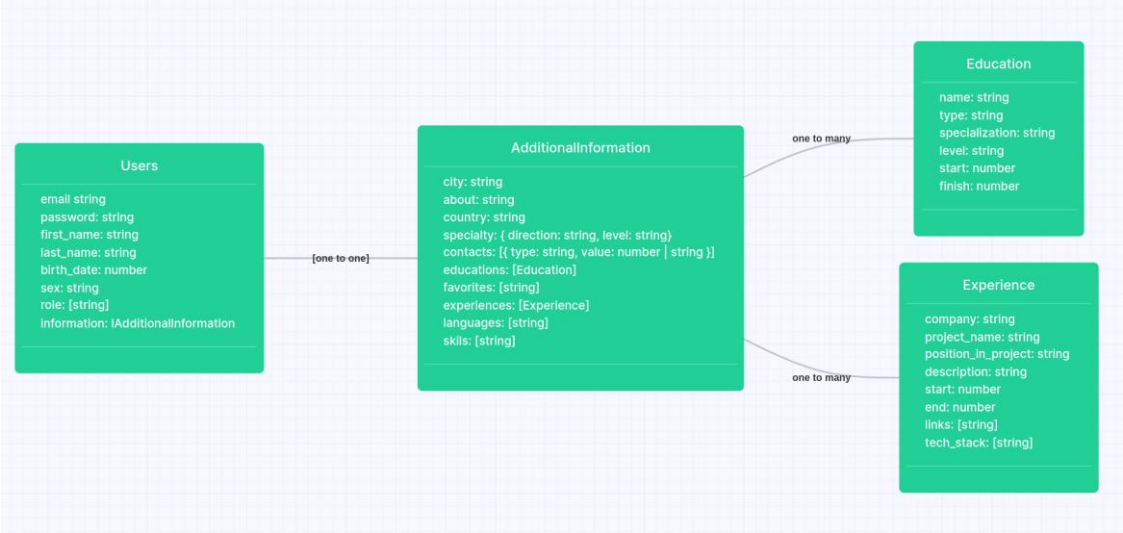

Рисунок 2 - Нереляционная модель данных

Коллекция Users:

```
User {
    email: string                       //   почта пользователя
    password: string                    //   хэш пароля
    first_name: string                  //   имя
    last_name: string                   //   фамилия
    birth_date: number                  //   дата рождения
    sex: string                         //   пол
    role: string[]                      //   роль пользователя
    information: IAdditionalInformation //   ссылка на дополнительную
информацию
}
```

Фактический размер сущности:

$$\mathbf{12}\text{Б} + \mathbf{61}\text{Б} + \mathbf{31}\text{Б} + \mathbf{31}\text{Б} + \mathbf{31}\text{Б} + \mathbf{8}\text{Б} + \mathbf{11}\text{Б} + \boldsymbol{N_R} * \mathbf{31}\text{Б} + \mathbf{12}\text{Б}$$

$$= \mathbf{197} + \mathbf{31} * \boldsymbol{N_R}, \boldsymbol{N_R} - \text{количество ролей}$$

Чистый размер сущности:

$$\mathbf{61}\text{Б} + \mathbf{31}\text{Б} + \mathbf{31}\text{Б} + \mathbf{31}\text{Б} + \mathbf{8}\text{Б} + \mathbf{11}\text{Б} + \boldsymbol{N_R} * \mathbf{31}\text{Б} = \mathbf{173} + \mathbf{31} * \boldsymbol{N_R}$$

Коллекция AdditionalInformation:

```
AdditionalInformation {                 //
    city: string                        //   город проживания
    about: string                       //   информация "о себе"
    country: string                     //   страна проживания
    specialties: Specialty[]            //   специализации пользователя
```

```
    contacts: Contacts[]                    //   список контактов
    educations: Education[]                 //   список образований
    favorites: string[]                     //   список избранных пользователей
    experiences: IExperience[]              //   список проектов (опыт работы)
    languages: string[]                     //   список языков
    skills: string[]                        //   список навыков
}

// sub-document
Specialty {
    direction: string                       //   направление специальности
    level: string                           //   уровень специальности
}

// sub-document
Contacts {
    type: string                            //   тип контакта (телефон, почта,
...)
    value: string                           //   контакт пользователя
}
```

Фактический размер сущности:

$$\mathbf{12}\text{Б} + \mathbf{61}\text{Б} + \mathbf{256}\text{Б} + \mathbf{61}\text{Б} + \boldsymbol{N_S} * (\mathbf{61}\text{Б} + \mathbf{30}\text{Б}) + \boldsymbol{N_C} * (\mathbf{31}\text{Б} + \mathbf{120}\text{Б}) +$$

$$+\boldsymbol{N_E} * \mathbf{12}\text{Б} + \boldsymbol{N_F} * \mathbf{12}\text{Б} + \boldsymbol{N_{Ex}} * \mathbf{12}\text{Б} + \boldsymbol{N_{Lang}} * \mathbf{21}\text{Б} + \boldsymbol{N_{Sk}} * \mathbf{31}\text{Б} =$$

$$= 390 + 91 * N_S + 151 * N_C + 12 * (N_E + N_F + N_{Ex}) + N_{Lang} * 21\text{Б} + N_{Sk} * 31\text{Б},$$

N_S – количество специализаций,

N_C – количество контактов,

N_E – количество образований,

N_F – количество избранных,

N_{Ex} – количество проектов,

N_{Lang} – количество языков,

N_{Sk} – количество навыков

Чистый размер сущности:

$$\mathbf{61}\text{Б} + \mathbf{256}\text{Б} + \mathbf{61}\text{Б} + \boldsymbol{N_S} * (\mathbf{61}\text{Б} + \mathbf{30}\text{Б}) + \boldsymbol{N_C} * (\mathbf{31}\text{Б} + \mathbf{120}\text{Б}) +$$

$$= \boldsymbol{N_{Lang}} * \mathbf{21}\text{Б} + \boldsymbol{N_{Sk}} * \mathbf{31}\text{Б}$$

$$= \mathbf{378} + \mathbf{91} * \boldsymbol{N_S} + \mathbf{151} * \boldsymbol{N_C} + \boldsymbol{N_{Lang}} * \mathbf{21}\text{Б} + \boldsymbol{N_{Sk}} * \mathbf{31}\text{Б}$$

Коллекция Experience:

```
Experience {                                //
    company: string                         //   название компании, в которой
```

```
                                        //   работал пользователь (если это
не
                                        //   личный проект)
    project_name: string                //   название проекта
    position_in_project: string         //   должность пользователя в
проекте
    description: string                 //   описание проекта
    start: number                       //   дата начала работы над
проектом
    end: number                         //   дата конца работы над проектом
    links: string[]                     //   ссылки на проект
    tech_stack: string[]                //   технологический стек проекта
}
```

Фактический размер сущности:

$$\boldsymbol{12Б + 256Б + 61Б + 256Б + 8Б + 8Б + N_L * 256Б + N_{TS} * 30Б}$$
$$\boldsymbol{= 601Б + N_L * 256Б + N_{TS} * 30Б},$$

$\boldsymbol{N_L}$ – количество ссылок,

$\boldsymbol{N_{TS}}$ – количество технологий

Чистый размер сущности:

$$\boldsymbol{256Б + 61Б + 256Б + 8Б + 8Б + N_L * 256Б + N_{TS} * 30Б}$$
$$\boldsymbol{= 589Б + N_L * 256Б + N_{TS} * 30Б}$$

Коллекция Education:

```
Education {
    name: string                        //   название учебного заведения
    type: string                        //   тип учебного заведения
    specialization: string              //   специальность
    level: string                       //   уровень
    start: number                       //   дата начало обучения
    finish: number                      //   дата окончания обучения
}
```

Фактический размер сущности:

$$\boldsymbol{12Б + 256Б + 61Б + 121Б + 31Б + 8Б + 8Б = 497Б}$$

Чистый размер сущности:

$$\boldsymbol{256Б + 61Б + 121Б + 31Б + 8Б + 8Б = 485Б}$$

Оценка удельного объема информации, хранимой в модели

U - число пользователей

$\boldsymbol{AI}$ - число дополнительной информации

$\boldsymbol{Exp}$ - число проектов

$\boldsymbol{Edu}$ - число образований

Зависимость параметров от числа пользователей:

$$\boldsymbol{AI = U, Exp = 5U, Edu = U}$$

Для расчета будем использовать средние значения переменных:

$$\boldsymbol{N_R = 1}$$

$$\boldsymbol{N_S = 1}$$

$$\boldsymbol{N_C = 5}$$

$$\boldsymbol{N_E = 1}$$

$$\boldsymbol{N_F = 0}$$

$$\boldsymbol{N_{Lang} = 5}$$

$$\boldsymbol{N_{Sk} = 15}$$

$$\boldsymbol{N_{Ex} = 5}$$

$$\boldsymbol{N_L = 1}$$

$$\boldsymbol{N_{TS} = 8}$$

Чистый объем

$$S_{Clean} = U * S_{User} + AI * S_{AdditionalInformation} + Exp * S_{Experience} + Edu * S_{Education} = U * 204 + 5U * 1794 + U * 1085 + U * 485 = 10744U$$

$$\boldsymbol{S_{User} = 173 + 31 * N_R = 204Б}$$

$$\boldsymbol{S_{AdditionalInformation} = 378 + 91 * N_S + 151 * N_C + N_{Lang} * 21Б + N_{Sk} * 31Б = 1794Б}$$

$$\boldsymbol{S_{Experience} = 589Б + N_L * 256Б + N_{TS} * 30Б = 1085Б}$$

$$\boldsymbol{S_{Education} = 485Б}$$

Фактический объем

$$S_{Fact} = U * S_{User} + AI * S_{AdditionalInformation} + Exp * S_{Experience} + Edu * S_{Education} = U * 228 + 5U * 1878 + U * 1097 + U * 497 = 11212U$$

$$S_{User} = 197 + 31 * N_R = 228Б$$

$$S_{AdditionalInformation} = 390 + 91 * N_S + 151 * N_C + 12 * (N_E + N_F + N_{Ex}) + N_{Lang} * 21Б + N_{Sk} * 31Б = 1878Б$$

$$S_{Experience} = 601Б + N_L * 256Б + N_{TS} * 30Б = 1097Б$$

$$S_{Education} = 497Б$$

Избыточность модели

$$\frac{S_{Fact}}{S_{Clean}} = \frac{11212U}{10744U} = 1.04$$

Направление роста модели при увеличении количества объектов каждой сущности

Рост модели линейный.

Запросы к модели, с помощью которых реализуются сценарии использования

1. Поиск людей, у которых в списке языков программирования указан ‘C++’:

```
await UsersModel.aggregate([
        {
          $lookup: {
            from: "additionalinformations",
            localField: "information",
            foreignField: "_id",
            as: "information",
          },
        },
        {
          $match: { "information.languages": { $in: ["C++"] } }
        },
      ])
```

Ответ:

```
[
  {
    "_id": "6366af384567ca6adf27b838",
    "email": "test@email.com",
```

```
"password": "123456789",
"first_name": "иван",
"last_name": "иванов",
"birth_date": 21546464,
"sex": "MALE",
"role": [
  "USER"
],
"information": [
  {
    "_id": "6366b3964567ca6adf27b841",
    "country": "Russia",
    "city": "St.Petersburg",
    "specialties": [
      {
        "direction": "Backend",
        "level": "senior"
      }
    ],
    "contacts": [
      {
        "type": "PHONE",
        "value": "012 345 67 89"
      },
      {
        "type": "TELEGRAM",
        "value": "012 345 67 89"
      }
    ],
    "about": "bla bla-bla",
    "experiences": [
      "6366b6454567ca6adf27b84b",
      "6366b6674567ca6adf27b84c"
    ],
    "education": [
      "6366b45d4567ca6adf27b848"
    ],
    "languages": [
      "JavaScript",
      "TypeScript",
      "Python",
      "C++",
      "Java"
    ],
    "skills": [
      "Git",
      "express",
      "mongodb",
      "MySql",
      "Nestjs",
      "CI/CD",
      "docker",
      "kubernetes"
    ]
  }
]
```

```
  }
]
```

Количество запросов: 1

Задействовано коллекций: 2

2. Просмотр статистики - количество человек, обладающих навыком ‘Git’:

```
await UsersModel.aggregate([
        {
          $lookup: {
            from: "additionalinformations",
            localField: "information",
            foreignField: "_id",
            as: "information",
          },
        },
        {
          $match: {
            "information.skills": { $in: ["Git"] },
          },
        },
        {
          $count: "count",
        },
      ])
```

Ответ:

```
[
  {
    "count": 2
  }
]
```

Количество запросов: 1

Задействовано коллекций: 2

3. Добавление нового опыта работы

```
const user = await UsersModel.findOne({
  email: "test@email.com",
}).exec();

await AdditionalInformationModel.findById(user.information);

const experience = new ExperiencesModel({
  company: "mail.ru",
  project_name: "аля супер проект",
  position_in_project: "junior frontend developer",
  start: new Date("01.01.2020").getTime(),
  end: new Date("01.01.2021").getTime(),
  description: "...",
```

```
  links: [],
  tech_stack: ["React", "TypeScript", "Mongodb", "docker"],
});

await AdditionalInformationModel.findByIdAndUpdate(user.information, {
  $push: { experiences: experience._id },
}).exec();
```

Ответ:

```
{
  "company": "mail.ru",
  "project_name": "аля супер проект",
  "position_in_project": "junior frontend developer",
  "description": "...",
  "start": 1577826000000,
  "end": 1609448400000,
  "links": [],
  "tech_stack": [
    "React",
    "TypeScript",
    "Mongodb",
    "docker"
  ],
  "_id": "6366d4c7a54b9069d81fc42d",
  "__v": 0
}
```

Количество запросов: 4

Задействовано коллекций: 2

Примеры хранения записей для одного пользователя

Коллекция User:

Рисунок 3 - Пример хранения User

Коллекция AdditionalInformation:

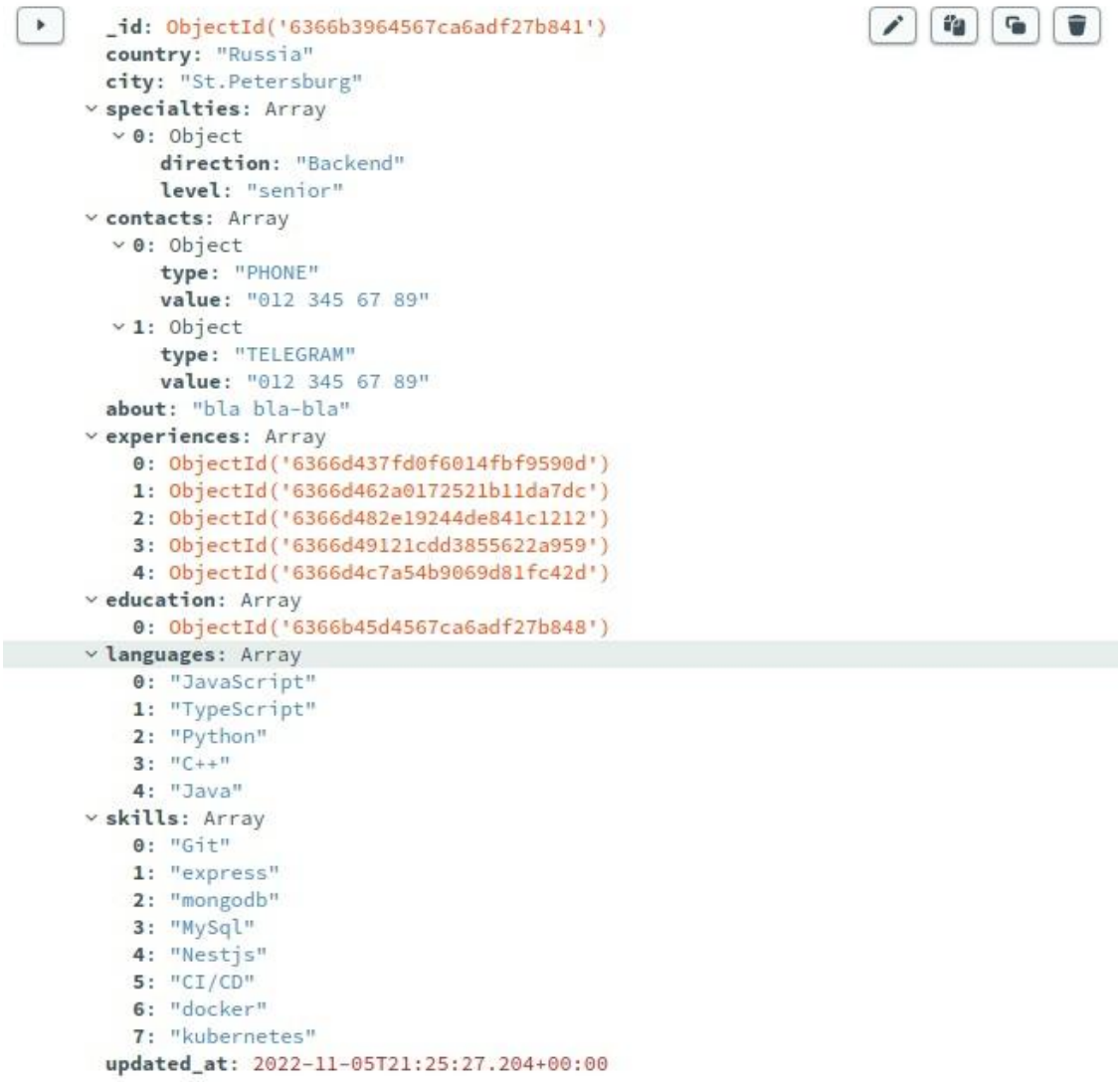

Рисунок 4 - Пример хранения AdditionalInformation

Коллекция Education:

Рисунок 5 - Пример хранения Education

Коллекция Experience:

```
_id: ObjectId('6366d482e19244de841c1212')
company: "mail.ru"
project_name: "аля супер проект"
position_in_project: "junior frontend developer"
description: "..."
start: 1577826000000
end: 1609448400000
links: Array
tech_stack: Array
  0: "React"
  1: "TypeScript"
  2: "Mongodb"
  3: "docker"
created_at: 2022-11-05T21:24:18.718+00:00
updated_at: 2022-11-05T21:24:18.718+00:00
__v: 0
```


Рисунок 6 - Пример хранения Experience

3.2. Реляционная модель данных

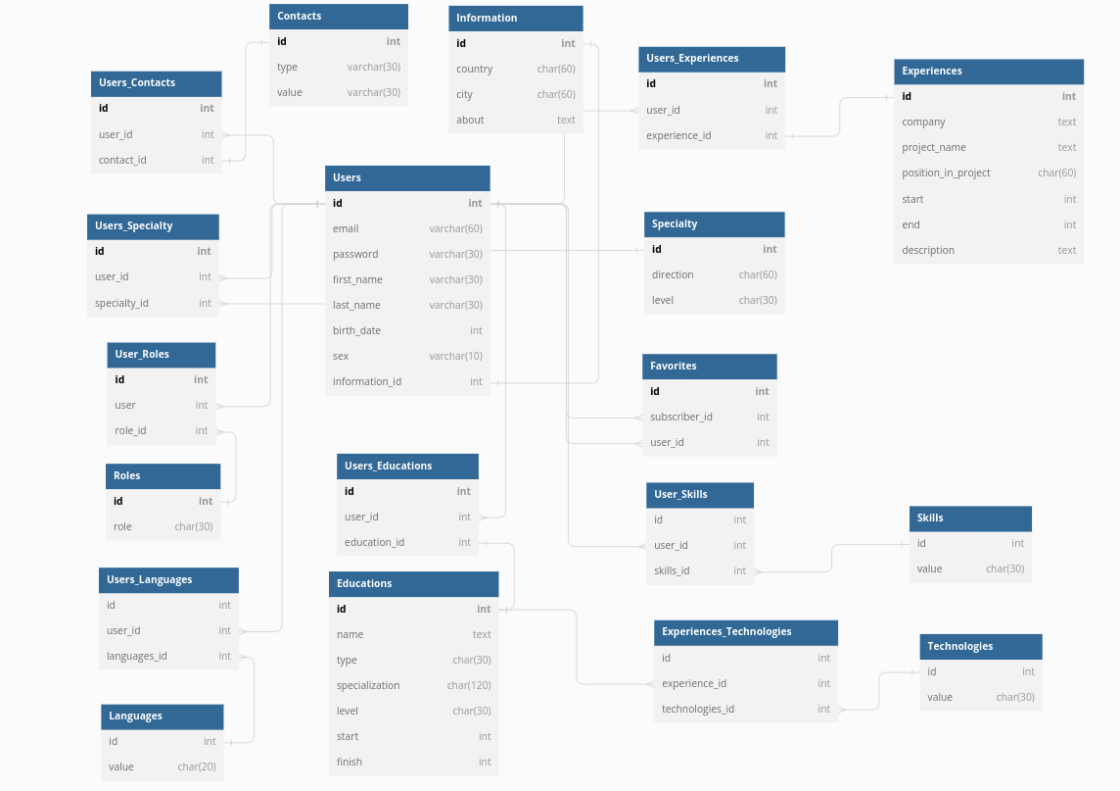

Рисунок 7 - Реляционная модель данных

Users:

```
Table Users  {
  id int          /*  */
  email varchar(60)
  password varchar(30)
  first_name varchar(30)
  last_name varchar(30)
  birth_date int
  sex varchar(10)
  information_id int
}
```

Фактический размер:

4 + (2 + 60) + (2 + 30) + (2 + 30) + (2 + 30) + 4 + (2 + 10) + 4 = 182Б

Чистый размер:

(2 + 60) + (2 + 30) + (2 + 30) + (2 + 30) + 4 + (2 + 10) = 174Б

Users_Contacts

```
Table Users_Contacts {
  id int
  user_id int
  contact_id int
}
```

Фактический размер:

4 + 4 + 4 = 12Б

Contacts

```
Table Contacts {
  id int
  type varchar(30)
  value varchar(30)
}
```

Фактический размер:

4 + 2 + 30 + 2 + 30 = 68Б

Чистый размер:

2 + 30 + 2 + 30 = 64Б

Information

```
Table Information
{
  id int
  country char(60)
  city char(60)
  about text
}
```

Фактический размер:

4 + 60 + 60 + (255 + 4) = 383Б

Чистый размер:

60 + 60 + (255 + 4) = 379Б

Experiences

```
Table Experiences
{
  id int
  company text
```

```
  project_name text
  position_in_project char(60)
  start int
  end int
  description text
}
```

Фактический размер:

4 + (255 + 4)*3 + 60 + 4 + 4 = 849Б

Чистый размер:

845Б

Users_Experiences

```
Table Users_Experiences
{
  id int
  user_id int
  experience_id int
}
```

Фактический размер:

4 + 4 + 4 = 12Б

Specialty

```
Table Specialty
{
  id int
  direction char(60)
  level char(30)
}
```

Фактический размер:

30 + 60 + 4 = 94Б

Чистый размер:

90Б

Users_Specialty

```
Table Users_Specialty
{
  id int
  user_id int
  specialty_id int
}
```

Фактический размер:

4 + 4 + 4 = 12Б

Educations

```
Table Educations
{
  id int
  name text
  type char(30)
  specialization char(120)
  level char(30)
  start int
  finish int
}
```

Фактический размер:

4 + (4 + 255) + 30 + 120 + 30 + 4 + 4 = 451Б

Чистый размер:

448Б

Users_Educations

```
Table Users_Educations
{
  id int
  user_id int
  education_id int
}
```

Фактический размер:

4 + 4 + 4 = 12Б

Favorites

```
Table Favorites
{
  id int
  subscriber_id int
  user_id int
}
```

Фактический размер:

4 + 4 + 4 = 12Б

Roles

```
Table Roles
{
  id int
  role char(30)
}
```

Фактический размер:

4 + 30 = 34Б

Чистый размер:

30Б

User_Roles

```
Table User_Roles
{
  id int
  user int
  role_id int
}
```

Фактический размер:

4 + 4 + 4 = 12Б

Skills

```
Table Skills {
  id int
  value char(30)
}
```

Фактический размер:

4 + 30 = 34Б

Чистый размер:

30Б

User_Skills

```
Table User_Skills {
  id int
  user_id int
  skills_id int
}
```

Фактический размер:

4 + 4 + 4 = 12Б

Languages

```
Table Languages {
  id int
  value char(20)
}
```

Фактический размер:

4 + 20 = 24Б

Чистый размер:

20Б

User_Languages

```
Table User_Languages {
  id int
  user_id int
  language_id int
}
```

Фактический размер:

4 + 4 + 4 = 12Б

Experiences_Technologies

```
Table Experiences_Technologies {
  id int
  experience_id int
  technologies_id int
}
```

Фактический размер:

4 + 4 + 4 = 12Б

Technologies

```
Table Technologies {
  id int
  value char(30)
}
```

Фактический размер:

4 + 30 = 34Б

Чистый размер:

30Б

Оценка удельного объема информации, хранимой в модели

U - число пользователей

$\boldsymbol{AI}$ - число дополнительной информации

$\boldsymbol{Exp}$ - число проектов пользователя

$\boldsymbol{Edu}$ - число образований пользователя

$\boldsymbol{C}$ - число контактов

$\boldsymbol{S}$ - число специальностей пользователя

$\boldsymbol{F}$ - число избранных пользователей

$\boldsymbol{UR}$ - число ролей пользователя

$\boldsymbol{ULang}$ - число языков пользователя

$\boldsymbol{USk}$ - число навыков пользователя

$\boldsymbol{ETS}$ - число технологий в проекте

Зависимость параметров от числа пользователей:

$$\boldsymbol{AI = U, Exp = 5U, Edu = U, C = 5U, S = U, F = 0U, UR = U, ULang = 5U, USk = 15U, ETS = 8 * 5U = 40U}$$

Следующие параметры не зависят от числа пользователей:

$\boldsymbol{R = 2}$ - количество ролей

$\boldsymbol{Lang = 256}$ - количество языков

$\boldsymbol{TS = 1256}$ - количество технологий

$\boldsymbol{Sk = 1000}$ - количество навыков

Чистый объем

$$S_{Clean} = U * S_{User} + AI * S_{Information} + Exp * S_{Experiences} + Edu * S_{Education} + C * S_{Contacts} + S * S_{Specialty} + F * S_{Favorites} + R * S_{Roles} + +UR * S_{Users_Roles} + Lang * S_{Languages} + Sk * S_{Skills} + TS * S_{Technologies} = 5636U + 72860$$

Фактический объем

$$S_{Fact} = U * S_{User} + AI * S_{Information} + Exp * \left(S_{Experiences} + S_{Users_Experiences}\right) + Edu * \left(S_{Education} + S_{Users_Education}\right) + C * (S_{Contacts} + +S_{Users_Contacts}) + S * \left(S_{Specialty} + S_{Users_Specialty}\right) + F * S_{Favorites} + R * S_{Roles} + UR * S_{Users_Roles} + Lang * S_{Languages} + ULang * S_{User_Languages} +$$

$$\boldsymbol{+Sk * S_{Skills} + USk * S_{User_Skills} + ETS * S_{Experiences_Technologies} + TS * S_{Technologies} = 6571U + 82916}$$

Избыточность модели

$$\boldsymbol{\frac{S_{Fact}}{S_{Clean}} = \frac{6571U + 82916}{5636U + 72860}}$$

Направление роста модели при увеличении количества объектов каждой сущности

Рост модели линейный

Запросы к модели, с помощью которых реализуются сценарии использования

1. Поиск людей, у которых в списке языков программирования указан ‘C++’

```
SELECT
    *
FROM
    users
INNER JOIN
    users_information
    ON users.information = users_information.information_id
    INNER JOIN
        users_languages
        ON users_languages.user_id = user.id
        INNER JOIN
            languages
            ON  users_languages.languages_id = languages.id
WHERE languages.value like "C++";
```

Количество запросов: 1

Задействовано таблиц: 4

2. Просмотр статистики - количество человек, обладающих навыком ‘Git’

```
SELECT
    count(*)
FROM
    users
INNER JOIN
    users_skills
    ON users_skills.user_id = user.id
    INNER JOIN
        skills
        ON  users_skills.skills_id = skills.id
WHERE skills.value like "Git";
```

Количество запросов: 1

Задействовано таблиц: 3

3. Добавление нового опыта работы

```
DECLARE @react_id INT;
DECLARE @typescript_id INT;
```

```
DECLARE @mongodb_id INT;
DECLARE @docker_id INT;
DECLARE @experience_id INT;
DECLARE @user_id INT;

SELECT @user_id = (SELECT id FROM Users WHERE mail LIKE "test@email.com")
SELECT @react_id = (SELECT id FROM Technologies WHERE Technologies.value LIKE
"React");
SELECT @typescript_id = (SELECT id FROM Technologies WHERE Technologies.value
LIKE "TypeScript");
SELECT @mongodb_id = (SELECT id FROM Technologies WHERE Technologies.value
LIKE "Mongodb");
SELECT @docker_id = (SELECT id FROM Technologies WHERE Technologies.value
LIKE "docker");
INSERT INTO Experiences VALUES (
  "mail.ru",
  "аля супер проект",
  "junior frontend developer",
  892747298427,
  891823789793,
  "..."
);

SELECT @experience_id = (SELECT LAST_INSERT_ID() FROM Experiences);

INSERT INTO Experiences_Technologies VALUES (
  @experience_id,
  @react_id
);

INSERT INTO Experiences_Technologies VALUES (
  @experience_id,
  @typescript_id
);

INSERT INTO Experiences_Technologies VALUES (
  @experience_id,
  @mongodb_id
);

INSERT INTO Experiences_Technologies VALUES (
  @experience_id,
  @docker_id
);

INSERT INTO Users_Experiences VALUES (
  @user_id,
  @experience_id
);
```

Количество запросов: 12

Задействовано таблиц: 4

3.3. Сравнение моделей

Пусть число пользователей U = 1000.

Реляционная модель:

- Чистый объём: $\mathbf{5636} * \mathbf{1000} + \mathbf{72860} = \mathbf{5}.7\boldsymbol{ГБ}$
- Фактический объём: $6571 * 1000 + 82916 = 6.7\boldsymbol{ГБ}$
- Избыточность: $\mathbf{1}.\mathbf{17}$

Нереляционная модель:

- Чистый объём: $\mathbf{10744} * 1000 = 10.7\boldsymbol{ГБ}$
- Фактический объём: $11212 * 1000 = 11.2\boldsymbol{ГБ}$
- Избыточность: $\mathbf{1}.\mathbf{04}$

Сравнение показало, что NoSQL модель занимает примерно в 2 раза больше памяти, нежели SQL модель, это происходит из-за дублирования данных. Но при этом избыточность меньше. При запросах в реляционной моделе использовалось большее число операций и задействованных таблиц, что усложняет работу с данной моделью. Исходя из всего этого, можно сделать вывод, что для удобства работы лучше выбрать NoSQL модель.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

В системе предусмотрены 2 роли – пользователь (USER) и администратор (ADMIN). После регистрации человек получает роль USER.

Пользователь может осуществлять поиск других людей при помощи фильтрации по определенным тегам, а также просматривать их профили, чтобы увидеть больше информации.

4.2. Используемые технологии

Для серверной части используются следующие технологии:

- Язык программирования TypeScript
- NodeJS
- Express
- СУБД MongoDB

Для клиентской части используются следующие технологии:

- Язык программирования TypeScript
- React
- SCSS

Для сборки и развертывания приложения используются Docker и Docker-compose.

В приложении А представлена документация по сборке и развертывания приложения.

В приложении Б представлены снимки экранов приложения.

4.3. Ссылки на Приложение

Ссылка на репозиторий: https://github.com/moevm/nosql2h22-it-profile

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы был разработан прототип приложения сервиса портфолио для программистов. Приложение позволяет пользователям зарегистрироваться и просмотреть профили других пользователей, используя поиск с фильтрацией.

В дальнейшем планируется расширить функционал приложения, добавив отображения статистики, возможность добавлять и редактировать пользовательские данные через веб-интерфейс.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация MongoDB: https://www.mongodb.com/docs/

ПРИЛОЖЕНИЕ А
ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

1. Скачать приложение из репозитория: https://github.com/moevm/nosql2h22-it-profile
2. Выполнить в командной строке команду:
   docker-compose -f ./docker-composes/prod.yaml up --build -d
3. Перейти по адресу http://localhost:8080

ПРИЛОЖЕНИЕ Б
СНИМКИ ЭКРАНОВ ПРИЛОЖЕНИЯ

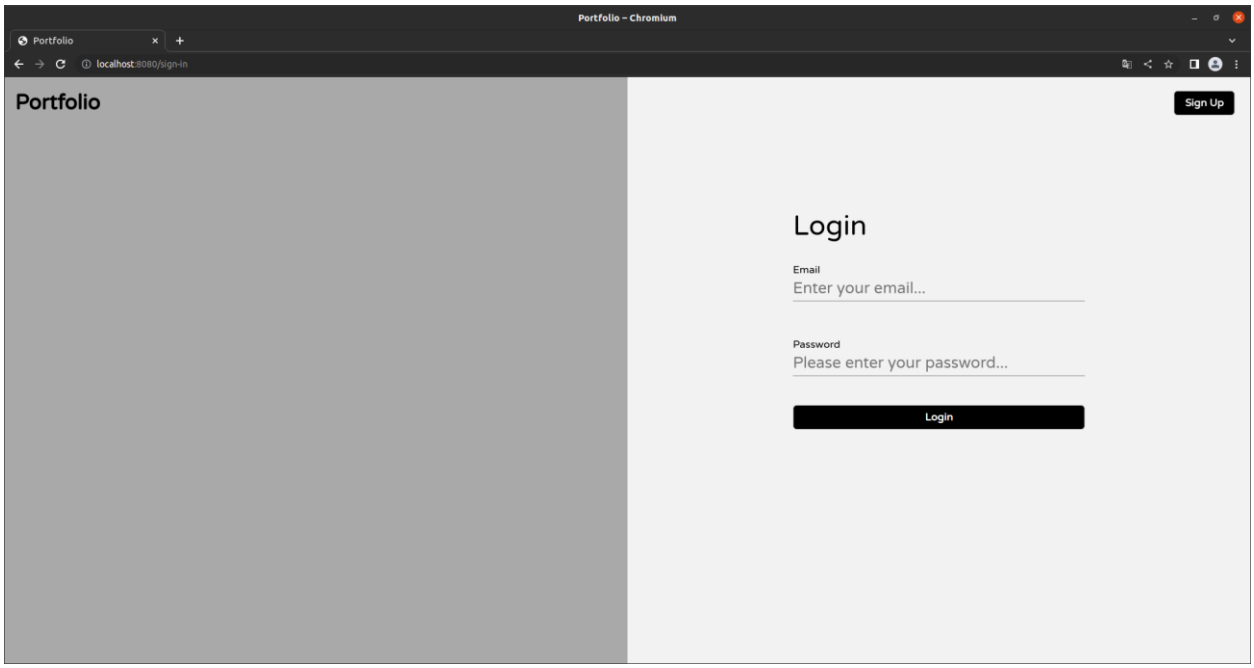

Рисунок 8 - Страница авторизации

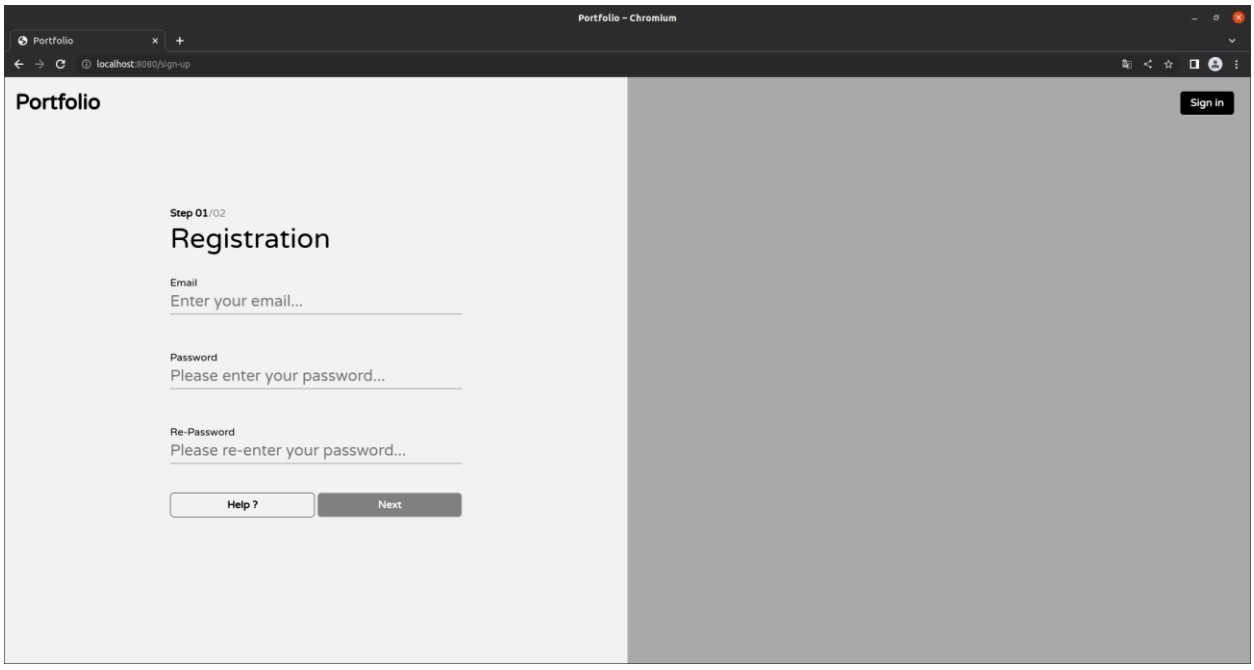

Рисунок 9 - Страница регистрации (1)

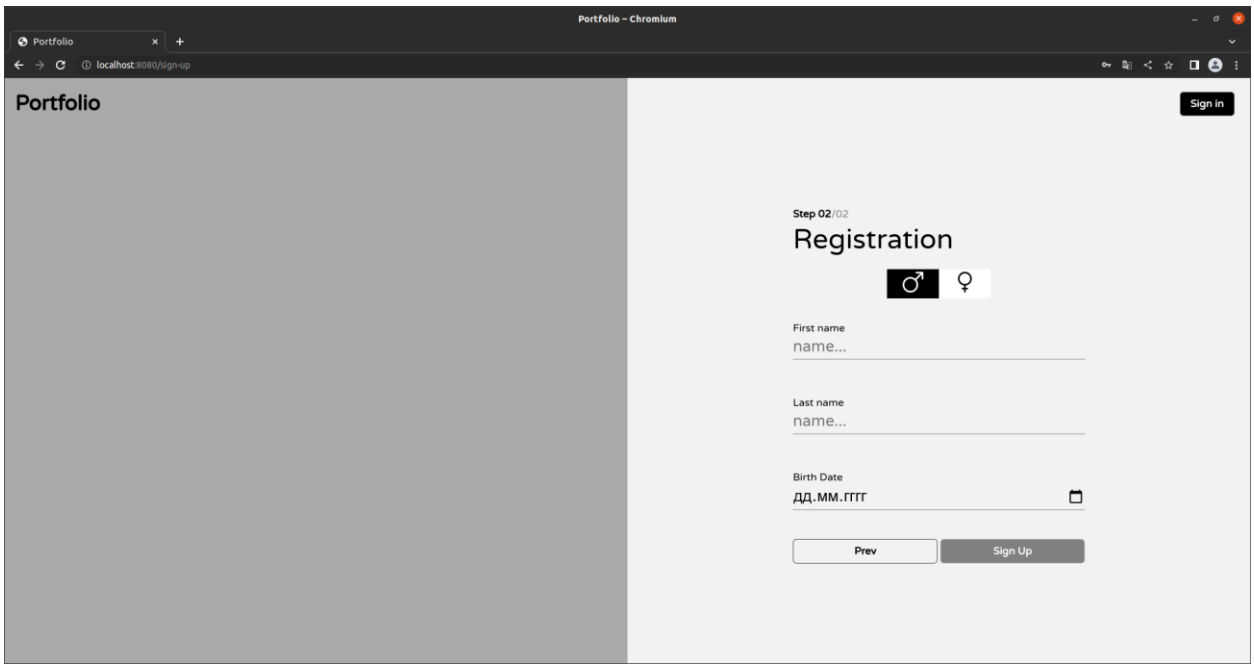

Рисунок 10 - Страница регистрации (2)

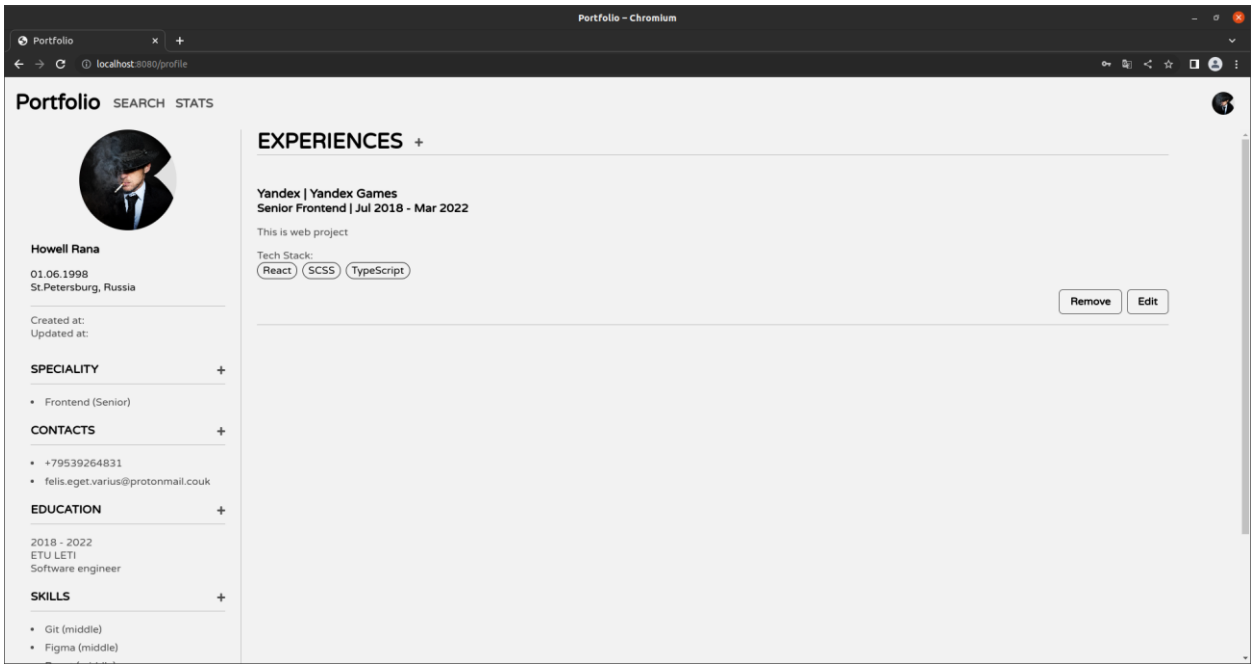

Рисунок 11 - Страница профиля

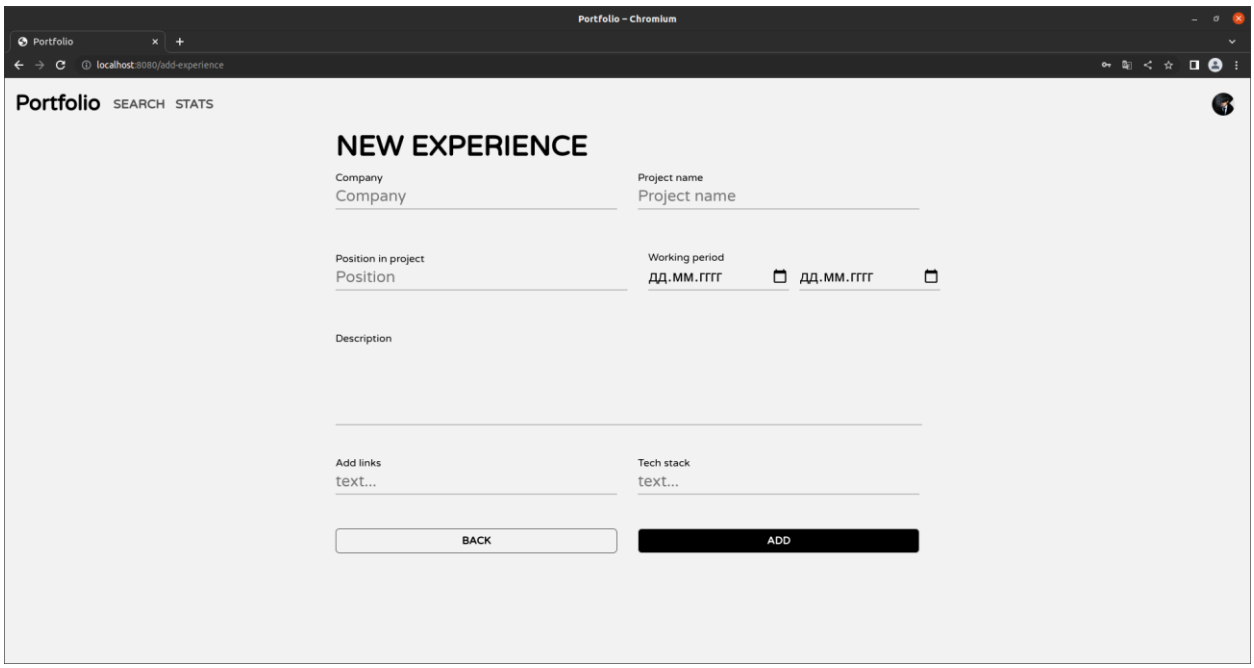

Рисунок 12 - Страница добавления нового проекта в портфолио

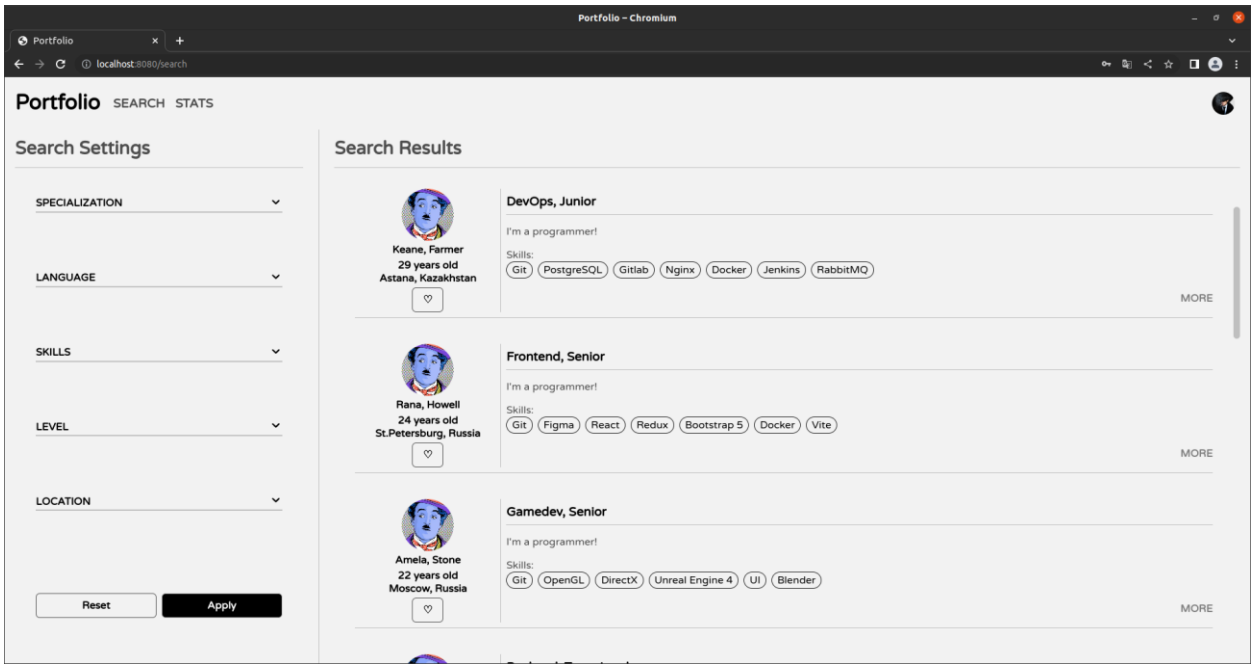

Рисунок 13 - Страница поиска

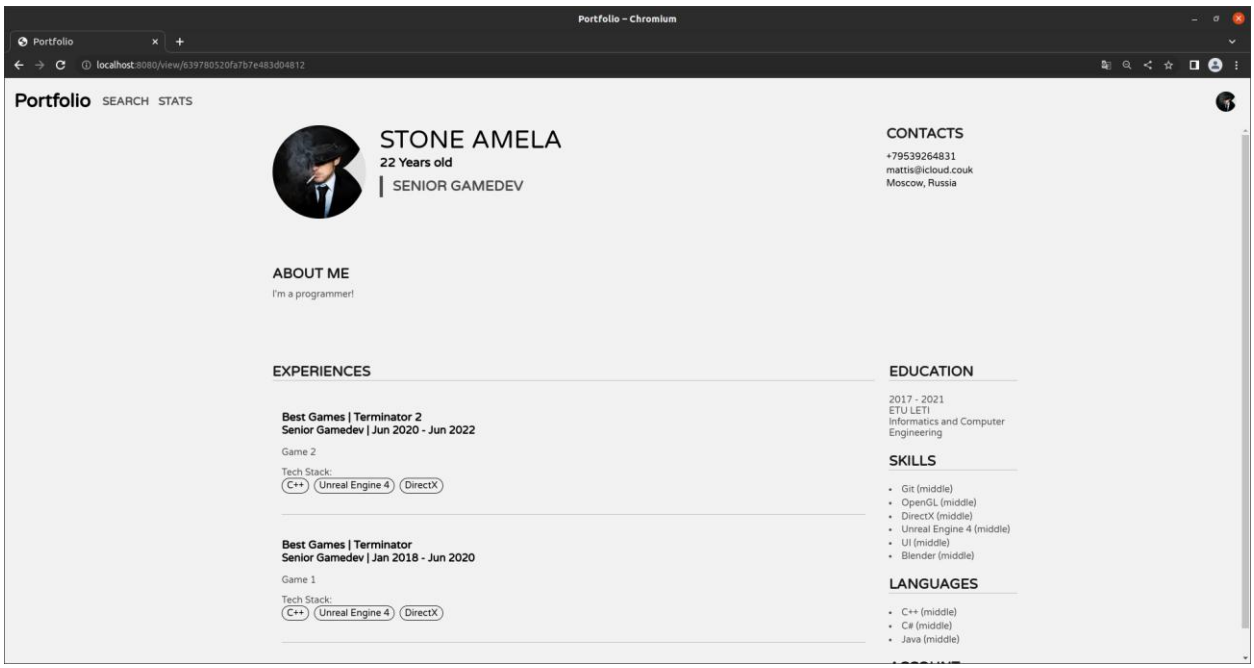

Рисунок 14 - Страница просмотра другого профиля

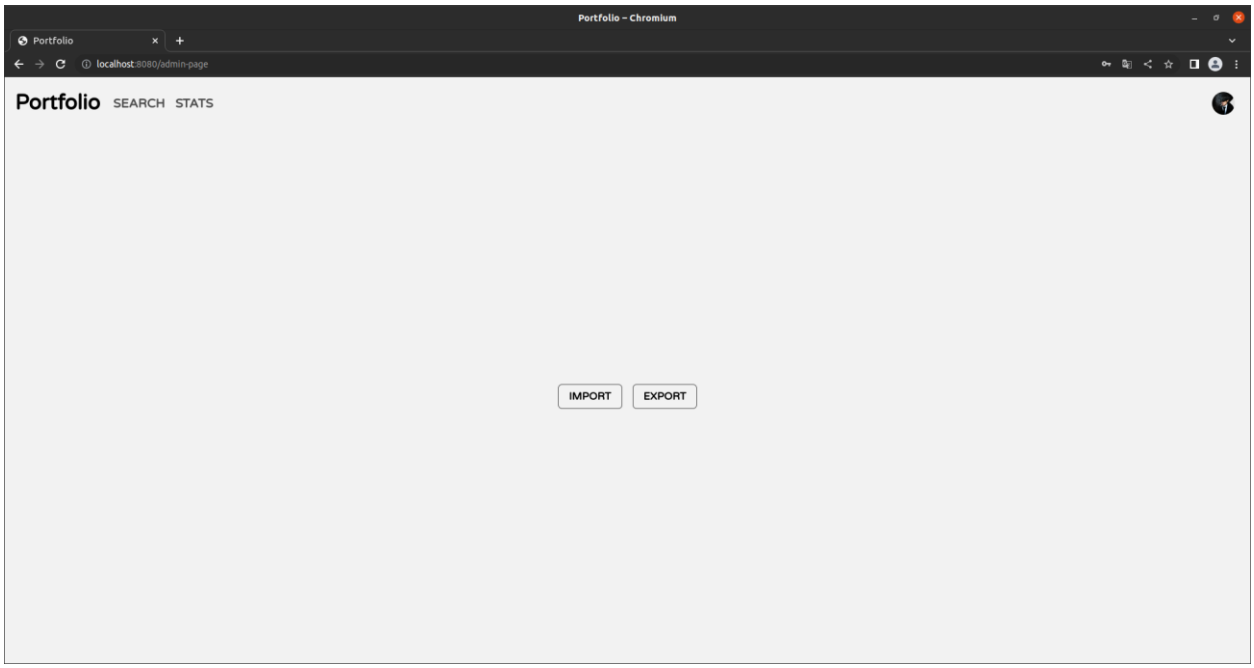

Рисунок 15 - Страница администратора (недоступна обычным пользователям)