

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

Отчет

Лабораторная работа №1

Выполнил:

Егоров Мичил

Группа

K33401

Проверил:

Добряков Д. И.

Санкт-Петербург

2022 г.

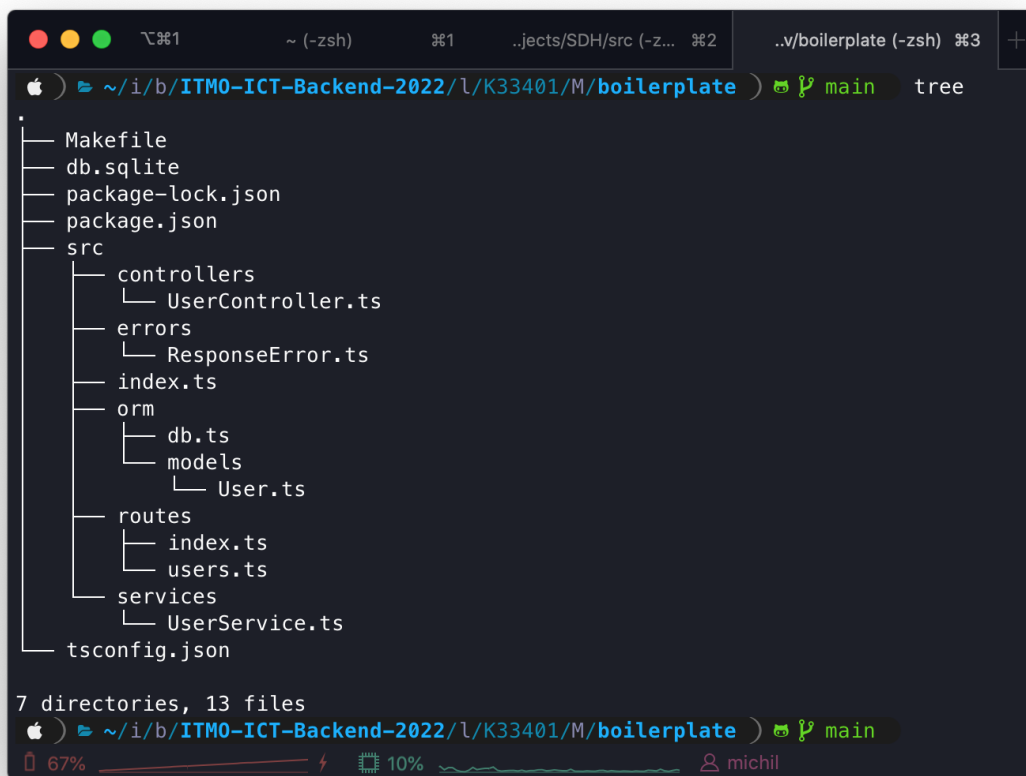
## Задача

Нужно написать свой boilerplate на express + sequelize / TypeORM + typescript. Должно быть явное разделение на:

- модели
- контроллеры
- роуты
- сервисы для работы с моделями (реализуем паттерн “репозиторий”)

## Ход работы

### 1. Структура приложения

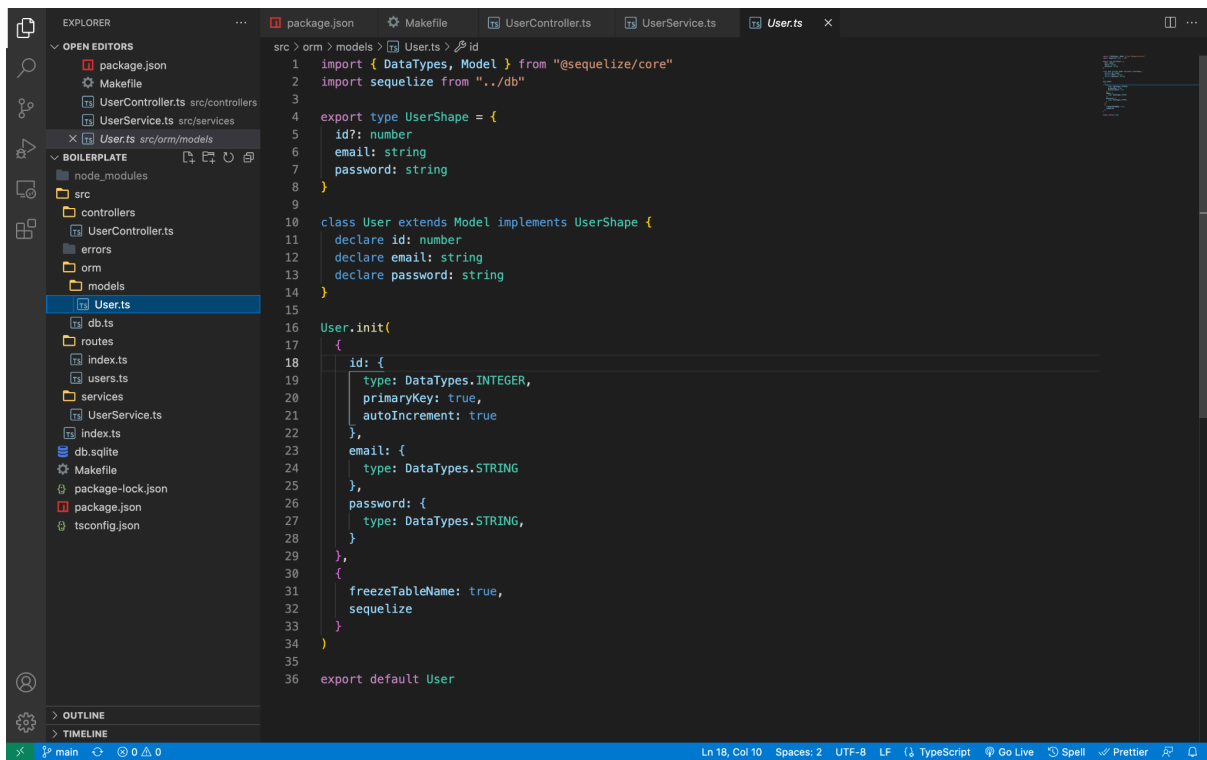


```
tree
.
├── Makefile
├── db.sqlite
├── package-lock.json
├── package.json
├── src
│   ├── controllers
│   │   └── UserController.ts
│   ├── errors
│   │   └── ResponseError.ts
│   ├── index.ts
│   ├── orm
│   │   ├── db.ts
│   │   ├── models
│   │   │   └── User.ts
│   ├── routes
│   │   ├── index.ts
│   │   └── users.ts
│   └── services
│       └── UserService.ts
└── tsconfig.json

7 directories, 13 files
```

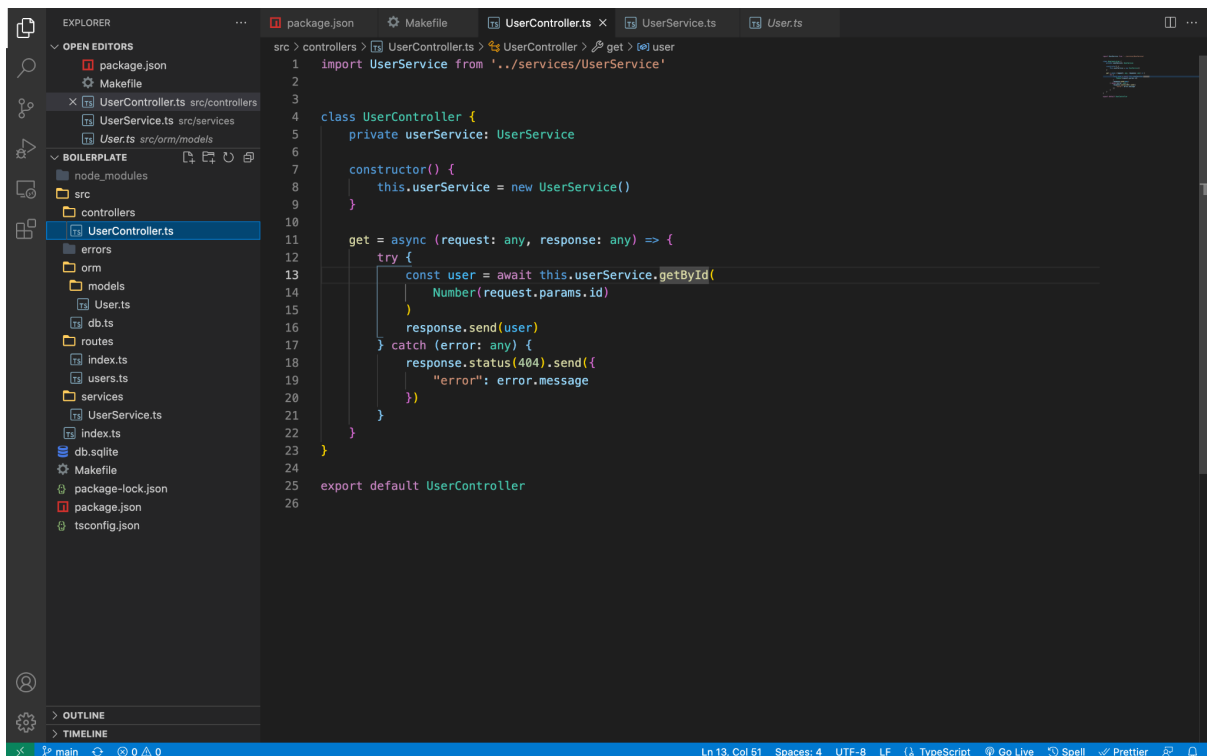
The screenshot shows a terminal window with the command `tree` executed in the directory `~/i/b/ITMO-ICT-Backend-2022/l/K33401/M/boilerplate`. The output displays the directory structure of the boilerplate, including files like `Makefile`, `db.sqlite`, `package-lock.json`, `package.json`, and a `src` directory containing `controllers`, `errors`, `index.ts`, `orm` (with `db.ts` and `models`), `routes`, and `services`. The `orm/models` directory contains `User.ts`. The `routes` directory contains `index.ts` and `users.ts`. The `services` directory contains `UserService.ts`. The terminal also shows `tsconfig.json` at the root of the `src` directory. The status bar at the bottom indicates 67% battery, 10% CPU usage, and the user `michil`.

### 2. Модели



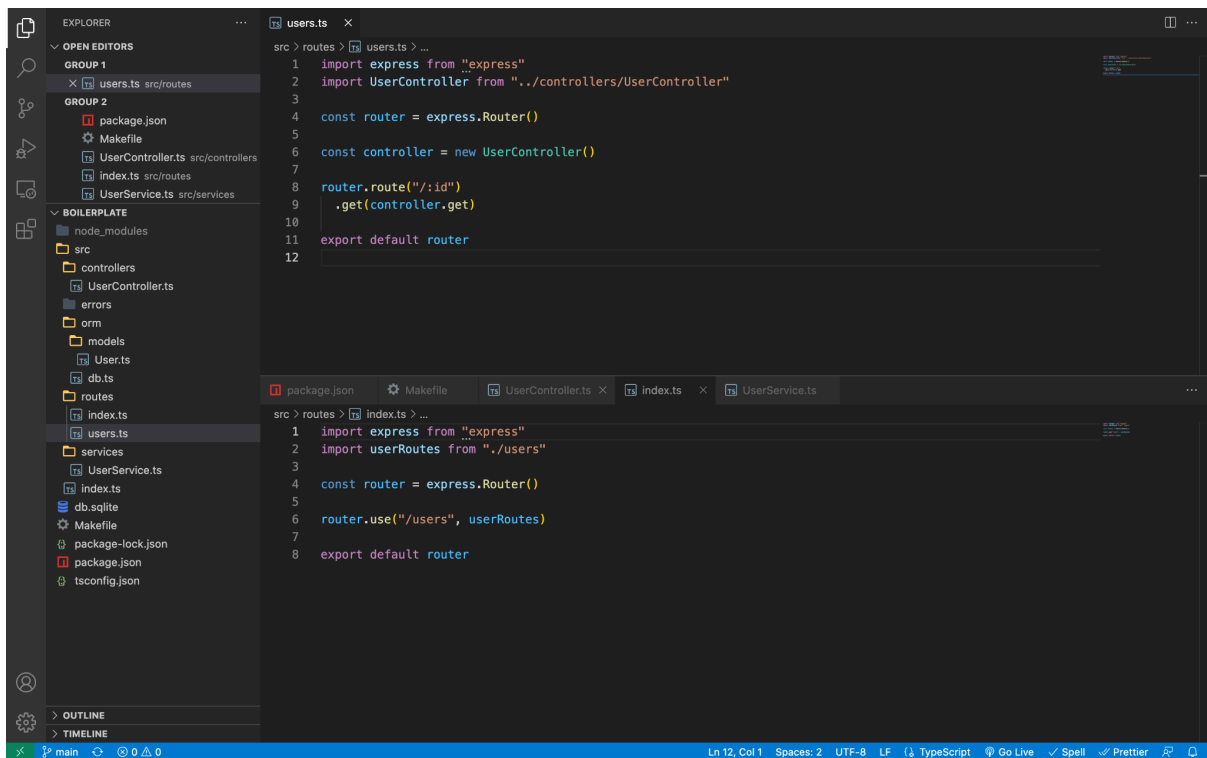
```
src > orm > models > User.ts > id
1 import { DataTypes, Model } from "@sequelize/core"
2 import sequelize from "../db"
3
4 export type UserShape = {
5   id?: number
6   email: string
7   password: string
8 }
9
10 class User extends Model implements UserShape {
11   declare id: number
12   declare email: string
13   declare password: string
14 }
15
16 User.init({
17   {
18     id: {
19       type: DataTypes.INTEGER,
20       primaryKey: true,
21       autoIncrement: true
22     },
23     email: {
24       type: DataTypes.STRING
25     },
26     password: {
27       type: DataTypes.STRING,
28     }
29   },
30   {
31     freezeTableName: true,
32     sequelize
33   }
34 })
35
36 export default User
```

### 3. Контроллеры

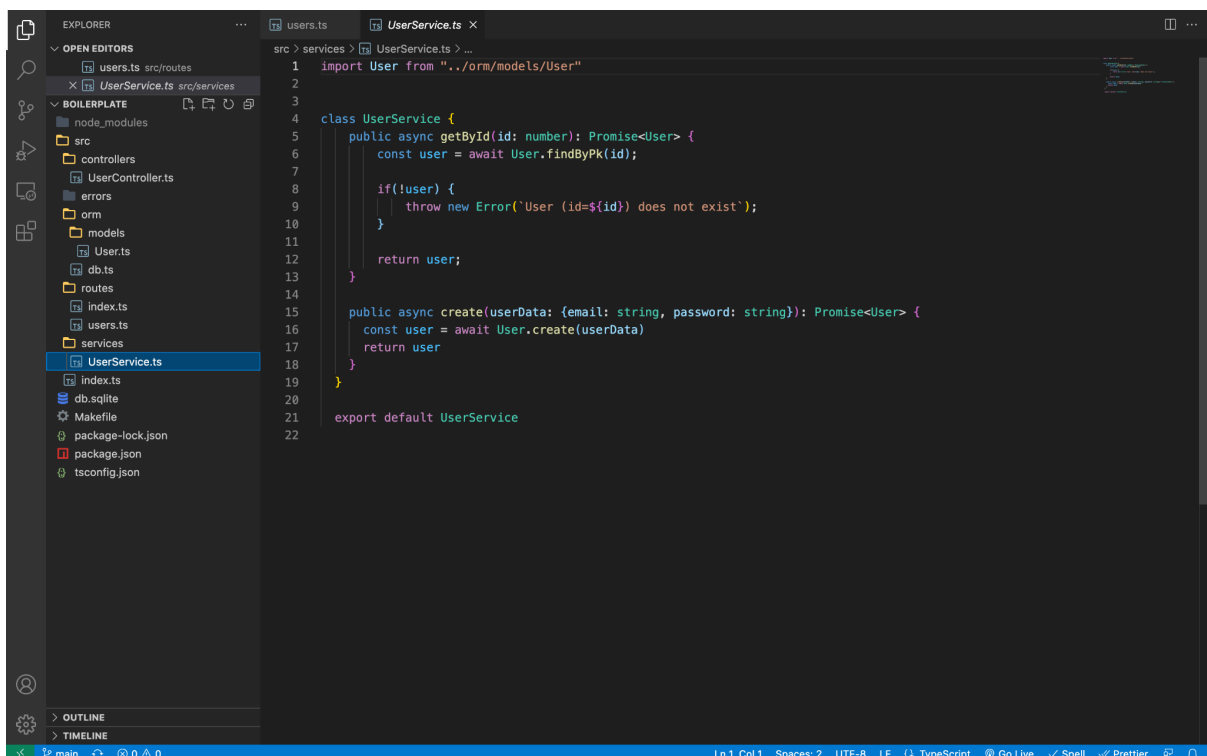


```
src > controllers > UserController.ts > UserController > get > user
1 import UserService from '../services/UserService'
2
3
4 class UserController {
5   private userService: UserService
6
7   constructor() {
8     this.userService = new UserService()
9   }
10
11   get = async (request: any, response: any) => {
12     try {
13       const user = await this.userService.getById(
14         Number(request.params.id)
15       )
16       response.send(user)
17     } catch (error: any) {
18       response.status(404).send({
19         "error": error.message
20       })
21     }
22   }
23 }
24
25 export default UserController
26
```

### 4. Routes



## 5. Сервисы



## Вывод

В ходе данной лабораторной работы мы написали свой шаблон backend-приложения на node. Реализовали паттерн репозиторий для

взаимодействия с базой данных и описанными моделями. Научились писать контроллеры для end-point-ов.