

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ САНКТ-ПЕТЕРБУРГСКИЙ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ  
Факультет информационных технологий и программирования

## **Лабораторная работа №4**

«Создание контроллеров страниц и спецификации»  
по дисциплине «Web программирование»

Выполнил: студент группы М33122 Белоногов Е.В.

Добавлены модули приложения:

- users – модуль для работы с пользователями
- comments – модуль для работы с комментариями, которые могут оставлять пользователи
- passwords – модуль для работы с паролями учетных записей пользователей (?)

```

  ✓ src
    ✓ comments
      ✓ dto
        TS create-comment.dto.ts
      TS comment.entity.ts
      TS comments.controller.ts
      TS comments.module.ts
      TS comments.service.ts
    ✓ passwords
      ✓ dto
        TS create-password.dto.ts
      TS password.entity.ts
      TS passwords.controller.ts
      TS passwords.module.ts
      TS passwords.service.ts
    ✓ users
      ✓ dto
        TS create-user.dto.ts
      TS user.entity.ts
      TS users.controller.ts
      TS users.module.ts
      TS users.service.ts
    TS app.controller.spec.ts
    TS app.controller.ts
    TS app.module.ts
    TS app.service.ts
    TS main.ts
```

Модуль users:

```
TS app.module.ts    TS users.module.ts ×    TS main.ts
src > users > TS users.module.ts > ...
 1  import { Module } from '@nestjs/common';
 2  import { TypeOrmModule } from '@nestjs/typeorm';
 3  import { UsersService } from './users.service';
 4  import { UsersController } from './users.controller';
 5  import { User } from './user.entity';
 6
 7
 8  @Module({
 9    imports: [TypeOrmModule.forFeature([User])],
10    providers: [UsersService],
11    controllers: [UsersController],
12  })
13  export class UsersModule {}
```

## Контроллер UsersController:

```
TS app.module.ts TS users.controller.ts M X TS main.ts
src > users > TS users.controller.ts > UsersController > create
1  import { Body, Controller, Delete, Get, Param, Post } from '@nestjs/common';
2  import { CreateUserDto } from '../dto/create-user.dto';
3  import { User } from '../user.entity';
4  import { UsersService } from '../users.service';
5  import { ApiTags, ApiOperation, ApiResponse } from '@nestjs/swagger'
6
7
8  @ApiTags('users')
9  @Controller('users')
10 export class UsersController {
11     constructor(private readonly usersService: UsersService) {}
12
13     @ApiOperation({ summary: 'create new user' })
14     @ApiResponse({ status: 201, description: 'created' })
15     @ApiResponse({ status: 400, description: 'bad request' })
16     @ApiResponse({ status: 403, description: 'forbidden' })
17     @ApiResponse({ status: 501, description: 'not implemented' })
18     @Post()
19     async create(@Body() createUserDto: CreateUserDto): Promise<User> {
20         return await this.usersService.create(createUserDto);
21     }
22
23
24     @ApiOperation({ summary: 'get all users' })
25     @ApiResponse({ status: 200, description: 'all users found' })
26     @ApiResponse({ status: 401, description: 'not authorized' })
27     @ApiResponse({ status: 403, description: 'forbidden' })
28     @ApiResponse({ status: 501, description: 'not implemented' })
29     @Get()
30     async findAll(): Promise<User[]> {
31         return await this.usersService.findAll();
32     }
33
34
35     @ApiOperation({ summary: 'get user by id' })
36     @ApiResponse({ status: 200, description: 'user found' })
37     @ApiResponse({ status: 204, description: 'no content'})
38     @ApiResponse({ status: 401, description: 'not authorized' })
39     @ApiResponse({ status: 403, description: 'forbidden' })
40     @ApiResponse({ status: 501, description: 'not implemented' })
41     @Get('/:id')
42     async findOne(@Param('id') id: number): Promise<User> {
43         return await this.usersService.findOne(id);
44     }
45
46
47     @ApiOperation({ summary: 'delete user by id' })
48     @ApiResponse({ status: 200, description: 'user deleted' })
49     @ApiResponse({ status: 204, description: 'user not found'})
50     @ApiResponse({ status: 401, description: 'not authorized' })
51     @ApiResponse({ status: 403, description: 'forbidden' })
52     @ApiResponse({ status: 501, description: 'not implemented' })
53     @Delete('/:id')
54     async remove(@Param('id') id: number): Promise<void> {
55         return await this.usersService.remove(id);
56     }
57 }
```

## UserService:

```
TS app.module.ts TS users.controller.ts M TS users.service.ts X TS main.ts
src > users > TS users.service.ts > UsersService > create
1  import { Injectable, NotImplementedException } from '@nestjs/common';
2  import { InjectRepository } from '@nestjs/typeorm';
3  import { Repository } from 'typeorm';
4  import { CreateUserDto } from '../dto/create-user.dto';
5  import { User } from '../user.entity';
6
7
8  @Injectable()
9  export class UserService {
10     constructor(
11         @InjectRepository(User)
12         private readonly usersRepository: Repository<User>,
13     ) {}
14
15     create(createUserDto: CreateUserDto): Promise<User> {
16         throw new NotImplementedException();
17
18         // const user = new User();
19         // user.name = createUserDto.name;
20         // user.surname = createUserDto.surname;
21         // user.password = createUserDto.password;
22
23         // return this.usersRepository.save(user);
24     }
25
26     findAll(): Promise<User[]> {
27         return this.usersRepository.find();
28     }
29
30     findOne(id: number): Promise<User> {
31         return this.usersRepository.findOne(id);
32     }
33
34     async remove(id: number): Promise<void> {
35         await this.usersRepository.delete(id);
36     }
37 }
```

## Описание сущности user – user.entity:

```
TS app.module.ts TS users.controller.ts M TS user.entity.ts M X TS main.ts
src > users > TS user.entity.ts > ...
1 import { Entity, Column, PrimaryGeneratedColumn, OneToOne, JoinColumn, OneToMany } from 'typeorm';
2 import { ApiProperty } from '@nestjs/swagger';
3 import { Password } from '../passwords/password.entity';
4 import { Comment } from '../comments/comment.entity';
5
6
7 @Entity('Users')
8 export class User {
9   @PrimaryGeneratedColumn()
10   id: number;
11
12
13   @ApiProperty({
14     example: 'Ivan',
15     description: 'User\'s name'
16   })
17   @Column()
18   name: string;
19
20
21   @ApiProperty({
22     example: 'Ivanov',
23     description: 'User\'s surname'
24   })
25   @Column()
26   surname: string;
27
28
29   @ApiProperty({
30     example: 'superivan',
31     description: 'User\'s unique nickname for login'
32   })
33   @Column()
34   login: string;
35
36
37   @ApiProperty({
38     example: 'true',
39     description: 'Online/not online status'
40   })
41   @Column()
42   isOnline: boolean;
43
44
45   @ApiProperty({
46     description: 'User\'s password'
47   })
48   @OneToOne(() => Password)
49   @JoinColumn()
50   password: Password;
51
52
53   @ApiProperty({
54     description: 'User\'s comments'
55   })
56   @OneToMany(() => Comment, comments => comments.user)
57   comments: Comment[];
58 }
```

Реализация остальных модулей аналогична.

Для создания документации проекта был установлен пакет swagger:

```
npm i @nestjs/swagger swagger-ui-express
```

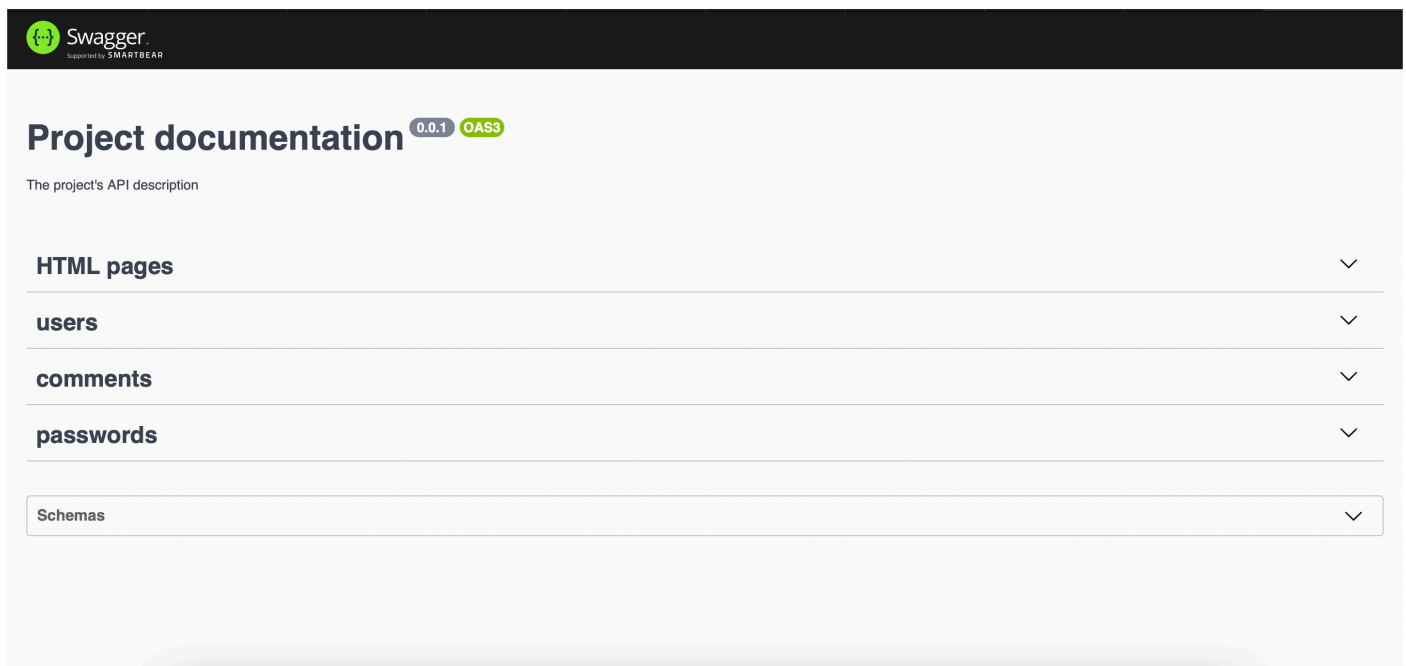
Инициализация swagger в функции bootstrap (main.ts):

```
19  const config = new DocumentBuilder()
20    .setTitle('Project documentation')
21    .setDescription('The project\'s API description')
22    .setVersion('0.0.1')
23    .build();
24  const documentation = SwaggerModule.createDocument(app, config);
25  SwaggerModule.setup('api', app, documentation);
```

Теперь документация доступна по маршруту /api. Для ее заполнения использовал декораторы:

- @ApiTags
- @ApiResponse
- @ApiProperty
- @ApiOperation

В результате:



## HTML pages



GET / get the main HTML page

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	OK	No links
308	redirect to the main page (index.html)	No links

GET /index.html get the main HTML page

GET /page.html get page.html

GET /plants.html get plants.html

GET /index.html get the main HTML page

Parameters

Cancel

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:3000/index.html' \
-H 'accept: */*'

```

Request URL

```
http://localhost:3000/index.html

```

Server response

Code	Details
200	Response body <pre>&lt;!DOCTYPE html&gt; &lt;html lang="ru"&gt; &lt;head&gt;   &lt;title&gt;&lt;/title&gt;   &lt;meta charset="utf-8"&gt;   &lt;meta name="author" content="Belonogov Evgeny"&gt;   &lt;meta name="description" content=" </pre>



# Project documentation

0.0.1 OAS3

The project's API description

## HTML pages

### users

POST	/users	create new user
GET	/users	get all users
GET	/users/{id}	get user by id
DELETE	/users/{id}	delete user by id

### comments

### passwords

### Schemas

## HTML pages

### users

POST	/users	create new user
Parameters		Try it out
No parameters		
Request body <small>required</small>		application/json
Example Value   Schema		
<pre>{   "name": "Ivan",   "surname": "Ivanov",   "login": "superivan",   "password": "qwerty" }</pre>		
Responses		
Code	Description	Links
201	created	No links

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:3000/users' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "Ivan",
    "surname": "Ivanov",
    "login": "superivan",
    "password": "qwerty"
  }'
```

Request URL

http://localhost:3000/users

Server response

Code

Details

501

Error: Not Implemented

Response body

```
{
  "statusCode": 501,
  "message": "Not Implemented"
}
```

Download

Response headers

```
connection: keep-alive
content-length: 46
```

CreateUserDto >

User ▾ {

name\*

string

example: Ivan

User's name

surname\*

string

example: Ivanov

User's surname

login\*

string

example: superivan

User's unique nickname for login

isOnline

boolean

example: true

Online/not online status

password\*

▾ {

description:

User's password

password\*

string

user's password

}

comments

▾ [

string

User's comments

]

}

CreateCommentDto >