САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Backend-Разработка

Отчет

Лабораторная работа 1

Выполнил:
Бункута Натан

Группа:
К33412

Проверил:
Добрияков Д.И

Санкт-Петербург

2023 г.

Задание:

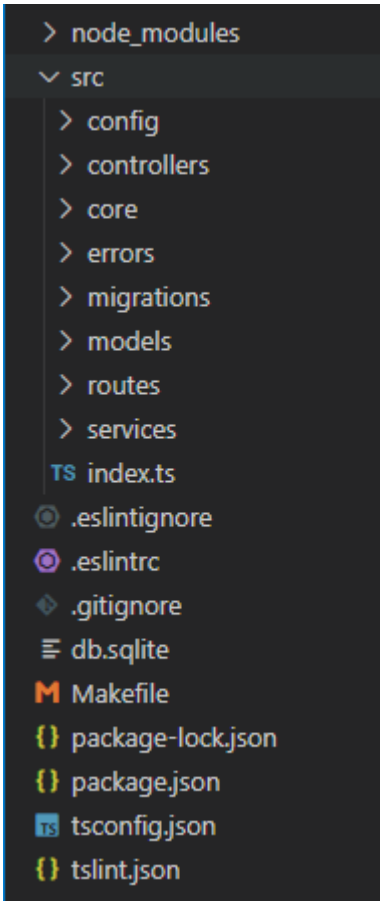
Нужно написать свой boilerplate на express + sequelize / TypeORM + typescript.

Должно быть явное разделение на:

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

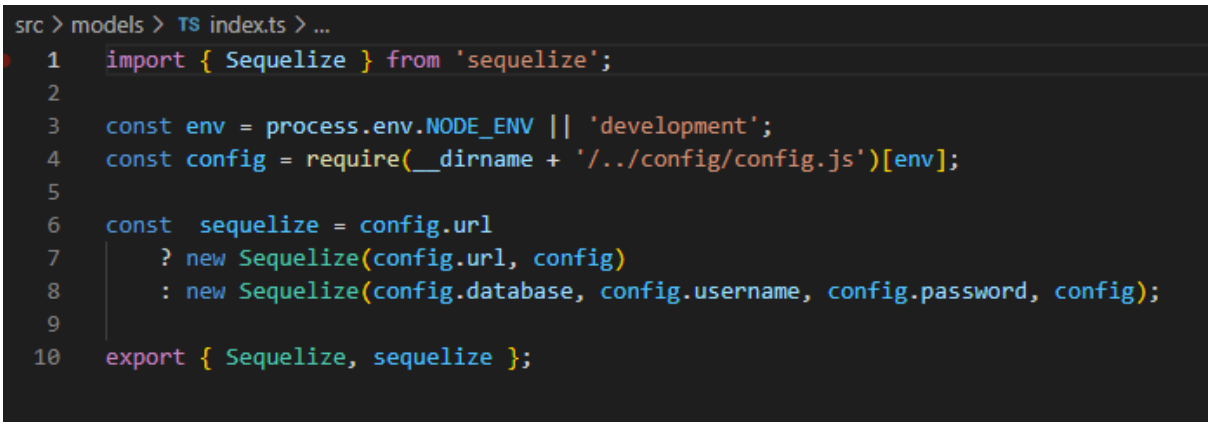
Ход работы:

Boilerplate components



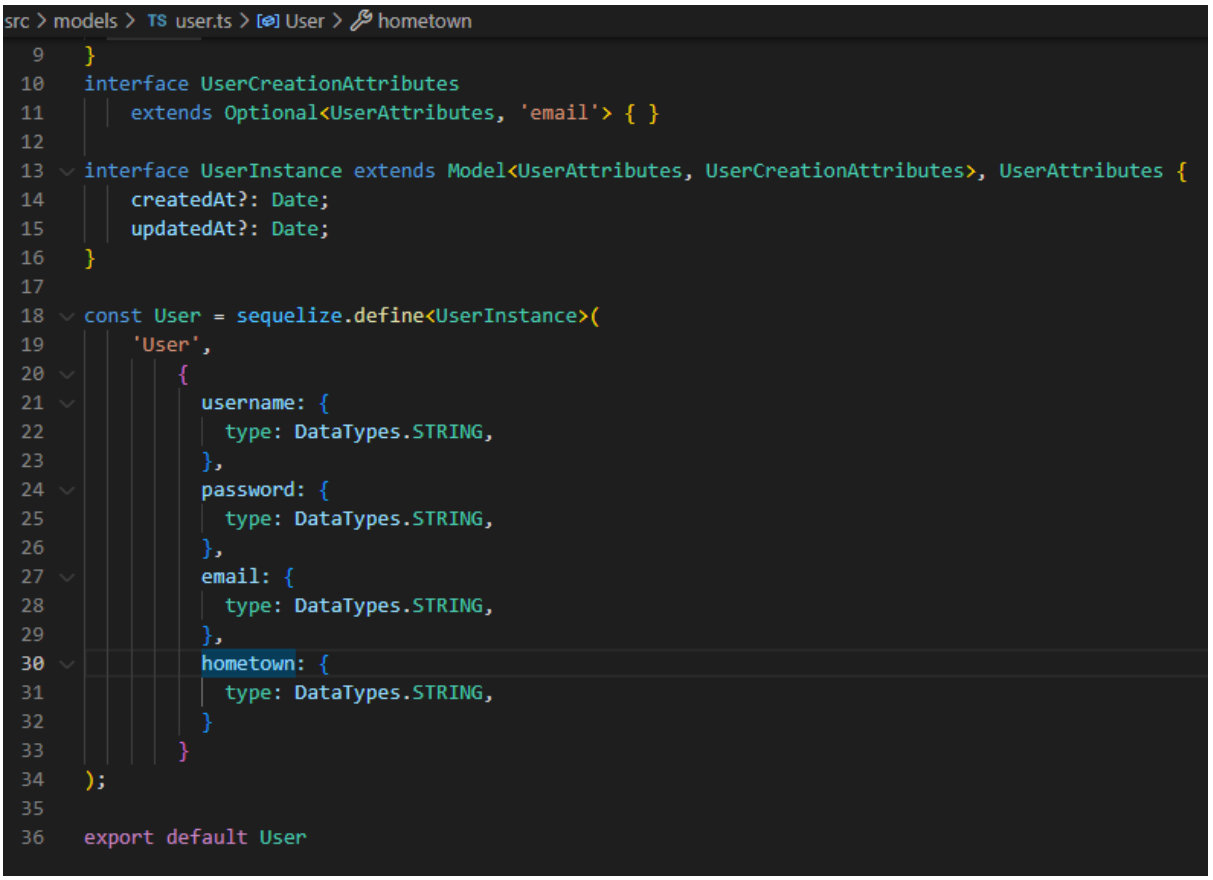
Models

models/index.ts



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

const env = process.env.NODE_ENV || 'development';
const config = require(__dirname + '/../config/config.js')[env];

const  sequelize = config.url
    ? new Sequelize(config.url, config)
    : new Sequelize(config.database, config.username, config.password, config);

export { Sequelize, sequelize };
```

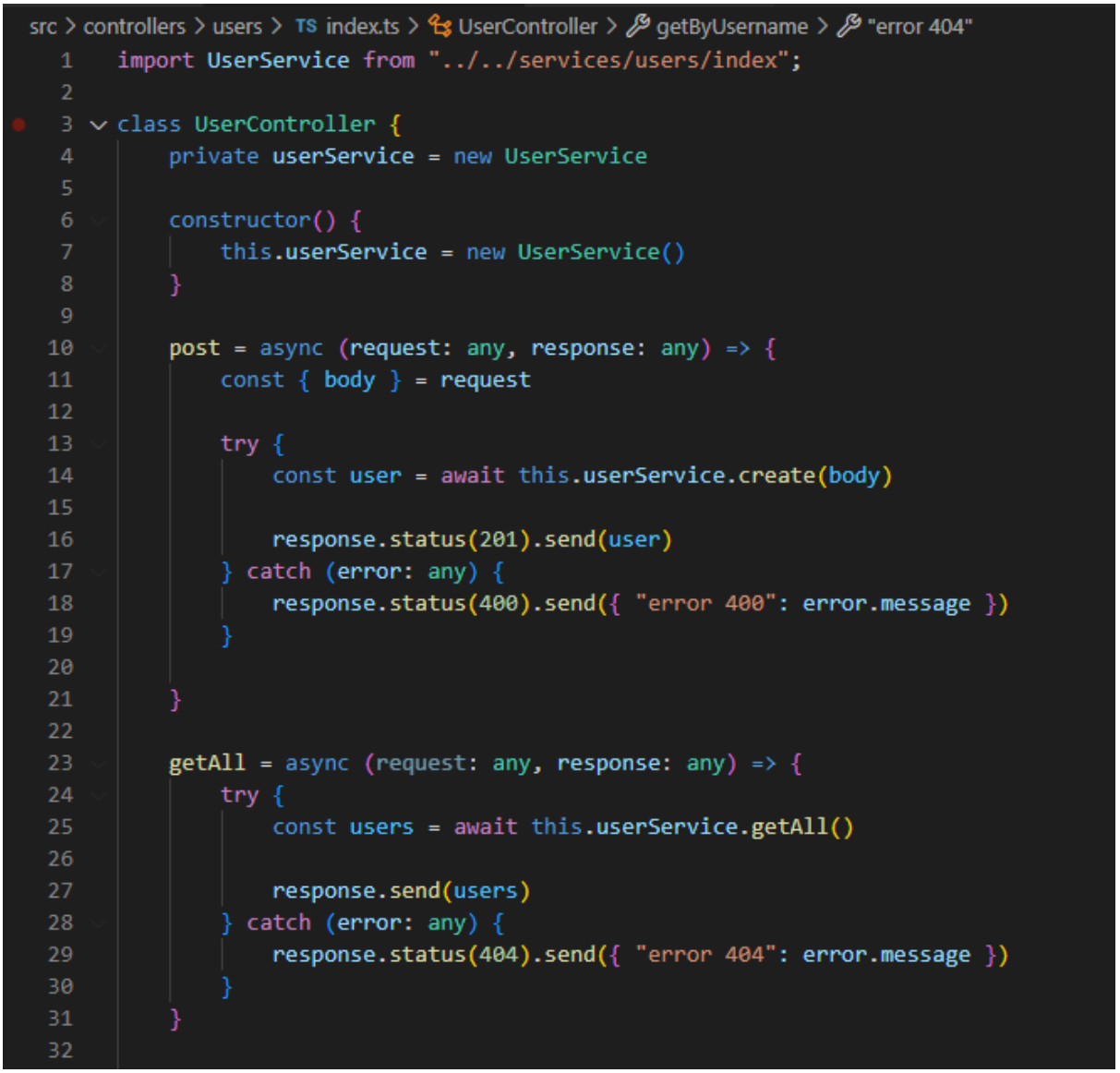
models/user.ts



```ts
}
interface UserCreationAttributes
    extends Optional<UserAttributes, 'email'> { }

interface UserInstance extends Model<UserAttributes, UserCreationAttributes>, UserAttributes {
    createdAt?: Date;
    updatedAt?: Date;
}

const User = sequelize.define<UserInstance>(
    'User',
        {
          username: {
            type: DataTypes.STRING,
          },
          password: {
            type: DataTypes.STRING,
          },
          email: {
            type: DataTypes.STRING,
          },
          hometown: {
            type: DataTypes.STRING,
          }
        }
);

export default User
```

Controllers

index.ts



```
    getById = async (request: any, response: any) => {
        try {
            const user = await this.userService.getById(
                Number(request.params.id)
            )

            response.send(user)
        } catch (error: any) {
            response.status(404).send({ "error 404": error.message })
        }
    }

    getByUsername = async (request: any, response: any) => {
        try {
            const user = await this.userService.getByUsername(
                request.params.username
            )

            response.send(user)
        } catch (error: any) {
            response.status(404).send({ "error 404": error.message })
        }
    }
}

export default UserController
```

Routes

routes/index.ts

```
src > routes > v1 > users > TS index.ts > ...
import express from "express";
import UserController from "../../../controllers/users/index";

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

const controller: UserController = new UserController()

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

router.route('/profiles')
    .get(controller.getAll)

router.route('/profile/id/:id')
    .get(controller.getById)

router.route('/profile/username/:username')
    .get(controller.getByUsername)


export default router
```

Services

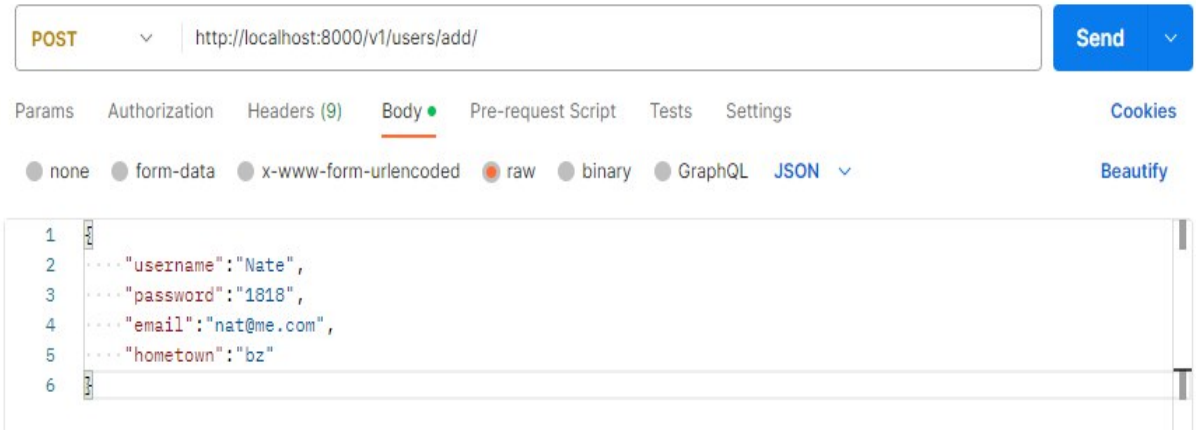
users/index.ts

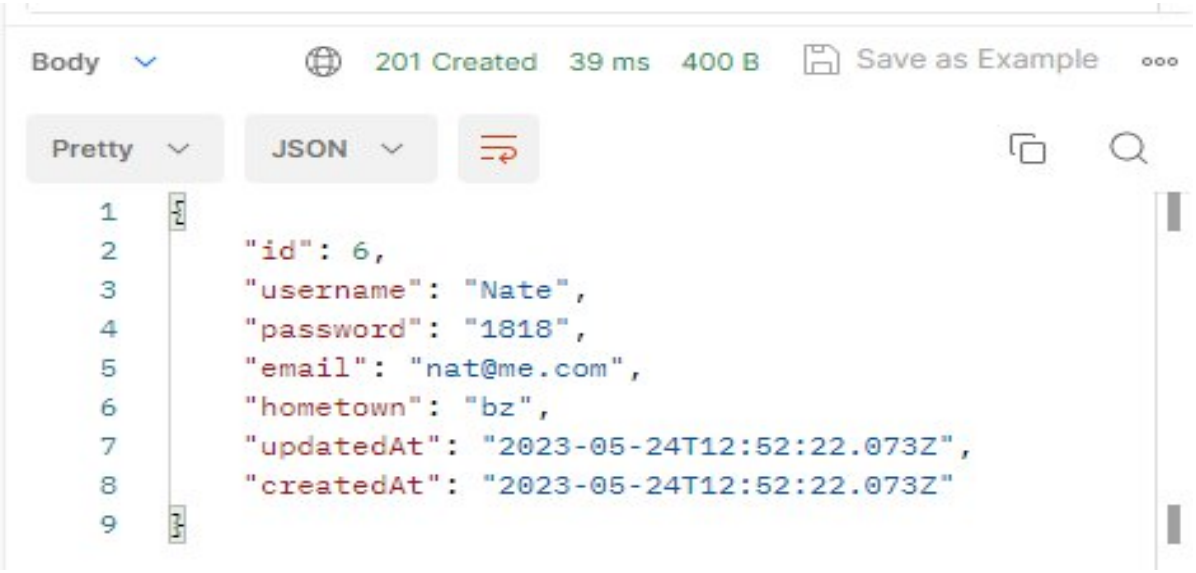
src > services > users > TS index.ts > ...

```ts
import UserError from "../../errors/users/index";
import User from "../../models/user"

class UserService {

    async create(userData: any) {
        try {
            console.log(userData)
            const user = await User.create(userData)

            return user.toJSON()
        } catch (e: any) {
            console.log(e)
            const errors = e.errors.map((error: any) => error.message)

            throw new UserError(errors)
        }
    }

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

        if (users) return users

        throw new UserError('Sorry, No user found!')
    }

    async getById(id: number) {
        const user = await User.findByPk(id)

        if (user) return user.toJSON()

        throw new UserError('Sorry, No such id user found !')
    }

    async getByUsername(username: string) {
        const user = await User.findOne({
            where: {
                username: username
            }
        })

        if (user) return user.toJSON()

        throw new UserError('Sorry, such username not found !')
    }
}

export default UserService
```

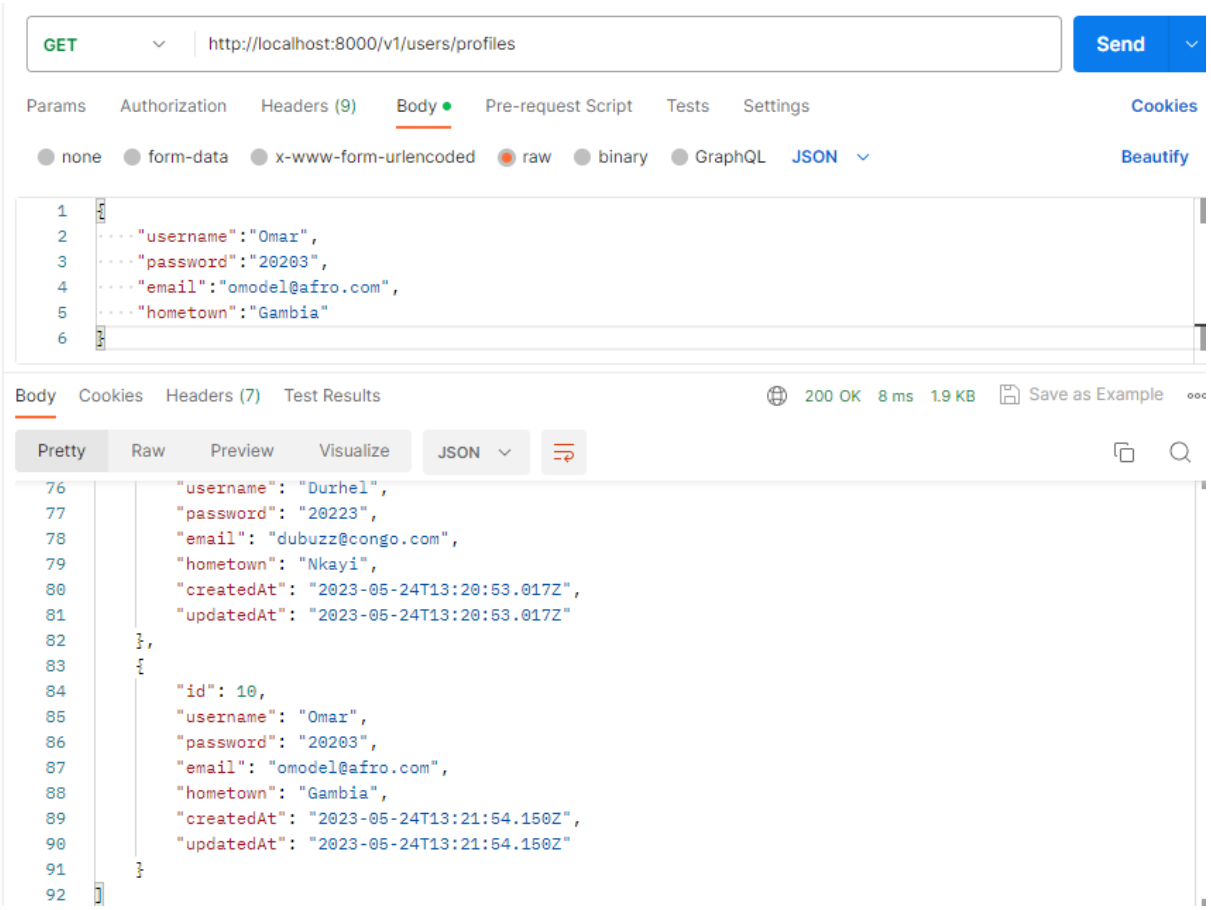
Проверка в Postman

Add a new user

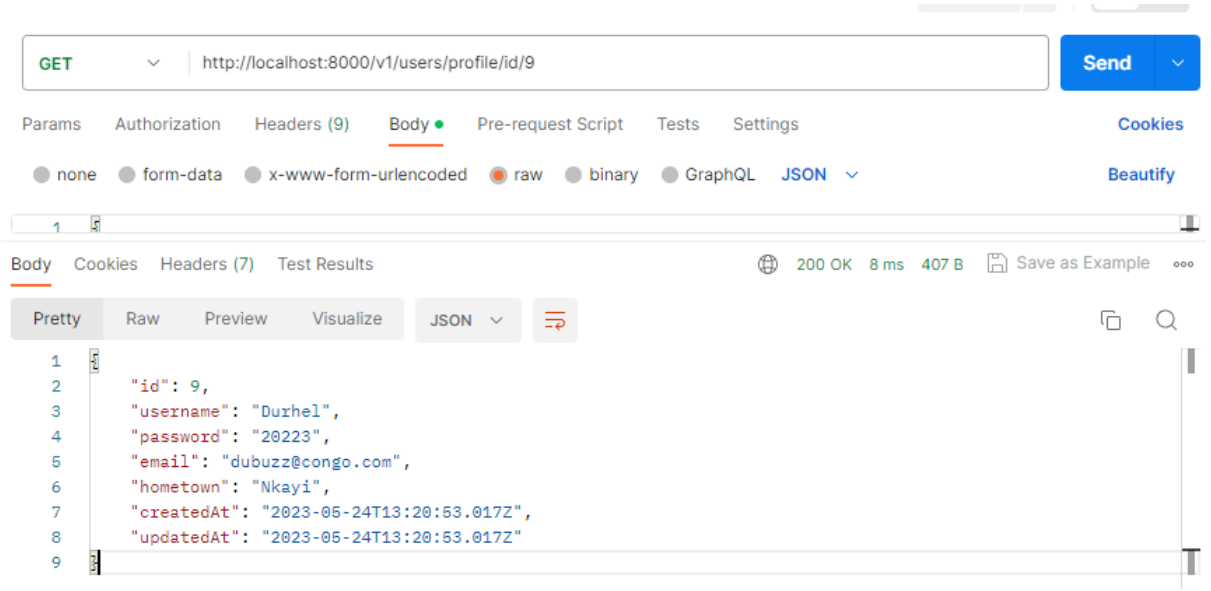




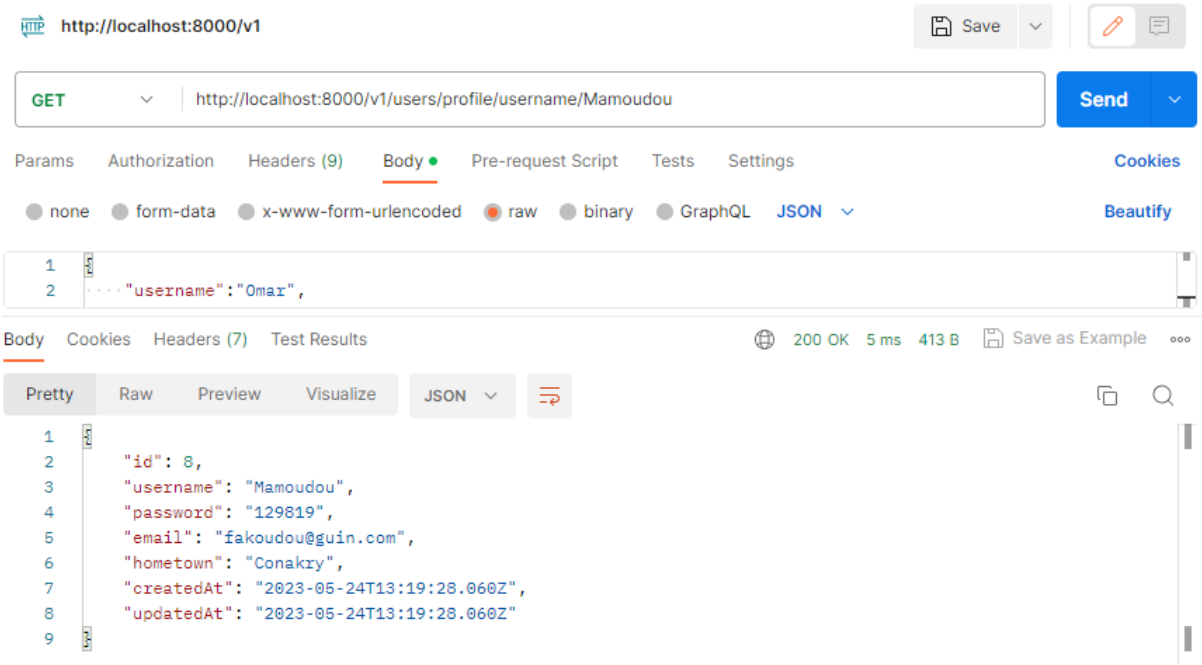
Get list of all users



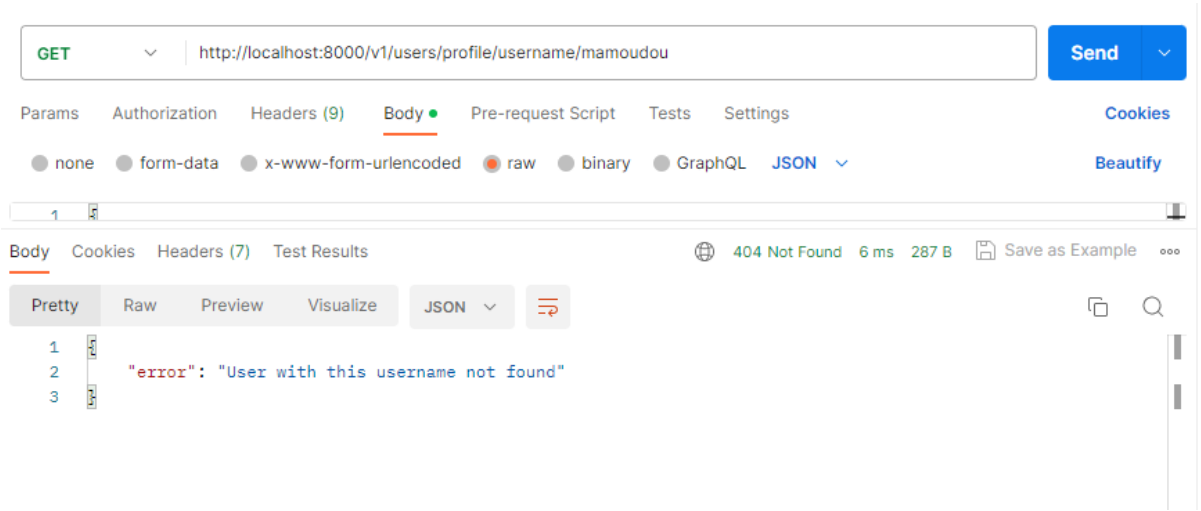
Find user by id



Find user by his username



Potential error exemple



Вывод:

При выполнения данной лабораторной работы, мы научили писать простой boilerplate с исползованием средств TypeScript, Express + Squelize. Были разделены компоненты как моделы, контроллеры, роуты и сервисы.