**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

# **Дисциплина:** Бэк-энд разработка

Отчет

Лабораторная работа № 3

Выполнил:

Гуторова Инна

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. Выделение самостоятельных модулей

Исходное монолитное приложение было разделено на следующие микросервисы:

1. **User Service** – отвечает за аутентификацию и управление пользователями.
2. **Travel Service** – управляет данными о путешествиях, достопримечательностях, маршрутах и типах поездок.
3. **Booking Service** – обрабатывает бронирования и избранные записи пользователей.
4. **API Gateway** – обеспечивает единую точку входа для клиентов и маршрутизацию запросов к соответствующим сервисам.

Получившаяся структура всего проекта:  
  
Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Каждый сервис (кроме api-gateway) в отдельности имеет вид:

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

2. Docker (подробнее в ЛР4)

Для каждого сервиса создан собственный Dockerfile и настроена оркестрация через docker-compose.yml.

**Основные параметры:**

* Каждый сервис работает в отдельном контейнере.
* Общая сеть travel-network для взаимодействия между сервисами.
* Использование PostgreSQL в качестве единой базы данных

3. Сетевое взаимодействие между сервисами

* Сервисы общаются через HTTP-запросы.
* API Gateway перенаправляет запросы на соответствующие сервисы (user-service, travel-service, booking-service).
* Для аутентификации используется JWT, проверяемый через user-service.

Для авторизации в travel-service и booking-service добавлен authClint. который отправляет запрос на верификацию токена в user-service.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

В user-service добавлен эндпоинт:

authRoutes.get('/verify-token', AuthController.verifyToken);

В связи с этим адаптированы методы в middleware в сервисах кроме user-service

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Похожим образом реализована верификация данных для booking-service, добавлены userClient и travelClient, в которых по id можно получить информацию из других сервисов.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

4. Описание сервисов

4.1. User Service

**Порт:** 3000  
**Функционал:**

* Регистрация и аутентификация пользователей.
* Управление профилями (CRUD).
* Генерация JWT-токенов.

**Зависимости:**

* PostgreSQL для хранения данных пользователей.

4.2. Travel Service

**Порт:** 3001  
**Функционал:**

* Управление достопримечательностями (/api/attractions).
* Работа с медиафайлами (/api/media).
* Управление маршрутами (/api/routes).
* Типы путешествий (/api/travel-types).
* Поездки (/api/trips).

**Зависимости:**

* PostgreSQL для хранения данных.
* Интеграция с user-service для проверки аутентификации.

4.3. Booking Service

**Порт:** 3002  
**Функционал:**

* Бронирования (/api/bookings).
* Избранные записи (/api/favorite).

**Зависимости:**

* PostgreSQL.
* Интеграция с user-service для аутентификации.

4.4. API Gateway

**Порт:** 3003  
**Функционал:**

* Единая точка входа для всех запросов.
* Маршрутизация:
  + /api/auth, /api/users → user-service.
  + /api/attractions, /api/trips и др. → travel-service.
  + /api/bookings, /api/favorite → booking-service.
* Обработка ошибок и CORS.

6. Вывод

В ходе работы монолитное приложение было успешно разделено на микросервисы. Настроено взаимодействие между ними через API Gateway.

**Итог:** Микросервисная архитектура реализована, все сервисы работают корректно.