САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 1: Технический дизайн микросервисов

Выполнил:

Кузнецов Артур

K3440

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Необходимо спроектировать набор следующих диаграмм:

– общая архитектура решения (сервисы и их взаимосвязи, клиент-серверное взаимодействие);

– диаграмма компонентов;

– диаграммы БД по каждому сервису;

– диаграммы основных пользовательских сценариев (те сценарии, которые позволяют вашим приложением полноценно воспользоваться, пройти весь путь).

Ход работы

1. **Общая архитектура решения**

Сервис для обмена рецептами и кулинарных блогов

– Вход

– Регистрация

– Личный кабинет пользователя (сохраненные рецепты, публикации)

– Поиск рецептов с фильтрацией по типу блюда, сложности, ингредиентам

– Страница рецепта с фото, пошаговыми инструкциями и видео

– Социальные функции (комментарии, лайки, подписки на кулинаров)

Приложение разделено на три микросервиса: auth-service, recipe-service и social-service.

Auth-service отвечает за регистрацию и вход пользователей, генерирует JWT-токены при аутентификации. Auth-middleware из common-service проверяет токены в запросах к другим сервисам.

Recipe-service хранит и обрабатывает рецепты, ингредиенты, шаги рецепта, типы блюд и уровни сложности.

Social-service обрабатывает социальные функции (комментарии, лайки и подписки).

Каждый сервис использует собственную базу данных PostgreSQL (auth-db, recipe-db, social-db) и взаимодействует с другими через HTTP-запросы. Проверка токенов выполняется общим модулем authMiddleware из common-service. Клиент получает JWT через auth-service и использует его для авторизованных запросов к остальным сервисам.

Все сервисы контейнеризированы с помощью Docker. В docker-compose.yml описаны их соединения, отдельные базы данных, порты (3000, 3001, 3002) и общая сеть app-network, обеспечивающая взаимодействие между компонентами.

2. Диаграмма компонентов

Диаграмма отображает основные точки входа клиента, использование баз данных и зависимость сервисов от общих модулей (рисунок 1).

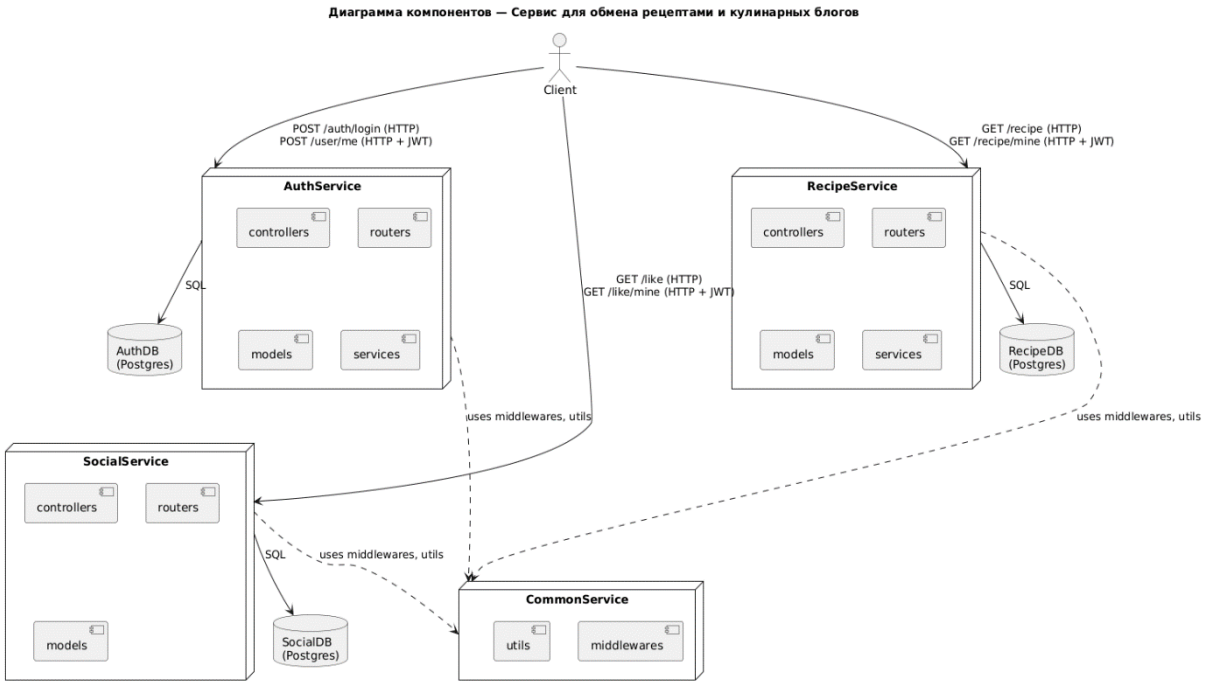

Рисунок 1 – Диаграмма компонентов

3. Диаграммы БД по каждому сервису

Диаграмма показывает структуру баз данных трёх микросервисов: AuthServiceDB, RecipeServiceDB и SocialServiceDB (рисунок 2).

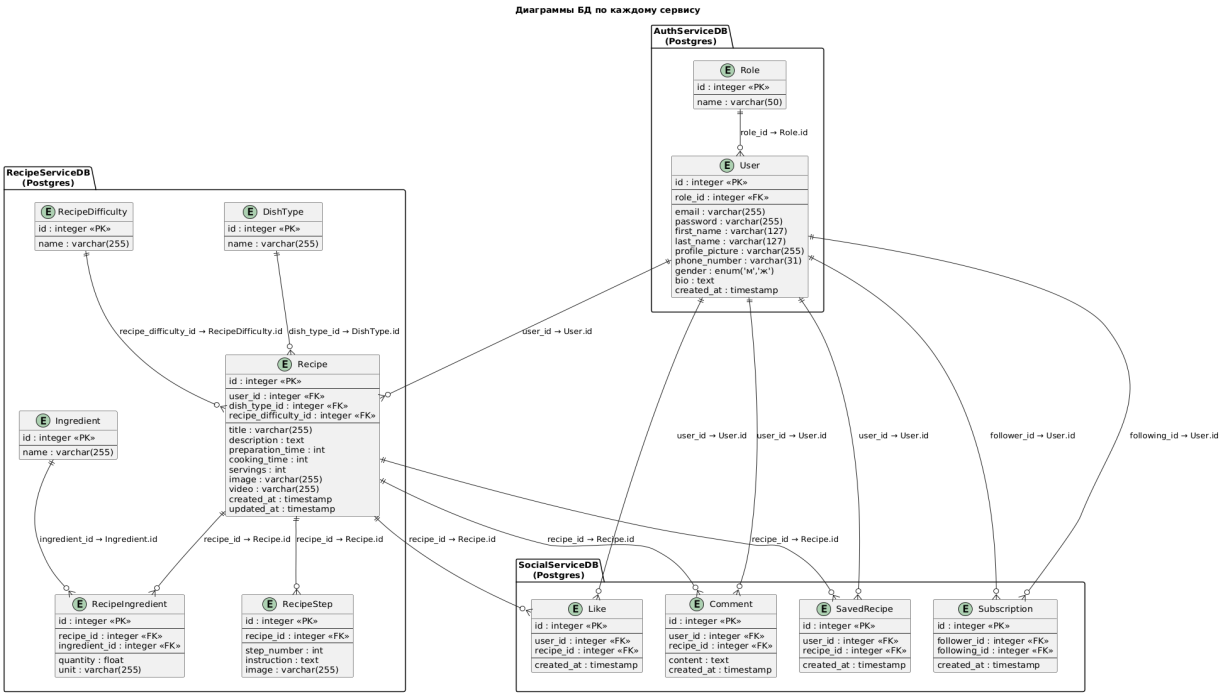

Рисунок 2 – Диаграммы БД по каждому сервису

AuthServiceDB содержит таблицы Role и User, где каждый пользователь связан с ролью (role_id → Role.id).

RecipeServiceDB включает таблицы для хранения рецептов: DishType, RecipeDifficulty, Ingredient, Recipe, RecipeIngredient и RecipeStep. Таблица Recipe ссылается на пользователя из AuthService (user_id → User.id) и связывается с типом блюда, уровнем сложности, шагами рецепта и ингредиентами через внешние ключи.

SocialServiceDB содержит социальные сущности: Comment, Like, SavedRecipe и Subscription. Все они связаны с пользователями (user_id, follower_id, following_id) и рецептами (recipe_id) из других сервисов.

Диаграмма отражает как внутренние связи таблиц внутри каждого сервиса, так и кросс-сервисные зависимости между микросервисами, обеспечивая целостность данных и поддержку микросервисной архитектуры.

4. Диаграммы основных пользовательских сценариев

1) Пользовательский сценарий - Регистрация

Пользователь (Client) отправляет запрос на регистрацию через AuthService с обязательными данными (email, пароль, имя и фамилия). Сервис проверяет уникальность email, получает роль пользователя из базы AuthDB и сохраняет нового пользователя. После успешного сохранения генерируется JWT-токен, который возвращается клиенту вместе с данными пользователя (id, имя, фамилия).

Пользовательский сценарий представлен на рисунке 3.

Рисунок 3 – Регистрация

2) Пользовательский сценарий - Создание и получение рецепта

Пользователь создаёт рецепт, отправляя данные в RecipeService. Сервис проверяет существование пользователя (UserRepo), тип блюда (DishTypeRepo) и уровень сложности (RecipeDifficultyRepo). После валидации рецепт сохраняется в базе RecipeDB и возвращается клиенту.

Для получения рецепта клиент обращается к сервису по id рецепта. RecipeService извлекает рецепт с деталями (тип блюда, сложность, ингредиенты, шаги) из базы и возвращает клиенту.

Пользовательский сценарий представлен на рисунке 4.

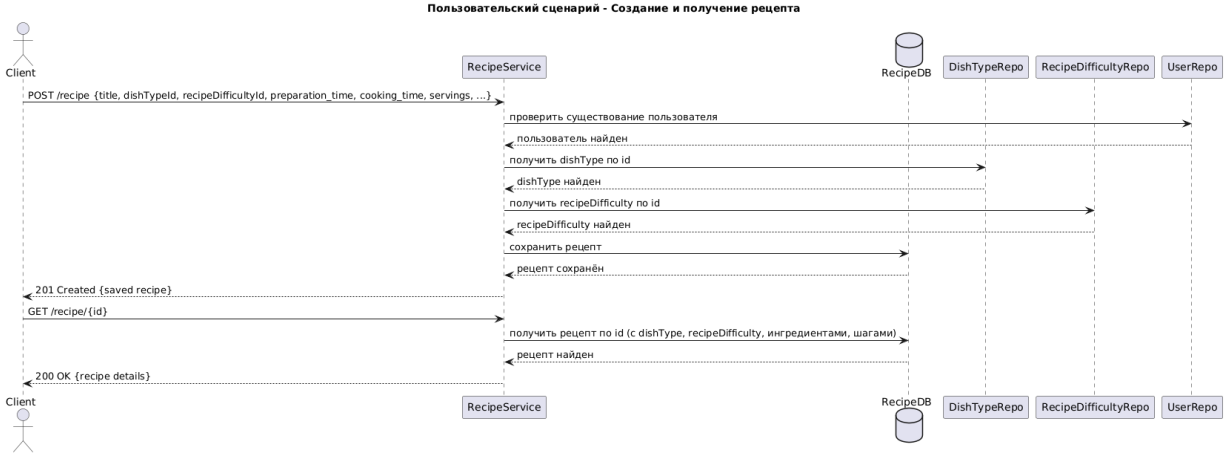

Рисунок 4 – Создание и получение рецепта

3) Пользовательский сценарий - Поставить лайк и сохранить рецепт

Пользователь отправляет запрос на лайк или сохранение рецепта через SocialService. Сервис проверяет существование пользователя (UserRepo) и рецепта (RecipeDB). После успешной проверки создаются записи Like или SavedRecipe в базе SocialDB. Сервис возвращает подтверждение клиенту с данными созданного лайка или сохранённого рецепта.

Пользовательский сценарий представлен на рисунке 5.

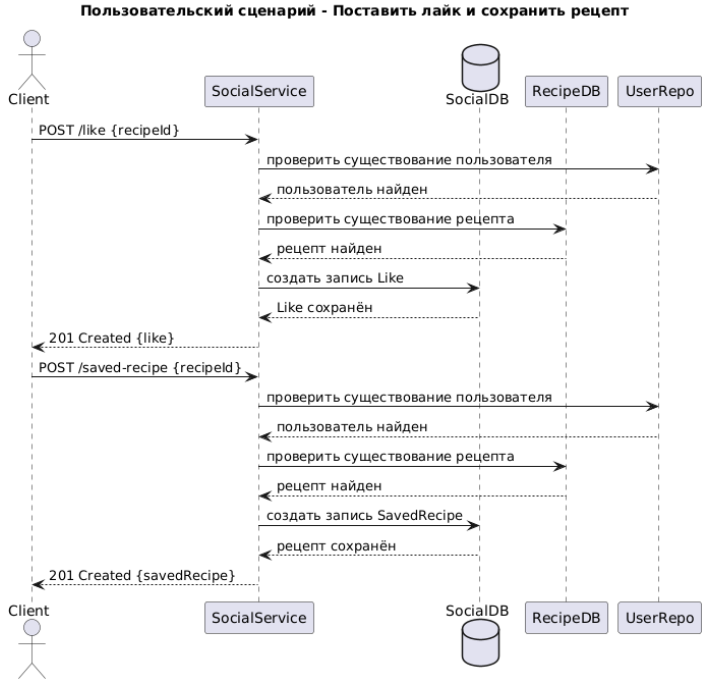

Рисунок 5 – Поставить лайк и сохранить рецепт

Вывод

В ходе выполнения домашней работы была спроектирована архитектура микросервисного приложения для обмена рецептами и ведения кулинарных блогов, включающая три основных сервиса: auth-service, recipe-service и social-service. Были определены их взаимосвязи, клиент-серверное взаимодействие и использование JWT для авторизации пользователей.

Созданы диаграммы компонентов, показывающие точки входа клиента, использование баз данных и зависимости сервисов от общих модулей. Также разработаны диаграммы баз данных каждого сервиса, отражающие структуру таблиц, связи между ними и кросс-сервисные зависимости, что обеспечивает целостность данных и поддерживает микросервисную архитектуру.

Особое внимание было уделено пользовательским сценариям, позволяющим пройти весь путь использования приложения: регистрация пользователя, создание и получение рецептов, а также взаимодействие с социальными функциями (лайки и сохранение рецептов). Для каждого сценария были описаны последовательности действий между клиентом, сервисами и базами данных, что наглядно демонстрирует логику работы приложения.

В результате работы были получены все необходимые диаграммы и описание сценариев, что позволяет полноценно представить структуру и функциональность приложения, а также использовать их для дальнейшей разработки, тестирования и документирования проекта.