САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №3

Выполнил:

Кадникова Екатерина

Группа К3341

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. Реализация микросервисов

Было решено выделить следующие сервисы:

- Auth-service: работа с авторизацией/аутентификацией;
- User-service: работа и хранение с сущностью пользователя;
- Property-service: работа с сущностью недвижимости;
- Rental-service: работа с сущностью аренды недвижимости;
- Chat-service: работа с сущностью сообщения.

Для каждого сервиса было создано отдельное приложение со своими моделями, ДТО, сервисами, контроллерами, middleware-ы и роутами (структуру итогового проекта можно увидеть на Рисунке 1). Каждое приложение запускается на отдельном порту (например, чтобы отдельно отправить запрос сервису работы с недвижимостью можно воспользоваться http://localhost:3002/properties/, а чтобы напрямую отправить запрос для регистрации Auth сервису - http://localhost:3000/auth/).

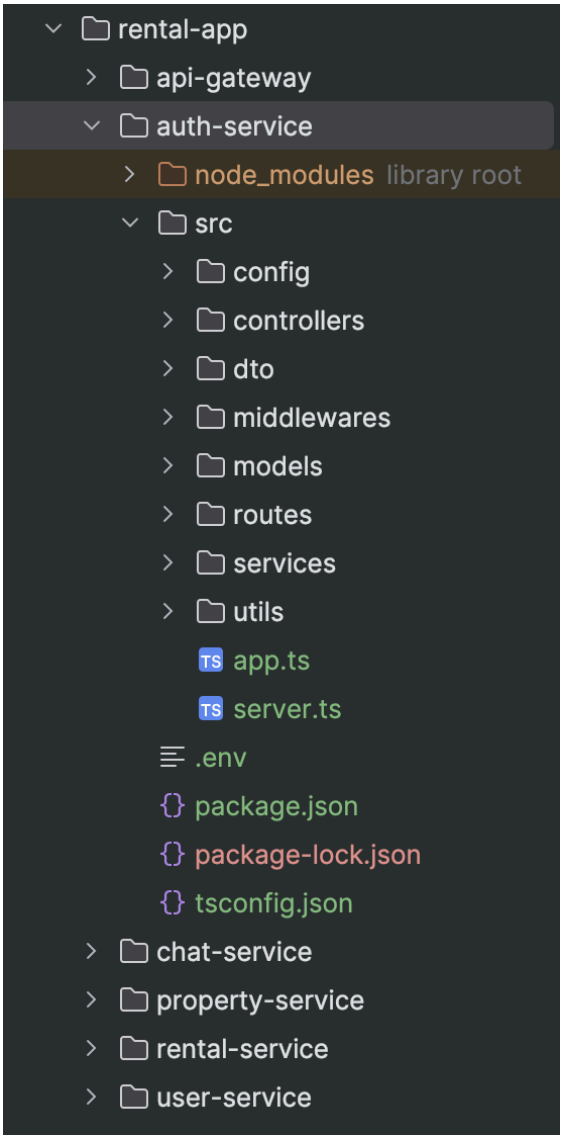

Рисунок 1 - Структура проекта после разделения на микросервисы

2. Настройка API-Gateway

Для единой точки входа в приложения был реализован API-Gateway (см. Листинг 1). В маршрутизации были выделены следующие прокси-правила:

- /auth - маршрутизация на Auth-service;
- /user - маршрутизация на User-service;
- /properties - маршрутизация на Property-service;
- /favorites - маршрутизация на Property-service;
- /messages - маршрутизация на Chat-service;

- /rentals - маршрутизация на Rental-service.

Листинг 1 - API-Gateway

```
import { Express } from 'express';
import { createProxyMiddleware } from 'http-proxy-middleware';
import SETTINGS from '../config/settings';

function createProxy(target: string) {
   return createProxyMiddleware({
       target,
       changeOrigin: true,
       pathRewrite: {
           '^/api': ''
       },
       onError: (err, req, res) => {
           console.error(err);
           if (!res.headersSent) {
                  res.status(502).json({ error: 'Bad gateway', details:
err.message });
           }
       },
       onProxyReq: (proxyReq, req) => {
           if (req.body) {
               const bodyData = JSON.stringify(req.body);
               proxyReq.setHeader('Content-Type', 'application/json');
                                       proxyReq.setHeader('Content-Length',
Buffer.byteLength(bodyData));
               proxyReq.write(bodyData);
           }
       }
   });
}

export function setupProxy(app: Express) {
   app.use('/auth', createProxy(SETTINGS.AUTH_SERVICE_URL));

   app.use('/users', createProxy(SETTINGS.USER_SERVICE_URL));

   app.use('/properties', createProxy(SETTINGS.PROPERTY_SERVICE_URL));
   app.use('/favorites', createProxy(SETTINGS.PROPERTY_SERVICE_URL));

   app.use('/messages', createProxy(SETTINGS.CHAT_SERVICE_URL));

   app.use('/rentals', createProxy(SETTINGS.RENTAL_SERVICE_URL));
}
```

В результате схема взаимодействия выглядит следующим образом: клиент делает запрос на API Gateway, Gateway перенаправляет запрос на нужны микросервис в соответствие с настройками маршрутизации, сервис обрабатывает запрос и возвращает ответ через Gateway клиенту.

3. Взаимодействие между микросервисаими

Микросервисы общаются между собой с помощью HTTP запросов через axios. Также для запросов, требующих авторизации, в заголовках передается оригинальный JWT-токен.

Для каждого сервиса реализована абстракция (например, user.client) для межсервисного взаимодействия, который отдельно отвечает за запрос на другой сервис, пробрасывает аутентификацию.

Вывод

В рамках работы в приложении были выделены самостоятельные модули, реализовано и проверено разделение API на микросервисы и сетевое взаимодействие между микросеврисами.