**Developing REST APIS**

|  |  |
| --- | --- |
| **SPRINGBOOT** | **NESTJS** |
| @RestController(“users”) | @controller(‘users’) |
| @GetMapping 🡪 get all the users | @Get() |
| @GetMapping(“/{id}”)  @PathVariable Integer id) | @Get(‘/:id’)  @Param() params) 🡪 console.log(params.id);  @Param(‘id’) id: int 🡪 console.log(id); |
| @PostMapping  @RequestBody User user | @Post()  @Body() user: User |
| @DeleteMapping, @UpdateMapping, @PatchMapping | @delete(), @update(), @patch() |
|  |  |
|  |  |

NestJS

----------------------------

Nest (NestJS) is a framework for building efficient, scalable Node.js server-side applications. It uses progressive JavaScript,

is built with and fully supports TypeScript (yet still enables developers to code in pure JavaScript) and

combines elements of OOP (Object Oriented Programming), FP (Functional Programming), and FRP (Functional Reactive Programming).

Under the hood, Nest makes use of robust HTTP Server frameworks like Express (the default) and optionally can be configured to use Fastify as well!

npm i -g @nestjs/cli

nest new project-name

npm run start

http://localhost:3000/

npm run start:dev (To watch for changes in your files)

nest g controller tasks -->To create a controller using the CLI, simply execute the

nest g module tasks

nest g service tasks

npm install uuid

import { v4 as uuidv4 } from 'uuid';

uuidv4(); --> id: uuidv4(),

@Post() -- send parametrs in x-www-form-urlencoded

public createTask(

@Body('name') name,

@Body('description') description,

): Task {

return this.\_taskService.createTask(name, description);

}

}

@Get('/:id')

public getTaskById(@Param('uuid', new ParseUUIDPipe()) uuid: string): Task {

return this.\_taskService.getTaskById(uuid);

}

@Post()

public createTask(@Body() createTaskDto: CreateTaskDto): Task {

return this.\_taskService.createTask(createTaskDto);

}

@Patch('/:id/status')

public updateTaskStatus(

@Param() params,

@Body('status') status: TaskStatus,

): Task {

return this.\_taskService.updateTaskStatus(params.id, status);

}

validations and pipes

----------------------------

npm i --save class-validator class-transformer

use this in main.ts --> app.useGlobalPipes(new ValidationPipe()); --> import { ValidationPipe } from '@nestjs/common';

TypeORM

--------------------------

npm install typeorm @nestjs/typeorm pg