

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа № 1

Выполнил:

Казанков Илья К33402

Проверил: Добряков Д. И.

Санкт-Петербург

2024 г.

Задача

Нужно написать свой boilerplate на express + sequelize / TypeORM + typescript.

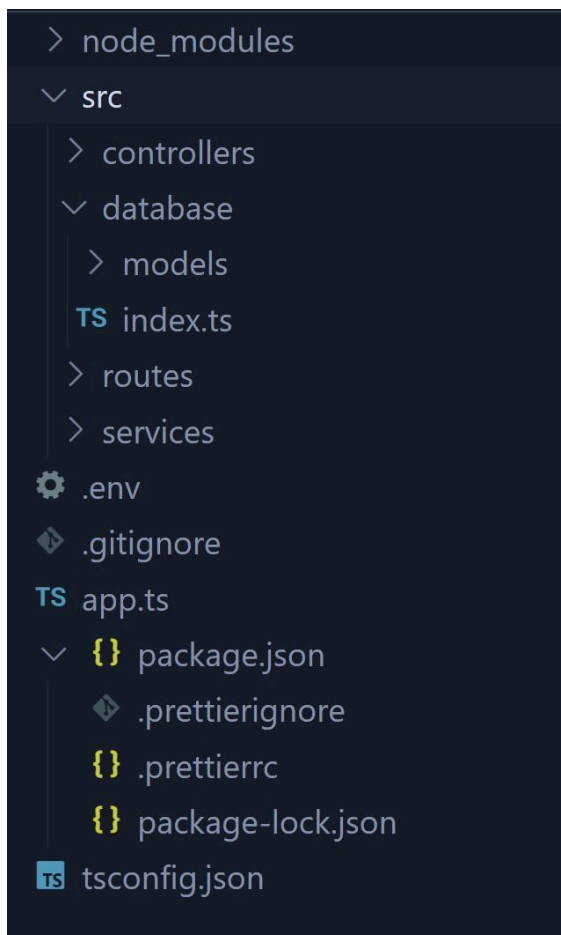
Должно быть явное разделение на:

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

Ход работы

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.

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

При подключении базы данных используем переменные окружения.

```
src > database > TS index.ts > ...
1  import { Sequelize } from 'sequelize-typescript'
2
3  import User from './models/User'
4
5  const sequelize = new Sequelize({
6     database: process.env.DB_NAME || 'database123',
7     username: process.env.DB_USER || 'username123',
8     password: process.env.DB_PASSWORD || 'password123',
9     dialect: 'postgres',
10    host: process.env.DB_HOST || 'localhost',
11    port: Number(process.env.DB_PORT || '5432'),
12    repositoryMode: true,
13    logging: console.log,
14 })
15
16 sequelize.addModels([User])
17
18 sequelize.sync().then(() => {
19     console.log('sync sequelize')
20 })
```

Создаём модель пользователя, используя sequelize-typescript и декораторы

```
src > database > models > TS User.ts > ...
1  import { Column, Model, Table } from 'sequelize-typescript'
2
3  @Table
4  class User extends Model<User> {
5      @Column
6      username!: string
7
8      @Column({ unique: true })
9      email!: string
10
11     @Column({ unique: true })
12     password!: string
13 }
14
15 export default User
16
```

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

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

```
src > services > TS UserService.ts > ...
1  import sequelize from 'src/database'
2
3  import User from '../database/models/User'
4
5  const userRepository = sequelize.getRepository(User)
6
7  export default class UserService {
8      static async getAllUsers() {
9          return userRepository.findAll()
10     }
11
12     static async getUserById(id: number) {
13         return userRepository.findByPk(id)
14     }
15
16     static async createUser(userData: any) {
17         return userRepository.create(userData)
18     }
19
20     static async updateUser(id: number, userData: any) {
21         const user = await userRepository.findByPk(id)
```

Вывод

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