Ministerul Educației și Cercetării al Republicii Moldova

Colegiul Universității Tehnice al Moldovei

**RAPORT**

**Disciplina:** asistenta pentru servere-side

Tema lucrării: salon auto

Elevul: Dorogoi Iulian

Grupa:   PAPP-221

Specialitatea:   Programarea si analiza produselor program

Chișinău 2025

Cuprins

[Laboratorul 1 2](#_Toc208907078)

[Laboratorul 2 2](#_Toc208907079)

[Laboratorul 3 2](#_Toc208907080)

[Laboratorul 4 2](#_Toc208907081)

[Laboratorul 5 2](#_Toc208907082)

# Laboratorul 1

Scopul:

Scopul acestui laborator este dezvoltarea unei aplicații NestJS pentru administrarea configurărilor unui salon auto, implementând rutare modulară, filtre, separare Admin/Public și controlul accesului pe roluri.

Obiectivele:

 Crearea modulelor și controller-elor pentru entitățile principale (configurări, utilizatori, rezervări, manageri).

 Implementarea endpoint-urilor REST: listare, detalii, filtrare.

 Separarea rutelor Admin/Public și implementarea middleware-ului pentru roluri.

 Testarea aplicației folosind Postman.

Realizarea sarcinilor pas cu pas:

**Nivelul 5 – Listă completă**

*  Endpoint: /configurari/list
* Cod controller:

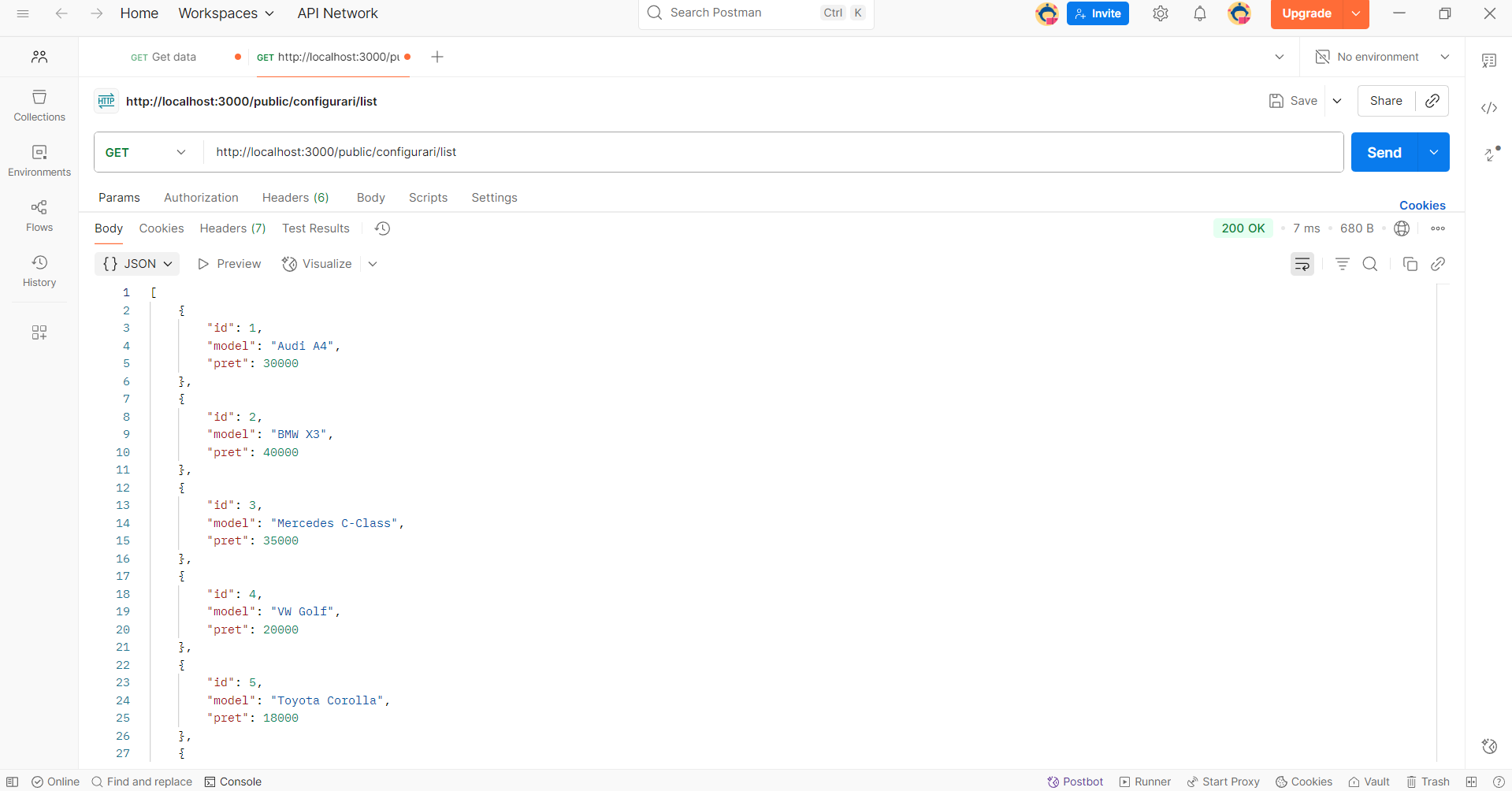
@Get('list')

findAll() {

return this.configurariService.findAll();

}

Testare Postman: URL: GET http://localhost:3000/configurari/list

****

**Nivelul 6 – Detalii după ID**

* Endpoint: /configurari/details/:id
* Cod controller:

@Get('details/:id')

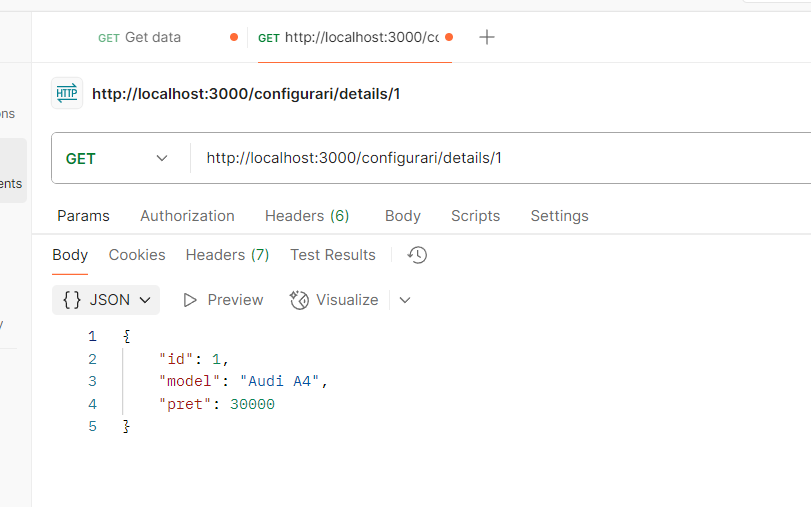
findOne(@Param('id') id: string) {

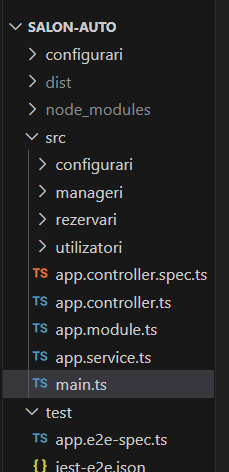
return this.configurariService.findOne(+id);

}

Testare Postman:

* URL: GET <http://localhost:3000/configurari/details/1>



**Nivelul 7 – Modularizare**

* Module create: configurari, utilizatori, rezervari, manageri
* Comenzi NestJS:

nest generate module utilizatori

nest generate module rezervari

nest generate module manageri

 Fiecare modul are propriul controller și service.

 **Poza recomandată:** structură foldere VSCode pentru module.

**Nivelul 8 – Endpoint /search cu query params**

* Endpoint: /configurari/search
* Cod controller:

@Get('search')

search(

@Query('model') model: string,

@Query('minPrice') minPrice: string,

@Query('maxPrice') maxPrice: string

) {

return this.configurariService.search(model, +minPrice, +maxPrice);

}

**Cod service :**

search(model?: string, min?: number, max?: number) {

return this.configurari.filter(c =>

(!model || c.model.toLowerCase().includes(model.toLowerCase())) &&

(!min || c.pret >= min) &&

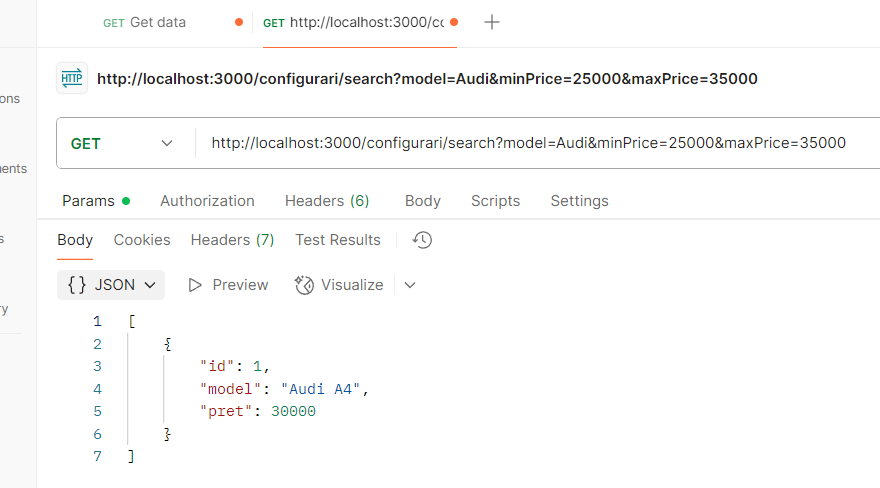
(!max || c.pret <= max)

);

}

**Testare Postman:**

* URL: GET <http://localhost:3000/configurari/search?model=Audi&minPrice=25000&maxPrice=35000>



**Nivelul 9 – Separare Admin/Public**

* Controller Public: configurari-public.controller.ts

@Controller('public/configurari')

export class ConfigurariPublicController {

@Get('list')

list() {

return this.configurariService.findAll();

}

}

* Controller Admin: configurari-admin.controller.ts

@Controller('admin/configurari')

export class ConfigurariAdminController {

@Get('edit/:id')

edit(@Param('id') id: string) {

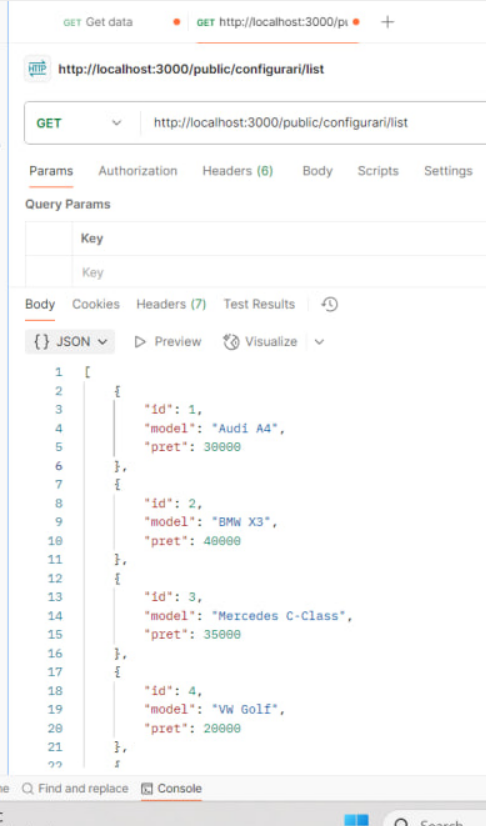
return `Edit configurare ${id}`;

}

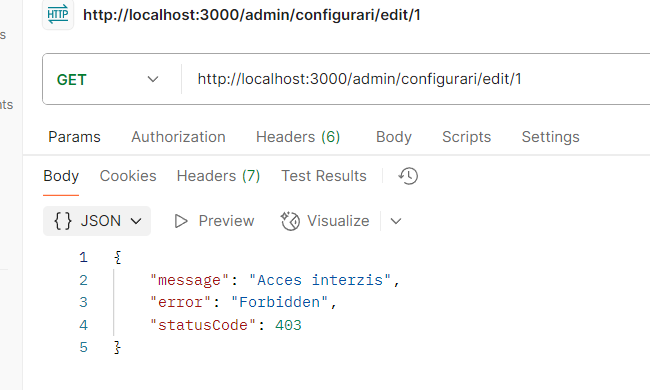
}

Testare Postman:

* /public/configurari/list → acces liber



* /admin/configurari/edit/1 → acces Admin



**Nivelul 10 – Middleware pentru roluri**

* Middleware: roles.middleware.ts

@Injectable()

export class RolesMiddleware implements NestMiddleware {

use(req: Request, res: Response, next: NextFunction) {

const role = req.headers['role'];

if (!role || role !== 'admin') {

throw new ForbiddenException('Acces interzis');

}

next();

}

}

* Aplicare în modul Configurari:

export class ConfigurariModule implements NestModule {

configure(consumer: MiddlewareConsumer) {

consumer

.apply(RolesMiddleware)

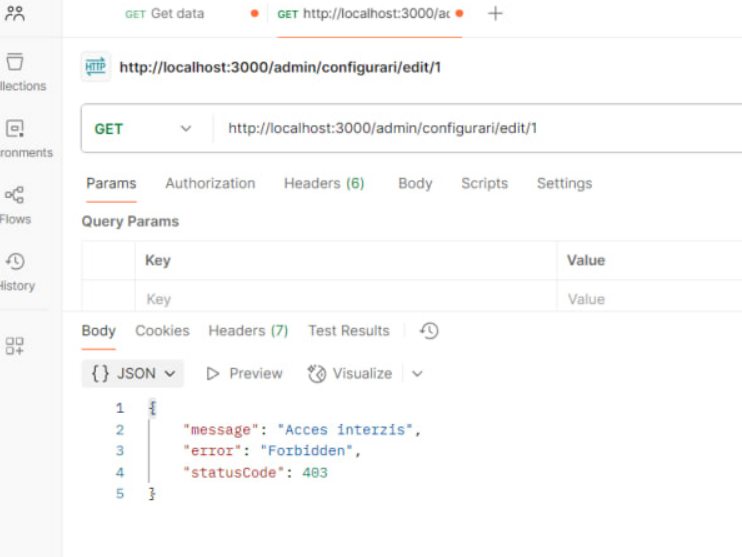
.forRoutes(ConfigurariAdminController);

}

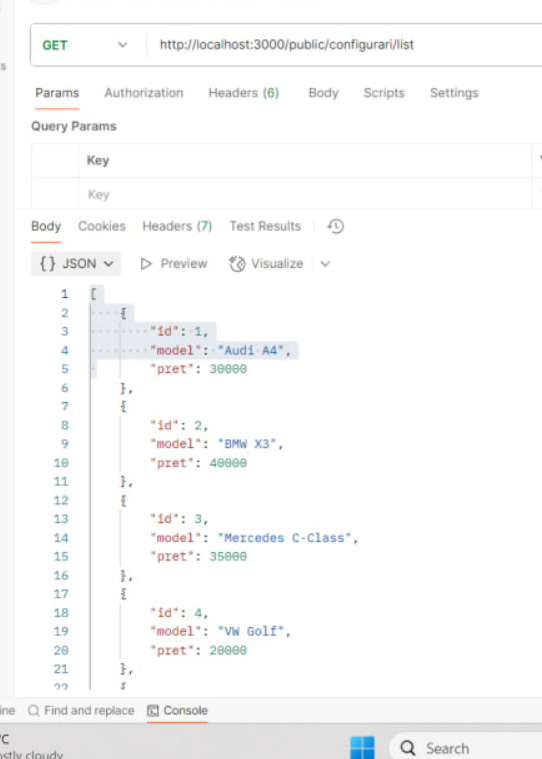
}

Testare Postman:

Header role=user sau lipsă → 403 Forbidden



* Header role=admin → acces permis



# Laboratorul 2

**Sarcina nr 1 Căutare după nume + Pipe Uppercase**

* **Pipe (uppercase.pipe.ts):**

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

@Injectable()

export class UppercasePipe implements PipeTransform {

transform(value: any, metadata: ArgumentMetadata) {

if (typeof value === 'string') {

return value.toUpperCase();

}

return value;

}

}

* **Controller:**

@Get('search-by-name')

searchByName(@Query('name', UppercasePipe) name: string) {

return this.configurariService.searchByName(name);

}

* **Service:**

searchByName(name?: string) {

return this.configurari.filter(c =>

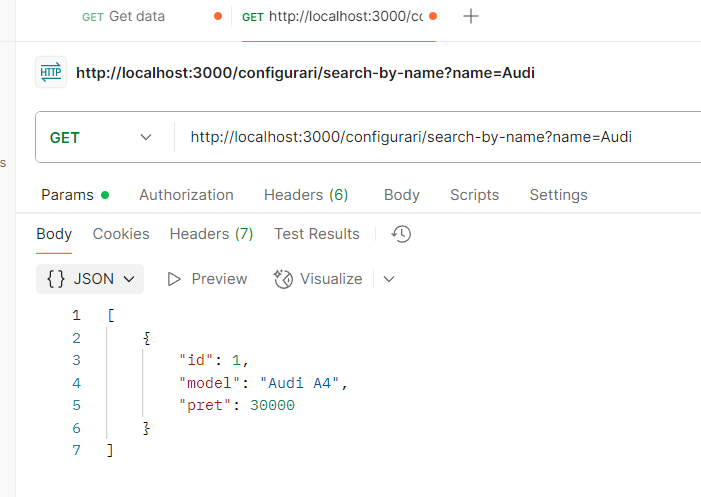
!name || c.model.toLowerCase().includes(name.toLowerCase())

);

}

**Rezultate obținute:**

* Toate endpoint-urile funcționează conform cerințelor.
* Testele Postman pentru public, admin, search și search-by-name funcționează.
* Middleware-ul pentru roluri protejează rutele admin.



**Sarcina nr 2 : Creează un pipe care transformă numele în majuscule**

**Scopul**

Transformarea unui parametru de tip string într-un format standardizat (majuscule) folosind un **Pipe** în NestJS.

**Obiectivele**

* Crearea unui pipe personalizat (UppercasePipe).
* Aplicarea pipe-ului pe un endpoint de tip GET.
* Verificarea funcționalității prin consola serverului și Postman.

**Crearea pipe-ului UppercasePipe**  
Fișier: users/uppercase.pipe.ts

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

@Injectable()

export class UppercasePipe implements PipeTransform {

transform(value: any, metadata: ArgumentMetadata) {

console.log('Before UppercasePipe:', value); // valoarea inițială

if (typeof value === 'string') {

const upper = value.toUpperCase();

console.log('After UppercasePipe:', upper); // valoarea după transformare

return upper;

}

return value;

}

}

**Crearea UsersModule**  
Fișier: users/users.module.ts

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

import { UsersController } from './users.controller';

@Module({

controllers: [UsersController],

})

export class UsersModule {}

**Importarea UsersModule în AppModule**  
Fișier: app.module.ts

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

import { ConfigurariModule } from './configurari/configurari.module';

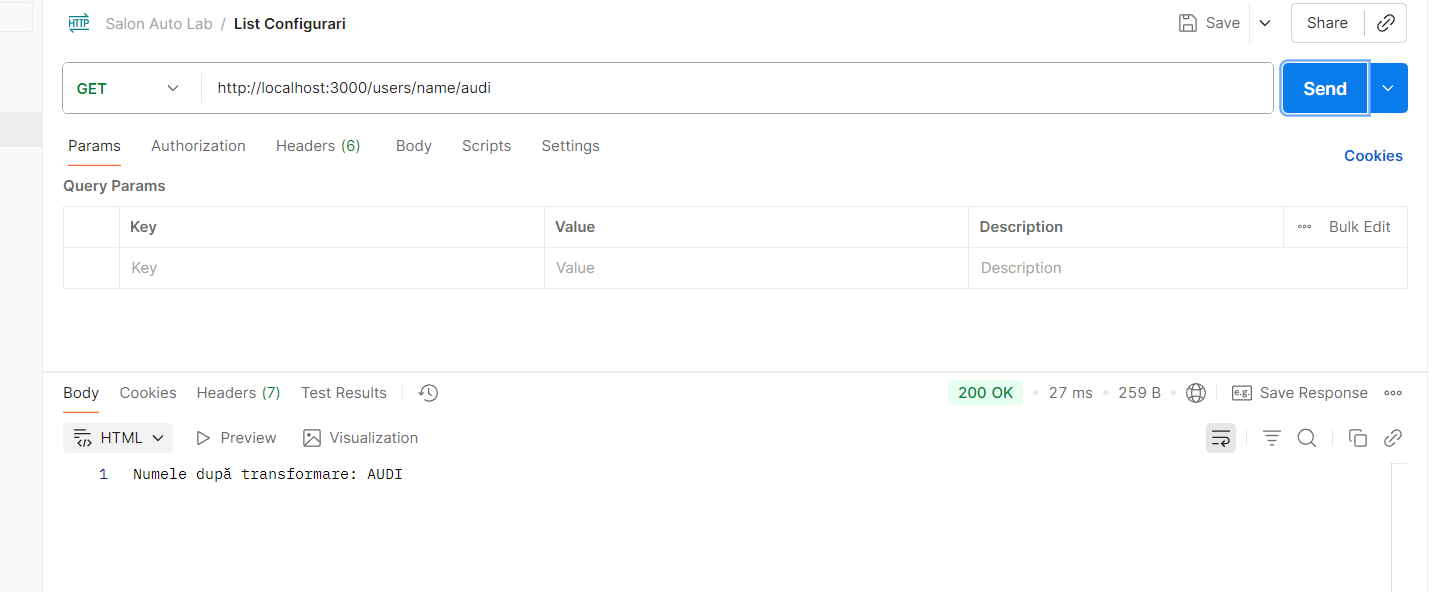
import { UsersModule } from './users/users.module';

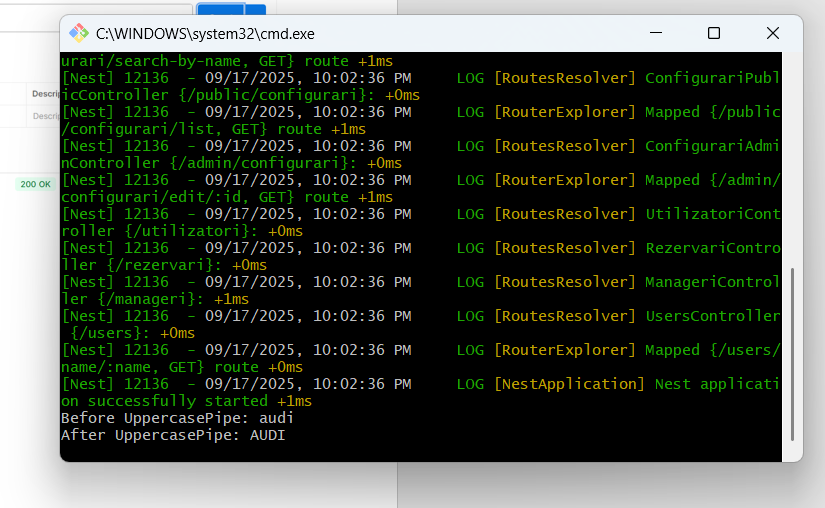
@Module({

imports: [ConfigurariModule, UsersModule],

})

export class AppModule {}

Rezultatul :  
  




**Rezultate obținute:**

 Aplicația returnează listele și detaliile configurărilor corect.

 Filtrarea funcționează pe baza query params.

 Rutele Admin/Public sunt separate corect.

 Middleware-ul controlează accesul Admin/User.

**Concluzie:**

Laboratorul a permis înțelegerea completă a:

* Modularizării NestJS
* Creării de endpoint-uri REST
* Separării rutelor pe roluri
* Testării prin Postman

Aplicația este pregătită pentru extindere cu module noi și autentificare reală.

# Laboratorul 3

Scopul:

Obiectivele:

Realizarea sarcinilor pas cu pas:

Rezultate obținute:

Concluzie:

# Laboratorul 4

Scopul:

Obiectivele:

Realizarea sarcinilor pas cu pas:

Rezultate obținute:

Concluzie:

Bibliografie:

# Laboratorul 5

Scopul:

Obiectivele:

Realizarea sarcinilor pas cu pas:

Rezultate obținute:

Concluzie:

Bibliografie: