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

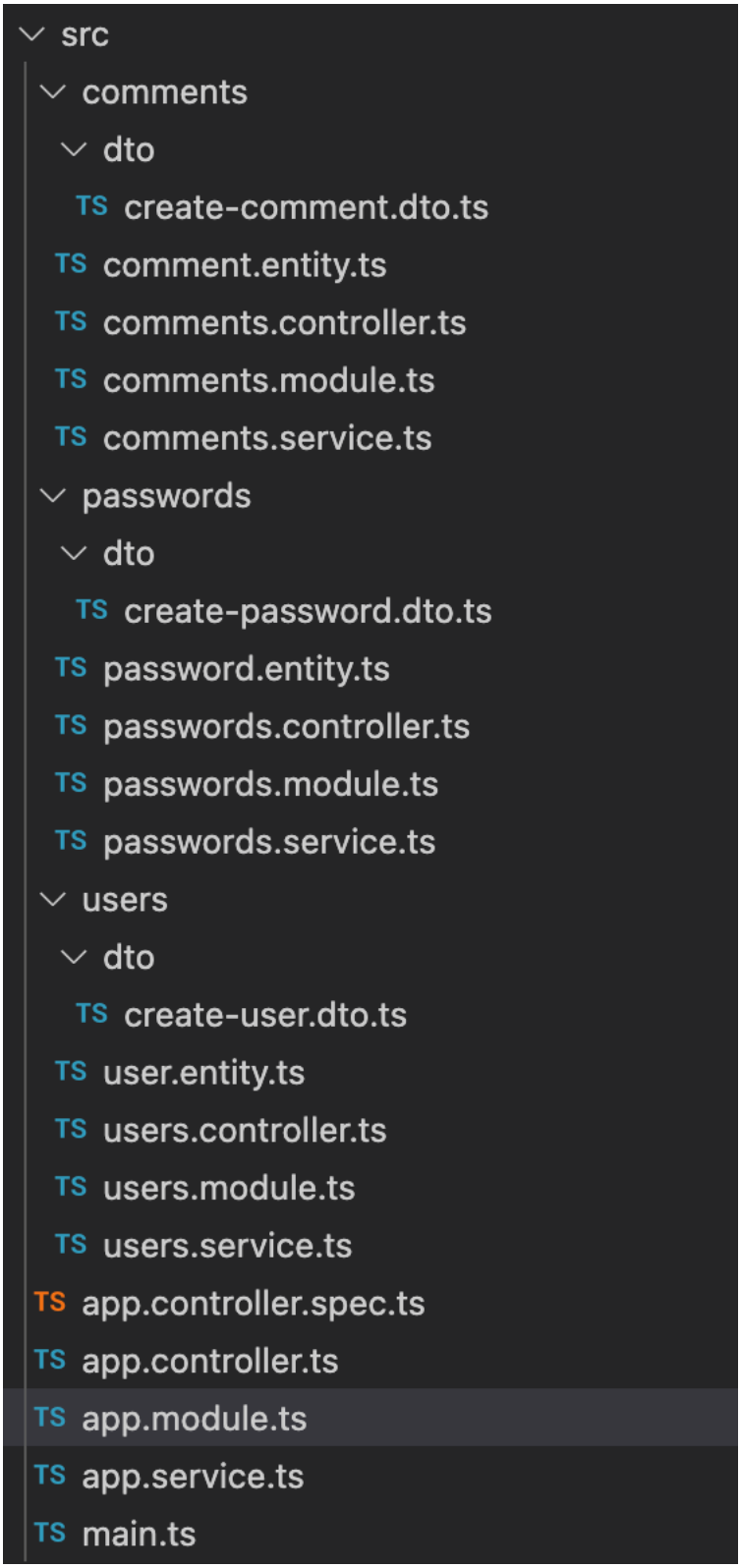
«Создание контроллеров страниц и спецификации»
по дисциплине «Web программирование»

Выполнил: студент группы M33122 Белоногов Е.В.

Санкт-Петербург
2022

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 (?)



Модуль users:

Контроллер UsersController:

UsersService:

app.module.ts | users.controller.ts M | users.service.ts | main.ts

src > users > users.service.ts > UsersService > create

```ts
import { Injectable, NotImplementedException } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { CreateUserDto } from './dto/create-user.dto';
import { User } from './user.entity';


@Injectable()
export class UsersService {
  constructor(
    @InjectRepository(User)
    private readonly usersRepository: Repository<User>,
  ) {}

  create(createUserDto: CreateUserDto): Promise<User> {
    throw new NotImplementedException();

    // const user = new User();
    // user.name = createUserDto.name;
    // user.surname = createUserDto.surname;
    // user.password = createUserDto.password;

    // return this.usersRepository.save(user);
  }

  findAll(): Promise<User[]> {
    return this.usersRepository.find();
  }

  findOne(id: number): Promise<User> {
    return this.usersRepository.findOne(id);
  }

  async remove(id: number): Promise<void> {
    await this.usersRepository.delete(id);
  }
}
```

Описание сущности user – user.entity:


Реализация остальных модулей аналогична.

Для создания документации проекта был установлен пакет swagger:

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

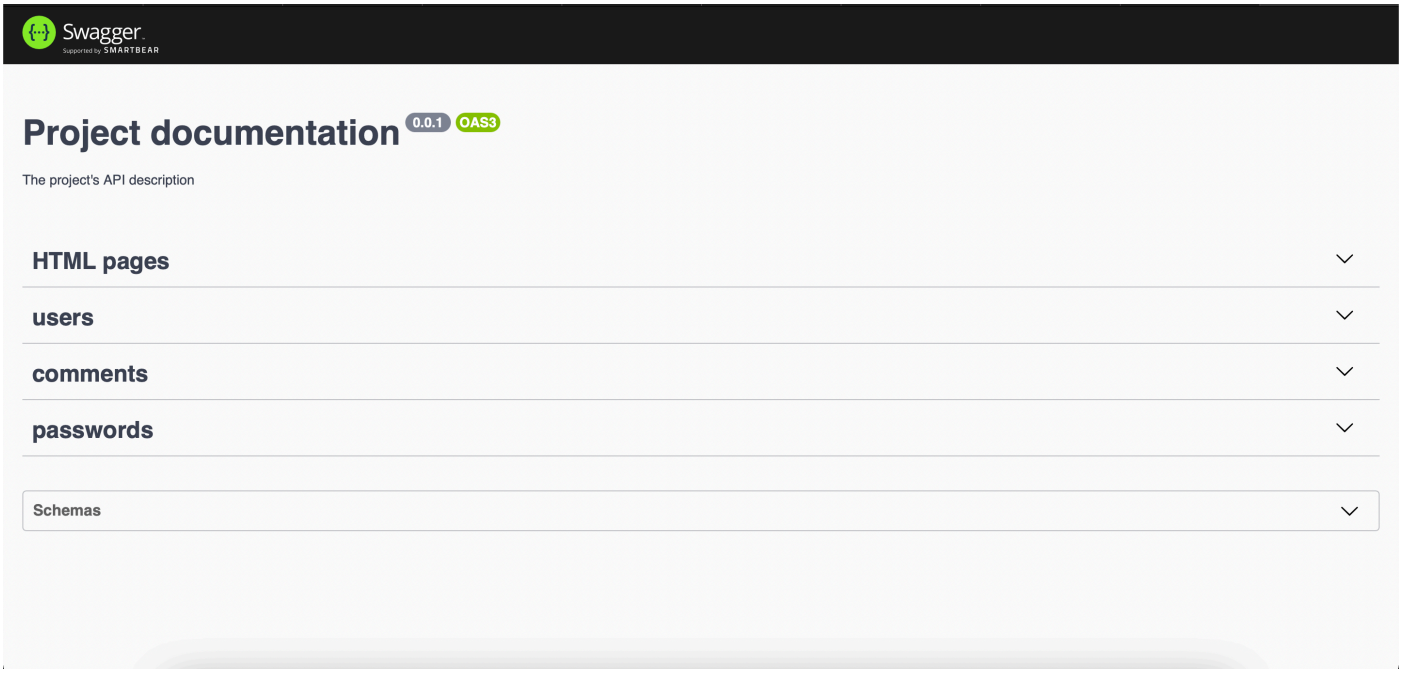
Инициализация swagger в функции bootstrap (main.ts):

```
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

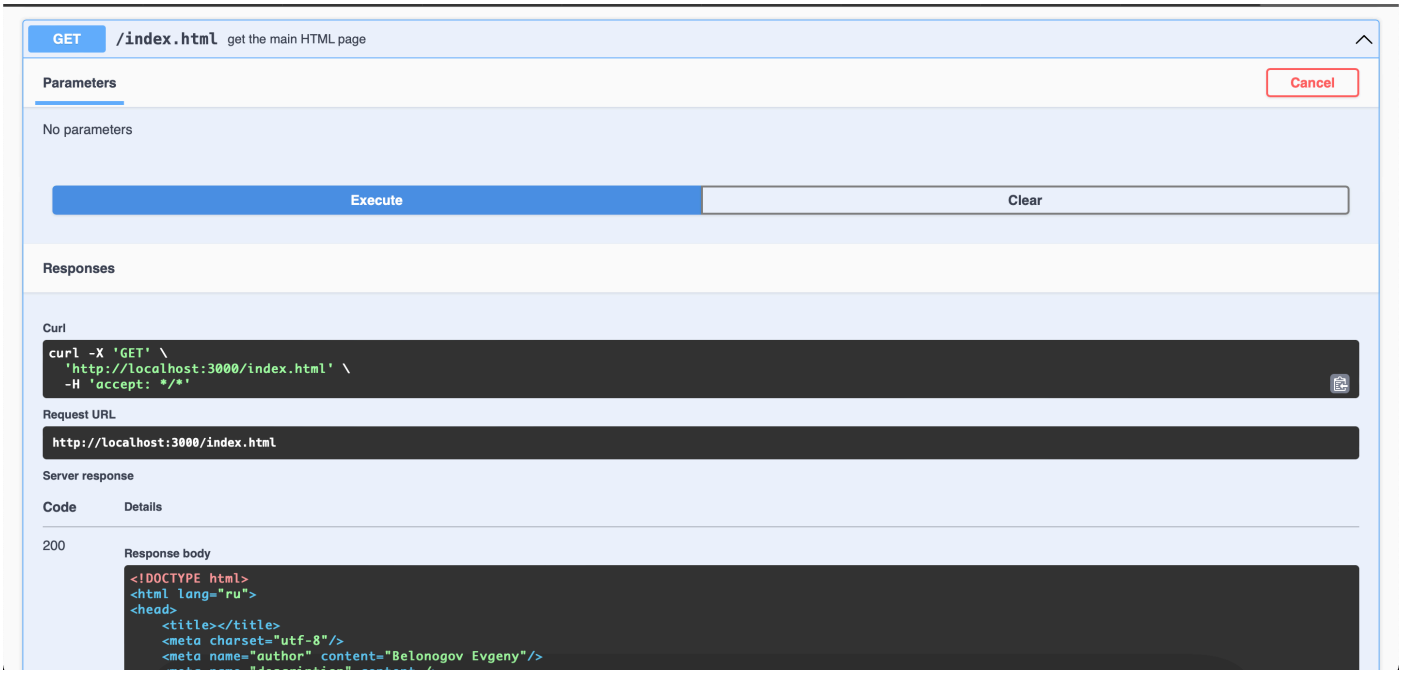
Responses

| 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



# 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 required application/json

Example Value | Schema

```
{
  "name": "Ivan",
  "surname": "Ivanov",
  "login": "superivan",
  "password": "qwerty"
}
```

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<br>*example: Ivan* | User's name |
| surname* | string<br>*example: Ivanov* | User's surname |
| login* | string<br>*example: superivan* | User's unique nickname for login |
| isOnline | boolean<br>*example: true* | Online/not online status |
| password* | ⌄ { description: User's password<br>password* string user's password<br>} | |
| comments | ⌄ [<br>string] | User's comments |

}

**CreateCommentDto** ›