САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №3

Выполнил:

Шалунов Андрей

Группа К3340

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Выделить самостоятельные модули в вашем приложении;

Провести разделение своего API на микросервисы (минимум, их должно быть 3);

Настроить сетевое взаимодействие между микросервисами.

Ход работы

1. Анализ существующего монолита

Выявлены основные домены: аутентификация, управление пользователями, объекты недвижимости, фотографии объектов, бронирования и обмен сообщениями.

Решено выделить минимум три микросервиса:

1. Auth-Service – отвечает за регистрацию, логин, выдачу JWT.
2. User-Service – хранит профили пользователей (таблица User).
3. Property-Service – CRUD для объектов недвижимости (Property, PropertyPhoto, Booking).
4. Photo-Service – загрузка и хранение URL фотографий.
5. Booking-Service – управление бронированиями.
6. Message-Service – обмен сообщениями между арендатором и владельцем.

2. Реализация микросервисов

Для каждого сервиса создано отдельное Node.js/Express (или Routing-Controllers) приложение со своей БД Postgres (через TypeORM).

В каждой службе определены свои сущности (@Entity), DTO и контроллеры.

Настроена валидация входящих DTO с помощью class-validator.

Сервис Auth-Service генерирует JWT, User-Service и Booking-Service защищены общим middleware, проверяющим токен.

3. Настройка API-Gateway

Выбран легковесный Gateway на базе Express + express-http-proxy.

В маршрутизации Gateway прописаны прокси-правила:

/api/auth → Auth-Service

/api/users → User-Service

/api/properties → Property-Service

/api/photos → Photo-Service

/api/bookings → Booking-Service

/api/messages → Message-Service

Gateway переписывает заголовки и при необходимости URL-prefix.

4. Настройка взаимодействия между сервисами

Сервис Message-Service для обогащения данных пользователей и бронирования делает HTTP-клиентские вызовы (axios) к User-Service и Booking-Service, подтягивает все необходимые поля и сохраняет в локальную БД (для кэша).

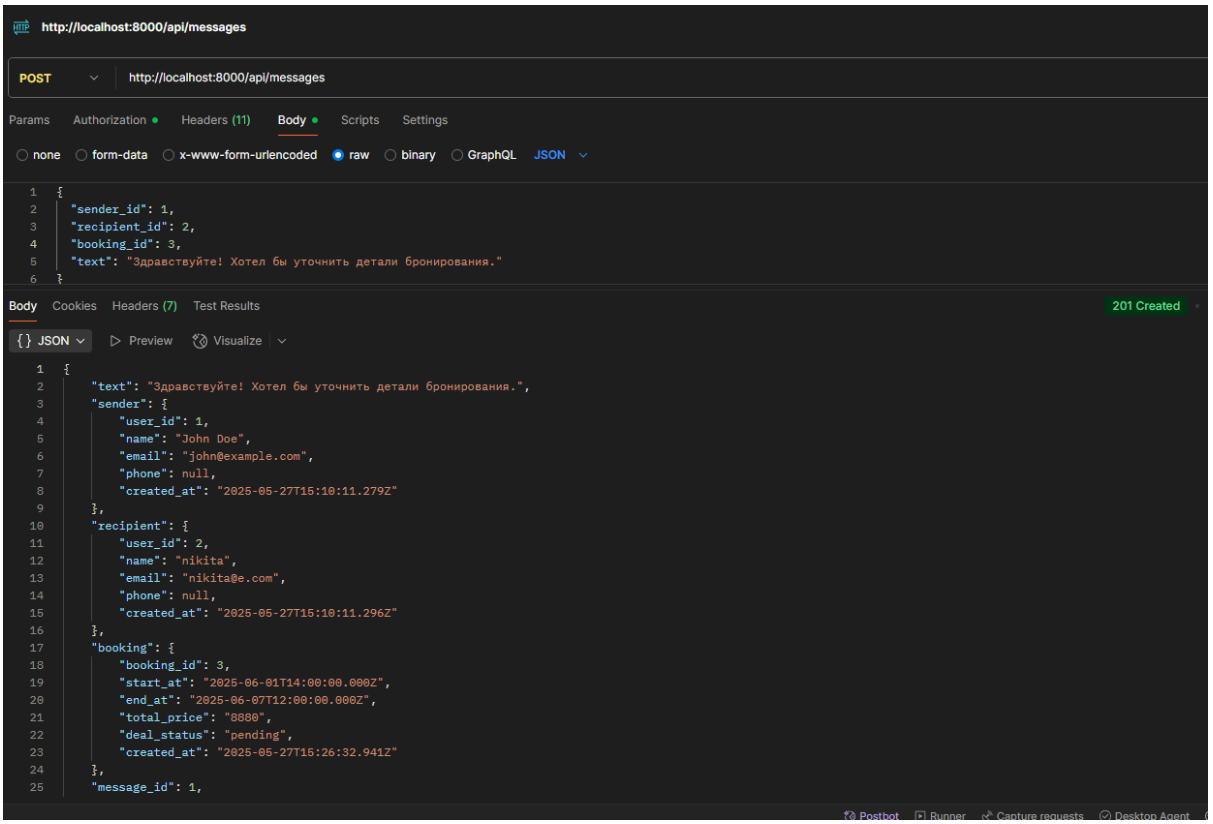
5. Тестирование

Запущены все сервисы и Gateway локально на разных портах.

С помощью Postman проверено:

1. создание пользователя, авторизация, получение токена;
2. CRUD-операции в Property, Photo, Booking, Message через единый http://localhost:8000/api/…
3. межсервисные вызовы (Message-Service корректно подтягивает пользователя и бронирование).

Пример создания сообщения между пользователями:



Вывод

Приложение разбито на независимые сервисы с чёткой ответственностью, что повысило модульность, упростило масштабирование и развертывание. Единый API-Gateway обеспечивает централизованную маршрутизацию и авторизацию, а межсервисные вызовы гарантируют согласованность данных. Такая архитектура делает систему более гибкой, надежной и удобной.