**First assignment: Basic CRUD API**

1. Create a basic nest project using nest CLI
2. Create new nest module (players module) using the nest CLI

Q: Name all places the module is declared.

1. Create new controller (players controller) using the nest CLI

Q: Name all places the controller is declared.

Q: Why the controller was not declared in the app module?

1. Create new service (players service) using the nest CLI

Q: Name all places the service is declared.

Q: Why the service was not declared in the app module?

Code:

1. Create a get API – get allPlayers
2. Create a get API – get playerbById
3. Create post API – create player – (use DTO)
4. Create put API – update player - (use DTO)
5. Create delete API – delete player

Notes:

1. Don’t use real data, create a JSON with 3 objects in a file and use it as a mock DB
2. Use as many decorators as you can:

Graphical user interface, application, Teams

Description automatically generated

Table

Description automatically generated with low confidence

Second assignment: Basic CRUD validations

1. Create a pipe validates the ID of the players contains only numbers
2. Create a DTO validation (without the id):
   1. First name: mandatory, only string
   2. Last name: mandatory, only string
   3. PPG – mandatory, only number

**First assignment: Basic CRUD API - cheat sheet:**

**1.First lets create the project using the command:**

**nest new nba-project**

**cd ./nba-project**

**2.secondly lets generate the module, controller and service using the nest cli:**

nest g module players

nest g controller players

nest g service players

**3. now let’s create the CRUD in the controller**

**src/players/players.controller.ts**

import { Body, Controller, Delete, Get, Param, ParseIntPipe, Post, Put } from '@nestjs/common';

import { CreatePlayerDto } from './dto/create-player.dto';

import { PlayersService } from './players.service';

import { UpdatePlayerDto } from './dto/update-player.dto';

import { Player } from './entities/player.entity';

@Controller('players')

export class PlayersController {

constructor(private playersService: PlayersService){}

@Get()

getAllPlayers(): Player[]{

return this.playersService.getAllPlayers()

}

@Get(':id')

getPlayerById(@Param('id', ParseIntPipe) id: number): Player{

return this.playersService.getPlayerById(id)

}

@Post()

createPlayer(@Body() createPlayerDto: CreatePlayerDto): Player[]{

return this.playersService.createPlayer(createPlayerDto)

}

@Put(':id')

updatePlayer(

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

@Body() updatePlayerDto: UpdatePlayerDto

){

return this.playersService.updatePlayer(id,updatePlayerDto)

}

@Delete(':id')

removePlayer(

@Param('id') id: number,

){

return this.playersService.removePlayer(id)

}

}

**4.Then lets create the playersDto (data transfer object)**

**src/players/dto/create-player.dto.ts**

export class CreatePlayerDto{

readonly firstName: string

readonly lastName: string

readonly ppg : string

}

**src/players/dto/update-player.dto.ts**

npm i @nestjs/mapped-types

import { PartialType } from '@nestjs/mapped-types';

import { CreatePlayerDto } from './create-player.dto';

export class UpdatePlayerDto extends PartialType(CreatePlayerDto) {}

**5.create the player entity**

nest g class player

and change the name of the file to player.entity.ts

**src/players/entities/player.entity.ts**

export class Player {

id: number

firstName: string

lastName: string

ppg: number

}

**6.Lastly lets create the players service**

import { Injectable } from '@nestjs/common';

import { CreatePlayerDto } from './dto/create-player.dto';

import { UpdatePlayerDto } from './dto/update-player.dto';

import { Player } from './entities/player.entity';

@Injectable()

export class PlayersService {

players: Player[] = [

{

id: 1,

firstName: "Michael",

lastName: "Jordan",

ppg: 30

},

{

id: 2,

firstName: "Lebron",

lastName: "James",

ppg: 27

},

{

id: 3,

firstName: "Kobe",

lastName: "Brayent",

ppg: 25

}

]

getAllPlayers(): Player[]{

return [...this.players]

}

getPlayerById(id:number): Player{

const player = this.findPlayerById(id)

return {...player}

}

createPlayer(createPlayerDto: CreatePlayerDto): Player[]{

const player: Player = {

id: this.createId(),

firstName: createPlayerDto.firstName,

lastName: createPlayerDto.lastName,

ppg: parseInt(createPlayerDto.ppg)

}

this.players.push(player)

return [...this.players]

}

updatePlayer(id:number, updatePlayerDto: UpdatePlayerDto): Player{

let player = this.findPlayerById(id)

const playerIdx = this.players.findIndex((player) => {

return player.id === id

})

console.log(updatePlayerDto);

console.log(player);

player = this.checkIfPlayerNeedsUpdate(updatePlayerDto, player);

this.players[playerIdx] = player;

return {...this.players[playerIdx]}

}

removePlayer(id:number): Player[]{

const playerIdx = this.players.findIndex((player)=>{

return player.id === id

})

this.players.splice(playerIdx,1);

return [...this.players]

}

private createId(){

return Math.floor(Math.random() \* Math.floor(1000))

}

private checkIfPlayerNeedsUpdate(updatePlayerDto: UpdatePlayerDto, player: Player ){

if(updatePlayerDto.firstName){

player.firstName = updatePlayerDto.firstName

}

if(updatePlayerDto.lastName){

player.lastName = updatePlayerDto.lastName

}

if(updatePlayerDto.ppg){

player.ppg = parseInt(updatePlayerDto.ppg)

}

return player

}

private findPlayerById(id){

return this.players.find((player)=>{

return player.id === id

})

}

}

**7.Test!**

**Second assignment: Basic CRUD API - cheat sheet:**

1.

npm i class-transformer class-validator

2.

import { ValidationPipe } from '@nestjs/common';

import { NestFactory } from '@nestjs/core';

import { AppModule } from './app.module';

async function bootstrap() {

const app = await NestFactory.create(AppModule);

**app.useGlobalPipes(new ValidationPipe());**

await app.listen(3000);

}

bootstrap();

3.

src/players/dto/create-player.dto.ts

import { IsString, IsNotEmpty, Matches, MinLength } from 'class-validator';

export class CreatePlayerDto{

@IsNotEmpty()

@IsString()

readonly firstName: string

@IsNotEmpty()

@IsString()

readonly lastName: string

@IsNotEmpty()

@IsString()

readonly ppg : string

}

4.

Nest g pipe idvalidation

src/players/pipes/idvalidation.pipe.ts

import {BadRequestException, Injectable, PipeTransform } from '@nestjs/common';

@Injectable()

export class IdvalidationPipe implements PipeTransform {

transform(value: any) {

if(value < 1 || value > 1000 ){

throw new BadRequestException('id parameter is not a valid ID');

}

return value;

}

}

5. update the controller

import { Body, Controller, Delete, Get, Param, ParseIntPipe, Post, Put, UsePipes } from '@nestjs/common';

import { CreatePlayerDto } from './dto/create-player.dto';

import { PlayersService } from './players.service';

import { UpdatePlayerDto } from './dto/update-player.dto';

import { Player } from './entities/player.entity';

import { IdvalidationPipe } from './pipes/idvalidation.pipe';

@Controller('players')

@UsePipes(IdvalidationPipe)

export class PlayersController {

constructor(private playersService: PlayersService){}

@Get()

getAllPlayers(): Player[]{

return this.playersService.getAllPlayers()

}

@Get(':id')

// @UsePipes(IdvalidationPipe)

getPlayerById(@Param('id', ParseIntPipe) id: number): Player{

return this.playersService.getPlayerById(id)

}

@Post()

createPlayer(@Body() createPlayerDto: CreatePlayerDto): Player[]{

return this.playersService.createPlayer(createPlayerDto)

}

@Put(':id')

// @UsePipes(IdvalidationPipe)

updatePlayer(

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

@Body() updatePlayerDto: UpdatePlayerDto

){

return this.playersService.updatePlayer(id,updatePlayerDto)

}

@Delete(':id')

// @UsePipes(IdvalidationPipe)

removePlayer(

@Param('id') id: number,

){

return this.playersService.removePlayer(id)

}

}

**6.Test!**