САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа

Выполнил:

Андронов Денис

Группа

K3342

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Лабораторная работа 2: взаимодействие с внешним API

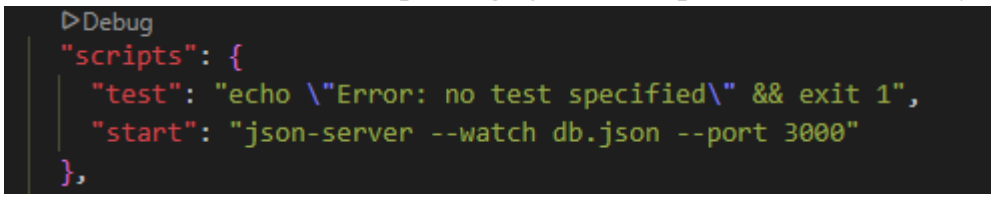
Варианты остаются прежними. Теперь Вам нужно привязать то, что Вы делали в ЛР1 к внешнему API средствами fetch/axios/xhr. Реализуйте моковое API средствами JSON-сервера и подключите к нему авторизацию, как в примерах, которые мы рассматривали в рамках тем "Иммитация работы с API".

Например, для приложения для просмотра прогнозов погоды задание выглядит следующим образом:

Реализовать получение погоды (прогноз на ближайшие 7 дней) из открытого API OpenWeatherMap, взависимости от геолокации пользователя. Реализовать вывод полученного прогноза в виде 7 карточек в три ряда (первый ряд - крупная карточка, второй ряд - три карточки в меньшем размере, третий ряд - четыре карточки в маленьком размере).

Ход работы

Сначала я установил NodeJS, затем локально в проект установил json-сервер командами npm init –y и npm install json-server --save-dev. Затем я перенес из html файла данные о ресторанах, букингах и пользователях в db.json. Также дописал в package.json скрипт для запуска сервера.



Далее я убрал список ресторанов с главной страницы и перенес логику на доставание их скриптом main.js. Чтобы не повторяться командами я вынес всю логику с запросами к API в api.js – доставание ресторанов по ID, логин, регистрация, доставание букинга пользователя и его создание.

В main.js у меня теперь загрузка ресторанов, рендер карточек и фильтрация. Далее в login.html и register.html я чуть поменял формы логина/регистрации. Логика входа и регистрации у меня теперь в auth.js. С помощью функции getelementbyid мы понимаем, какая это форма – регистрации или логина, делаем проверки введенных данных и используем опять же api.js для POST или GET соответственно. Также при успешном входе я сохраняю restorator_user в localStorage, чтобы пользователя не выкидывало из аккаунта.

На страницу профиля я добавил список бронирований и подтягивание их через API, подтягивая их через user из localStorage, и кнопку выхода.

На странице ресторана я добавил подгрузку страницы ресторана в зависимости от id и форму бронирования, которая затем добавляется в db.json через файлик restaurant.js методом POST.

Немного поправил формы в конце. Затем заметил баг, что меня после входа в профиль выкидывало обратно на страницу регистрации, это было связано с тем, что были разные переменные в localStorage по названию, и другая страничка считала, что пользователь не вошел. Поправил название переменной, и также для понятности, что пользователь уже вошел в аккаунт – я решил убирать кнопку входа и регистрации с навигационной панели, если вход произведен. Для этого я добавил id кнопкам в navbar и написал

файлик authnav.js, который достает информацию из localstorage и скрывает ненужные кнопки.

Вывод

В данной лабораторной работе я учился использовать моковое API.