**Full-Stack разработка веб приложений на JavaScript**

**Онлайн продуктовый магазин**

**(online-grocery-store)**

**BACKEND (Server)**

**npm run dev**

**STACK технологий – BACKEND**  
1. NODE JS – платформа для написания backend на языке js  
2. Express – фреймворк для написания backend, т.е для серверной части на node js

3. PostgreSQL – СУБД

4. Sequelize – ORM для реляционных БД на node js. ORM – технология, позволяющая связывать программный код с БД (не пишем запросы к БД напрямую, а создаем какую-то функцию, описывающие модели таблиц БД)

5. Postman- для тестирования методов http запросов. Клиент, который позволяет отправлять запросы на сервер

**Заметки**

dev nodemon – автоматически будет перезагружать сервер при изменении данных

cors – пакет для запросов с браузера

routes – маршруты

express-fileupload – для загрузки img

uuid – пакет для генерации случайных id (нужен для генерации id для img)

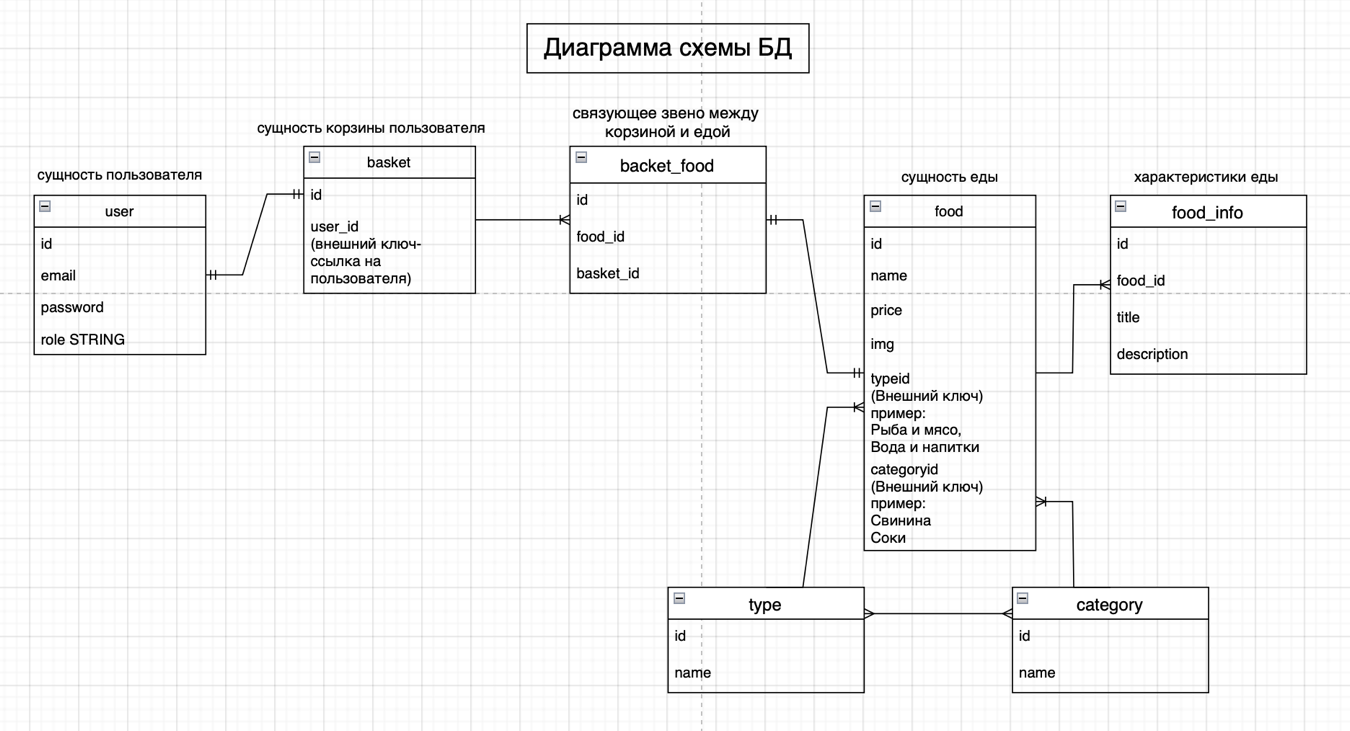
path – модуль node js который адаптирует путь к ОС

**Задачи**

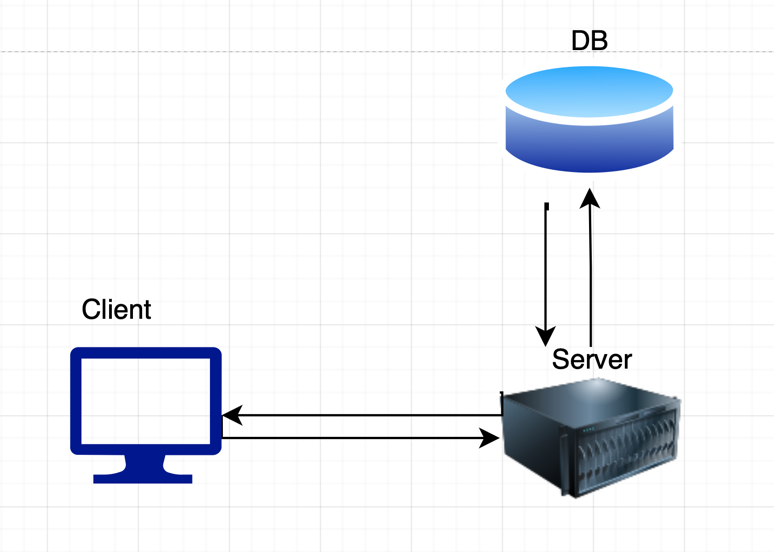
1. Построить диаграмму БД из 8 таблиц +

2. Реализовать авторизацию пользователя по JWT токену +

3. Полноценное Rest API онлайн продуктового магазина. (Сможем получать, добавлять, удалять данные) - **Нет удаления users , basket, basket\_food**



**Клиент серверное взаимодействие**



**Описание файлов и папок в папке server**

.env – Переменное окружение для настройки

db.js – Подключение к БД

models.js – Описание моделей схемы БД и типы связей между ними

**Каркас:**

Роутинг: папка routes – маршруты для сущностей

Post – создание; get – получение

Логика маршрутов: папка controllers

**Обработка ошибок:**

Error/ApiError.js – универсальная обработка ошибок

**Как воспользоваться?**

1) Импортировать класс ApiError

2) Задать условие на проверку

3) Добавить параметр к тестируемой функции next  
4) Вызвать метод next  
5) В параметре к методу указать ApiError   
6) У класса ApiError вызвать нужный метод   
7) Добавить параметр сообщение к методу

8) Добавить return

**Папка middleware**

ErrorHandlingMiddleware,js – проверяет следующее условие:

-если при загрузке api ошибка имеет класс ApiError, то она будет выведена

AuthMiddleware.js – декодирование токена и проверка на валидность (проверяющий авторизован ли пользователь)

**Как пользоваться?**

1) POST registration пользователя, затем переходим в login  
2) логинимся и получаем токен, копируем его

3) Выбираем GET, затем Headers, в ячейке выбираем Authorization

4) Вставляем токен, перед токеном Bearer

checkRoleMiddleware.js – проверка роли пользователя

**CRUD операции для выполнения SQL команд**

Происходит в controllers, они получают данные из тела запроса, затем выполняют функцию выполнения SQL команд

Папка static - в нее будут перемещаться все файлы, которые будут отправлять с клиента

Фультрация по категории или типу в foodController.js

**Авторизация происходит по JWT Token**

jsonwebtoken – модуль для генерации веб токена

bcrypt – для хэширования паролей (чтобы не хранить в БД)

**\*сделать : нормальная валидация для node js при регистрации**

**\*разобраться как работает jwt token**

**\*сделать корзину**

Создавать, удалять товары, категории, типы могут только пользователи ADMINS. (У USER нет доступа к type, category, food)

**Задачи:**

**1) Добавить удаление товаров  
2) Сделать basket, basket\_food**

**3) Нормальная валидация при регистрации и авторизации**

**FRONTEND (Client)**

**npm start**

**STACK технологий – FRONTEND**

1. React JS

2. React bootstrap – графика (чтобы не тратить время на верстку)

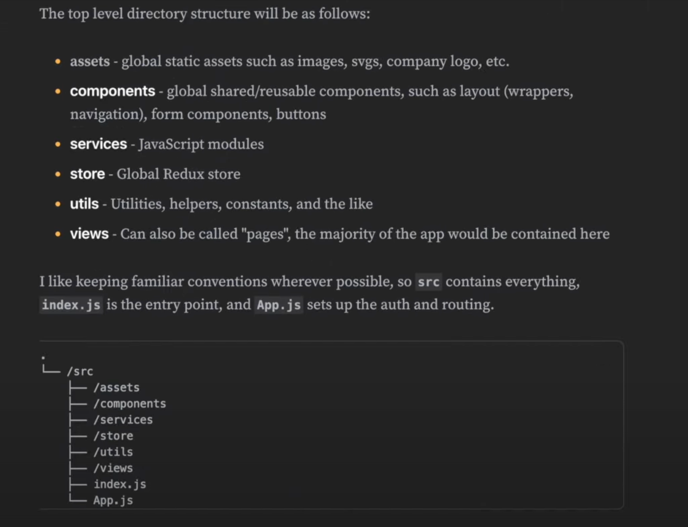
3. Axios – запросы к серверу (библиотека)

4. React-router-dom – подстраничная навигация

5. MobX – стейт менеджер

Mobx-react-lite – чтобы связать mobx с функциональными элементами реакта

**Структура приложения:**



App.js - основной компонент приложения

**Папка store** – взаимодействие с mobx и хранение данных

**Папка pages** – корневые компоненты, которые будут являться страницами

Auth.js – компонент с авторизацией

Shop.js – основная страница магазина

Food.js – карточка еды

Admin.js – админ панель

Basket.js - корзина

**Папка components** – для navbar и тд.

AppRouter – в нем описана логика, навигация по страницам

routes.js – все маршруты к страницам

**Задачи**1) Сделать полноценную верстку

2) Сделать модальное окно регистрации/авторизации (/login /registration)

**Сделано:**

Страница регистрации, авторизации

Главная страница: список категорий, список типов +