САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая/Лабораторная работа

Выполнил:

Пластун Елизавета Олеговна

Группа K33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

Нужно написать свой boilerplate на express + sequelize + typescript.

Должно быть явное разделение на:

– модели

– контроллеры

– роуты

– сервисы для работы с моделями (реализуем паттерн “репозиторий”)

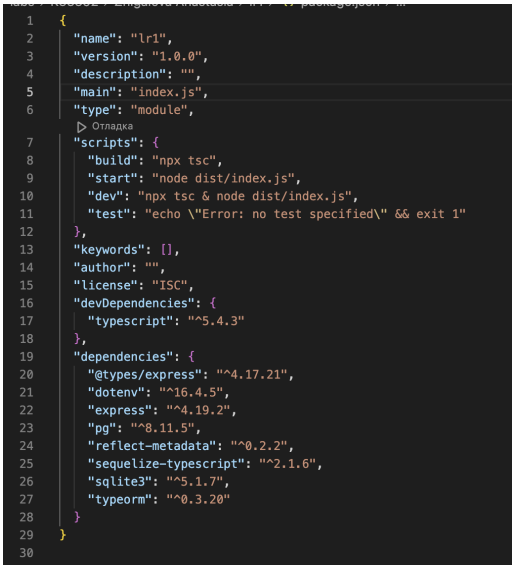
Ход работы

Написали файл для запуска сервера.

1) Инициализируем модуль и устанавливаем все необходимое
2) tsconfig.json

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

3) package.json



```
{
  "name": "lr1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  ▷ Отладка
  "scripts": {
    "build": "npx tsc",
    "start": "node dist/index.js",
    "dev": "npx tsc & node dist/index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "typescript": "^5.4.3"
  },
  "dependencies": {
    "@types/express": "^4.17.21",
    "dotenv": "^16.4.5",
    "express": "^4.19.2",
    "pg": "^8.11.5",
    "reflect-metadata": "^0.2.2",
    "sequelize-typescript": "^2.1.6",
    "sqlite3": "^5.1.7",
    "typeorm": "^0.3.20"
  }
}
```

4) модель юзера

```ts
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
}
```

5) методы для юзера

```ts
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)
        }
    }
}
```

6) контроллеры

```ts
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' })
      }
    }
}
```

7) роуты

```
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
```

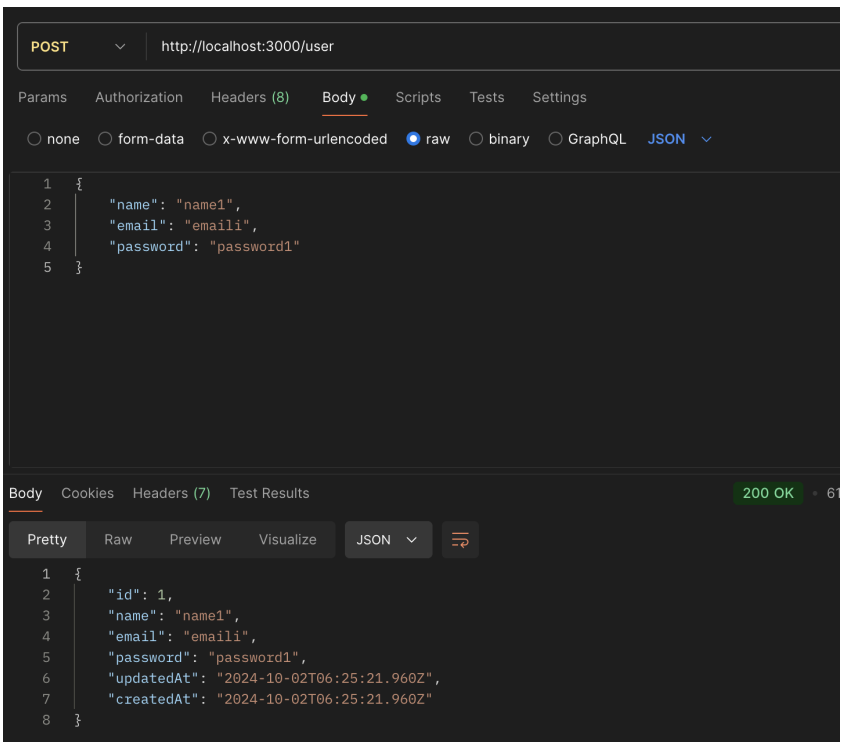
8) index.ts

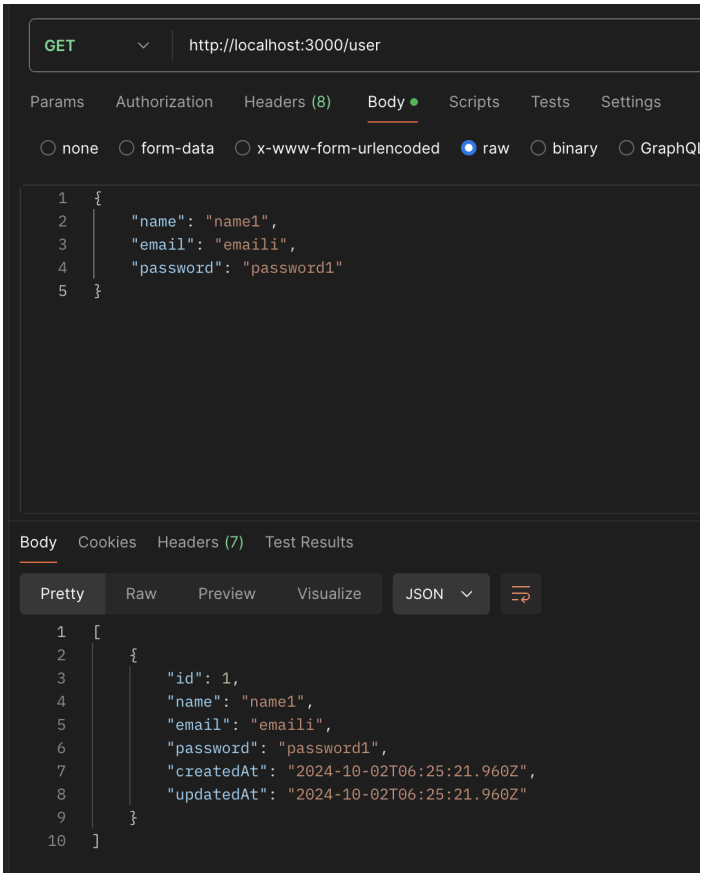
```
import express from "express";
import userRoutes from "./routes/users/User.js"
import Sequelize from "./providers/db.js"
import dotenv from 'dotenv'


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

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

Результаты:





Вывод

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