МИНИСТЕРСТВО НАУКИ И 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 №04

По дисциплине «Web-программирование»

Создание контроллеров страниц и спецификации

Выполнила студентка группы №M33081

Ахмедова Лейла Таги кызы

Проверил

Приискалов Роман Андреевич

САНКТ-ПЕТЕРБУРГ

2022

Генерация для наших моделей

Модулей:

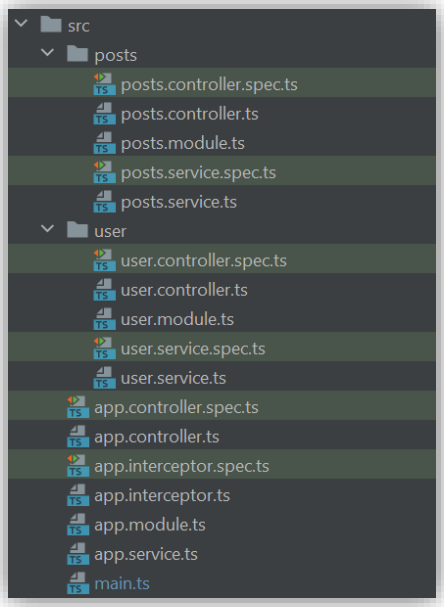
```
nest g module user
nest g module posts
```

Контроллеров:

```
nest g controller user
nest g controller posts
```

Сервисов:

```
nest g service user
nest g service posts
```



Добавляем новые модули в **app.modules.ts**

```ts
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { UserModule } from './user/user.module';
import { PostsModule } from './posts/posts.module';

@Module( metadata: {
  imports: [UserModule, PostsModule],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}
```

Создаем контроллеры для каждой модели

User

```ts
import { Get, Post, Delete, Param, Controller } from '@nestjs/common';
import { ApiOperation, ApiResponse } from '@nestjs/swagger';
import { User } from '@prisma/client';
import { UserService } from './user.service';

@ApiResponse( options: {
  status: 501,
  description: 'The method is not implemented.',
})
@Controller( prefix: 'users')
export class UserController {
  constructor(private readonly userService: UserService) {}

  @ApiOperation( options: { summary: 'Finding a user by ID and name' })
  @Get( path: ':user')
  async getUser(@Param( property: 'User id') id: number, username: string): Promise<User> {
    return await this.userService.findUser(id, username);
  }

  @Post( path: 'create')
  async createUser(email: string, username: string): Promise<User> {
    return await this.userService.createUser(email, username);
  }

  @Delete( path: ':user/delete')
  async deleteUser(
    @Param( property: 'User id') id: number,
    username: string,
  ): Promise<User> {
    return await this.userService.deleteUser(id, username);
  }
}
```

Posts

```ts
import { Get, Post, Delete, Param, Controller } from '@nestjs/common';
import { ApiOperation, ApiResponse } from '@nestjs/swagger';
import { Posts } from '@prisma/client';
import { PostsService } from './posts.service';

@ApiResponse( options: {
  status: 501,
  description: 'The method is not implemented.',
})
@Controller( prefix: 'posts')
export class PostsController {
  constructor(private readonly postsService: PostsService) {}

  @ApiOperation( options: { summary: 'Finding a post by ID and title' })
  @Get( path: ':post')
  async getPost(@Param( property: 'Post id') id: number, title: string): Promise<Posts> {
    return await this.postsService.findPost(id, title);
  }

  @Post( path: 'create')
  async createPost(title: string, content: string): Promise<Posts> {
    return await this.postsService.createPost(title, content);
  }

  @Delete( path: ':post/delete')
  async deletePost(
    @Param( property: 'Post id') id: number,
    title: string,
  ): Promise<Posts> {
    return await this.postsService.deletePost(id, title);
  }
}
```

Создаем сервисы для каждой модели

User

```ts
import { Injectable, NotImplementedException } from '@nestjs/common';
import { User } from '@prisma/client';

@Injectable()
export class UserService {
  async findUser(id: number, username: string): Promise<User> {
    throw new NotImplementedException();
  }

  async createUser(email: string, username: string): Promise<User> {
    throw new NotImplementedException();
  }

  async deleteUser(id: number, username: string): Promise<User> {
    throw new NotImplementedException();
  }
}
```

Posts

```ts
import { Injectable, NotImplementedException } from '@nestjs/common';
import { Posts } from '@prisma/client';

@Injectable()
export class PostsService {
  async findPost(id: number, title: string): Promise<Posts> {
    throw new NotImplementedException();
  }

  async createPost(title: string, content: string): Promise<Posts> {
    throw new NotImplementedException();
  }

  async deletePost(id: number, title: string): Promise<Posts> {
    throw new NotImplementedException();
  }
}
```

Устанавливаем Swagger

npm install –save @nestjs/swagger swagger-ui-express

Чтобы swagger работал,
нужно дописать **main.ts**

```ts
async function bootstrap() {
  const app = await NestFactory.create<NestExpressApplication>(AppModule);
  // eslint-disable-next-line @typescript-eslint/no-var-requires
  const hbs = require('hbs');
  app.useStaticAssets(join(__dirname, '..', 'public'));
  app.setBaseViewsDir(join(__dirname, '..', 'views'));
  hbs.registerPartials(join(__dirname, '..', 'views/partials'));
  const config = new DocumentBuilder()
    .setTitle('Posts API')
    .setDescription('An API for Posts')
    .setVersion('0.1')
    .addTag( name: 'Posts')
    .build();
  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup( path: 'api', app, document);
  app.setViewEngine('hbs');
  await app.listen(process.env.PORT || 3000);
}
bootstrap();
```

Запускаем проект, открываем http://localhost:12345/api