

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 3: Миграция написанного API на
микросервисную архитектуру

Выполнил:

Лазебный Всеволод

Группа К3344

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Нужно вынести из Api 3 микросервиса и пробросить связь между друг-другом.

Должно быть явное разделение на:

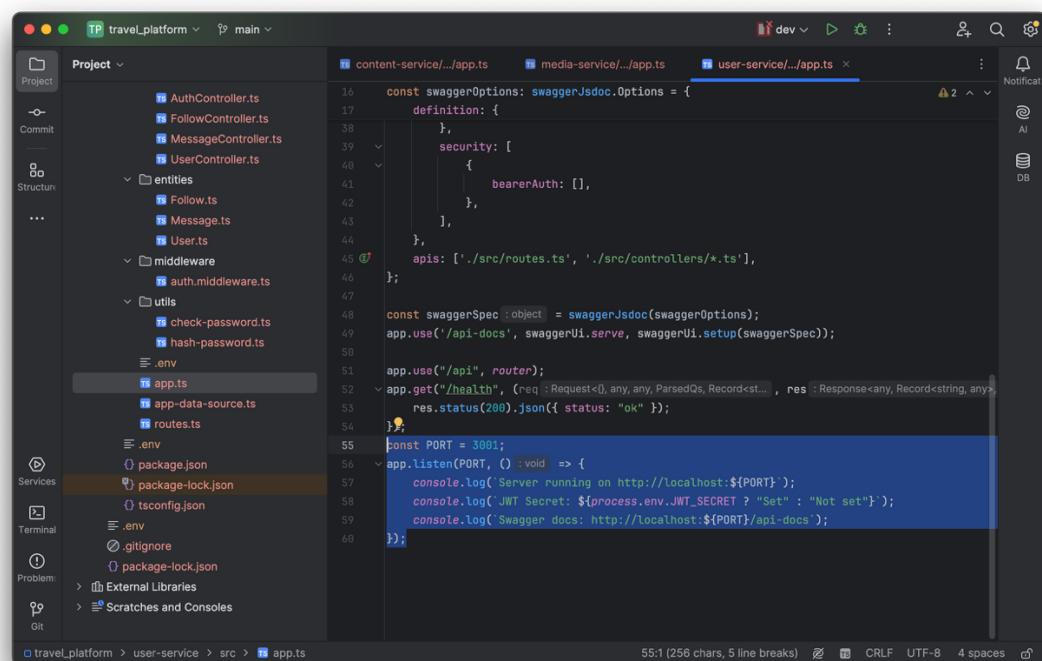
- модели
- контроллеры
- роуты

Ход работы

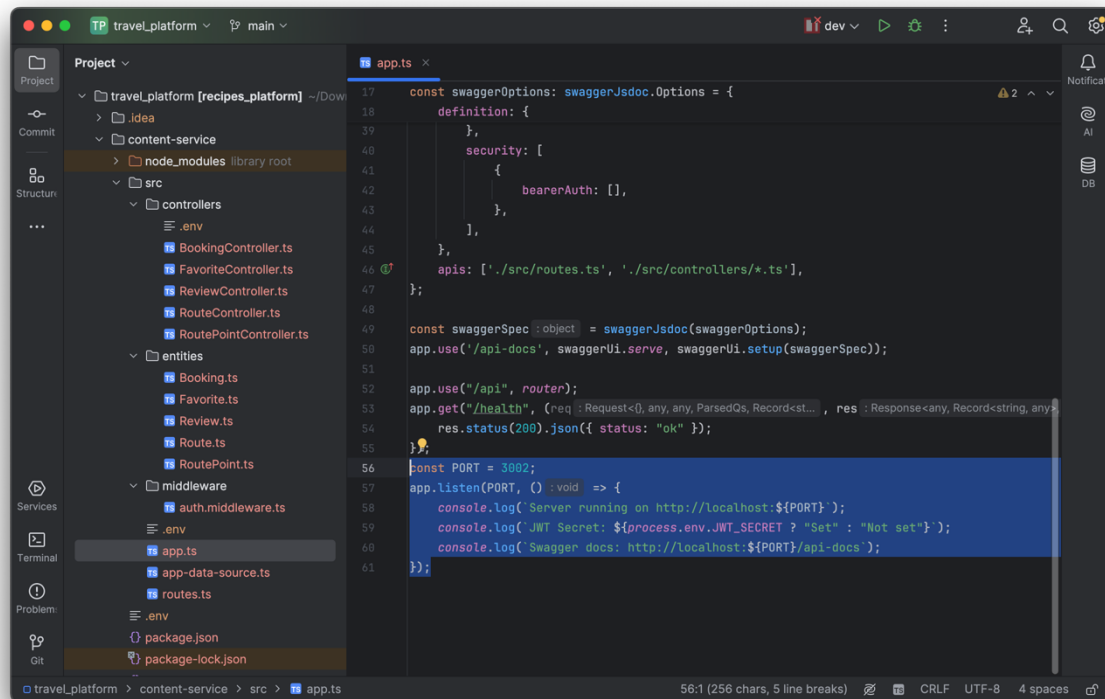
В моей работе смог выделить три микросервиса:

- user-service – сервис в котором происходит регистрация, вход, проброс запросов по поиску пользователей и возможность подписки на путешествия;
- content-service – сервис в котором происходит работа с основными сущностями и объектами;
- media-service – сервис в котором обрабатываются ссылки на медиа из интернета.

user-service:



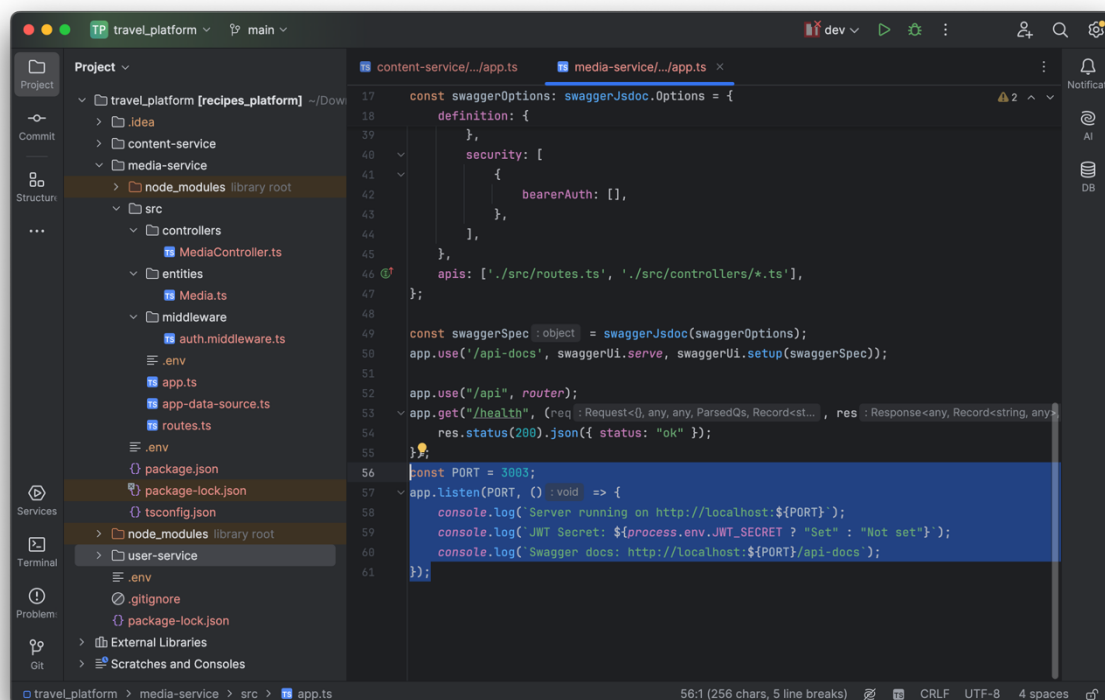
content-service:



The screenshot shows the VS Code editor with the 'content-service' project selected in the sidebar. The main editor displays the 'app.ts' file. The code defines Swagger options, sets up a security scheme for BearerAuth, and configures the application to listen on port 3002. The status bar at the bottom indicates the file is 56 lines long, containing 256 characters with 5 line breaks, using CRLF line endings, UTF-8 encoding, and 4 spaces for indentation.

```
17 const swaggerOptions: swaggerJsdoc.Options = {
18   definition: {
39     },
40     security: [
41       {
42         bearerAuth: [],
43       },
44     ],
45   },
46   apis: ['./src/routes.ts', './src/controllers/*.ts'],
47 };
48
49 const swaggerSpec: object = swaggerJsdoc(swaggerOptions);
50 app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerSpec));
51
52 app.use("/api", router);
53 app.get("/health", (req: Request<0, any, any, ParsedQs, Record<string, any>, res: Response<any, Record<string, any>, res.status(200).json({ status: "ok" });
54
55
56 const PORT = 3002;
57 app.listen(PORT, () => {
58   console.log(`Server running on http://localhost:${PORT}`);
59   console.log(`JWT Secret: ${process.env.JWT_SECRET ? "Set" : "Not set"}`);
60   console.log(`Swagger docs: http://localhost:${PORT}/api-docs`);
61 });
```

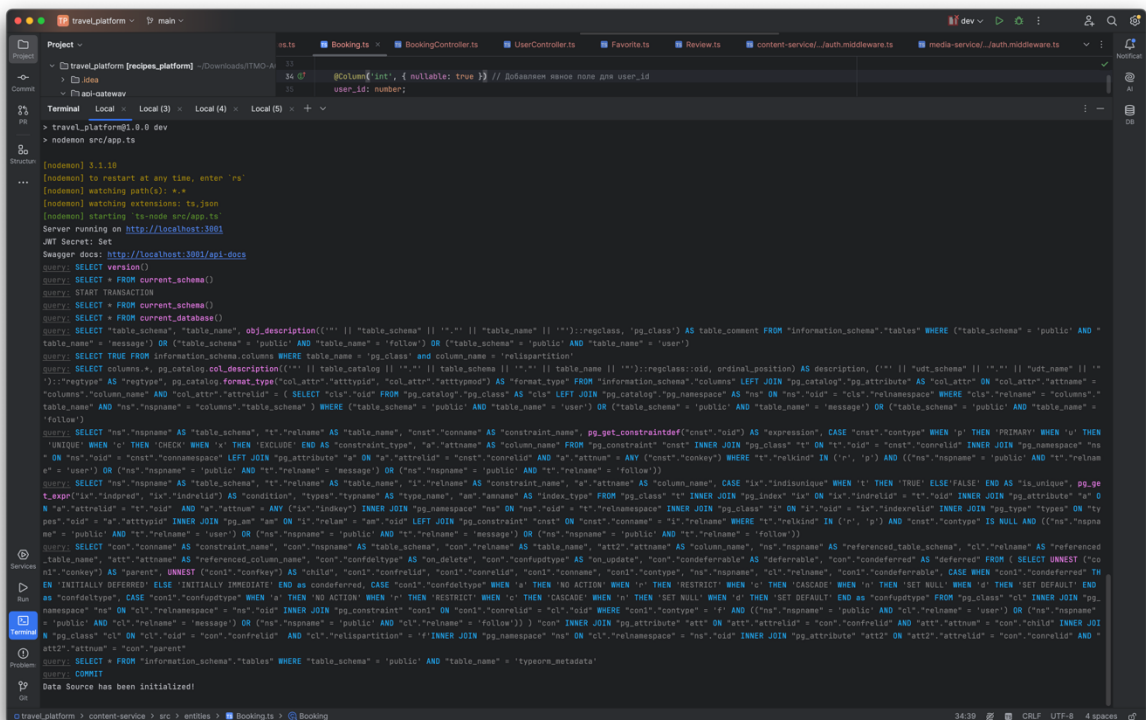
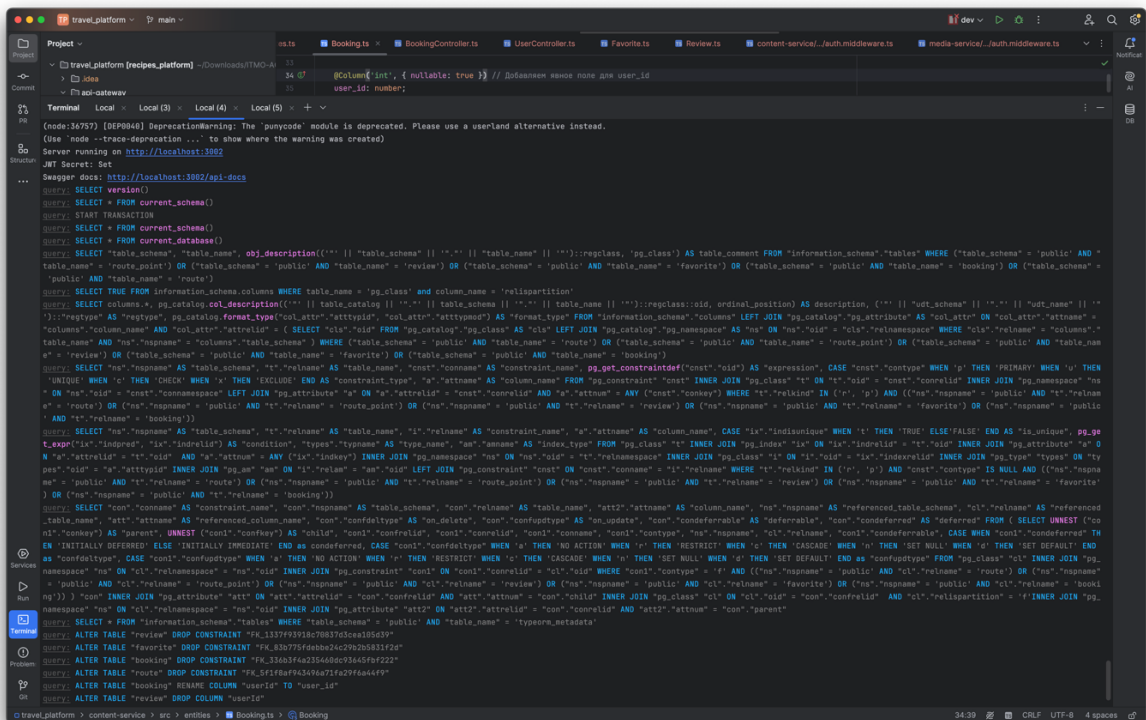
Media-service:



The screenshot shows the VS Code editor with the 'media-service' project selected in the sidebar. The main editor displays the 'app.ts' file. The code defines Swagger options, sets up a security scheme for BearerAuth, and configures the application to listen on port 3003. The status bar at the bottom indicates the file is 56 lines long, containing 256 characters with 5 line breaks, using CRLF line endings, UTF-8 encoding, and 4 spaces for indentation.

```
17 const swaggerOptions: swaggerJsdoc.Options = {
18   definition: {
39     },
40     security: [
41       {
42         bearerAuth: [],
43       },
44     ],
45   },
46   apis: ['./src/routes.ts', './src/controllers/*.ts'],
47 };
48
49 const swaggerSpec: object = swaggerJsdoc(swaggerOptions);
50 app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerSpec));
51
52 app.use("/api", router);
53 app.get("/health", (req: Request<0, any, any, ParsedQs, Record<string, any>, res: Response<any, Record<string, any>, res.status(200).json({ status: "ok" });
54
55
56 const PORT = 3003;
57 app.listen(PORT, () => {
58   console.log(`Server running on http://localhost:${PORT}`);
59   console.log(`JWT Secret: ${process.env.JWT_SECRET ? "Set" : "Not set"}`);
60   console.log(`Swagger docs: http://localhost:${PORT}/api-docs`);
61 });
```

Каждый микросервис создал свои хранилища в одной базе данных



В ходе этой лабораторной были реализованы 3 микросервиса и задана связь между ними. Каждый сервис получил свой порт, что поможет в дальнейшем.