Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

О Т Ч Е Т

по лабораторной работе №3
«Миграция написанного API на микросервисную архитектуру»
курса «Бэкенд разработка»

Выполнили:
Бахарева М.А., K3342
Привалов К.А., K3342
Проверил:
Добряков Д.И.

Санкт-Петербург,
2025

Содержание

Ход работы

Задание

Лабораторная работа включает в себя следующие задачи:

- выделение самостоятельных модулей в приложении;
- разделение API на микросервисы (минимум, их должно быть 3);
- настройка сетевого взаимодействие между микросервисами.

В процессе работы были выполнены следующие этапы:

- “Распилили” монолит на три доменных микросервиса: пользователи, недвижимость, чаты.
- Реализовали Gateway для работы с микросервисами через единую точку входа.

Основная часть

Разделение монолита

Для начала необходимо выделить самостоятельные модули, которые могут спокойно существовать без определенных моделей. В нашем случае это эндпоинты, связанные с пользователями (логин, регистрация и т.д.); с недвижимость (объекты недвижимости, запросы на бронь, жалобы); с коммуникацией между арендатором и арендодателем (чаты и сообщения).

Инициализировали Node.JS проект для каждого микросервиса и шлюза. Установили необходимые зависимости, выполнили правки от преподавателя с предыдущей ЛР, перенесли модели и другие компоненты из монолита в отдельные микросервисы. Структура проекта каждого микросервиса:

Users-microservice

```
.
├── package-lock.json
├── package.json
├── src
│   ├── app.ts
│   ├── config
│   │   ├── databaseConfig.ts
│   │   └── ProcessEnv.d.ts
│   ├── controllers
│   │   ├── BaseController.ts
│   │   └── UserController.ts
│   ├── entities
│   │   └── User.ts
│   ├── index.ts
│   ├── middleware
│   │   ├── checkJwt.ts
│   │   ├── errorHandler.ts
│   │   └── validator
│   │       ├── validatorChangePassword.ts
│   │       ├── validatorLogin.ts
│   │       ├── validatorRegister.ts
│   │       └── validatorUpdateUser.ts
│   ├── migrations
│   │   ├── 1746898783675-CreateTypeUserRole.ts
│   │   └── 1746898840634-CreateTableUser.ts
```

```
│   ├── routes
│   │   ├── index.ts
│   │   └── users.ts
│   ├── services
│   │   └── UserService.ts
│   ├── types
│   │   └── express
│   │       └── index.d.ts
│   └── utils
│       ├── jwt.ts
│       └── password.ts
└── tsconfig.json
```

Properties-service

```
.
├── package-lock.json
├── package.json
├── src
│   ├── app.ts
│   ├── config
│   │   ├── databaseConfig.ts
│   │   └── ProcessEnv.d.ts
│   ├── controllers
│   │   ├── BaseController.ts
│   │   ├── BookingRequestController.ts
│   │   ├── ComplaintController.ts
│   │   ├── FavoriteController.ts
│   │   ├── PropertyController.ts
│   │   └── PropertyImageController.ts
│   ├── entities
│   │   ├── BookingRequest.ts
│   │   ├── Complaint.ts
│   │   ├── Favorite.ts
│   │   ├── Property.ts
│   │   ├── PropertyImage.ts
│   │   └── UserRole.ts
│   ├── index.ts
│   ├── middleware
│   │   ├── checkJwt.ts
│   │   ├── errorHandler.ts
│   │   └── validator
│   │       ├── validatorBookingRequest.ts
│   │       ├── validatorComplaint.ts
│   │       └── validatorProperty.ts
│   ├── migrations
```

```
│   │   ├── 1746898803014-CreateTypeBookingRequestStatus.ts
│   │   ├── 1746898813338-CreateTypeComplaintStatus.ts
│   │   ├── 1746898848453-CreateTableProperty.ts
│   │   ├── 1746898867125-CreateTableFavorite.ts
│   │   ├── 1746898873399-CreateTableBookingRequest.ts
│   │   ├── 1746898884411-CreateTableComplaint.ts
│   │   └── 1746898897868-CreateTablePropertyImages.ts
│   ├── routes
│   │   ├── booking-requests.ts
│   │   ├── complaints.ts
│   │   ├── favorites.ts
│   │   ├── index.ts
│   │   ├── properties.ts
│   │   └── property-images.ts
│   ├── services
│   │   ├── BookingRequestService.ts
│   │   ├── ComplaintService.ts
│   │   ├── FavoriteService.ts
│   │   ├── PropertyImageService.ts
│   │   └── PropertyService.ts
│   ├── types
│   │   └── express
│   │       └── index.d.ts
│   └── utils
│       ├── jwt.ts
│       └── password.ts
├── tsconfig.json
└── uploads
    └── property-images
```

Communication-service

```
.
├── package-lock.json
├── package.json
├── src
│   ├── app.ts
│   ├── config
│   │   ├── databaseConfig.ts
│   │   └── ProcessEnv.d.ts
│   ├── controllers
│   │   ├── BaseController.ts
│   │   └── ChatController.ts
│   ├── entities
│   │   ├── Chat.ts
│   │   ├── Message.ts
```

```
│   │   └── UserRole.ts
│   ├── index.ts
│   ├── middleware
│   │   ├── checkJwt.ts
│   │   ├── errorHandler.ts
│   │   └── validator
│   │       └── validatorChat.ts
│   ├── migrations
│   │   ├── 1746898854512-CreateTableChat.ts
│   │   └── 1746898861382-CreateTableMessage.ts
│   ├── routes
│   │   ├── chats.ts
│   │   └── index.ts
│   ├── services
│   │   └── ChatService.ts
│   ├── types
│   │   └── express
│   │       └── index.d.ts
│   └── utils
│       ├── jwt.ts
│       └── password.ts
└── tsconfig.json
```

Реализация шлюза

Для того, чтобы было удобно работать с микросервисами реализуем единую точку входа - Gateway. Будем использовать концепцию Reverse Proxy, чтобы проксировать запросы клиента к сервисам.

Пропишем пути для каждого сервиса и целевой узел, на котором лежит сервис.

```
app.use('/api/users', createProxyMiddleware({
 target: 'http://localhost:3001',
 changeOrigin: true,
 pathRewrite: { '^/': '/api/users/' },
}));

app.use('/api/properties', createProxyMiddleware({
 target: 'http://localhost:3002',
 changeOrigin: true,
 pathRewrite: { '^/': '/api/properties/' },
```

```
}));

app.use('/api/favorites', createProxyMiddleware({
 target: 'http://localhost:3002',
 changeOrigin: true,
 pathRewrite: { '^/': '/api/favorites/' },
}));

app.use('/api/booking-requests', createProxyMiddleware({
 target: 'http://localhost:3002',
 changeOrigin: true,
 pathRewrite: { '^/': '/api/booking-requests/' },
}));

app.use('/api/complaints', createProxyMiddleware({
 target: 'http://localhost:3002',
 changeOrigin: true,
 pathRewrite: { '^/': '/api/complaints/' },
}));

app.use('/api/property-images', createProxyMiddleware({
 target: 'http://localhost:3002',
 changeOrigin: true,
 pathRewrite: { '^/': '/api/property-images/' },
}));

app.use('/api/chats', createProxyMiddleware({
 target: 'http://localhost:3003',
 changeOrigin: true,
 pathRewrite: { '^/': '/api/chats/' },
}));
```

Вывод

В ходе работы монолитное приложение было успешно декомпозировано на три отдельных сервиса: пользовательский, сервис недвижимости и сервис коммуникации. Для каждого микросервиса инициирован отдельный Node.js-проект с необходимыми зависимостями и перенесёнными компонентами из монолита, что упростило поддержку и масштабирование кода.