САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэкэнд разработка

Отчет

Лабораторная работа №5

Выполнил:

Габов Михаил

Группа K3440

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Цель работы

Декомпозировать монолитное приложение на микросервисную архитектуру с использованием Node.js, TypeScript, Express и Docker. Реализовать межсервисное взаимодействие через HTTP REST API и настроить API Gateway для единой точки входа.

Описание

Проект представляет собой платформу для публикации и взаимодействия с кулинарными рецептами. В рамках лабораторной работы была реализована микросервисная архитектура с разделением функциональности на независимые сервисы.

Реализованные микросервисы

1. API Gateway (:3000)

Маршрутизация:
*/users/** → auth-users-service:3001*
*/follows/** → auth-users-service:3001*
*/recipes/** → recipes-service:3002*
*/ingredients/** → recipes-service:3002*
*/comments/** → interactions-service:3003*
*/likes/** → interactions-service:3003*
*/favorites/** → interactions-service:3003*

2. Auth Users Service (:3001)

Функциональность:

- Регистрация пользователей с хешированием паролей (bcrypt)
- Аутентификация и генерация JWT токенов
- Управление профилем пользователя
- Система подписок (follow/unfollow)
- Поиск пользователей по username или ID
- Просмотр списка подписчиков и подписок

Основные эндпоинты:
POST /users/register - Регистрация нового пользователя
POST /users/login - Вход и получение JWT токена

GET /users/:id - Получение профиля пользователя
GET /users/search/by - Поиск по username или id
POST /follows/:targetId - Подписаться на пользователя
DELETE /follows/:targetId - Отписаться от пользователя
GET /follows/followers/:id - Получить список подписчиков
GET /follows/following/:id - Получить список подписок

Пример создания пользователя:

```
@Post('/register')
public async register(
  @Body() body: { username: string; password: string; email: string }
): Promise<{ userId: number }> {
  const hashedPassword = await bcrypt.hash(body.password, 10);
  const user = this.userRepository.create({
    username: body.username,
    passwordHash: hashedPassword,
    email: body.email
  });
  await this.userRepository.save(user);
  return { userId: user.id };
}
```

3. Recipes service (:3002)

Функциональность:

- CRUD операции для рецептов
- Многоуровневая структура рецептов (шаги приготовления)
- Управление ингредиентами
- Связывание ингредиентов с рецептами (количество и единица измерения)
- Фильтрация рецептов по ингредиентам и уровню сложности
- Межсервисное получение username от auth-users-service

Основные эндпоинты:
POST /recipes - Создание нового рецепта
GET /recipes/:id - Получение рецепта по ID
GET /recipes - Список рецептов (с фильтрами)
PUT /recipes/:id - Обновление рецепта

DELETE /recipes/:id - Удаление рецепта
POST /ingredients - Создание ингредиента
GET /ingredients - Список всех ингредиентов

Пример структуры рецепта


```
{
  "id": 1,
  "title": "Паста Карбонара",
  "description": "Классическая итальянская паста",
  "difficulty": "medium",
  "userId": 5,
  "username": "chef_mario",
  "steps": [
    {"stepNumber": 1, "instruction": "Отварите спагетти..."},
    {"stepNumber": 2, "instruction": "Обжарьте бекон..."}
  ],
  "ingredients": [
    {"name": "Спагетти", "quantity": 400, "unit": "г"},
    {"name": "Бекон", "quantity": 200, "unit": "г"}
  ]
}
```


4. Interactions Service (:3003)

Функциональность:

- Лайки рецептов (один пользователь может поставить только один лайк)
- Комментарии к рецептам
- Управление избранным (добавление/удаление рецептов)
- Получение статистики по рецепту (количество лайков, комментариев)

Основные эндпоинты:

POST /likes - Поставить лайк рецепту
DELETE /likes/:recipeId - Убрать лайк
GET /likes/recipe/:id - Список пользователей, лайкнувших рецепт

POST /comments - Добавить комментарий
GET /comments/recipe/:id - Комментарии к рецепту
POST /favorites - Добавить рецепт в избранное
DELETE /favorites/:recipeId - Убрать из избранного
GET /favorites - Список избранных рецептов

Заключение

В рамках лабораторной работы успешно реализована микросервисная архитектура для платформы кулинарных рецептов.