САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №1

Выполнил:
Вахрушева Ксения

Группа:
K33412

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”)

Структура проекта:

config: имеется конфигурация, которая помогает миграции узнать о базе данных.

control: это класс, который содержит функции для разных маршрутизаторов, например, если ввести какой-либо путь, эти функции будут запущены.

core: подробности о сервере, такие как порт, хост, экспресс-конфигурация, express.use и настройка прослушивания сервера, находятся в этом файле.

database: здесь находится наша БД, а также наша конфигурация базы данных.

migrations: содержит файлы миграций.

модели: содержит модели для модели пользователя и токенов.

rout: здесь расположены все наши пути и здесь вызываются функции наших контроллеров

.env: это наши окружения, которые мы используем в разных частях нашего проекта, например, в конфигурации базы данных в папке базы данных.

main: вызывает наш сервер для запуска.

tsconfig.json: этот файл содержит конфигурации машинописного текста.

1) Модель пользователя:

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

@Table
class User extends Model {
    @Column
    name: string

    @Unique
    @Column
    email: string

    @AllowNull(false)
    @Column
    password: string

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

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

export default User
```

Модель токена:

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

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

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

export default Tokens;
```

2) Контроллеры:

```
import express from 'express'
import User from '../models/user';
import userproduct from '../models/userproducts'
import generateToken from '../util/TokenCreater';
import checkPassword from '../util/checkPassword';
import generateProduct from '../util/ProduntCreter';

export class Controller{

    //Get home
    public static Home = (req:express.Request,res:express.Response)=>{
        res.send(
            `<h1> Welcome to the Application </h1>

            <pre>
             http://localhost:4000  for Home page(current Page)
             http://localhost:4000/Register  for Registeration
             http://localhost:4000/login  for login
             http://localhost:4000/update/(numOfId)  for Update by Id
             http://localhost:4000/users/(numOfId)  for delete by Id
             http://localhost:4000/FindUserById/(numOfId)  for FindUser by Id
             http://localhost:4000/users  for showing all users <pre>
            `
        )
    }


    //POST registeration
    public static Register =async (req:express.Request,res:express.Response) => {
        const {name,password,email}= req.body;
        const exist = await User.findOne({
            where:{
                email
            }
```

3) Роуты:

```
import express from 'express'
const rout = express.Router()
import { Controller } from '../control/controllers'

rout.get('/', Controller.Home)
rout.post('/Login',Controller.Login)
rout.post('/Register',Controller.Register)
rout.get('/users',Controller.AllUsers)
rout.post('/userProducts',Controller.ProductCheck)
rout.get('/getProducts/:id',Controller.UserProducts)  //find products according to userid
rout.get('/FindUserById/:id',Controller.FindById)
rout.delete('/users/:id',Controller.DeleteUser)
rout.put('/update/:id',Controller.UpdateUser)









export default rout;
```

Вывод:

В ходе выполнения работы был создан boilerplate на express, sequelize и typescript. Этот boilerplate можно использовать для создания backend приложения.