TP-11

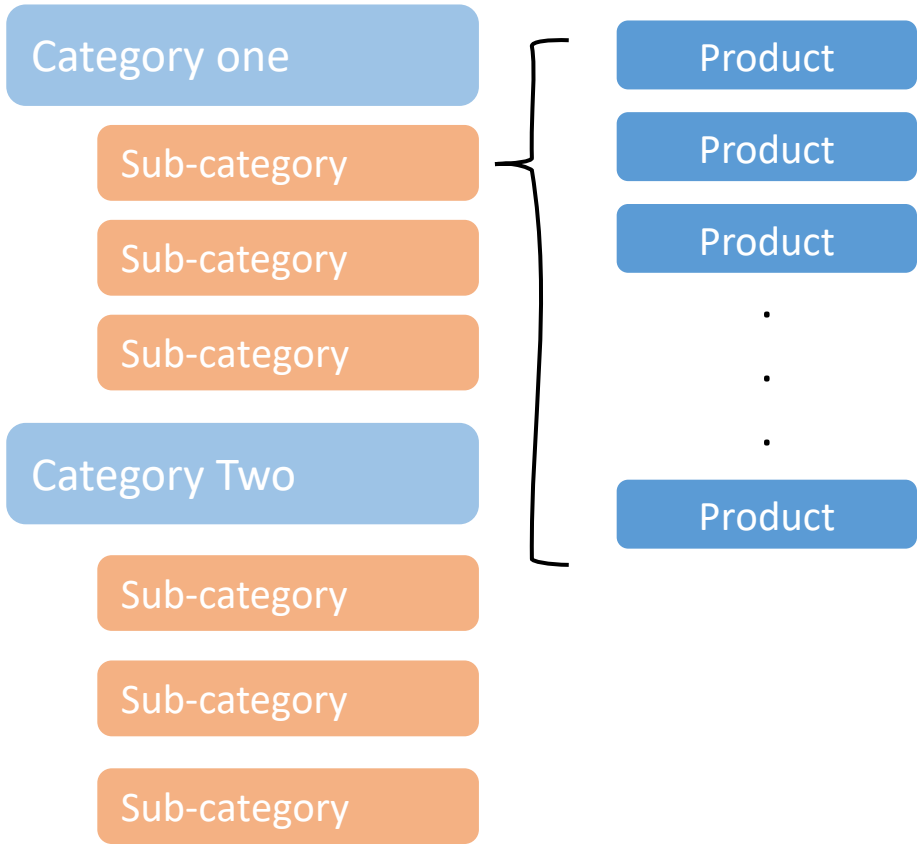
VueJS, NodeJS

CRUD APIs, Mongoose

TP11 Exercise

TP11.1: CRUD Apis

Create APIs to manage products by categorizing/sub-categorizing.



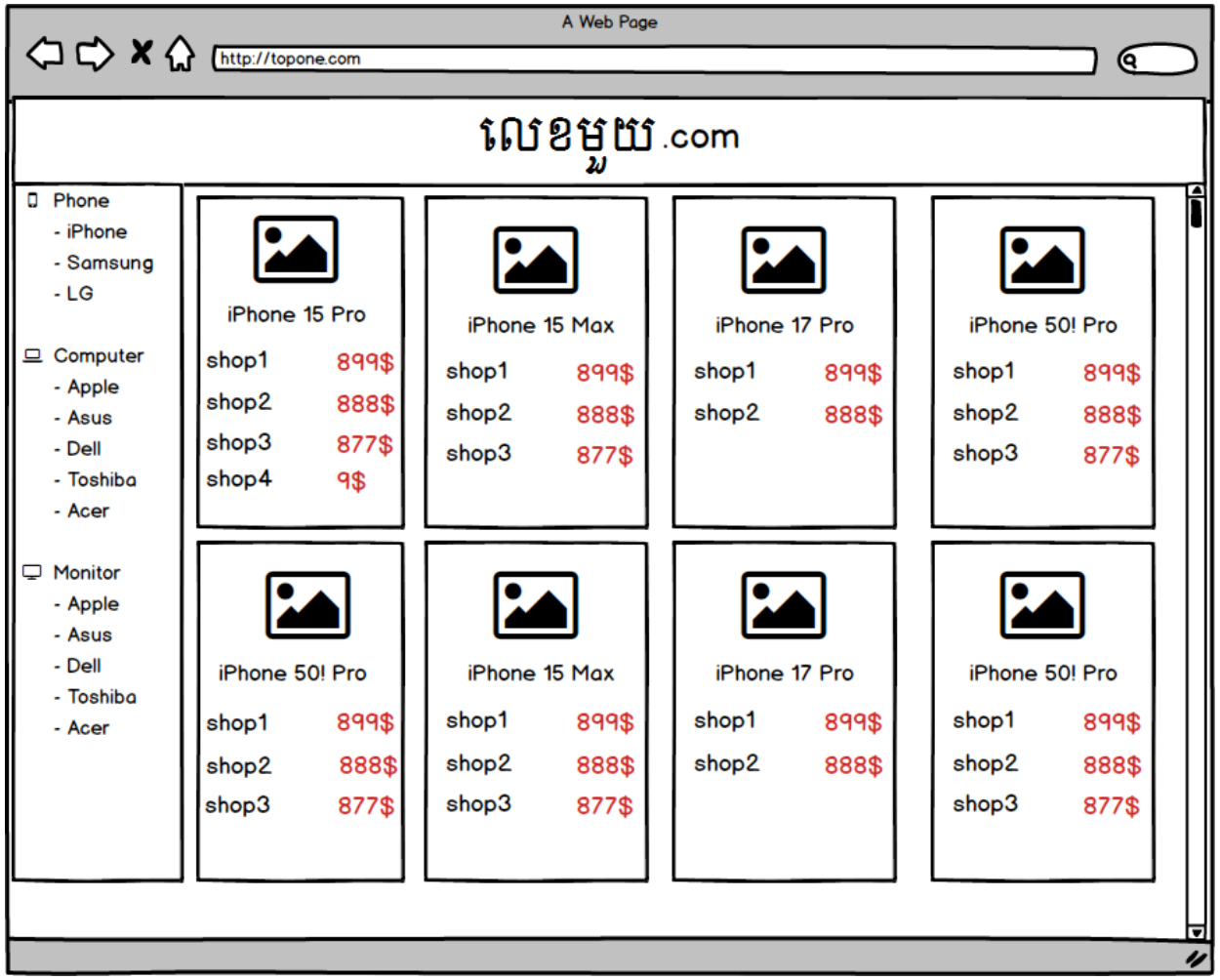
Makes sure:

- You have CRUD APIs for category
- You have CRUD APIs for sub-category
- You have CRUD APIs for product

TP11 Exercise

TP11.2: TopOne app

Design a basic website template to compare product price tags from different sources.



Makes sure:

☞ Category and Sub-category are listed according to the data from APIs

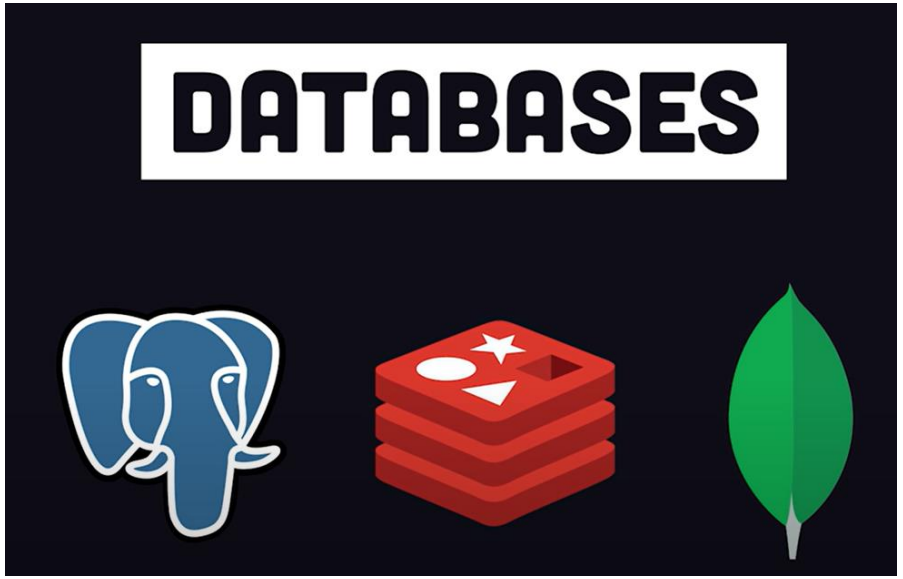
☞ All products listed according to the data from APIs (*price tags can be ignored*)

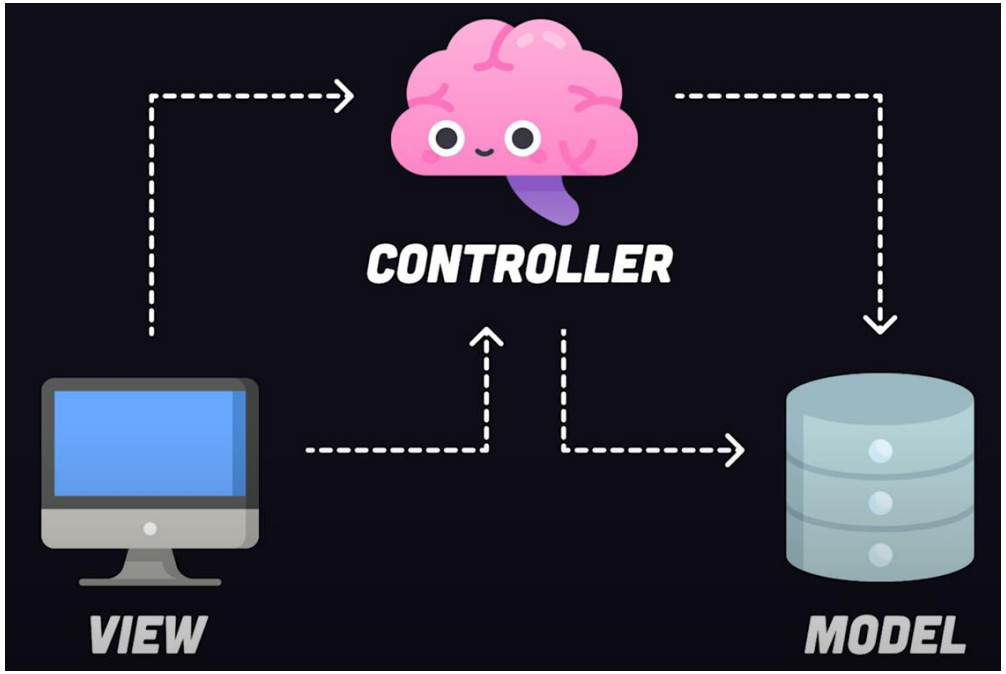
Getting to learn another new Thing 🤤

“NestJS (Node Server Framework)”

Why NestJS?? 🤔

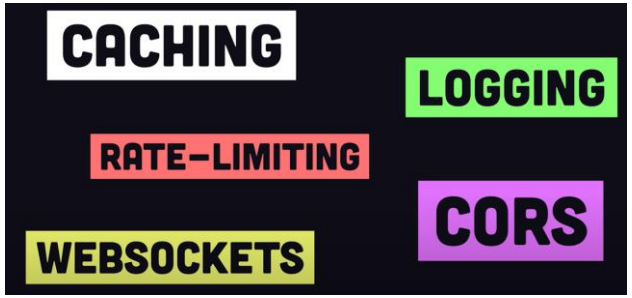

API
REST

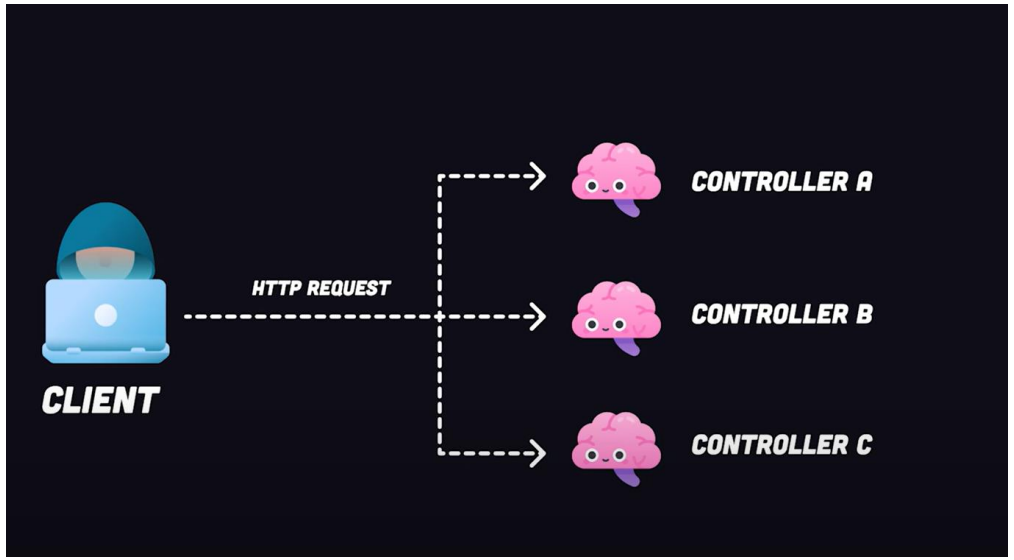

DATABASES

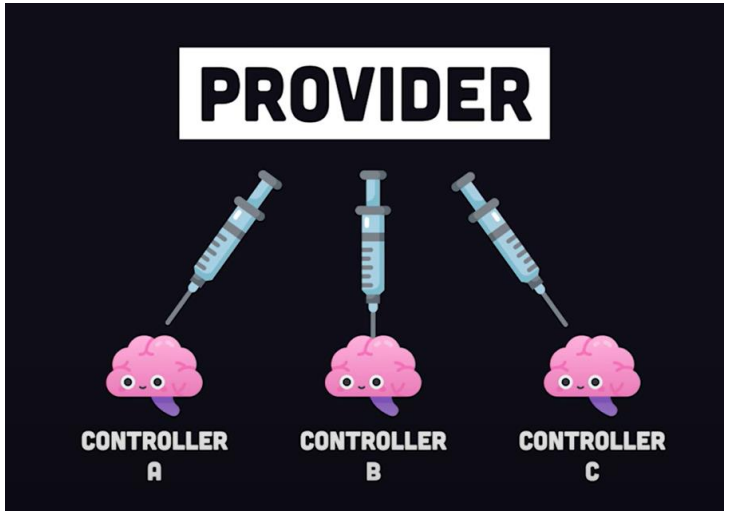

CONTROLLER
VIEW
MODEL


STREAMING

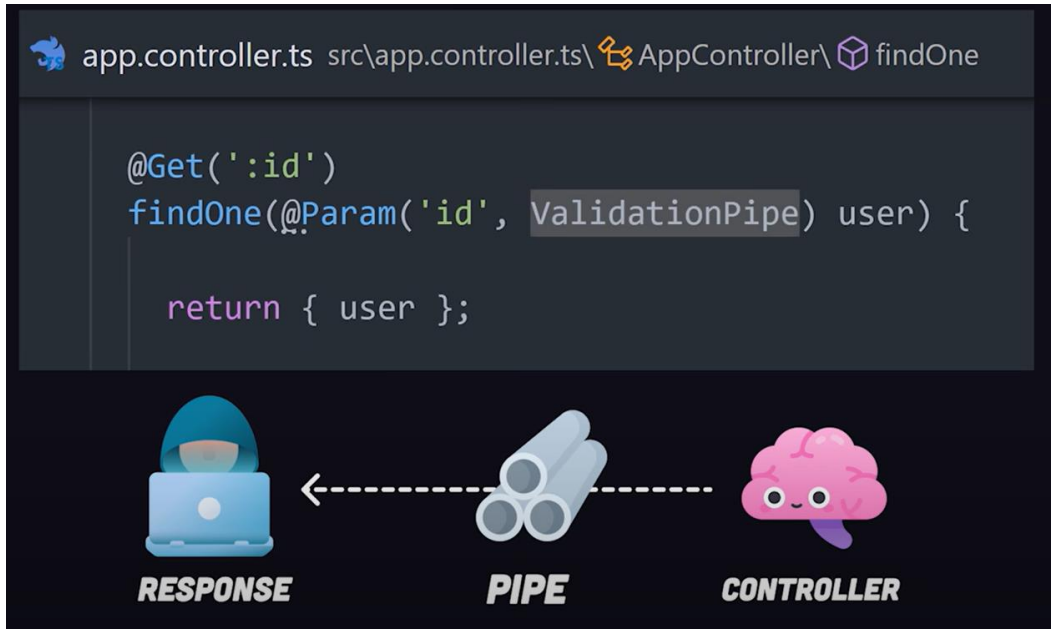

SECURITY


CACHING
LOGGING
RATE-LIMITING
CORS
WEBSOCKETS


CONTROLLER A
HTTP REQUEST
CONTROLLER B
CLIENT
CONTROLLER C


PROVIDER
CONTROLLER A
CONTROLLER B
CONTROLLER C


auth.guard.ts 1 src\auth.guard.ts\ AuthGuard\ canActivate
@Injectable()
export class AuthGuard implements CanActivate {
canActivate() {
if (isAdmin) {
return true;
}
REQUEST
GUARD
CONTROLLER


app.controller.ts src\app.controller.ts\ AppController\ findOne
@Get(':id')
findOne(@Param('id', ValidationPipe) user) {
return { user };
RESPONSE
PIPE
CONTROLLER


@Module({
imports: [],
controllers: [AppController, DogController],
providers: [AppService],
})
export class AppModule {}
MODULE
ORGANIZE AND LAZY-LOAD

Get started with a basic understanding

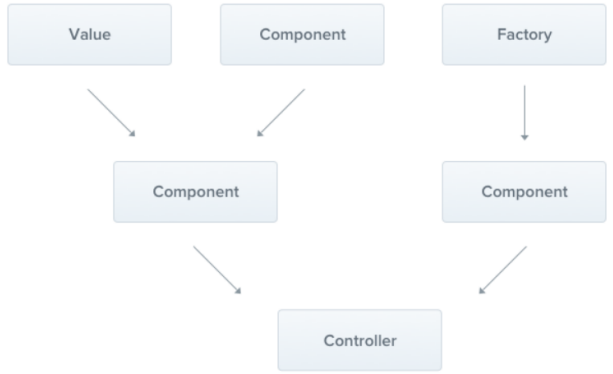
https://docs.nestjs.com/

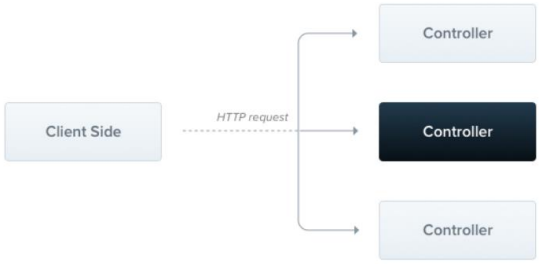
https://www.digitalocean.com/community/tutorials/getting-started-with-nestjs

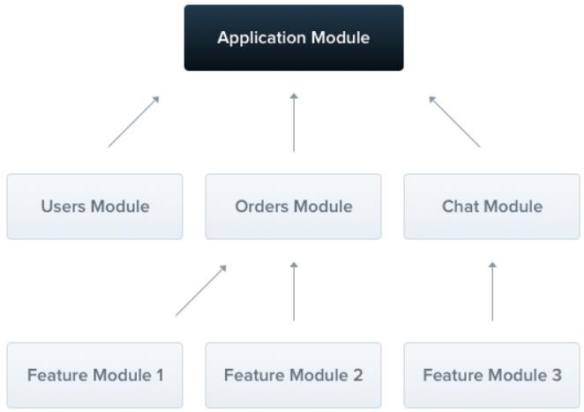
+ Understanding the Building blocks of Nest.js

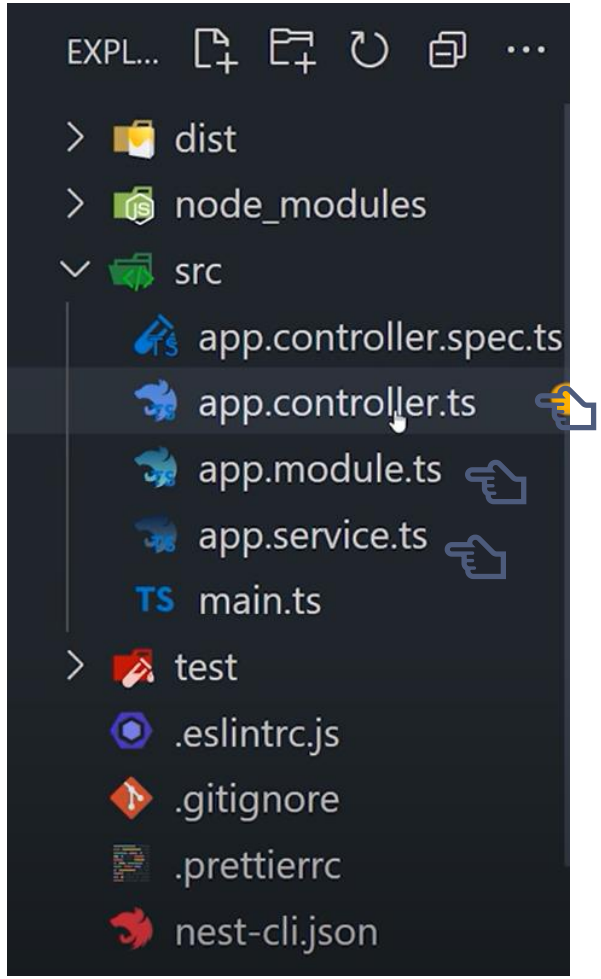
The following are the building blocks used when building Nest.js applications:

- **Controllers**
- **Providers**
- **Modules**









- Controller

```
 [label users.controller.ts]

import { Controller, Get } from '@nestjs/common';

@Controller('users')
export class UsersController {
 @Get()
 findAll() {
   return 'This will return all the users';
 }
}
```

- Service (Provider)

```
import { Injectable } from '@nestjs/common';
import { User } from './interfaces/user.interface';

@Injectable()
export class UsersService {
  private readonly users: User[] = [];

  create(user: User) {
    this.users.push(user);   }

  findAll(): User[] {
    return this.users;
  }
}
```

- Module

```
...
import { UsersModule } from './users/users.module';

@Module({
  ...
})

export class AppModule { }
```

```
import { Module } from '@nestjs/common';
import { UsersController } from './users.controller.ts';
import { UsersService } from './users.service.ts';

@Module({
  controllers: [UsersController],
  providers: [UsersService]
})

export class UsersModule {}
```

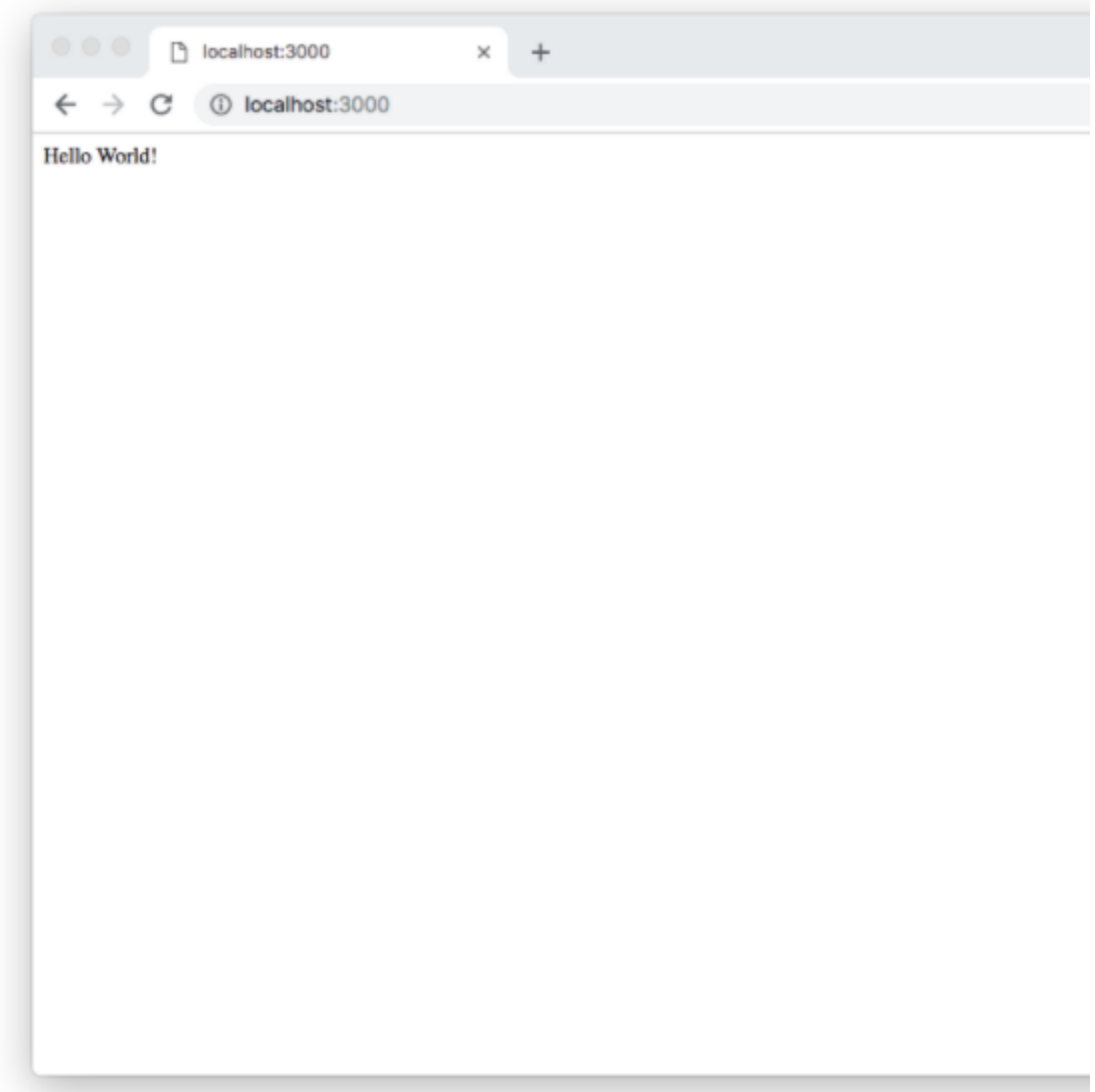
Step 1 – Installing Nest.js

```
$ npm install -g @nestjs/cli
```

```
$ nest new bookstore-nest
```

```
$ cd bookstore-nest
```

```
$ npm run start
```



- Development

```
// start the application using nodemon
npm run start:dev
```

Step 2 – Generating a Module

```
$ nest generate module books
```

src/books/books/module.ts

```
import { Module } from '@nestjs/common';
@Module({})
export class BooksModule {}
```

Step 3 – Setting up a Service

```
nest generate service books
```

[label src/mocks/books.mock.ts]

```
export const BOOKS = [
    { id: 1, title: 'First book', description: "This is the description for the f
    { id: 2, title: 'Second book', description: "This is the description for the
    { id: 3, title: 'Third book', description: "This is the description for the t
    { id: 4, title: 'Fourth book', description: "This is the description for the
    { id: 5, title: 'Fifth book', description: "This is the description for the f
    { id: 6, title: 'Sixth book', description: "This is the description for the s
];
```

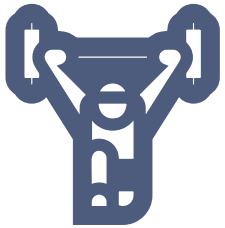
[label /src/books/books.service.ts]

```
  import { Injectable, HttpException } from '@nestjs/common';
  import { BOOKS } from '../mocks/books.mock';

  @Injectable()
  export class BooksService {
      books = BOOKS;

      getBooks(): Promise<any> {
          return new Promise(resolve => {
              resolve(this.books);
          });
      }
      getBook(bookID): Promise<any> {
          let id = Number(bookID);
          return new Promise(resolve => {
              const book = this.books.find(book => book.id === id);
              if (!book) {
                  throw new HttpException('Book does not exist!', 404);
              }
              resolve(book);
          });
      }
  }
```

```
import { Injectable, HttpException } from '@nestjs/common';
import { BOOKS } from '../mocks/books.mock';
@Injectable()
export class BooksService {
    books = BOOKS;
    ...
    addBook(book): Promise<any> {
        return new Promise(resolve => {
            this.books.push(book);
            resolve(this.books);
        });
    }
}
```

Step 4 – Creating Routes and Controllers/ Injecting the Service into the Controller

```
$ nest generate controller books
```



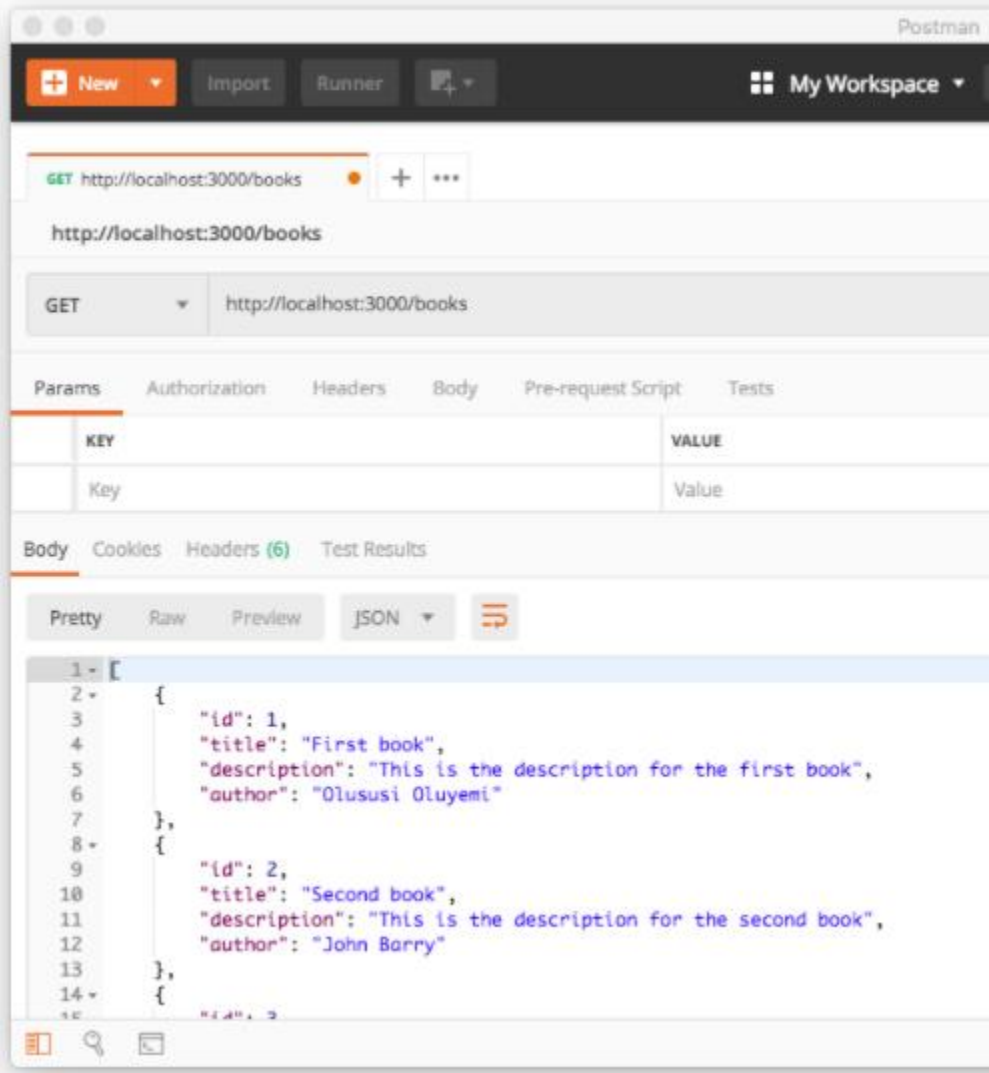
```
import { Controller, Get, Param, Post, Body, Query, Delete } from '@nestjs/common';
import { BooksService } from './books.service';
import { CreateBookDTO } from './dto/create-book.dto';

@Controller('books')
export class BooksController {
    constructor(private booksService: BooksService) { }

    @Get()
    async getBooks() {
        const books = await this.booksService.getBooks();
        return books;
    }

    @Get(':bookID')
    async getBook(@Param('bookID') bookID) {
        const book = await this.booksService.getBook(bookID);
        return book;
    }

    @Post()
    async addBook(@Body() createBookDTO: CreateBookDTO) {
        const book = await this.booksService.addBook(createBookDTO);
        return book;
    }

    @Delete()
    async deleteBook(@Query() query) {
        const books = await this.booksService.deleteBook(query.bookID);
        return books;
    }
}
```

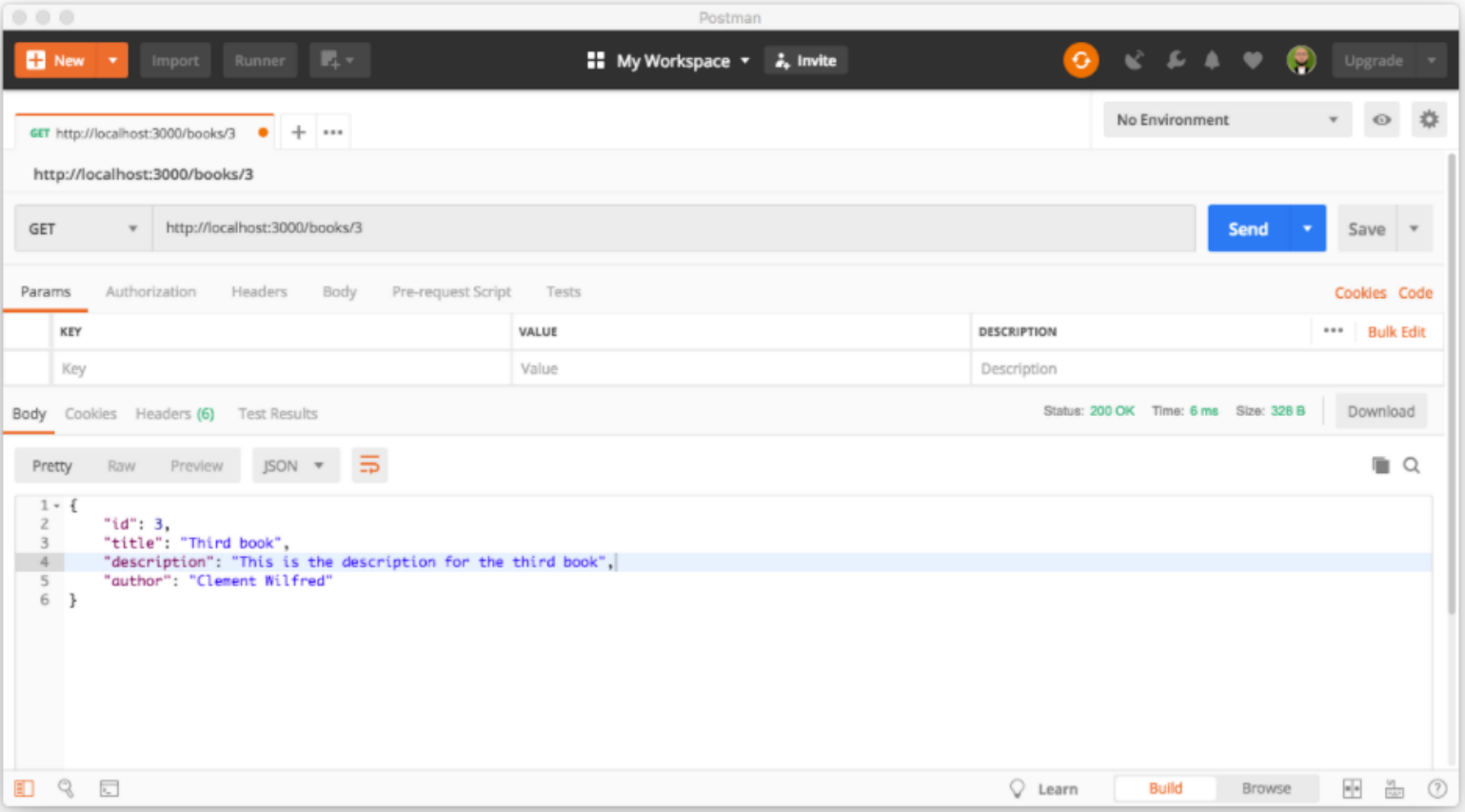
Step 5 – Importing Controller/Service into the Module

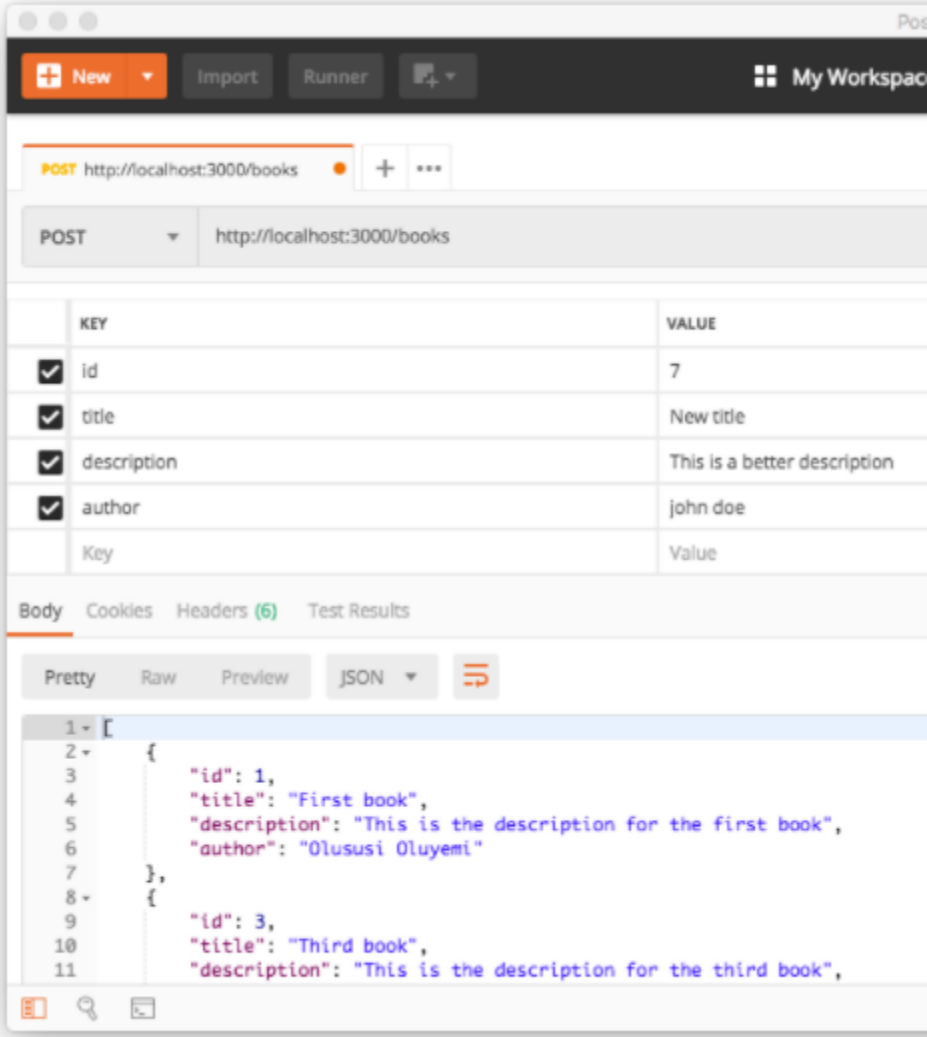
```
import { Module } from '@nestjs/common';
import { BooksController } from './books.controller';
import { BooksService } from './books.service';
@Module({
  controllers: [BooksController],
  providers: [BooksService]
})
export class BooksModule {}
```

Start the application again if it is not running at the moment with:

```
$ npm run start
```




Postman
New
Import
Runner
My Workspace
Invite
Upgrade
GET http://localhost:3000/books/3
No Environment
http://localhost:3000/books/3
GET
http://localhost:3000/books/3
Send
Save
Params
Authorization
Headers
Body
Pre-request Script
Tests
Cookies
Code
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (6)
Test Results
Status: 200 OK
Time: 6 ms
Size: 328 B
Download
Pretty
Raw
Preview
JSON
{
"id": 3,
"title": "Third book",
"description": "This is the description for the third book",
"author": "Clement Wilfred"
}
Learn
Build
Browse


New
Import
Runner
My Workspac
POST http://localhost:3000/books
POST
http://localhost:3000/books
KEY
VALUE
id
7
title
New title
description
This is a better description
author
john doe
Key
Value
Body
Cookies
Headers (6)
Test Results
Pretty
Raw
Preview
JSON
[
{
"id": 1,
"title": "First book",
"description": "This is the description for the first book",
"author": "Olususi Oluyemi"
},
{
"id": 3,
"title": "Third book",
"description": "This is the description for the third book",

Good luck 👌