САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэкенд разработка

Отчет

Лабораторная работа 1

Выполнил:
Омар Сизей

Группа: К33412

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

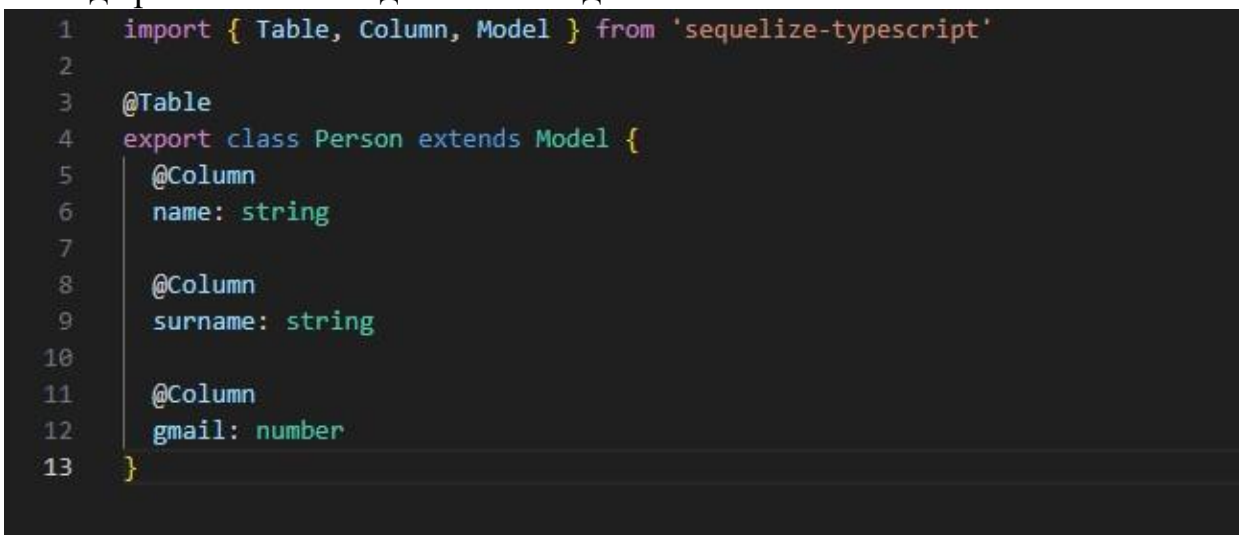
Нужно написать свой boilerplate на express + sequelize / TypeORM + typescript.

Должно быть явное разделение на:

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

Ход работы

В ходе работы был создан класс моделей:



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

@Table
export class Person extends Model {
  @Column
  name: string

  @Column
  surname: string

  @Column
  gmail: number
}
```

Использование моделей конфигурируется в config.ts:

config > TS config.ts > ...

```ts
import { Sequelize } from 'sequelize-typescript'
import { Person } from '../models/Person'

export const sequelize = new Sequelize({
  database: 'example_db',
  dialect: 'sqlite',
  username: 'root',
  password: '',
  storage: ':memory:',
  models: [Person],
  repositoryMode: true
})
```

Запросы к БД реализуются через класс Сервис:

services > TS index.ts > ...

```ts
import { Person } from '../models/Person'
import { sequelize } from '../config/config'

export class MainService {

    private repo = sequelize.getRepository(Person)

    add(name_par: string, surname_par: string, gmail_par: number) {
        this.repo.create({ name: name_par, surname: surname_par, gmail: gmail_par })
    }

    get() {
        const data = this.repo.findAll()
        return data
    }
}
```

С сервисом взаимодействует Контроллер:

```ts
import { MainService } from '../services/index'

class ExampleController {

    private service = new MainService()

    post = async (request: any, response: any) => {
        try {
            const body = request.body
            await this.service.add(body.name, body.surname, body.gmail)
            response.send('Added')
        } catch (error: any) {
            response.status(400).send(error.message)
        }
    }

    get = async (request: any, response: any) => {
        try {
            const data = await this.service.get()
            response.send(data)
        } catch (error: any) {
            response.status(400).send(error.message)
        }
    }
}

export default ExampleController
```

Роуты:

routes > TS index.ts > ...

```ts
import express from "express"
import
(alias) function express(): core.Express
(alias) namespace express
import express
Creates an Express application. The express() function is a top-level function exported by the express module.
const r

const e

router
  .route('/')
  .get(exampleController.get)
  .post(exampleController.post)

export default router
```

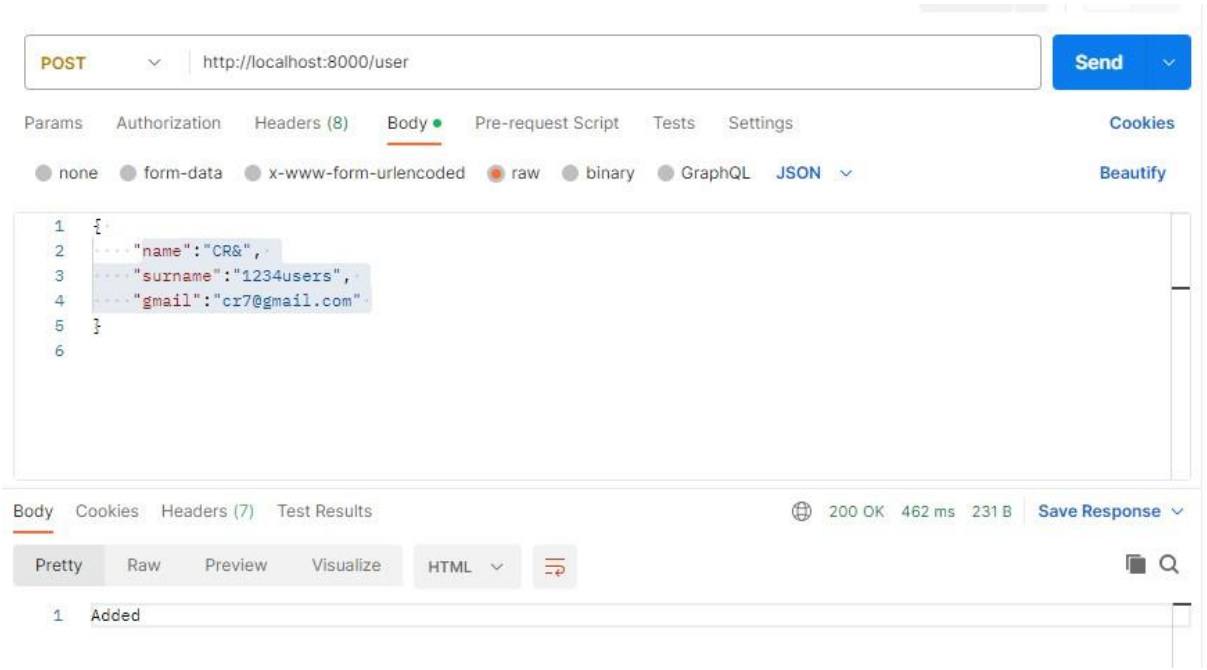
Получившийся Makefile:

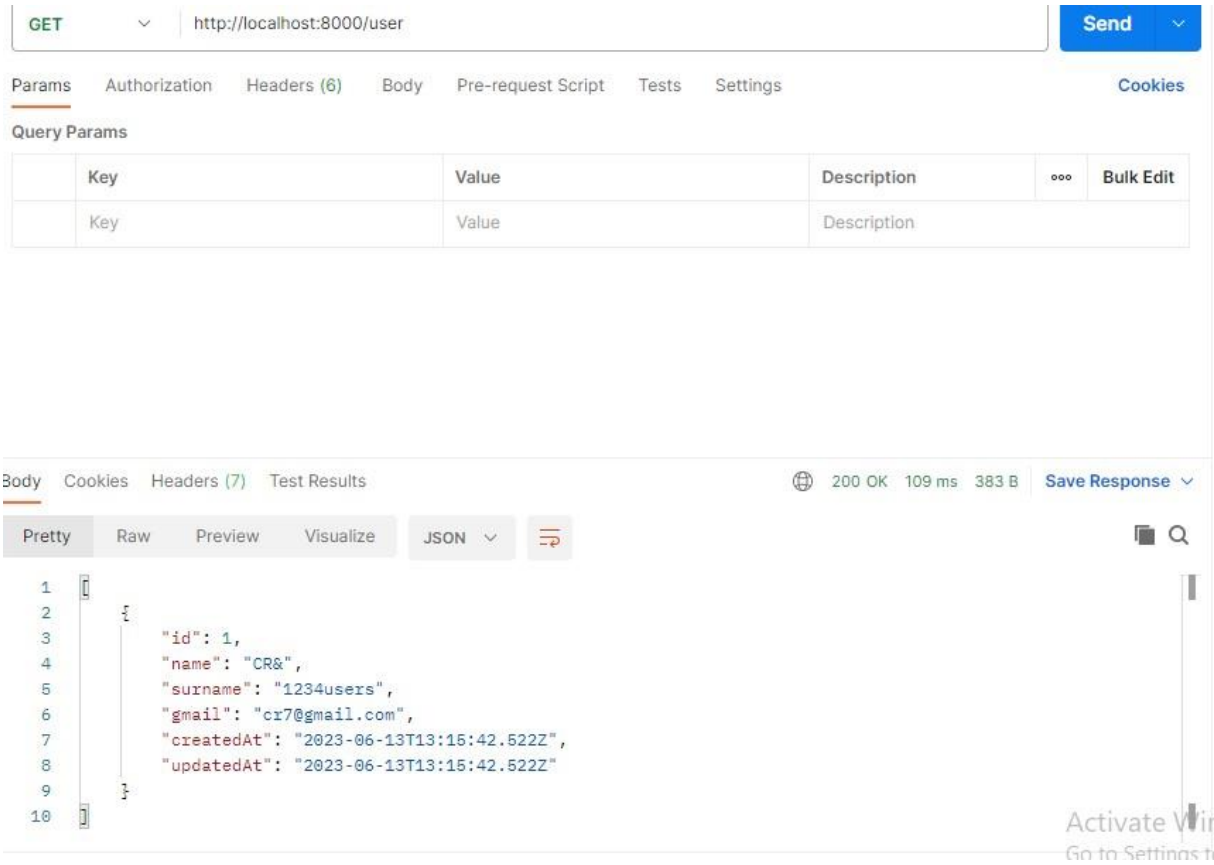
```
.PHONY: init
init:
    npm i
    npm run build

.PHONY: run
run:
    npm start

.PHONY: migrate
migrate:
    npx sequelize-cli db:migrate

.DEFAULT_GOAL := init
```





Вывод

В этой работе был создан бойлерплейт на express, sequelize и typescript. Получившийся код разделен на логические части – модели, контроллеры, роуты, сервис.