САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет
Лабораторная работа 1

Выполнила:
Герулайтите Габриэля
К33402

Проверил:
Добряков Д. И.

Санкт-Петербург
2022 г.

Задача

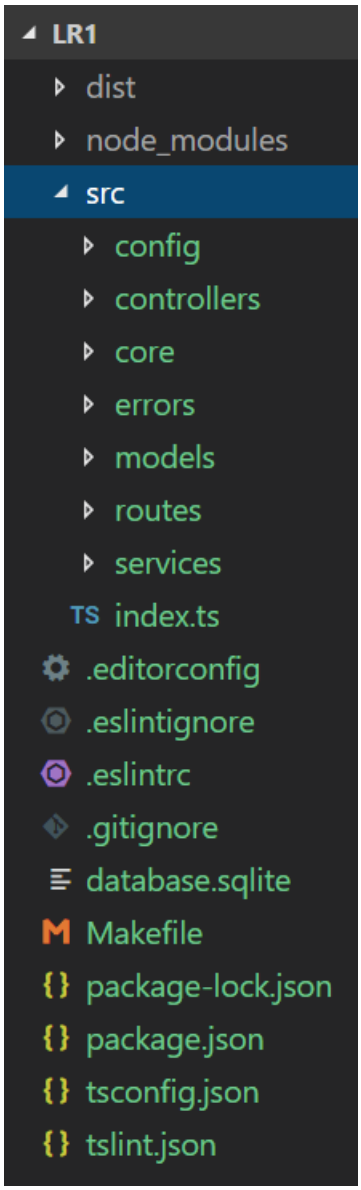
Написать свой boilerplate на express + sequelize + typescript.

Должно быть явное разделение на:

1. Модели
2. Контроллеры
3. Роуты
4. Сервисы для работы с моделями (реализуем паттерн “репозиторий”)

Ход работы

Структура:



Нужные конфигурации и зависимости отображаемые в packace.json

```
{
  "name": "lr1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "prestart": "npm run build",
    "start": "nodemon dist/index.js",
    "build": "npx tsc",
    "lint": "npx eslint . --ext .ts",
    "migrate": "npx sequelize-cli db:migrate"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "@types/express": "^4.17.13",
    "@types/node": "^17.0.31",
    "@types/uuid": "^8.3.4",
    "@typescript-eslint/eslint-plugin": "^5.21.0",
    "@typescript-eslint/parser": "^5.21.0",
    "eslint": "^8.14.0",
    "nodemon": "^2.0.15",
    "sequelize-cli": "^6.4.1",
    "ts-node": "^10.7.0",
    "tslint": "^6.1.3",
    "typescript": "^4.6.3"
  },
  "dependencies": {
    "express": "^4.18.0",
    "sequelize": "^6.19.0",
    "sqlite3": "^5.0.6",
    "uuid": "^8.3.2"
  }
}
```

В качестве ORM я решила использовать sequelize.

Модель пользователя в src/models/index.ts

```ts
import { DataTypes, Model } from "sequelize"
import db from "../config/database_config";

interface Attributes {
    id: string;
    firstName: string;
    lastName: string;
    email: string;
  }
class User extends Model<Attributes> {}

User.init(
    {
        id: {
            type: DataTypes.UUIDV4,
            allowNull: false,
            primaryKey: true
          },
          firstName: {
            type: DataTypes.STRING,
            allowNull: false
          },
          lastName: {
            type: DataTypes.STRING,
            allowNull: false
          },
          email: {
            type: DataTypes.STRING,
            allowNull: false,
            unique: true
          },
    },
    {
        sequelize:db,
    }
)

export default User
```

src/config/database_config.ts

```ts
import { Sequelize } from 'sequelize';

const db = new Sequelize('app', '', '',{
    storage:'./database.sqlite',
    dialect: 'sqlite',
    logging: false,
});

export default db;
```

В директории controllers созданы контроллеры, использующие UserServices для работы с пользователями – создание, вывод всех пользователей, удаление и обновление.

Src/controllers/user.ts

```ts
import User from "../models/index"
import { v4 as uuidv4 } from "uuid"
import UserService from "../services/user"
import router from "../routes/index"

class UserController {
    private userService: UserService

    constructor() {
        this.userService = new UserService()
    }

    get = async (request: any, response: any) => {
        try {
            const records = await this.userService.listUsers()
            return response.json(records);
        } catch (error) {
            response.status(404).send({ "error": 404 })
        }
    }
    post = async (request: any, response: any) => {
        const id = uuidv4()
        console.log(request)
        try {
            const record = await this.userService.create({ ...request.body, id})
            return response.json({ record, msg: 'Successfully create user' })
        } catch (error) {
            response.status(400).send({ "error": 404 })
        }
    }
```

```ts
    getbyID = async (request: any, response: any) => {
        try {
            const record = await this.userService.getById(request.params.id)
            return response.json(record);
        } catch (error) {
            response.status(404).send({ "error": 404 })
        }
    }
    put = async (request: any, response: any) => {
        try {
            const record = await this.userService.updateUser(request.params.id, request.body)
            return response.json({record, msg: 'Successfully update user' })
        } catch (error) {
            response.status(404).send({ "error": 404 })
        }
    }
    delete = async (request: any, response: any) => {
        try {
            const record = await this.userService.deleteUser(request.params.id)
            return response.json({msg: 'Successfully delete user' })
        } catch (error) {
            response.status(404).send({ "error": 404 })
        }
    }
}

export default UserController
```

Сервисы прописаны в src/services/user.ts

```ts
import { userInfo } from "os"
import UserError from "../errors/users/user"
import User from "../models/index"


class UserService {
    async getById(id: string){
        const user = await User.findByPk(id)

        if (user) return user.toJSON()

        throw new UserError('Not found!')

    }

    async create(user: any): Promise<User|Error>{
        try {
            const userData = await User.create(user)
            return userData
        } catch (e: any) {
            const errors = e.errors.map((error: any) => error.message)
            throw new UserError(errors)
        }

    }

    async listUsers(){
        const users = await User.findAll()

        if (users) return users

        throw new UserError('Not found!')
    }

```

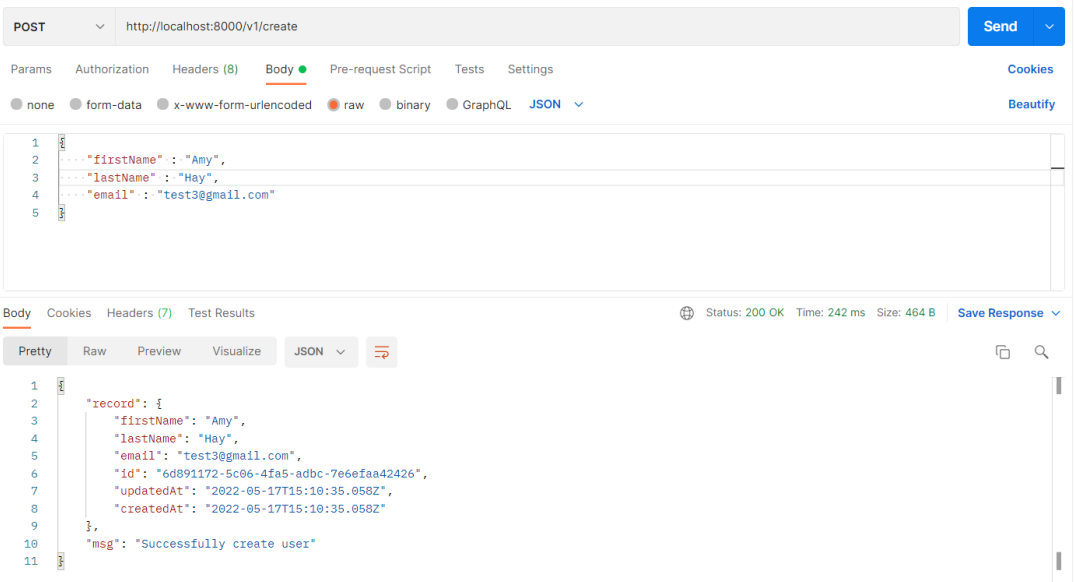
```ts
    async updateUser(id:string, data: any) {
        try {
            const user = await User.findByPk(id)
            if (user) {
                const result = await (user.update(data))
            }
            return user
        } catch (e: any) {
            const errors = e.errors.map((error: any) => error.message)

            throw new UserError(errors)
        }
    }

    async deleteUser(id:string) {
        try {
            await User.destroy({where: {id:id}})
        } catch (e: any) {
            const errors = e.errors.map((error: any) => error.message)

            throw new UserError(errors)
        }
    }

}

export default UserService
```

Все пути прописаны в роутере src/routes/index.ts

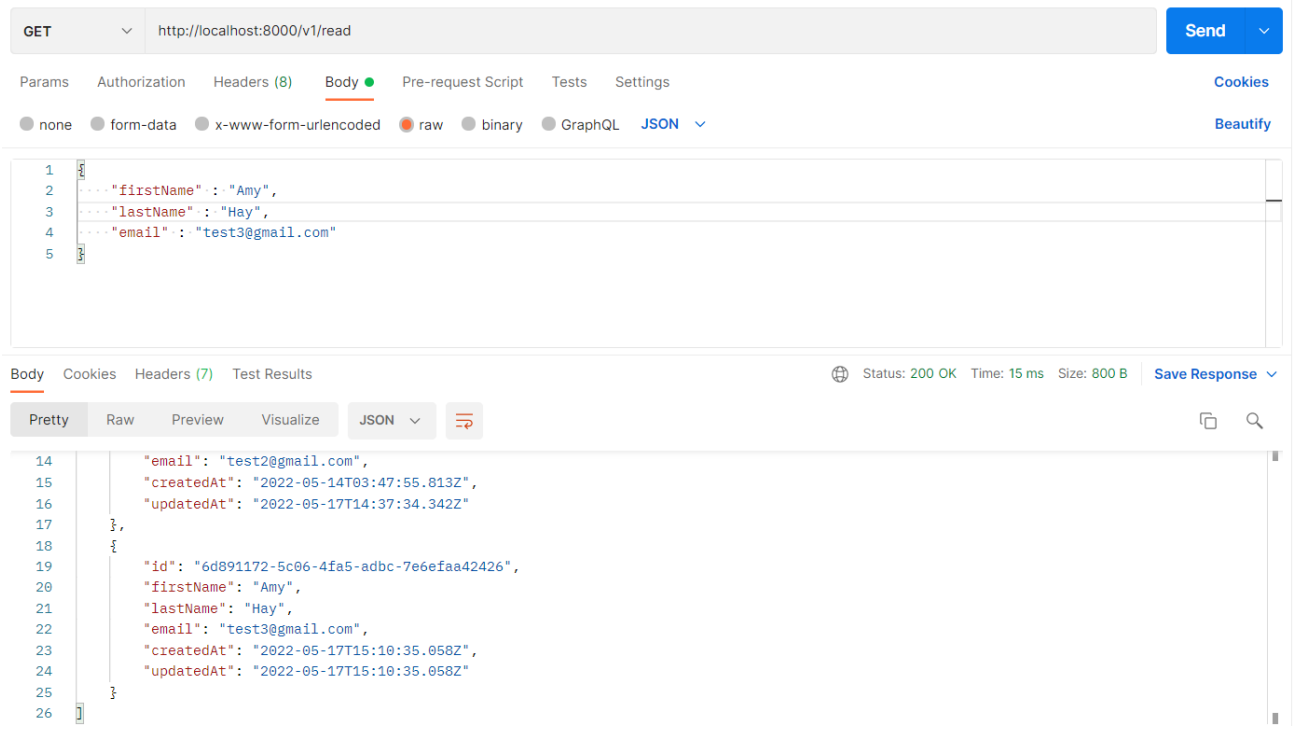
```ts
import express from "express"
import Controller from '../controllers/user'
import { Router } from 'express'


const router: express.Router = express.Router()

const controller = new Controller()

router.route('/read')
  .get(controller.get)

router.route('/create')
  .post(controller.post)

router.route('/user/:id')
  .get(controller.getbyID)

router.route('/update/:id')
  .put(controller.put)

router.route('/delete/:id')
  .delete(controller.delete)

export default router
```

Проверим работу:

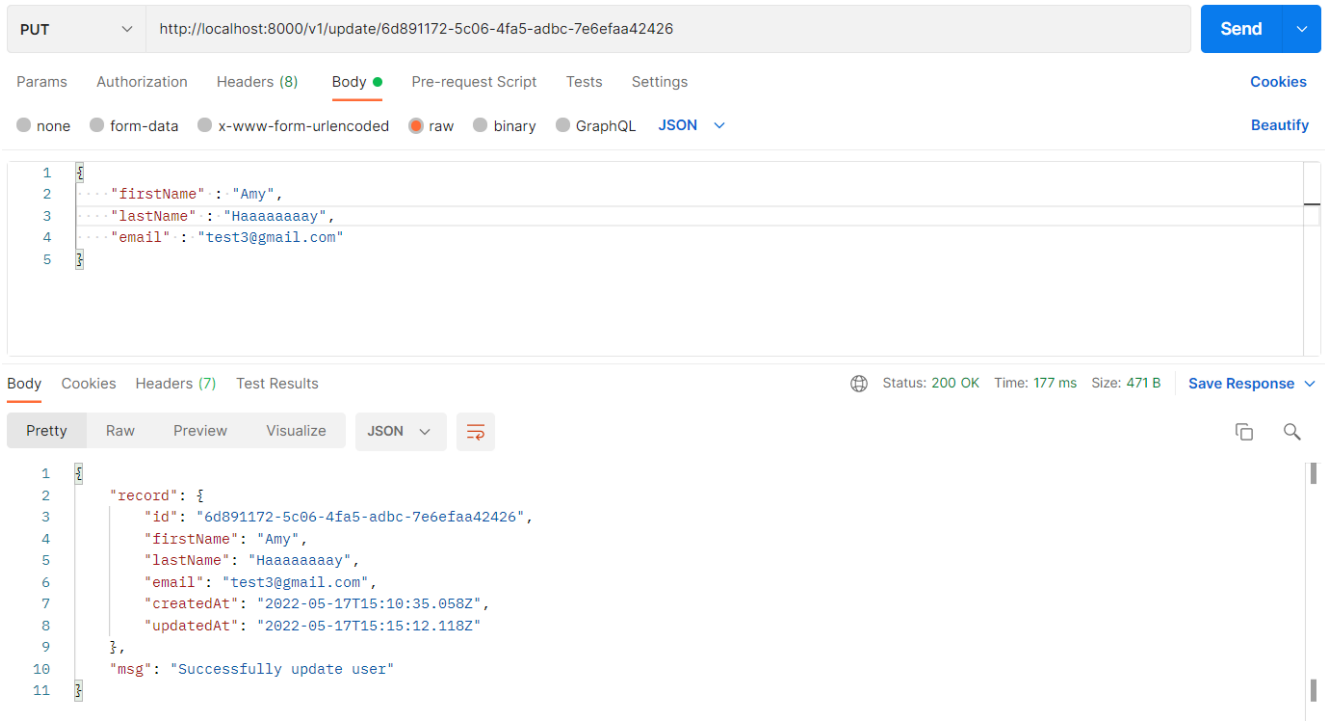
Создание пользователя



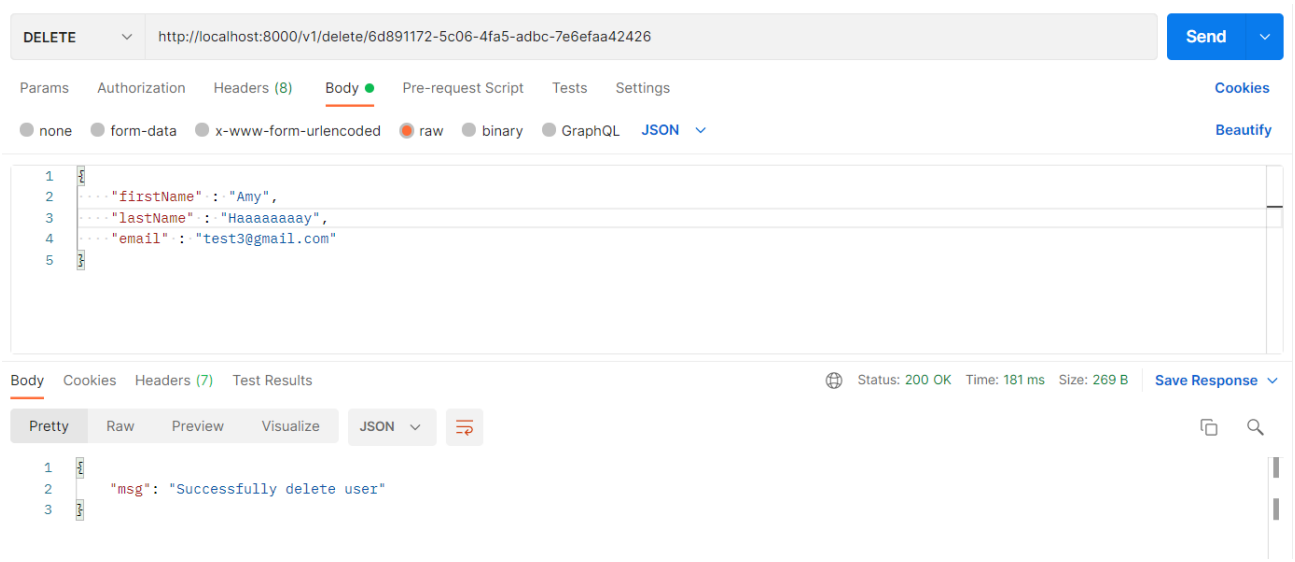
Вывод списка всех пользователей



Обновление информации пользователя



Удаление пользователя



Выводы:

В ходе лабораторной работы был написан свой boilerplate на express, sequelize и typescript. Проверена работа CRUD-методов в postman.