МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет информационных технологий механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа №4

По дисциплине «Web-программирование»

Выполнил:

Студент группы №M33091 Сабитов Искандэр

САНКТ-ПЕТЕРБУРГ

2022

Генерируем модули для пользователей, регистрации и постов:

```
PS C:\Users\Beresa\Desktop\web-back\portfolio-web> nest g resource users
? What transport layer do you use? REST API
? Would you like to generate CRUD entry points? Yes
CREATE src/users/users.controller.spec.ts (566 bytes)
CREATE src/users/users.controller.ts (894 bytes)
CREATE src/users/users.module.ts (247 bytes)
CREATE src/users/users.service.spec.ts (453 bytes)
CREATE src/users/users.service.ts (609 bytes)
CREATE src/users/dto/create-user.dto.ts (30 bytes)
CREATE src/users/dto/update-user.dto.ts (169 bytes)
CREATE src/users/entities/user.entity.ts (21 bytes)
UPDATE package.json (2249 bytes)
UPDATE src/app.module.ts (499 bytes)
√ Packages installed successfully.
```

users.service.ts:

```ts
import { Injectable } from '@nestjs/common';
import { CreateUserDto } from './dto/create-user.dto';
import { UpdateUserDto } from './dto/update-user.dto';
import { PrismaService} from "../prisma/prisma.service";

@Injectable()
export class UsersService {
  constructor(private prismaService: PrismaService) {
  }

  create(createUserDto: CreateUserDto) {
    return this.prismaService.user.create({ data: createUserDto });
  }

  findAll() {
    return this.prismaService.user.findMany();
  }

  findOne(id: number) {
    return this.prismaService.user.findUnique({where: {id}});
  }

  update(id: number, updateUserDto: UpdateUserDto) {
    return this.prismaService.user.update({ data: updateUserDto, where: { id } });
  }

  remove(id: number) {
    return this.prismaService.user.delete({ where: { id } });
  }
```

users.controller.ts:

```ts
@Controller( prefix: 'users')
export class UsersController {
  constructor(private readonly usersService: UsersService) {}

  @Post()
  create(@Body() createUserDto: CreateUserDto) {
    return this.usersService.create(createUserDto);
  }

  @Get()
  findAll() {
    return this.usersService.findAll();
  }

  @Get( path: ':id')
  findOne(@Param( property: 'id') id: number) {
    return this.usersService.findOne(id);
  }

  @Patch( path: ':id')
  update(@Param( property: 'id') id: number, @Body() updateUserDto: UpdateUserDto) {
    return this.usersService.update(id, updateUserDto);
  }

  @Delete( path: ':id')
  remove(@Param( property: 'id') id: number) {
    return this.usersService.remove(id);
  }
}
```

create-user.dto.service:

```ts
export class CreateUserDto {
  name: string

  email: string

  password: string
}
```

Устанавливаем модуль swagger

Редактируем контроллеры и dto:

```ts
@ApiTags( tags: 'users')
@Controller( prefix: 'users')
export class UsersController {
  constructor(private readonly usersService: UsersService) {}

  @ApiOperation( options: {summary: 'Create new user'})
  @Post()
  create(@Body() createUserDto: CreateUserDto) {
    return this.usersService.create(createUserDto);
  }

  @ApiOperation( options: {summary: 'Get all users'})
  @Get()
  findAll() {
    return this.usersService.findAll();
  }

  @ApiOperation( options: {summary: 'Get user by id'})
  @Get( path: ':id')
  findOneById(@Param( property: 'id') id: number) {
    return this.usersService.findOneById(id);
  }

  @ApiOperation( options: {summary: 'Get user by id'})
  @Get( path: ':id')
  findOneByEmail(@Param( property: 'id') email: string) {
    return this.usersService.findOneByEmail(email);
  }
```

```ts
import { ApiProperty } from '@nestjs/swagger';

export class CreateUserDto {
  @ApiProperty()
  name: string;

  @ApiProperty()
  email: string;

  @ApiProperty()
  password: string;
}
```

API:

Wooolf memes 1.0 OAS3

The wolves API description

wolves

default

- GET /
- GET /index.html
- GET /neurowolves.html
- GET /memes.html
- GET /login
- GET /register

users

- POST /users Create new user
- GET /users Get all users
- GET /users/{id} Get user by id
- PATCH /users/{id} Update info about user by id
- DELETE /users/{id} Delete user by id

posts

- POST /posts Create post
- GET /posts Get all posts
- GET /posts/{id} Get post by id
- PATCH /posts/{id} Update post by id
- DELETE /posts/{id} Delete post by id

authorization/registration

- POST /auth/login login
- POST /auth/user registration

Schemas

CreateUserDto {

name*	string
email*	string
password*	string

}

UpdateUserDto

CreatePostDto

UpdatePostDto