САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа

Выполнил:

Чебан Илья

Группа К3341

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

- Реализовать документацию средствами сваггер и постман
- |

Ход работы

Нужно сделать доку

Реализация в коде:

```
@ApiTags('deal')

@Controller('deal')

export class DealController {

 constructor(private readonly dealService: DealService) {}



 @Get()

 @ApiOperation({ summary: 'Получить все deal_service' })

 @ApiResponse({ status: 200, description: 'Список deals успешно получен' })

 @ApiBearerAuth()

 findAll() {

   return this.dealService.dealFindAll();

 }



 @Get(':id')

 @ApiOperation({ summary: 'Получить deal_service по ID' })

 @ApiResponse({ status: 200, description: 'deal_service успешно получена' })

 @ApiResponse({ status: 404, description: 'deal_service не найдена' })

 @ApiParam({ name: 'id', type: 'number', description: 'ID deal_service' })

 getUser(@Param('id', ParseIntPipe) id: number) {

   return this.dealService.dealGetById(id);

 }
```

```
@Post()
@UsePipes(new ValidationPipe())
@ApiOperation({ summary: 'Создать новую deal_service' })
@ApiResponse({ status: 201, description: 'deal_service успешно создана' })
@ApiResponse({ status: 400, description: 'Неверные данные' })
@ApiBody({ type: CreateDealDto })
@ApiBearerAuth()
create(@Body() dto: CreateDealDto) {
  return this.dealService.dealCreate(dto);
}


@Put(':id')
@UsePipes(new ValidationPipe())
@ApiOperation({ summary: 'Обновить deal_service' })
@ApiResponse({ status: 200, description: 'deal_service успешно обновлена' })
@ApiResponse({ status: 400, description: 'Неверные данные' })
@ApiResponse({ status: 404, description: 'deal_service не найдена' })
@ApiParam({ name: 'id', type: 'number', description: 'ID deal_service' })
@ApiBearerAuth()
update(@Param('id', ParseIntPipe) id: number, @Body() dto: TUpdateDealDto) {
  return this.dealService.dealUpdate(id, dto);
}


@Delete(':id')
@UsePipes(new ValidationPipe())
@ApiOperation({ summary: 'Удалить deal_service' })
@ApiResponse({ status: 200, description: 'deal_service успешно удалена' })
@ApiResponse({ status: 404, description: 'deal_service не найдена' })
@ApiParam({ name: 'id', type: 'number', description: 'ID deal_service' })
```

```
 @ApiBearerAuth()
 delete(@Param('id', ParseIntPipe) id: number) {
   return this.dealService.dealDelete(id);
 }
}
@ApiTags('message')
@Controller('message')
export class MessageController {
 constructor(private readonly messageService: MessageService) {}

 @Get()
 @ApiOperation({ summary: 'Получить все message_service' })
 @ApiResponse({
   status: 200,
   description: 'Список message_service успешно получен',
 })
 @ApiBearerAuth()
 findAll() {
   return this.messageService.messageFindAll();
 }

 @Get(':id')
 @ApiOperation({ summary: 'Получить message_service по ID' })
 @ApiResponse({ status: 200, description: 'message_service успешно получена' })
 @ApiResponse({ status: 404, description: 'message_service не найдена' })
 @ApiParam({ name: 'id', type: 'number', description: 'ID компании' })
 getUser(@Param('id', ParseIntPipe) id: number) {
   return this.messageService.messageGetById(id);
 }
```

```
@Post()
@UsePipes(new ValidationPipe())
@ApiOperation({ summary: 'Создать новую message_service' })
@ApiResponse({ status: 201, description: 'message_service успешно создана' })
@ApiResponse({ status: 400, description: 'Неверные данные' })
@ApiBody({ type: CreateMessageDto })
@ApiBearerAuth()
create(@Body() dto: CreateMessageDto) {
  return this.messageService.messageCreate(dto);
}


@Put(':id')
@UsePipes(new ValidationPipe())
@ApiOperation({ summary: 'Обновить message_service' })
@ApiResponse({
  status: 200,
  description: 'message_service успешно обновлена',
})
@ApiResponse({ status: 400, description: 'Неверные данные' })
@ApiResponse({ status: 404, description: 'message_service не найдена' })
@ApiParam({ name: 'id', type: 'number', description: 'ID message_service' })
@ApiBearerAuth()
update(
  @Param('id', ParseIntPipe) id: number,
  @Body() dto: TUpdateMessageDto,
) {
  return this.messageService.messageUpdate(id, dto);
}
```

```
 @Delete(':id')
 @UsePipes(new ValidationPipe())
 @ApiOperation({ summary: 'Удалить message_service' })
 @ApiResponse({ status: 200, description: 'message_service успешно удалена' })
 @ApiResponse({ status: 404, description: 'message_service не найдена' })
 @ApiParam({ name: 'id', type: 'number', description: 'ID message_service' })
 @ApiBearerAuth()
 delete(@Param('id', ParseIntPipe) id: number) {
   return this.messageService.messageDelete(id);
 }
}
@ApiTags('property')
@Controller('property')
export class PropertyController {
 constructor(private readonly propertyService: PropertyService) {}

 @Get()
 @ApiOperation({ summary: 'Получить все property_service' })
 @ApiResponse({
   status: 200,
   description: 'Список property_service успешно получен',
 })
 @ApiBearerAuth()
 findAll() {
   return this.propertyService.propertyFindAll();
 }

 @Get(':id')
```

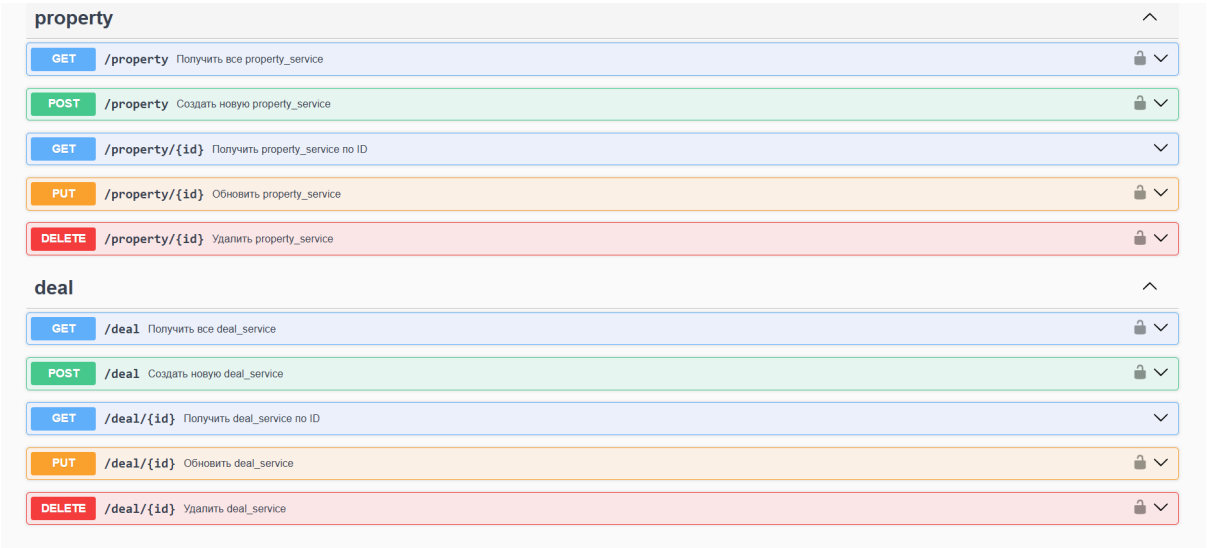
```
@ApiOperation({ summary: 'Получить property_service по ID' })
@ApiResponse({
  status: 200,
  description: 'property_service успешно получена',
})
@ApiResponse({ status: 404, description: 'property_service не найдена' })
@ApiParam({ name: 'id', type: 'number', description: 'ID property_service' })
getUser(@Param('id', ParseIntPipe) id: number) {
  return this.propertyService.propertyGetById(id);
}


@Post()
@UsePipes(new ValidationPipe())
@ApiOperation({ summary: 'Создать новую property_service' })
@ApiResponse({ status: 201, description: 'property_service успешно создана' })
@ApiResponse({ status: 400, description: 'Неверные данные' })
@ApiBody({ type: CreatePropertyDto })
@ApiBearerAuth()
create(@Body() dto: CreatePropertyDto) {
  return this.propertyService.propertyCreate(dto);
}


@Put(':id')
@UsePipes(new ValidationPipe())
@ApiOperation({ summary: 'Обновить property_service' })
@ApiResponse({
  status: 200,
  description: 'property_service успешно обновлена',
})
```

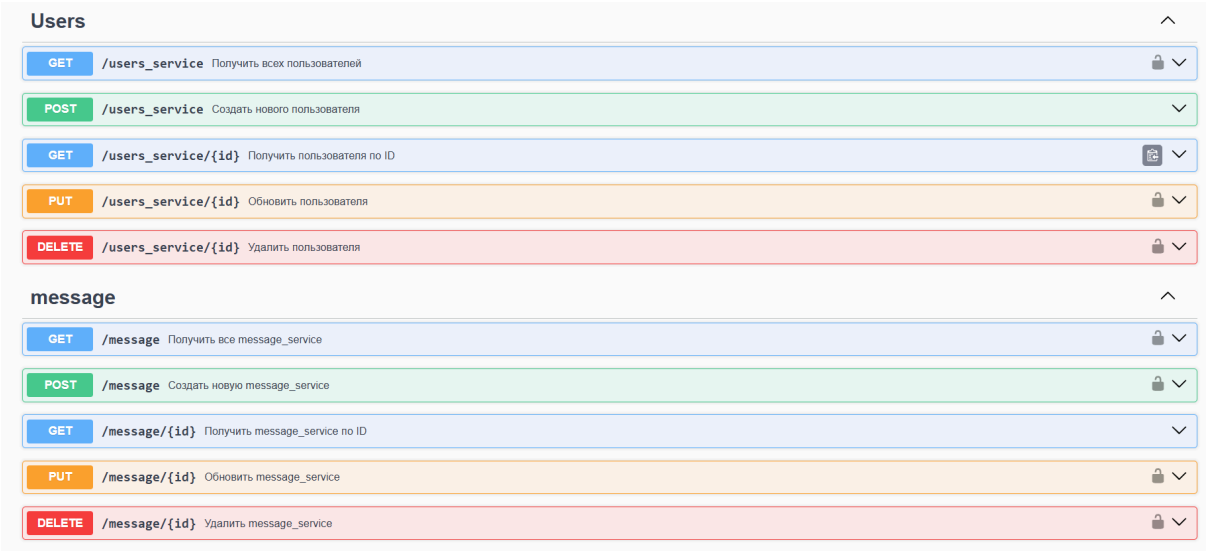
```
 @ApiResponse({ status: 400, description: 'Неверные данные' })
 @ApiResponse({ status: 404, description: 'property_service не найдена' })
 @ApiParam({ name: 'id', type: 'number', description: 'ID property_service' })
 @ApiBearerAuth()
 update(
   @Param('id', ParseIntPipe) id: number,
   @Body() dto: TUpdatePropertyDto,
 ) {
   return this.propertyService.propertyUpdate(id, dto);
 }


 @Delete(':id')
 @UsePipes(new ValidationPipe())
 @ApiOperation({ summary: 'Удалить property_service' })
 @ApiResponse({ status: 200, description: 'property_service успешно удалена' })
 @ApiResponse({ status: 404, description: 'property_service не найдена' })
 @ApiParam({ name: 'id', type: 'number', description: 'ID property_service' })
 @ApiBearerAuth()
 delete(@Param('id', ParseIntPipe) id: number) {
   return this.propertyService.propertyDelete(id);
 }
}
@ApiTags('Users')
@Controller('users_service')
export class UsersController {
 constructor(private readonly usersService: UsersService) {}


 @Get()
 @ApiOperation({ summary: 'Получить всех пользователей' })
```

```
@ApiResponse({
  status: 200,
  description: 'Список пользователей успешно получен',
})
@ApiBearerAuth()
findAll() {
  return this.usersService.userFindAll();
}

@Get(':id')
@ApiOperation({ summary: 'Получить пользователя по ID' })
@ApiResponse({ status: 200, description: 'Пользователь успешно получен' })
@ApiResponse({ status: 404, description: 'Пользователь не найден' })
@ApiParam({ name: 'id', type: 'number', description: 'ID пользователя' })
getUser(@Param('id', ParseIntPipe) id: number) {
  return this.usersService.userGetById(id);
}

@Post()
@UsePipes(new ValidationPipe())
@ApiOperation({ summary: 'Создать нового пользователя' })
@ApiResponse({ status: 201, description: 'Пользователь успешно создан' })
@ApiResponse({ status: 400, description: 'Неверные данные' })
@ApiBody({ type: CreateUsersDto })
create(@Body() dto: CreateUsersDto) {
  return this.usersService.userCreate(dto);
}

@Put(':id')
```

```
@UsePipes(new ValidationPipe())

@ApiOperation({ summary: 'Обновить пользователя' })

@ApiResponse({ status: 200, description: 'Пользователь успешно обновлен' })

@ApiResponse({ status: 400, description: 'Неверные данные' })

@ApiResponse({ status: 404, description: 'Пользователь не найден' })

@ApiParam({ name: 'id', type: 'number', description: 'ID пользователя' })

@ApiBearerAuth()

update(@Param('id', ParseIntPipe) id: number, @Body() dto: TUpdateUsersDto) {

  return this.usersService.userUpdate(id, dto);

}



@Delete(':id')

@UsePipes(new ValidationPipe())

@ApiOperation({ summary: 'Удалить пользователя' })

@ApiResponse({ status: 200, description: 'Пользователь успешно удален' })

@ApiResponse({ status: 404, description: 'Пользователь не найден' })

@ApiParam({ name: 'id', type: 'number', description: 'ID пользователя' })

@ApiBearerAuth()

delete(@Param('id', ParseIntPipe) id: number) {

  return this.usersService.userDelete(id);

}

}
```

Документация в сваггере:





Вывод: реализовали документацию в сваггере