МИНОБРНАУКИ РОССИИ

–––––––——————————–––––––

Санкт-Петербургский государственный  
электротехнический университет «ЛЭТИ»

————————————————————

**Web-программирование**

**Лабораторная № 5**

**Тема: НАПИСАНИЕ ЛОГИКИ ДЛЯ ФОРМЫ АВТОРИЗАЦИИ**

Выполнил: студент группы 3311 Шарпинский Денис

Проверил: Калмыков Михаил Александрович

Санкт-Петербург

Издательство СПбГЭТУ «ЛЭТИ»

2025

Цель работы: написание логики для сервера и клиента для создания авторизации.

Для авторизации пользователей я немного отклонюсь от методических указаний и буду хранить пользователей не просто в массиве в оперативной памяти сервера, а также в JSON файле в корневой директории проекта. При старте работы сервера я читаю этот файл и записываю в объект users. Объект имеет следующий тип:

type Tusers = {[key: string]: {name: string, password: string, color: string, id: string}}

Здесь ключом выступает id пользователя, а значением – информация о пользователе. Такая структура является стандартной для js и позволяет быстро удалять и добавлять новых пользователе (для сравнения при хранении пользователей в массиве приходилось бы каждый раз делать обход, чтобы найти нужного).

Я создал на сервере роут POST /api/auth (все обращения к серверу, которые не ведут на какую-то страницу будут начинаться на /api). В нём я провожу проверку, является ли пользователь уже авторизованным. Если так, то я проверяю введенные данные – логин и пароль, которые переданы в body запроса. Если данные верны – я возвращаю 200-ый статус код, иначе возвращаю 401 ошибку. Если пользователь не был авторизован, то я создаю нового пользователя и асинхронно сохраняю в файл.

Затем на клиенте я написал небольшой скрипт, который делает fetch на сервер с указанными в форме данными. После успешного выполнения запроса в window.location.href дописывается /profile с сохранением query параметра. Иначе, если статус код не попадает в диапазон [200, 300), поля ввода получают красную обводку – так я даю пользователю понять, что он ввел неверные данные.

Примеры работы приложения:

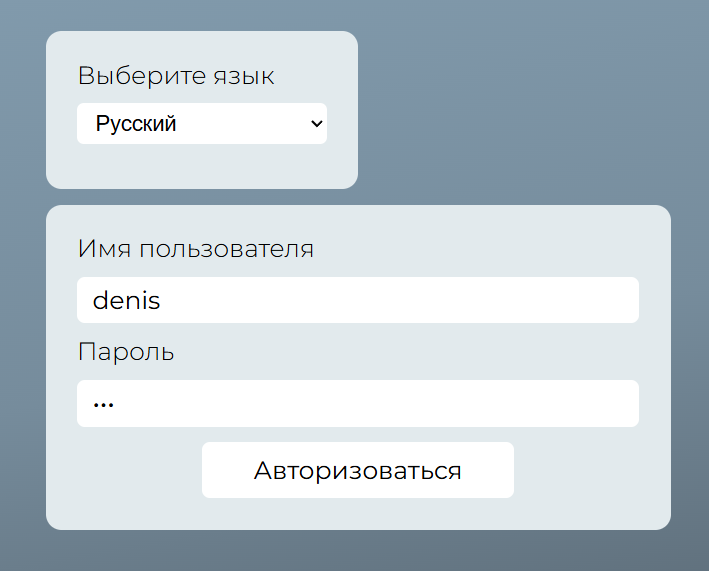


Рис 5.1 – заполнение формы авторизации

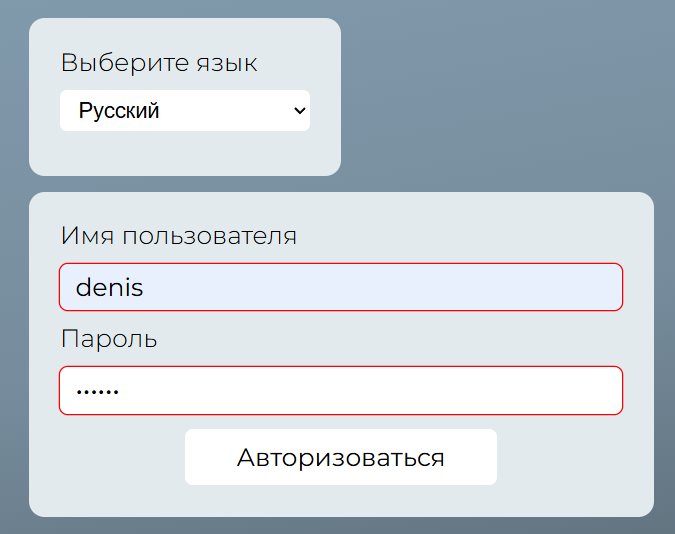


Рис 5.2 – неправильный пароль

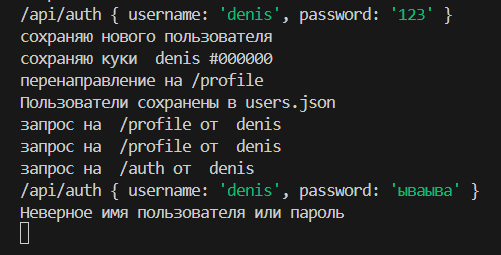


Рис 5.3 – логи сервера.

Выводы:

В ходе выполнения работы была реализована базовая логика авторизации пользователей на клиенте и сервере. Данные пользователей хранятся в JSON-файле и загружаются при запуске сервера, что позволяет сохранять их между сессиями. Реализованы сценарии успешной авторизации и обработки ошибок, а также визуальная обратная связь пользователю в случае неверного ввода.