**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

# **Дисциплина:** Бэк-энд разработка

Отчет

Лабораторная работа #2

Выполнил:

Игнатьев Алексей

К3340

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

**Задание:**

По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

**Ход работы**

1. **Определение необходимого набора методов API**

В ходе анализа бизнес требований проекта выявлены следующие необходимые методы API-

Авторизация

регистрация

Админские методы для создания / добавления / редактирования аукционов

Клиентские методы для просмотра, бронирования аукционов

Возможность сделать ставку в аукционе

Возможность добавлять отзыв о товаре победителем

Фильтрация аукционов по названию / категории / брендам

пополнение баланса для ставок

уведомления о предстоящих аукционах забронированных

1. **Реализация методов API средствами NestJS**

В ходе разработки создана реализация вышеперечисленных методов, в том числе подключены следующие сервисы – хранилище файлов minio, сервис рассылки писем RegRu, очереди отправки EMAIL писем в SMTP сервисы

Пример реализации контроллера аукциона  
import {  
 Body,  
 Controller,  
 Delete,  
 Get,  
 Param,  
 ParseIntPipe,  
 Patch,  
 Post,  
 Query,  
 Request,  
 UploadedFile,  
 UploadedFiles,  
 UseGuards,  
 UseInterceptors,  
} from '@nestjs/common';  
import { ConfigService } from '@nestjs/config';  
import { FileInterceptor, FilesInterceptor } from '@nestjs/platform-express';  
import { commonControllerFactory } from 'src/common/common.controller-default';  
import { StatusOKDto } from 'src/common/dto/status.dto';  
import { AuctionsService } from './auctions.service';  
import { CreateAuctionDto } from './dto/create-auction.dto';  
import { UpdateAuctionDto } from './dto/update-auction.dto';  
import { AuctionEntity } from './entities/auction.entity';  
import { AuctionParams } from './params/auction.params.dto';  
import {  
 ApiBasicAuth,  
 ApiBearerAuth,  
 ApiConsumes,  
 ApiQuery,  
 ApiResponse,  
} from '@nestjs/swagger';  
import {  
 AuctionCategory,  
 AuctionParticipant,  
 AuctionStatus,  
 AuctionType,  
} from '@prisma/client';  
import { BasicAuthGuard } from 'src/common/guards/basic-auth.guard';  
import { StorageService } from 'src/config/s3/s3.service';  
import { JwtAuthGuard } from 'src/auth/jwt.guard';  
import { JwtUserPayloadDto } from 'src/auth/dto/jwt.dto';  
import { BasicJwtCombineGuard } from 'src/common/guards/basic-jwt-combine.guard';  
  
const CommonController = commonControllerFactory<AuctionEntity>({  
 entity: AuctionEntity,  
});  
  
@Controller('auctions')  
export class AuctionsController extends CommonController {  
 constructor(private readonly auctionsService: AuctionsService) {  
 super(auctionsService);  
 }  
 @ApiBearerAuth('Bearer')  
 @ApiBearerAuth('Basic')  
 @UseGuards(BasicJwtCombineGuard)  
 @Get()  
 async findAll(  
 @Query() params: AuctionParams,  
 @Request() req?: { user?: JwtUserPayloadDto; admin?: boolean },  
 ): Promise<AuctionEntity[]> {  
 const isAdmin = req?.*admin* || false;  
  
 return await this.auctionsService.findAll(params, req?.*user*?.id, isAdmin);  
 }  
  
 @ApiResponse({ type: AuctionEntity })  
 @ApiBearerAuth('Bearer')  
 @ApiBearerAuth('Basic')  
 @UseGuards(BasicJwtCombineGuard)  
 @Get(':id')  
 async findOne(  
 @Param('id', ParseIntPipe) id: number,  
 @Request() req?: { user?: JwtUserPayloadDto },  
 ): Promise<AuctionEntity | null> {  
 return await this.auctionsService.findOne(id, req?.*user*?.id);  
 }  
  
 @ApiBearerAuth('Basic')  
 @ApiConsumes('multipart/form-data')  
 @Post()  
 @UseGuards(BasicAuthGuard)  
 @UseInterceptors(FilesInterceptor('images', 10))  
 async create(  
 @Body() createAuctionDto: CreateAuctionDto,  
 @UploadedFiles() files?: Express.Multer.File[],  
 ): Promise<AuctionEntity> {  
 return await this.auctionsService.create(createAuctionDto, files);  
 }  
 @ApiBearerAuth('Basic')  
 @UseGuards(BasicAuthGuard)  
 @Delete('participant/:id')  
 async deleteAuctionParticipant(  
 @Param('id', ParseIntPipe) participantId: number,  
 ): Promise<AuctionParticipant> {  
 return await this.auctionsService.deleteAuctionParticipant(participantId);  
 }  
  
 @ApiBearerAuth('Basic')  
 @ApiConsumes('multipart/form-data')  
 @Patch(':id')  
 @UseGuards(BasicAuthGuard)  
 @UseInterceptors(FilesInterceptor('images', 10))  
 async update(  
 @Param('id', ParseIntPipe) id: number,  
 updateAuctionDto: UpdateAuctionDto,  
 @UploadedFiles() files?: Express.Multer.File[],  
 ): Promise<AuctionEntity> {  
 return await this.auctionsService.update(id, updateAuctionDto, files);  
 }  
 @ApiBearerAuth('Bearer')  
 @Post('book-auction/:auctionId')  
 @UseGuards(JwtAuthGuard)  
 async bookAuction(  
 @Param('auctionId', ParseIntPipe) auctionId: number,  
 @Request() req: { user: JwtUserPayloadDto },  
 ) {  
 return await this.auctionsService.bookAuction(req.*user*.id, auctionId);  
 }  
 @ApiBearerAuth('Bearer')  
 @Post('make-rate/:auctionId')  
 @UseGuards(JwtAuthGuard)  
 async makeRate(  
 @Request() req: { user: JwtUserPayloadDto },  
 @Param('auctionId') auctionId: number,  
 ) {  
 return await this.auctionsService.makeRate(req.*user*.id, auctionId);  
 }  
 @ApiBearerAuth('Basic')  
 @Delete(':id')  
 @UseGuards(BasicAuthGuard)  
 async remove(@Param('id', ParseIntPipe) id: number): Promise<StatusOKDto> {  
 return await this.auctionsService.remove(id);  
 }  
  
 @ApiBearerAuth('Bearer')  
 @UseGuards(JwtAuthGuard)  
 @Post(':id/favorite')  
 async addToFavorites(  
 @Param('id', ParseIntPipe) id: number,  
 @Request() req: { user: JwtUserPayloadDto },  
 ) {  
 return this.auctionsService.addToFavorites(req.*user*.id, Number(id));  
 }  
  
 @ApiBearerAuth('Bearer')  
 @UseGuards(JwtAuthGuard)  
 @Delete(':id/favorite')  
 async removeFromFavorites(  
 @Param('id', ParseIntPipe) id: number,  
 @Request() req: { user: JwtUserPayloadDto },  
 ) {  
 return this.auctionsService.removeFromFavorites(req.*user*.id, Number(id));  
 }  
}

**Вывод**

* В ходе выполнения лабораторной работы выполнена реализация Restful api, с использованием функционала NestJS
* Подключены сервисы, использующиеся в бизнес логике (Minio и SMTP)