САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №2

Выполнил:

Пиотуховский Александр

K3441

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Необходимо привязать то, что было сделано в ЛР1, к внешнему API средствами fetch/axios/xhr. Реализуйте моковое API и подключите к нему авторизацию.

Ход работы

В качестве серверной части был разработан мок-сервер на базе фреймворка FastAPI. Данные хранятся в формате JSON, что позволяет имитировать работу реальной базы данных. Реализованы необходимые для веб-приложения эндпоинты, такие как авторизация и регистрация, получение рецептов, постов, подробная их информация, комментарии и другое.

Для организации взаимодействия клиентской части с сервером был разработан JavaScript класс ApiClient, представляющий собой обёртку над стандартным браузерным API fetch. Это решение позволило инкапсулировать логику отправки запросов, установку заголовков и обработку ошибок в одном месте, избегая дублирования кода.

Ключевой особенностью реализации является механизм авторизации на основе JWT. При успешном входе пользователя сервер возвращает токен, который сохраняется в localStorage браузера. Класс ApiClient автоматически добавляет этот токен в заголовок авторизации для запросов, требующих аутентификации.

На рисунке 1 представлен фрагмент кода класса ApiClient, отвечающий за формирование запроса и перехват ошибок.

Рисунок 1 – Реализация метода request в классе ApiClient

В методе реализована логика перехвата 401 ошибки. Если сервер возвращает такой статус, приложение автоматически очищает локальное хранилище и перенаправляет пользователя на страницу авторизации.

Наше приложение всё ещё находится в активной стадии разработки. При загрузке страницы клиент пытается обратиться к удалённому API. В случае недоступности сервера приложение автоматически переключается на использование локальных моков (старых JSON файлов из папки mocks),

тем самым обеспечивая работоспособность интерфейса даже без соединения с бэкендом.

Функционал сайта был разделен на публичный и приватный. Просмотр ленты рецептов и поиск доступны всем посетителям, однако действия, требующие идентификации (комментирование, просмотр своего профиля), защищены. Реализована проверка наличия токена: если неавторизованный пользователь попытается получить доступ к защищенному маршруту, он будет перенаправлен на страницу входа.

Для удобства пользователя интерфейс динамически реагирует на статус авторизации. Если токен валиден, в навигационной панели отображается аватар пользователя и его имя. В противном случае отображается кнопка «Войти».

Вывод

В рамках лабораторной работы была выполнена клиент-серверная архитектура веб-приложения. Использование fetch и async/await позволило настроить асинхронное взаимодействие с API без блокировки интерфейса. Внедрение класса-обертки `ApiClient` упростило масштабирование системы запросов. Были освоены принципы работы с локальным хранилищем браузера, что позволило превратить статичную вёрстку в динамическое веб-приложение.