САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэкенд разработка

Отчет

Лабораторная работа №1: Boilerplate

Выполнила:

Еремеева Арина

Группа К33412

Проверил:
Добряков Д. И.

Санкт-Петербург
2023г.

Цель: написать свой boilerplate на express + sequelize / TypeORM typescript

Задачи:

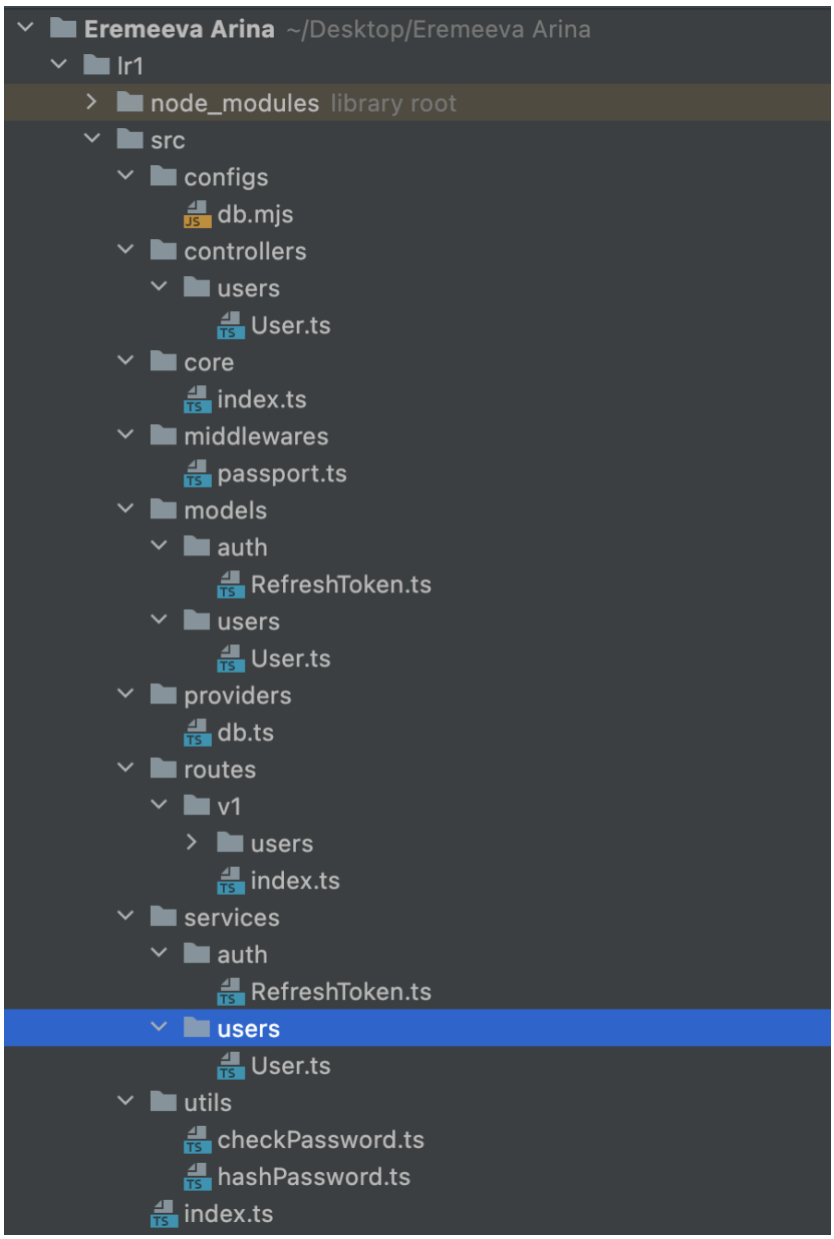
Создать явное разделение на следующие папки:

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

Ход работы:

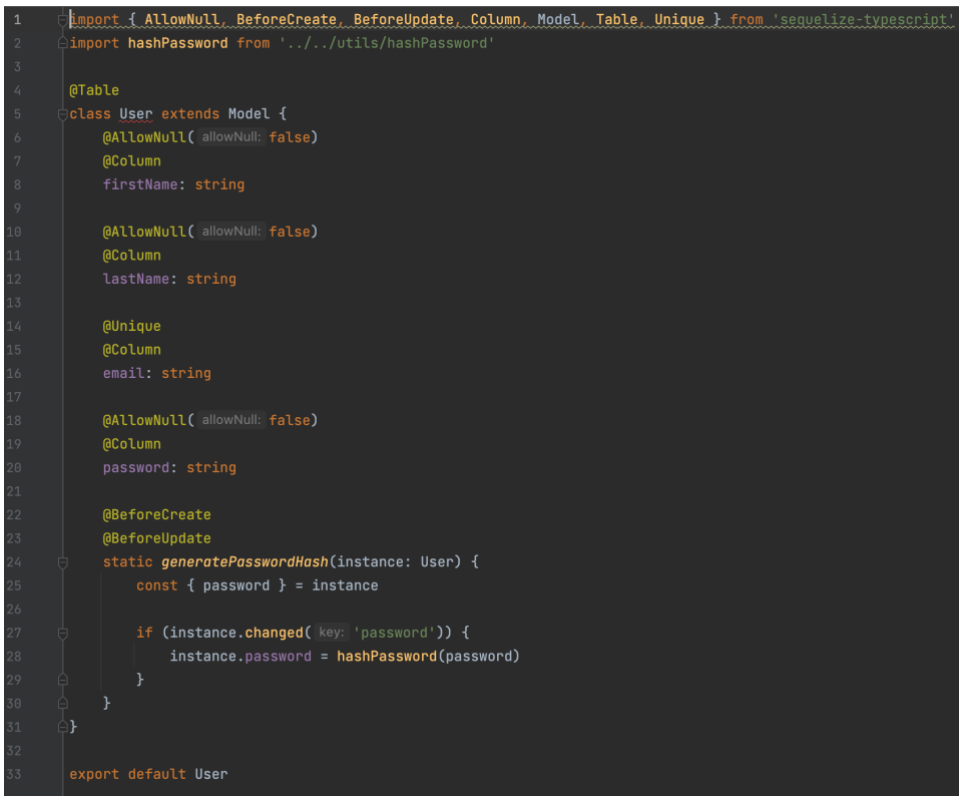
1. Для запуска проекта, мне необходимо было создать файл Makefile, в котором я реализовала следующие команды:
    - migrate
    - init
    - install_d
    - start
    - lint
    - build
2. Далее я инициализировала проект, установила зависимости, создала файл .env, и сделала make start

Получилась следующая структура папки src:



Затем мной были созданы следующие модели:

1. Модель пользователя:



```
import { AllowNull, BeforeCreate, BeforeUpdate, Column, Model, Table, Unique } from 'sequelize-typescript'
import hashPassword from '../../utils/hashPassword'

@Table
class User extends Model {
    @AllowNull( allowNull: false)
    @Column
    firstName: string

    @AllowNull( allowNull: false)
    @Column
    lastName: string

    @Unique
    @Column
    email: string

    @AllowNull( allowNull: false)
    @Column
    password: string

    @BeforeCreate
    @BeforeUpdate
    static generatePasswordHash(instance: User) {
        const { password } = instance

        if (instance.changed( key: 'password')) {
            instance.password = hashPassword(password)
        }
    }
}

export default User
```

2. Модель RefreshToken’a:

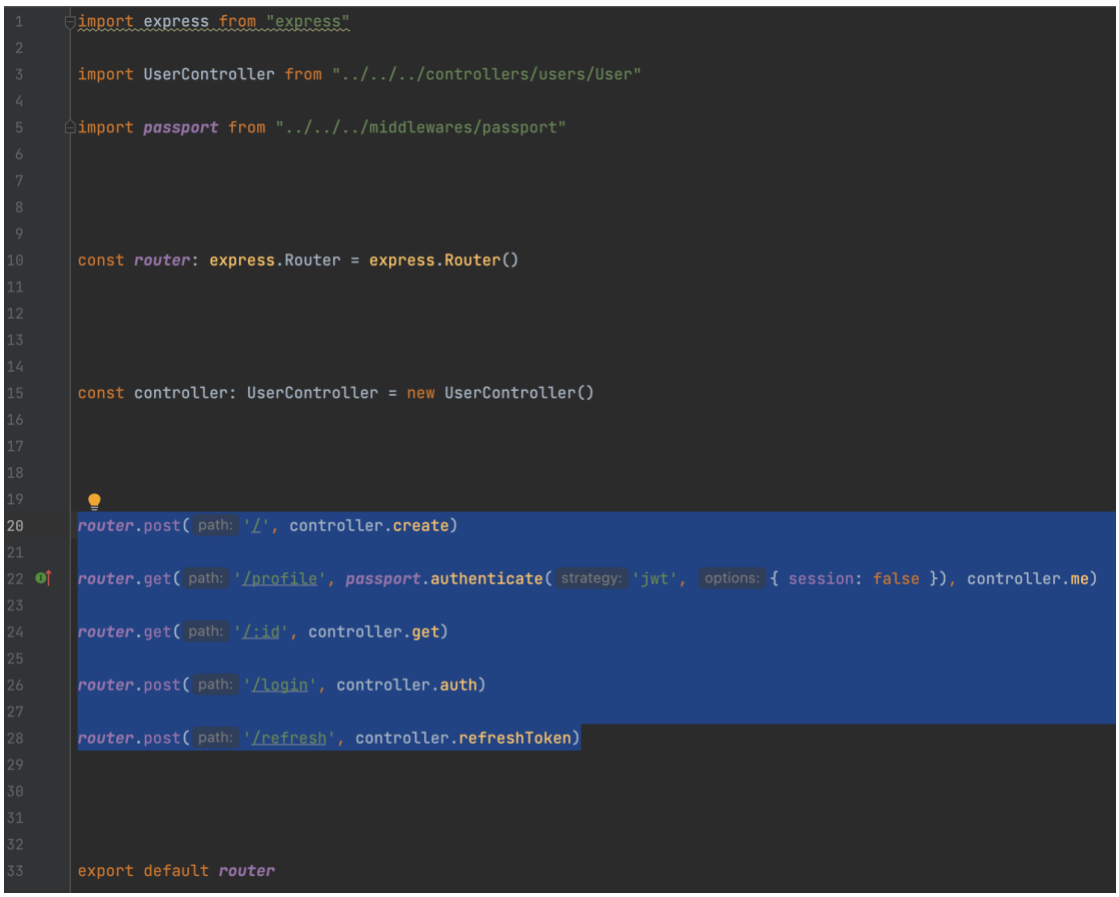
```
import { Table, Column, Model, Unique, AllowNull, ForeignKey } from 'sequelize-typescript'
import User from '../users/User'

@Table
class RefreshToken extends Model {
    @Unique
    @AllowNull( allowNull: false)
    @Column
    token: string

    @ForeignKey( relatedClassGetter: () => User)
    @Column
    userId: number
}

export default RefreshToken
```

Маршруты для различных get и post запросов можно найти в файле User.ts по следующим routes:



```
import express from "express"

import UserController from "../../../controllers/users/User"
import passport from "../../../middlewares/passport"



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



const controller: UserController = new UserController()



router.post( path: '/', controller.create)

router.get( path: '/profile', passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.me)

router.get( path: '/:id', controller.get)

router.post( path: '/login', controller.auth)

router.post( path: '/refresh', controller.refreshToken)



export default router
```

Вывод: в данной лабораторной работе мной был написан boilerplate на express + sequelize / TypeORM typescript.