САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа

Выполнил:

Котовщиков Андрей

Группа K3439

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

- Реализовать моковое API средствами JSON-сервера.
- Подключить авторизацию средствами json-server-auth.

Ход работы

Подготовка моковых данных

В начале были установлены необходимые пакеты (json-server и json-server-auth). Затем при помощи нейросети были сгенерированы моковые данные и записаны в файл db.json. В конце подготовки был запущен json-server при помощи команды npm start.

Авторизация и регистрация

Логика авторизации и регистрации расположена в файлах sign-in.js и sign-up.js соответственно. В них происходит обращение к json-server при помощи fetch (рисунок 1 и рисунок 2).

```
async function requestAuthCode({ email }) {
    const jsonBody = {
        email,
        password: "password",
    }

    const response = await fetch(API_URL, {
        method: "POST",
        body: JSON.stringify(jsonBody),
        headers: {
            "Content-Type": "application/json",
        },
    })

    const jsonResponse = await response.json()
    if (!response.ok) {
        return alert(jsonResponse)
    }

    const accessToken = jsonResponse.accessToken
    const authCode = generateAuthCode()

    return [accessToken, authCode]
}
```

Рисунок 1 — Запрос на авторизацию

```
async function signUp({ firstName, lastName, email }) {
    const jsonBody = {
        firstName,
        lastName,
        email,
        password: "password",
    }

    const response = await fetch(API_URL, {
        method: "POST",
        body: JSON.stringify(jsonBody),
        headers: {
            "Content-Type": "application/json",
        },
    })

    const jsonResponse = await response.json()
    if (!response.ok) {
        return alert(jsonResponse)
    }

    localStorage.setItem("accessToken", jsonResponse.accessToken)
    window.location.href = "restaurant-list.html"
}
```

Рисунок 2 — Запрос на регистрацию

Рендер карточек с ресторанами

Далее была реализована логика запросов списка ресторанов (файл restaurant-list.js) с возможностью фильтрации по цене, городу и кухне (рисунок 3), а также отрисовка полученных данных на главной странице (рисунок 4). Фильтрация происходит при помощи передачи соответствующих параметров в строке запроса.

```
async function fetchRestaurants({ city, cuisine, priceFrom }) {
    const url = new URL(API_URL)
    if (city) {
        url.searchParams.append("city", city)
    }

    console.log(cuisine)
    if (cuisine) {
        url.searchParams.append("cuisines_like", cuisine)
    }

    if (priceFrom) {
        url.searchParams.append("priceFrom_gte", priceFrom)
    }

    const response = await fetch(url, {
        method: "GET",
        headers: {
            "Content-Type": "application/json",
        },
    })

    if (!response.ok) {
        return []
    }

    return await response.json()
}
```

Рисунок 3 — Запрос списка ресторанов

```
function renderRestaurantCard({ id, name, description, city, cuisines, priceFrom }) {
    let cuisinesString = cuisines.join(", ")
    if (cuisinesString.length > 15) {
        cuisinesString = cuisinesString.substring(0, 15) + "..."
    }

    const container = document.querySelector(".card-list")
    const card = `
            <div class="col">
                <div class="card">
                    <img
                        src="https://zefirka.club/uploads/posts/2023-01/1673355119_22-zefirka-club-p-znak-voprosa-na-chernom-
                        class="card-img-top"
                        alt="${name}"
                    >
                    <div class="card-body">
                        <h5 class="card-title">${name}</h5>
                        <p class="card-text">${description}</p>
                        <ul class="list-group list-group-flush mb-3">
                            <li class="list-group-item">Город: ${city}</li>
                            <li class="list-group-item">Кухня: ${cuisinesString}</li>
                            <li class="list-group-item">Цена: От ${priceFrom} рублей</li>
                        </ul>
                        <a href="restaurant-detail.html?id=${id}&name=${name}" class="btn btn-info text-white">Подробнее</a>
                    </div>
                </div>
            </div>
        `

    container.innerHTML += card
}
```

Рисунок 4 — Отрисовка карточки ресторана

Рендер истории бронирования пользователя

В личном кабинете пользователя запрашивается его история бронирования (рисунок 5) и затем отрисовывается на странице профиля (рисунок 6). Токен авторизации берется из localStorage, куда он был сохранен ранее при авторизации.

```
async function fetchBookings({ accessToken }) {
    const response = await fetch(`${API_URL}/bookings`, {
        method: "GET",
        headers: {
            "Content-Type": "application/json",
            Authorization: `Bearer ${accessToken}`,
        },
    })

    if (!response.ok) {
        return []
    }

    return await response.json()
}
```

Рисунок 5 — Запрос истории бронирования

```
function renderBookingCard({ restaurantName, date, totalAmount }) {
    const dateObj = new Date(date)
    const container = document.querySelector(".booking-cards")
    const card = `
        <div class="list-group-item d-flex justify-content-between align-items-center">
            <div>
                <h6 class="mb-1">${restaurantName}</h6>
                <small>${dateObj.toLocaleString()}</small>
            </div>
            <span class="badge bg-danger rounded-pill">${totalAmount} ₽</span>
        </div>`

    container.innerHTML += card
}
```

Рисунок 6 — Отрисовка истории бронирования

Вывод

В рамках лабораторной работы 2 были изучены инструменты для создания мокового API, а также совренменные средства для выполнения AJAX запросов в браузере (fetch API).