

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №1

Выполнил:

Дорофеева Арина

Группа к33401

Проверил:

Добряков Д. И.

Санкт-Петербург

2022 г.

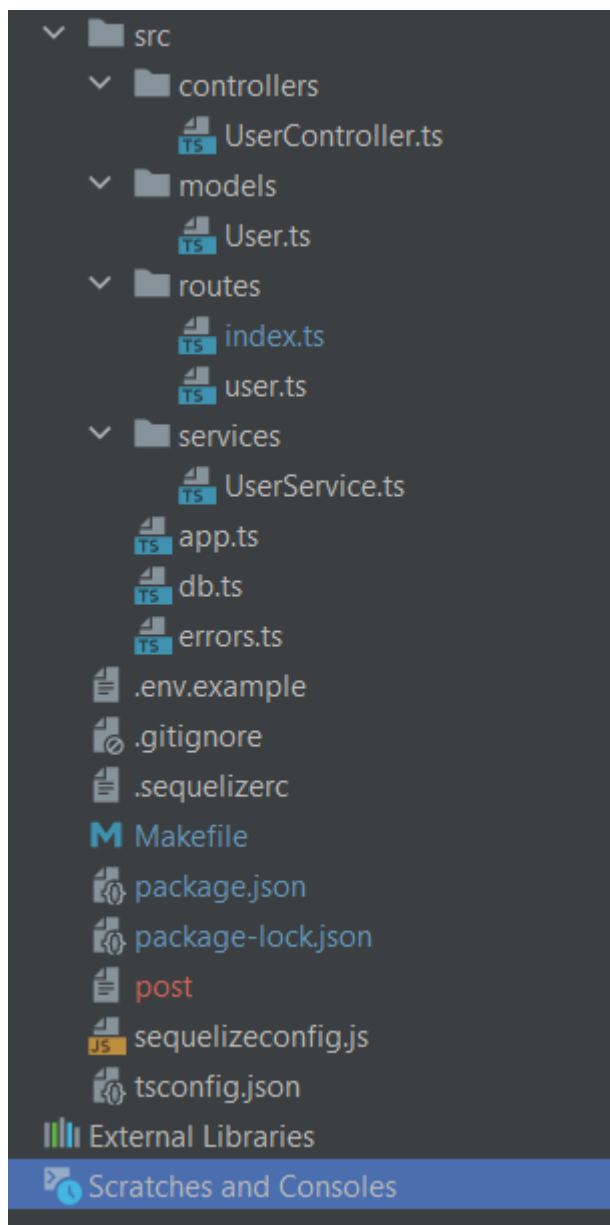
Задача

Написать свой boilerplate на express + sequelize + typescript.

Должно быть явное разделение на:

1. модели
2. контроллеры
3. роуты
4. сервисы для работы с моделями (реализуем паттерн “репозиторий”)

Ход работы



Контроллеры:

```
UserController.ts
1 import type { UserShape } from "../models/User"
2 import type { Request, Response } from "express"
3 import AppError, { handleGenericError } from "../errors"
4 import UserService from "../services/UserService"
5
6 type ResponseOrError<T> = Response<T | AppError>
7
8 class UserController {
9   private userService: UserService
10
11   public constructor() {
12     this.userService = new UserService()
13   }
14
15   public get = async (req: Request, res: ResponseOrError<UserShape>) => {
16     const { id } = req.params
17     this.userService
18       .get(Number(id)) Promise<User>
19       .then(user => res.status( code: 200 ).send(user)) Promise<Response<AppError | UserShape>>
20       .catch(e => handleGenericError(res, e))
21   }
22
23   public post = async (req: Request, res: ResponseOrError<UserShape>) => {
24     const { user } = req.body
25     this.userService
26       .create(user as UserShape) Promise<User>
27       .then(user => res.status( code: 200 ).send(user)) Promise<Response<AppError | UserShape>>
28       .catch(e => handleGenericError(res, e))
29   }
30 }
31
32 export default UserController
```

Модели:

```
User.ts
1 import { DataTypes, Model } from "@sequelize/core"
2 import bcrypt from "bcrypt"
3 import sequelize from "../db"
4
5 export type UserShape = {
6   id?: number
7   username: string
8   password: string
9 }
```

```

10
11 class User extends Model implements UserShape {
12     declare id: number
13     declare username: string
14     declare password: string
15 }
16
17 User.init(
18     attributes: {
19         id: {
20             type: DataTypes.INTEGER,
21             primaryKey: true,
22             autoIncrement: true
23         },
24         username: {
25             type: DataTypes.STRING
26         },
27         password: {
28             type: DataTypes.STRING,
29             set(value: string) {
30                 this.setDataValue("password", bcrypt.hashSync(value, saltOrRounds: 10))
31             }
32         }
33     },
34     options: {
35         freezeTableName: true,
36         sequelize
37     }
38 )
39
40 export default User
41

```

Роуты:

```

index.ts x user.ts x
1 import express from "express"
2 import userRoutes from "./user"
3
4 const router = express.Router()
5
6 router.use("/user", userRoutes)
7
8 export default router
9

```

```

1 import express from "express"
2 import UserController from "../controllers/UserController"
3
4 const router = express.Router()
5
6 const controller = new UserController()
7
8 router.route( prefix: "/:id" )
9   .get(controller.get)
10
11 router.route( prefix: "/" )
12   .post(controller.post)
13
14 export default router

```

Сервисы:

```

1 import User from "../models/User"
2 import type { UserShape } from "../models/User"
3
4 class UserService {
5   public async get(id: number): Promise<User> {
6     if (id !== +id) throw new Error("User id must be an integer.")
7     let user
8     try {
9       user = await User.findByPk(id)
10    } catch (e) {
11      throw new Error(`User with id=${ id } not found.`)
12    }
13    return user as User
14  }
15
16   public async create(userData: UserShape): Promise<User> {
17     const user = await User.create(userData)
18     return user
19   }
20 }
21
22 export default UserService

```

Вывод

В ходе работы я собрала шаблон boilerplate на express + sequelize + typescript, который буду использовать для создания бэкенд-приложения.