

**Министерство цифрового развития, связи и массовых коммуникаций  
Ордена трудового Красного Знамени  
федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Московский технический университет связи и информатики»**

Кафедра Математическая кибернетика и информационные технологии

Отчет по учебной практике (ознакомительной)

Выполнил:

студент группы БПИ-2301

Черненко Сергей Сергеевич

Руководитель: Тимчук А.В.

Москва 2024

## Оглавление

1 Введение.....	3
2 Ход работы.....	4-9
3 Тестирование .....	10
4 Используемая литература .....	11
5 Ссылки.....	11

## 1 Введение

Цель работы: разработка платформы парсинга данных о соискателях и вакансиях с платформ hh.ru, avito.ru, habr Карьера.

Задачи:

- 1) изучить существующие платформы для парсинга (beautiful soap, selenium или API платформ);
- 2) сформулировать функциональные требования к системе;
- 3) спроектировать базу данных;
- 4) написать инструкцию пользователя;
- 5) провести тестирование системы.

Требования к функционалу:

- 1) Возможность формирования запроса для парсинга данных по таким полям, как фео, название должности, навыки, формат работы и т.д.
- 2) Загрузка в базу данных информации по результатам парсинга и вывод аналитики по параметрам (например, сколько есть вакансий и сколько кандидатов на вакансию).

## Ход работы

### 1. Создание Backend-части приложения.

Подключим mongoDB через mongoose:

```
1 mongoose
2 .connect(
3   "mongodb+srv://admin:wwwwww@cluster0.t54d0pj.mongodb.net/parsing_hhru?retryWrites=true&w=majority&appName=Cluster0"
4 )
5 .then(() => {
6   console.log("DB ok");
7 })
8 .catch((err) => {
9   console.log("error DB");
10 });
```

Создадим схему:

```
1 import mongoose from "mongoose";
2
3 const VacanciesSchema = new mongoose.Schema({
4   _id: { type: String },
5   name: { type: String },
6   city: { type: String },
7   employment: { type: String },
8   salary_from: { type: Number },
9   salary_to: { type: Number },
10 });
11
12 export default mongoose.model("Vacansies", VacanciesSchema);
```

Разработаем методы для работы приложения, их будет 3:



```
1 app.get("/vacancies", getAllVacancies);
2 app.get("/names", getNames);
3 app.post("/vacancies/name", getVacanciesByName);
```

Get методы:

-/vacancies – метод для получения всех вакансий с бд.

-/names - метод для получения всех названий вакансий с бд.

Post метод:

-/vacancies/name – метод для получения вакансий по фильтрам с бд, но сначала он занесет сами данные с hh.ru в бд.

## 2. Создание Frontend-части приложения

При первой загрузке будут браться все вакансии, которые когда-то были занесены в бд и отрисовываться первые 6, затем при нажатии на кнопку 'Показать еще' следующие 6 и так далее.

Затем разработаем поиск нужных вакансий по фильтрам: зарплата, название.

Расскажу подробнее о разработке фильтра по зарплате:

Для этого я использую библиотеку react-slider. У нас будет 2 ползунка, где можно задать минимальное и максимальное значение.

```
1 <div className={styles.slider_block}>
2   <div className={styles.slider_main}>
3     <p>Зарплата</p>
4     <label>
5       <ReactSlider
6         name="age"
7         className={styles.slider}
8         thumbClassName={styles.slider_thumb}
9         trackClassName="slider_track"
10        onChange={({value: [number, number]}) =>
11          setSalary({ from: value[0], to: value[1] })
12        }
13        defaultValue={[salary.from, salary.to]}
14        min={0}
15        max={500000}
16        step={10000}
17        minDistance={50000}
18        renderThumb={({props, state}) => <div {...props}></div>}
19      />
20    </label>
21  </div>
```

**Как работает поиск: со всех фильтров мы получаем значения, сохраняем их, после чего передаем запрос на сервер со всеми эти значениями при нажатии на кнопку “Поиск ” и возвращаем результат.**

```
1  async function findVacancies() {
2    const tempValue = value ? (value.value ? value.value : value) : "";
3    console.log(tempValue);
4    try {
5      const response: { data: inVacancy[] } = await instance.post(
6        "/vacancies/name",
7        {
8          salary_from: salary.from,
9          salary_to: salary.to,
10         text: tempValue,
11         employ: employ,
12       }
13     );
14     if (response.data.name === "нет") {
15       setVacancies([]);
16     } else setVacancies(response.data);
17   } catch (err) {
18     console.log(err);
19   }
20 }
```

### 3. Обернем приложение в Docker Compose:

Создадим Dockerfile для frontend-части:

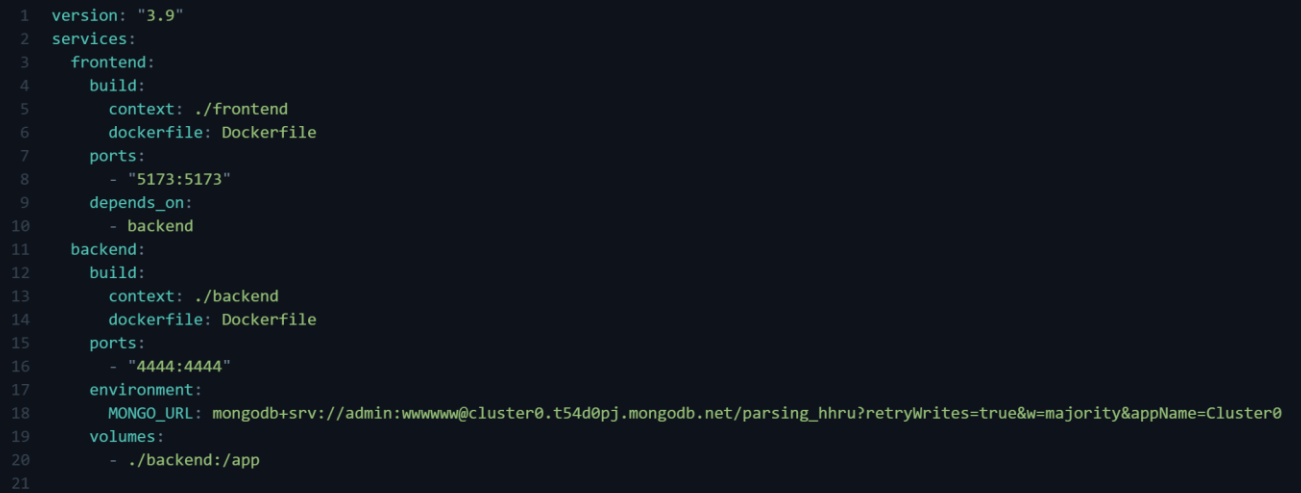
```
1 FROM node:21-alpine
2
3 WORKDIR /app
4
5 COPY package.json yarn.lock* package-lock.json* pnpm-lock.yaml* ./
6 RUN \
7     if [ -f yarn.lock ]; then yarn --frozen-lockfile; \
8     elif [ -f package-lock.json ]; then npm ci; \
9     elif [ -f pnpm-lock.yaml ]; then yarn global add pnpm && pnpm i --frozen-lockfile; \
10    else echo "Lockfile not found." && exit 1; \
11    fi
12
13 RUN yarn
14
15 COPY . .
16
17 CMD ["yarn", "dev", "--host", "0.0.0.0"]
```

Для Backend части:

```
1 FROM node:21-alpine
2
3 WORKDIR /app
4
5 COPY package*.json ./
6
7 RUN npm install
8
9 COPY . .
10
11 EXPOSE 4444
12
13 CMD ["npm", "run", "start:dev"]
```



**Создадим файл docker-compose.yml:**



```
1 version: "3.9"
2 services:
3   frontend:
4     build:
5       context: ./frontend
6       dockerfile: Dockerfile
7     ports:
8       - "5173:5173"
9     depends_on:
10      - backend
11  backend:
12    build:
13      context: ./backend
14      dockerfile: Dockerfile
15    ports:
16      - "4444:4444"
17    environment:
18      MONGO_URL: mongodb+srv://admin:wwwwww@cluster0.t54d0pj.mongodb.net/parsing_hhru?retryWrites=true&w=majority&appName=Cluster0
19    volumes:
20      - ./backend:/app
21
```

**Как запустить проект.**

- Открыть** папку с проектом в терминале
- Прописать в терминале `docker-compose up`

## Тестирование

### 1. Проверка корректности отображения диапазона зарплаты:

- Шаг 1: Проверить, что по умолчанию отображается текст "Не указана" для диапазона зарплаты.
- Шаг 2: Передвинуть ползунки слайдера зарплаты в произвольное положение.
- Шаг 3: Проверить, что текст отображает выбранный диапазон зарплаты в формате "от [минимальная зарплата] до [максимальная зарплата]".
- Шаг 4: Передвинуть ползунки слайдера в крайние положения (0 и 500000).
- Шаг 5: Проверить, что текст отображает правильный диапазон зарплаты, учитывая максимальное и минимальное значения слайдера.

Все успешно работает.

### 2. Проверка работы поиска по названию вакансии:

- Шаг 1: Ввести в поле "Введите название вакансии" название вакансии, которое присутствует в списке вариантов.
- Шаг 2: Проверить, что в списке вариантов автозаполнения отображаются варианты, содержащие введенный текст.
- Шаг 3: Выбрать из списка вариантов название вакансии.
- Шаг 4: Нажать на кнопку "Поиск".
- Шаг 5: Проверить, что в компоненте HomeFinds отображаются результаты поиска по выбранной вакансии.
- Шаг 6: Ввести в поле "Введите название вакансии" название вакансии, которое отсутствует в списке вариантов.
- Шаг 7: Проверить, что в списке вариантов автозаполнения отображаются варианты, содержащие введенный текст, даже если вакансия отсутствует в списке.
- Шаг 8: Проверить, что после нажатия на кнопку "Поиск" отображается соответствующее сообщение об отсутствии результатов.

Все успешно работает

## **Использованная литература**

<https://nodejs.org/api/all.html>

<https://www.mongodb.com/docs/>

<https://mongoosejs.com/docs/>

<https://expressjs.com/en/api.html>

<https://api.hh.ru/openapi/redoc>

## **Ссылки:**

**Гит репозиторий - <https://github.com/Quzzqq/parsing-hhru>**