

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

Факультет Инфокоммуникационных технологий

Образовательная программа 09.03.03 Мобильные и сетевые технологии

О Т Ч Е Т

по лабораторной работе № 1
по дисциплине «Бэк-энд разработка»

Выполнил:

Мосин З. И.

Проверил:

Добряков Д. И.



УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2024

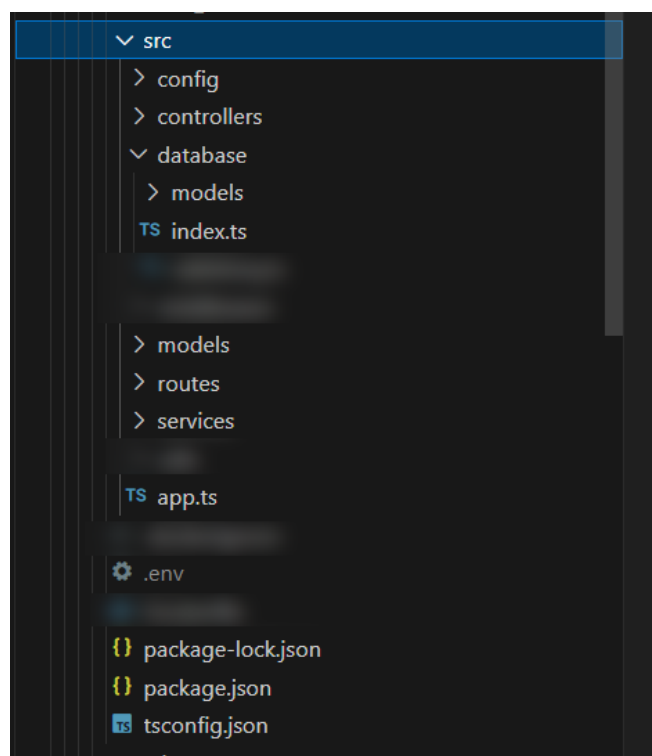
Цели работы

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

В начале работы инициализировал репозиторий. Добавил нужные зависимости, настроил Prettier для удобного форматирования кода, поработал с tsconfig.json файлом, подробно настроив TypeScript. Определил структуру файлов. Проект запускается через app.ts файл, а остальной код содержится в директории src, и разделён по принципам MVC.



В главном файле подключаем Express JS и Sequelize. С помощью bodyParser активируем работу энд поинтов.

```
labs > K33402 > Мосин_Захар > lab1 > app > src > TS app.ts > ...
1  import express from 'express'
2
3  import sequelize from './database/index'
4  import bodyParser from 'body-parser'
5
6  const PORT = Number(process.env.PORT) || 5000
7
8  const app = express()
9
10 app.use(bodyParser.json())
11
12 app.listen(PORT, () => {
13   sequelize // to not delete after compilation
14   console.log(`Server is running on port ${PORT}`)
15 })
16
```

При подключении базы данных используем переменные окружения.

```
10  dotenv.config()
11  console.log(process.env.DB_NAME)
12  console.log(process.env.DB_USERNAME)
13  console.log(process.env.DB_PASSWORD)
14  console.log(process.env.DB_HOST)
15  const sequelize = new Sequelize({
16    database: process.env.DB_NAME,
17    username: process.env.DB_USERNAME,
18    password: process.env.DB_PASSWORD,
19    dialect: 'postgres',
20    storage: 'db.postgres',
21    host: process.env.DB_HOST,
22    port: 5432,
23    repositoryMode: true,
24    logging: console.log,
25  })
26
27  sequelize.addModels([User, Book, UserHasBook, Exchange])
28
29  sequelize.sync().then(() => {
30    console.log('sync sequelize')
31  })
```

Создаём модель пользователя, используя sequelize-typescript и декораторы.

```
TS app.ts TS index.ts TS User.ts X
labs > K33402 > Мосин_Захар > lab1 > app > src > database > mod
1  import {
2    AutoIncrement,
3    Column,
4    DataType,
5    Model,
6    PrimaryKey,
7    Table,
8    Unique,
9  } from 'sequelize-typescript'
10
11  @Table({
12    tableName: 'User',
13  })
14  class User extends Model {
15    @PrimaryKey
16    @AutoIncrement
17    @Column(DataType.INTEGER)
18    declare id: number
19
20    @Column(DataType.STRING)
21    declare name: string
22
23    @Unique
24    @Column(DataType.STRING)
25    declare email: string
26
27    @Unique
28    @Column(DataType.STRING)
29    declare password: string
30  }
31
32  export default User
33
```

Ошибки обрабатываем внутри Controllers, а бизнес логика находится внутри Services и соответствует паттерну Repository.

```
labs > K33402 > Мосин_Захар > lab1 > app > src > controllers > TS UserController.ts > ...
1  import { Request, Response } from 'express'
2  import jwt, { JwtPayload } from 'jsonwebtoken'
3
4  import UserService from '../services/UserService'
5  import handleError from '../utils/handleError'
6
7  export default {
8    async getUserById(req: Request, res: Response) {
9      try {
10        const id = Number(req.params.id)
11        const user = await UserService.getUserById(id)
12        return res.status(201).json(user)
13      } catch (error) {
14        return handleError({ res, error, code: 500 })
15      }
16    },
17
18    async getAllUsers(req: Request, res: Response) {
19      try {
20        const users = await UserService.getAllUsers()
21        return res.status(201).json(users)
22      } catch (error) {
23        return res.status(500).json(error)
24      }
25    },
26  },
```

```

TS UserService.ts X
labs > K33402 > Мосин_Захар > lab1 > app > src > services > TS UserService.ts > ...
1  import sequelize from '../database/index'
2  import User from '../database/models/User'
3  import serviceHandleError from '../utils/serviceHandleError'
4
5  const userRepository = sequelize.getRepository(User)
6
7  type UserDataTypes = {
8      name: string
9      email: string
10     password: string
11 }
12
13 class UserService {
14     static async getUserById(id: number) {
15         return userRepository.findByPk(id)
16     }
17
18     static async getAllUsers() {
19         return userRepository.findAll()
20     }
21
22     static async createUser(userData: UserDataTypes) {
23         const { name, email, password } = userData
24         return userRepository.create({ name, email, password })
25     }
26
27     static async updateUser(id: number, userData: any) {
28         const user = await userRepository.findByPk(id)
29
30         if (!user) {
31             return serviceHandleError({ message: 'Пользователь не найден' })
32         }
33     }
34 }

```

ВЫВОД

В процессе работы создали boilerplate репозиторий с ExpressJs и sequelize, получили навыки составления структуры серверного проекта.