САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №2

Выполнила:

Гергокова Амина

Группа

K4242

Проверил:

Добряков Д. И.

Санкт-Петербург

2024 г.

Задача

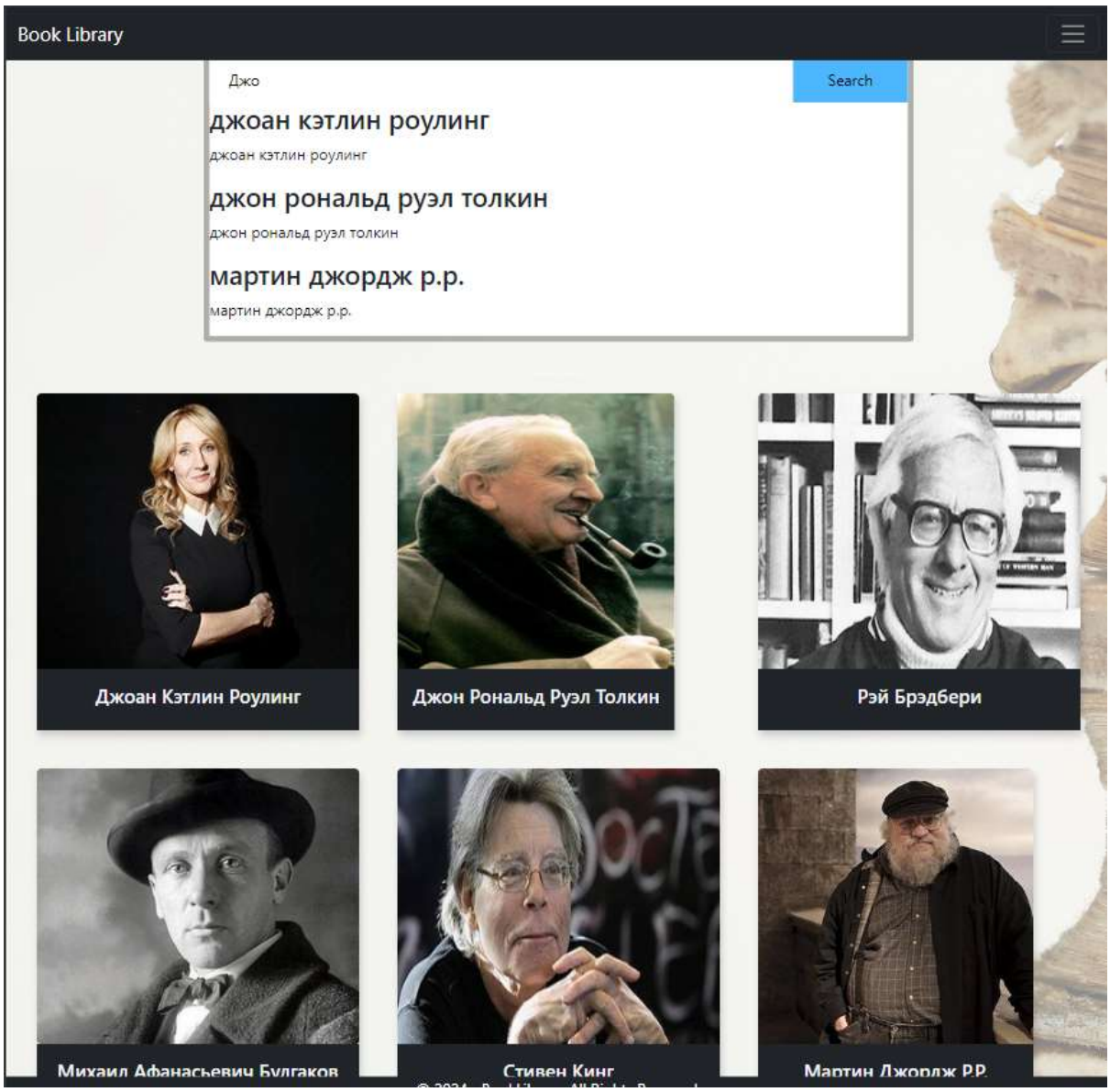
Взаимодействие с внешним API. Нужно реализовать имитацию API и запланированный функционал.

Ход работы

Вёрстка реализована при помощи HTML, CSS и Bootstrap.

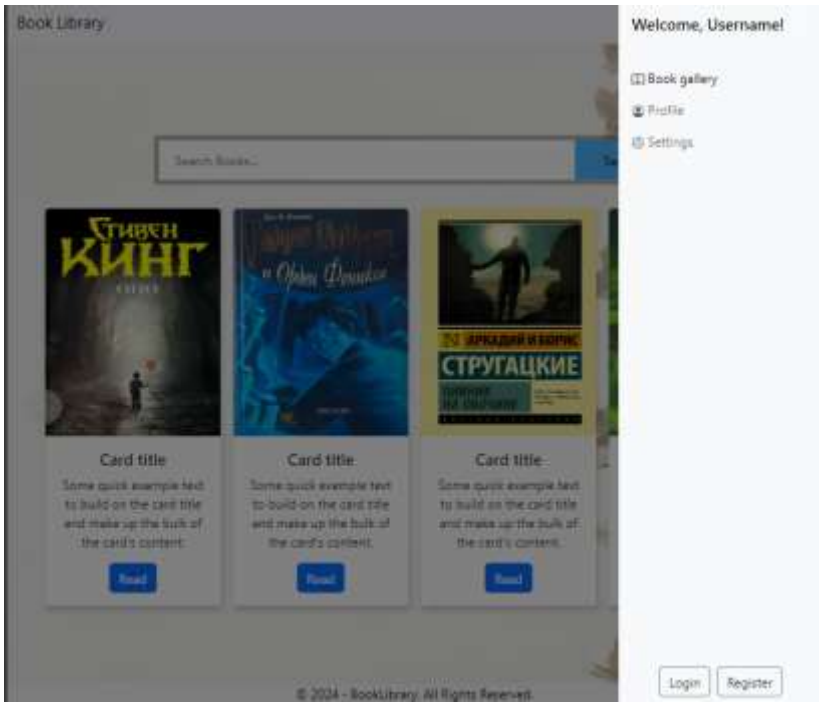
Основная страница:

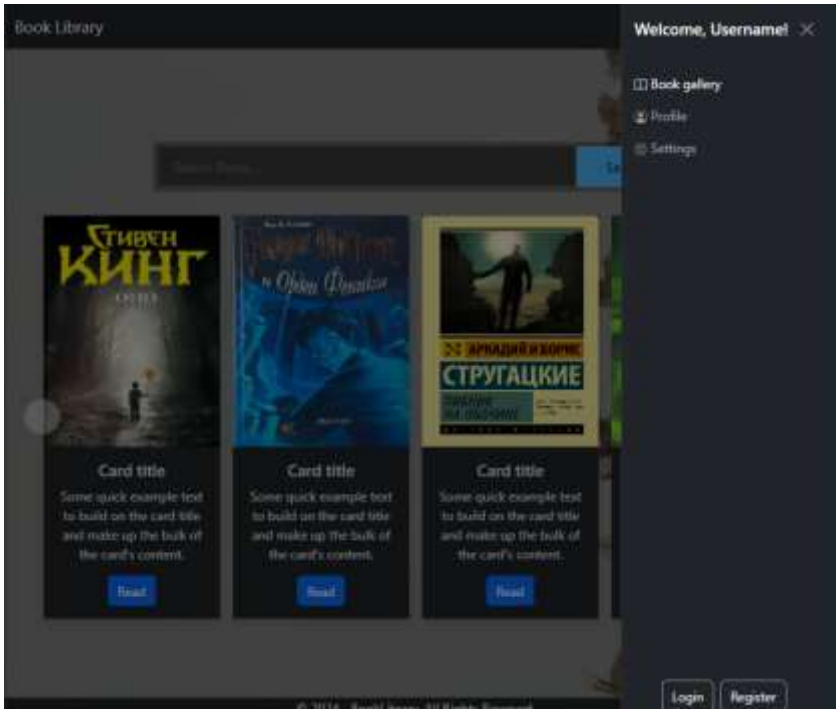
Реализована функция поиска по авторам.



```
function searchAuthors(event) {
  event.preventDefault();

  const searchInput = document.getElementById('searchInput').value.toLowerCase();
  const author = document.querySelectorAll('.editors-picks');

  const searchResults = [];

  author.forEach((author) => {
    const title = author.querySelector(".author").textContent.toLowerCase();

    if (title.includes(searchInput)) {
        console.log(title)
      searchResults.push({
        title: title,
      });
    }
  });

  displaySearchResults(searchResults, searchInput);
}

function displaySearchResults(results, searchInput) {
  const searchResultsContainer = document.getElementById('searchResults');
  searchResultsContainer.innerHTML = '';

  if (results.length === 0) {
    searchResultsContainer.innerHTML = 'No results found.';
    return;
  }

  results.forEach((result) => {
    const resultItem = document.createElement('div');
    const highlightedText = highlightSearchTerm(result.title, searchInput);
    resultItem.innerHTML = `<h3>${result.title}</h3><p>${highlightedText}</p>`;
    searchResultsContainer.appendChild(resultItem);
  });
}

function highlightSearchTerm(text, searchTerm) {
  const regex = new RegExp(`(${searchTerm})`, 'gi');
  return text.replace(regex, '<span class="highlighted">$1</span>');
}
```

Добавлена функция смены темы пользователем:

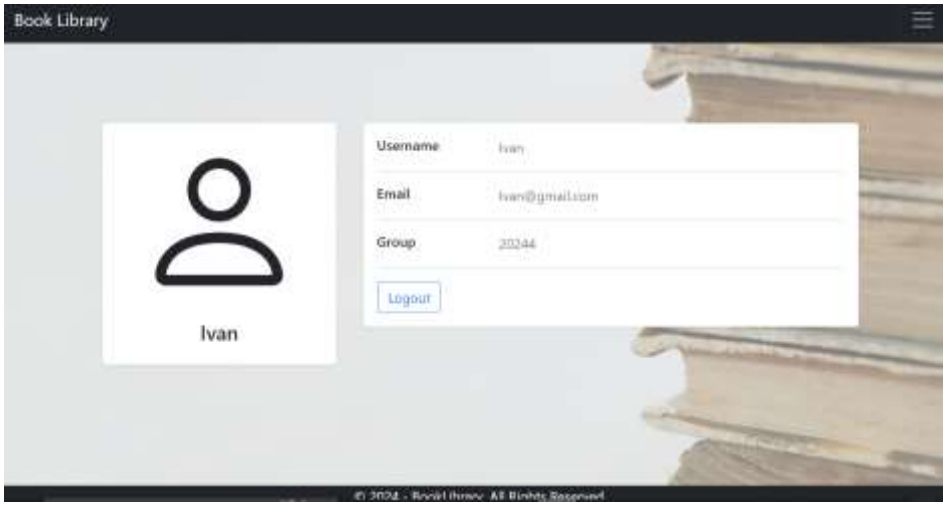




Страница регистрации:

Добавлена функция регистрации/авторизации нового пользователя

После успешной авторизации пользователь попадает на страницу профиля, где отображаются данные пользователя. И есть кнопка выхода (очищает local storage и перенаправляет на главную страницу)



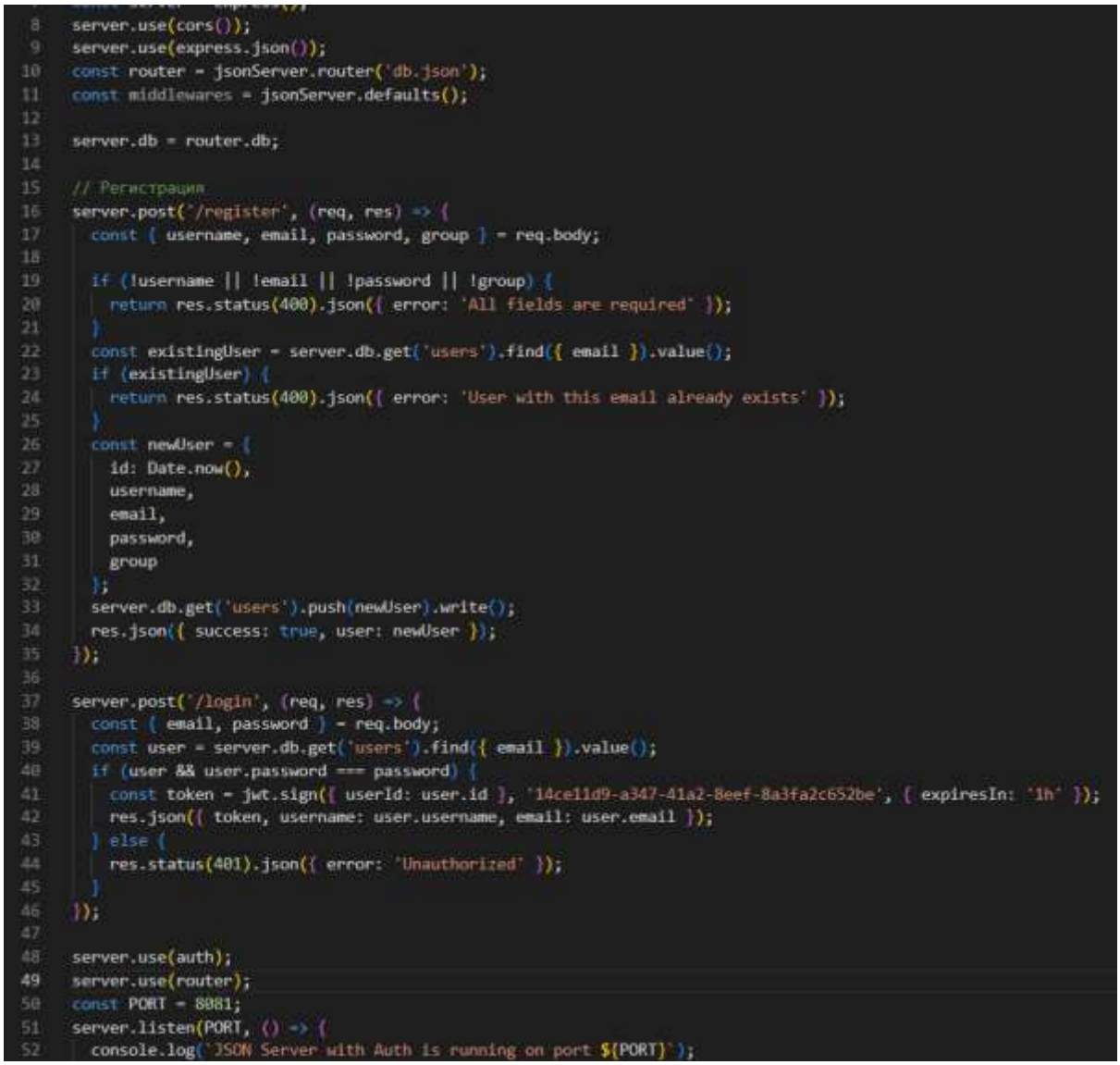
Json-файл, псевдо бд.

Реализация методов.

```
async function login(event) {
    event.preventDefault();

    const email = document.getElementById('inputEmail1').value;
    const password = document.getElementById('inputPassword1').value;

    try {
      const response = await fetch('http://localhost:8081/login', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        mode: 'cors',
        body: JSON.stringify({ email, password }),
      });

      if (response.ok) {
        const data = await response.json();
        console.log('Login successful', data);

        localStorage.setItem('token', data.token);
        localStorage.setItem('name', data.name);
        localStorage.setItem('email', data.email);

        window.location.href = `profile.html?name=${data.name}&email=${data.email}`;
      } else {
        const errorData = await response.json();
        console.error('Login failed:', errorData);
      }
    } catch (error) {
      console.error('Login failed:', error);
    }
  }

  async function register(event) {
    event.preventDefault();

    const name = document.getElementById('inputName2').value;
    const email = document.getElementById('inputEmail2').value;
    const password = document.getElementById('inputPassword2').value;
    const repeatPassword = document.getElementById('inputPassword3').value;

    if (password !== repeatPassword) {
      console.error('Passwords do not match');
      return;
    }
```

```
        localStorage.setItem('email', user.email);

        console.log('localStorage name:', localStorage.getItem('name'));
        console.log('localStorage email:', localStorage.getItem('email'));

        window.location.href = `profile.html?name=${user.name}&email=${user.email}`;
      } else {
        const errorData = await response.json();
        console.error('Registration failed:', errorData);
      }
    } catch (error) {
      console.error('Registration failed:', error);
    }
  }

function changeProfile() {
    const params = new URLSearchParams(window.location.search);
    const username = params.get('username');
    const email = params.get('email');
    const nameInput = document.getElementById('nameInput');
    const emailInput = document.getElementById('emailInput');
    if (username !== null && email !== null) {
        nameInput.value = username;
        emailInput.value = email;
        localStorage.setItem('profileName', username);
        localStorage.setItem('profileEmail', email);
    }
};

document.addEventListener('DOMContentLoaded', updateProfileView);
document.addEventListener('DOMContentLoaded', function () {
    const storedName = localStorage.getItem('profileName');
    const storedEmail = localStorage.getItem('profileEmail');
    const nameInput = document.getElementById('nameInput');
    const emailInput = document.getElementById('emailInput');
    if (storedName !== null && storedEmail !== null) {
        nameInput.value = storedName;
        emailInput.value = storedEmail;
    }
});

function logout() {
    localStorage.removeItem('profileName');
    localStorage.removeItem('profileEmail');
    window.location.href = 'index.html';
```

Небольшой сервер на express.js и json-server



```
server.use(cors());
server.use(express.json());
const router = jsonServer.router('db.json');
const middlewares = jsonServer.defaults();

server.db = router.db;

// Регистрация
server.post('/register', (req, res) => {
  const { username, email, password, group } = req.body;

  if (!username || !email || !password || !group) {
    return res.status(400).json({ error: 'All fields are required' });
  }
  const existingUser = server.db.get('users').find({ email }).value();
  if (existingUser) {
    return res.status(400).json({ error: 'User with this email already exists' });
  }
  const newUser = {
    id: Date.now(),
    username,
    email,
    password,
    group
  };
  server.db.get('users').push(newUser).write();
  res.json({ success: true, user: newUser });
});

server.post('/login', (req, res) => {
  const { email, password } = req.body;
  const user = server.db.get('users').find({ email }).value();
  if (user && user.password === password) {
    const token = jwt.sign({ userId: user.id }, '14ce11d9-a347-41a2-8eef-8a3fa2c652be', { expiresIn: '1h' });
    res.json({ token, username: user.username, email: user.email });
  } else {
    res.status(401).json({ error: 'Unauthorized' });
  }
});

server.use(auth);
server.use(router);
const PORT = 8081;
server.listen(PORT, () => {
  console.log(`JSON Server with Auth is running on port ${PORT}`);
```

Вывод

Во время выполнения данной лабораторной работы познакомились и изучили работу с API, express.js и json-server, а также доработали функционал.