САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет

Лабораторная работа №1

Выполнил:

Жигалова Анастасия
K33392

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

1. npm init – инициализируем модуль

```
D:\ITMO-ICT-Backend-2024\labs\K33392\Zhigalova Anastasia>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\ITMO-ICT-Backend-2024\labs\K33392\Zhigalova Anastasia\package.json:

{
  "name": "lr1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes) yes
```

2. npm i dotenv express sequelize-typescript sqlite3 @types/express – устанавливаем зависимости

```
D:\ITMO-ICT-Backend-2024\labs\K33392\Zhigalova Anastasia>npm i dotenv express sequelize-typescript sqlite3 @types/express
npm WARN deprecated @npmcli/move-file@1.1.2: This functionality has been moved to @npmcli/fs

added 222 packages, and audited 223 packages in 30s

25 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

3. tsconfig.json

tsconfig.json > ...

```json
{
  "compilerOptions": {
    "module": "NodeNext",
    "moduleResolution": "NodeNext",
    "target": "ES2020",
    "outDir": "dist",
    "sourceMap": true,
    "experimentalDecorators": true,
    "emitDecoratorMetadata": true
  },
  "include": ["src/**/*"]
}
```

4. package.json

```json
"scripts": {
  "build": "npx tsc",
  "start": "node dist/index.js",
  "dev": "npx tsc & node dist/index.js",
  "test": "echo \"Error: no test specified\" && exit 1"
},
```

Добавим, чтобы запускать по команде **npm run dev**

5. models/users/User.ts - добавим модель юзера

```typescript
import { Table, Column, Model, Unique, AllowNull } from 'sequelize-typescript';

@Table({
  tableName: 'Users'
})
export class User extends Model<User> {
    @Column
    name: string

    @Unique
    @Column
    email: string

    @AllowNull(false)
    @Column
    password: string
}
```

6. services/users/User.ts – прописываем методы, у меня – создание нового юзера и получение списка всех юзеров.

```
import { User } from '../../models/users/User.js'

export class UserRepository {
    async create(userData: any): Promise<User> {
        try {
            const user = await User.create(userData);
            return user;
        } catch (error) {
            console.error("Ошибка при создании пользователя:", error);
            throw new Error("Не удалось создать пользователя: " + error.message);
        }
    }


    async get(): Promise<User[]> {
        try {
            const users = await User.findAll()
            return users
        } catch (error) {
            console.error(error)
            throw new Error("Ошибка при получении списка пользователей: " + error.message)
        }
    }
}
```

7. controllers/users/User.ts – контроллеры

```
src > controllers > users > TS User.ts > ...
import { Request, Response } from "express"
import { UserRepository } from "../../services/users/User.js"


export class UserController {
    service: UserRepository

    constructor() {
      this.service = new UserRepository()
    }

    get = async (req: Request, res: Response) => {
        try {
            const user = await this.service.get()
            res.send(user)
        } catch (error) {
            console.error(error.message)
            res.status(404).send({ error: error.message })
        }
    };

    post = async (req: Request, res: Response) => {
      try {
            const user = await this.service.create(req.body)
            res.json(user);
      } catch {
        res.status(400).send({ error: 'Указанные неверные данные' })
      }
    }
  }
```

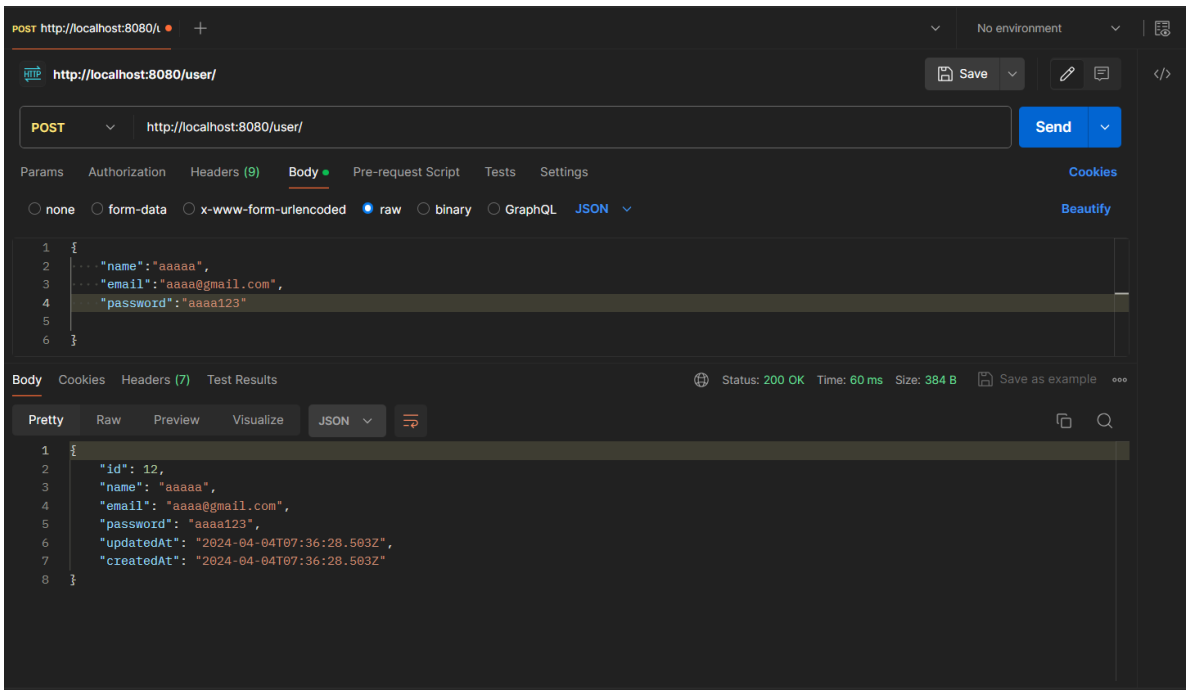
8. Создаем роуты - routes/users/User.ts

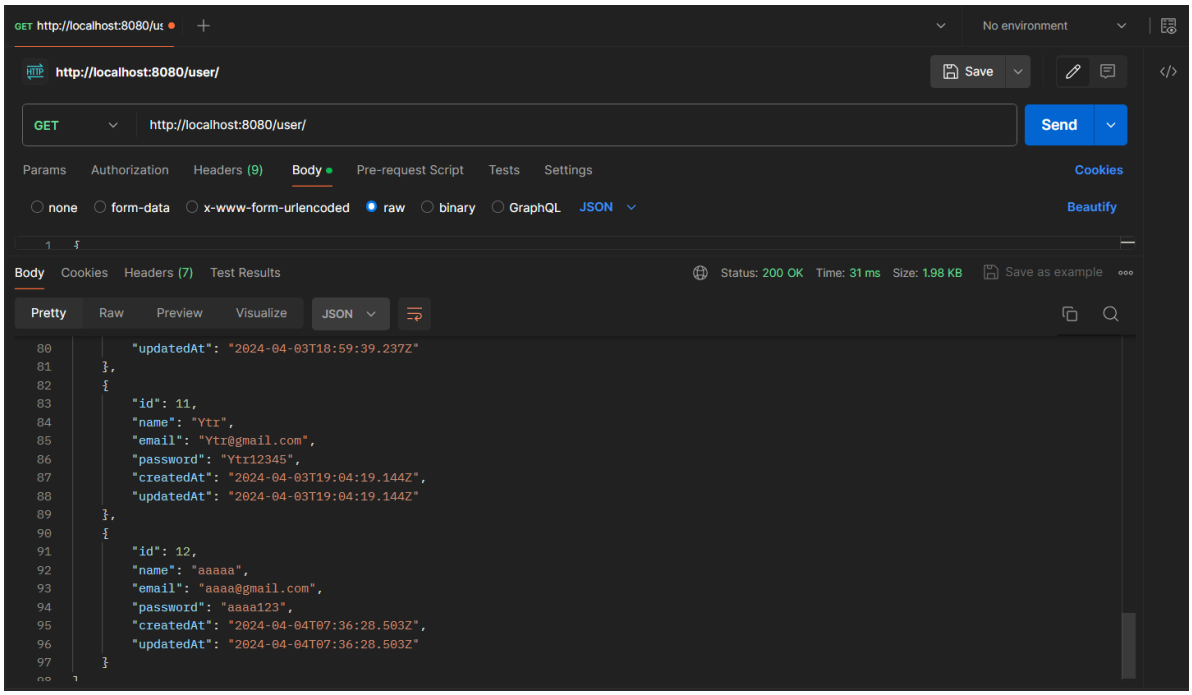
```ts
import { Router } from "express";
import { UserController } from "../../controllers/users/User.js";

const router = Router()
const controller = new UserController()

router.get('/user', controller.get)
router.post('/user', controller.post)

export default router
```

9. index.ts

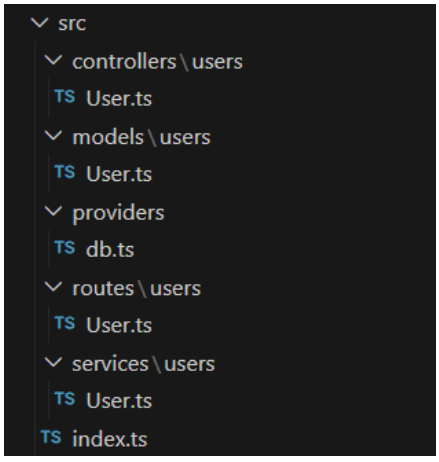
```ts
src > TS index.ts > ...
import express from "express";
import userRoutes from "./routes/users/User.js"
import Sequelize from "./providers/db.js"
import dotenv from 'dotenv'


dotenv.config()
const app = express()
app.use(express.json())
app.use('/', userRoutes)

app.listen(8080, () => {
  Sequelize
  console.log(`Listening on port 8080`)
})

```

Результаты работы





Структура кода



Вывод

В ходе лабораторной работы был написан свой boilerplate на express, sequelize и typescript.