

# DOCUMENTAȚIA PROIECTULUI

## PcGear - Magazin Online de Componente

### PC

Porfireanu Constantin Laurențiu  
Stanga George

23 iunie 2025

## Cuprins

<b>1</b>	<b>Prezentarea proiectului</b>	<b>2</b>
1.1	Descrierea generală . . . . .	2
1.2	Problemele rezolvate . . . . .	2
<b>2</b>	<b>Tehnologiile folosite</b>	<b>2</b>
<b>3</b>	<b>Baza de date</b>	<b>2</b>
3.1	Diagrama bazei de date . . . . .	2
<b>4</b>	<b>Prezentarea API-ului</b>	<b>3</b>
4.1	Autentificare (AuthController) . . . . .	4
4.2	Produse (ProductsController) . . . . .	4
4.3	Categorii și Producători . . . . .	4
4.4	Recenzii (ProductReviewsController) . . . . .	4
4.5	Documentarea API-ului cu Swagger . . . . .	4
<b>5</b>	<b>Utilizarea aplicației</b>	<b>4</b>
5.1	Tipuri de utilizatori . . . . .	4
5.2	Autentificare . . . . .	5
5.3	Funcționalități avansate . . . . .	5
<b>6</b>	<b>Concluzii și contribuții</b>	<b>6</b>
6.1	Împărțirea task-urilor . . . . .	6
6.2	Ce am învățat . . . . .	6
6.3	Link GitHub . . . . .	6

# 1 Prezentarea proiectului

## 1.1 Descrierea generală

PcGear este o aplicație web de tip magazin online specializată în vânzarea de componente şi accesorii pentru calculatoare personale. Proiectul implementează o arhitectură Clean Architecture cu separarea responsabilităţilor pe mai multe straturi.

## 1.2 Problemele rezolvate

Aplicația rezolvă următoarele probleme:

- **Gestionarea inventarului** - Administrarea produselor, categoriilor şi producătorilor
- **Autentificare şi autorizare** - Sistem securizat de logare cu roluri diferite (Admin/User)
- **Sistem de recenzii** - Clienții pot lăsa feedback pentru produse
- **Căutare avansată** - Filtrare, sortare şi paginare pentru o experiență îmbunătățită
- **Management stoc** - Actualizarea în timp real a disponibilității produselor

# 2 Tehnologiile folosite

Proiectul utilizează următoarele tehnologii moderne:

- **Backend Framework:** ASP.NET Core 9.0
- **Bază de date:** SQL Server cu Entity Framework Core 9.0.5
- **Autentificare:** JWT (JSON Web Tokens)
- **Documentare API:** Swagger/OpenAPI 7.2.0
- **Arhitectură:** Clean Architecture cu separarea în straturi
- **Pattern-uri:** Repository Pattern, Dependency Injection, DTO Pattern
- **Securitate:** Hashing cu salt pentru parole, autorizare bazată pe roluri

# 3 Baza de date

## 3.1 Diagrama bazei de date

Baza de date implementează următoarele entități principale:

**Entități şi relații:**

- **User** - Utilizatori ai sistemului (clienți şi administratori)
- **Category** - Categoriile de produse (ex: Plăci video, Procesoare)
- **Manufacturer** - Producători (ex: NVIDIA, AMD, Intel)

- **Product** - Produsele disponibile în magazin
- **ProductReview** - Recenziile lăsate de utilizatori pentru produse

Relațiile dintre tabele:

- Product **Many-to-One** Category (un produs aparține unei categorii)
- Product **Many-to-One** Manufacturer (un produs are un producător)
- ProductReview **Many-to-One** Product (mai multe recenzii pentru un produs)
- ProductReview **Many-to-One** User (un utilizator poate avea mai multe recenzii)

Toate entitățile moștenesc din **BaseEntity** care conține câmpurile comune: **Id**, **CreatedAt**, **ModifiedAt**, **DeletedAt** pentru implementarea soft delete.

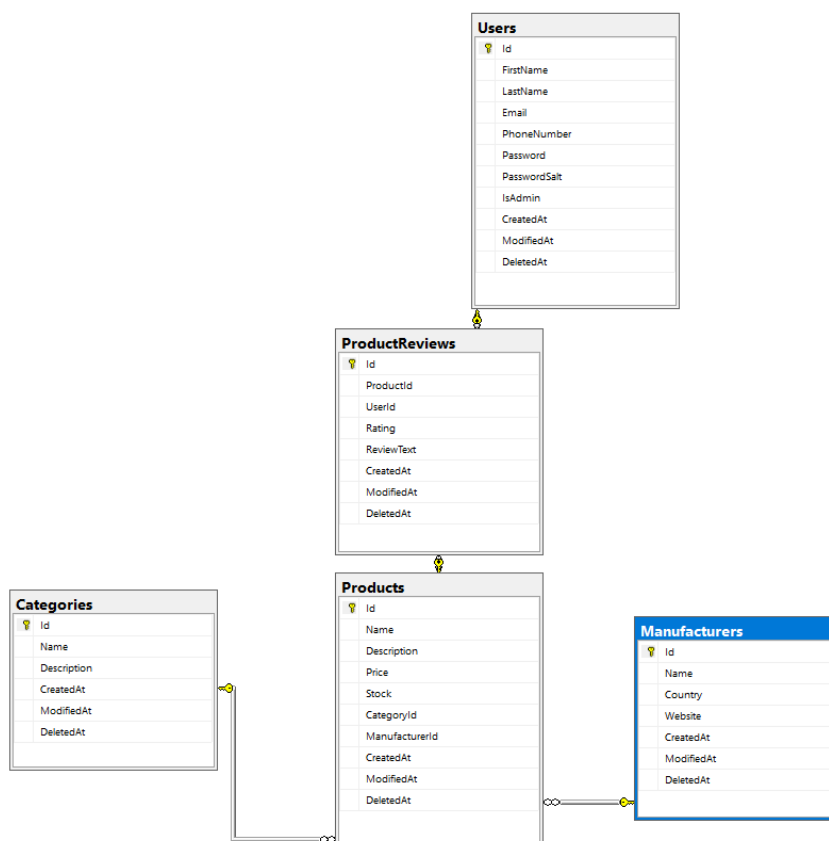


Figura 1: Diagrama entităților și relațiilor din baza de date PcGear

## 4 Prezentarea API-ului

API-ul este documentat complet cu Swagger și expune următoarele endpoint-uri:

## 4.1 Autentificare (AuthController)

- POST /api/auth/Login - Autentificare utilizator
- POST /api/auth/Register - Înregistrare utilizator nou
- GET /api/auth/profile - Obținerea profilului utilizatorului autentificat

## 4.2 Produse (ProductsController)

- POST /api/products/Add\_product - Adăugare produs nou
- GET /api/products/Get\_products - Listarea tuturor produselor
- GET /api/products/Get\_products\_by\_id:{id} - Obținere produs după ID
- GET /api/products/Get\_products\_with\_reviews{id} - Produs cu recenzii
- PUT /api/products/Update\_product{id} - Actualizare produs
- PATCH /api/products/Update\_product\_stock{id} - Actualizare stoc
- DELETE /api/products/Delete\_product{id} - Ștergere produs
- GET /api/products/Get\_paged\_and\_filter - Căutare avansată cu filtre

## 4.3 Categoriile şi Producători

Endpoint-uri similare pentru gestionarea categoriilor (/api/categories) şi producătorilor (/api/manufacturers) cu operații CRUD complete.

## 4.4 Recenzii (ProductReviewsController)

- POST /api/reviews/Add\_review - Adăugare recenzie
- GET /api/reviews/Get\_Reviews - Listarea recenziilor
- DELETE /api/reviews/Delete\_by\_id{id} - Ștergere recenzie

## 4.5 Documentarea API-ului cu Swagger

# 5 Utilizarea aplicației

## 5.1 Tipuri de utilizatori

- **Utilizatori obișnuiți:** Pot vizualiza produsele, lăsa recenzii, se pot înregistra şi autentifica
- **Utilizatori cu drepturi de administrator:** Orice utilizator poate avea şi rolul de administrator (câmpul IsAdmin), având astfel acces complet la toate funcționalitățile - gestionarea produselor, categoriilor, producătorilor şi utilizatorilor

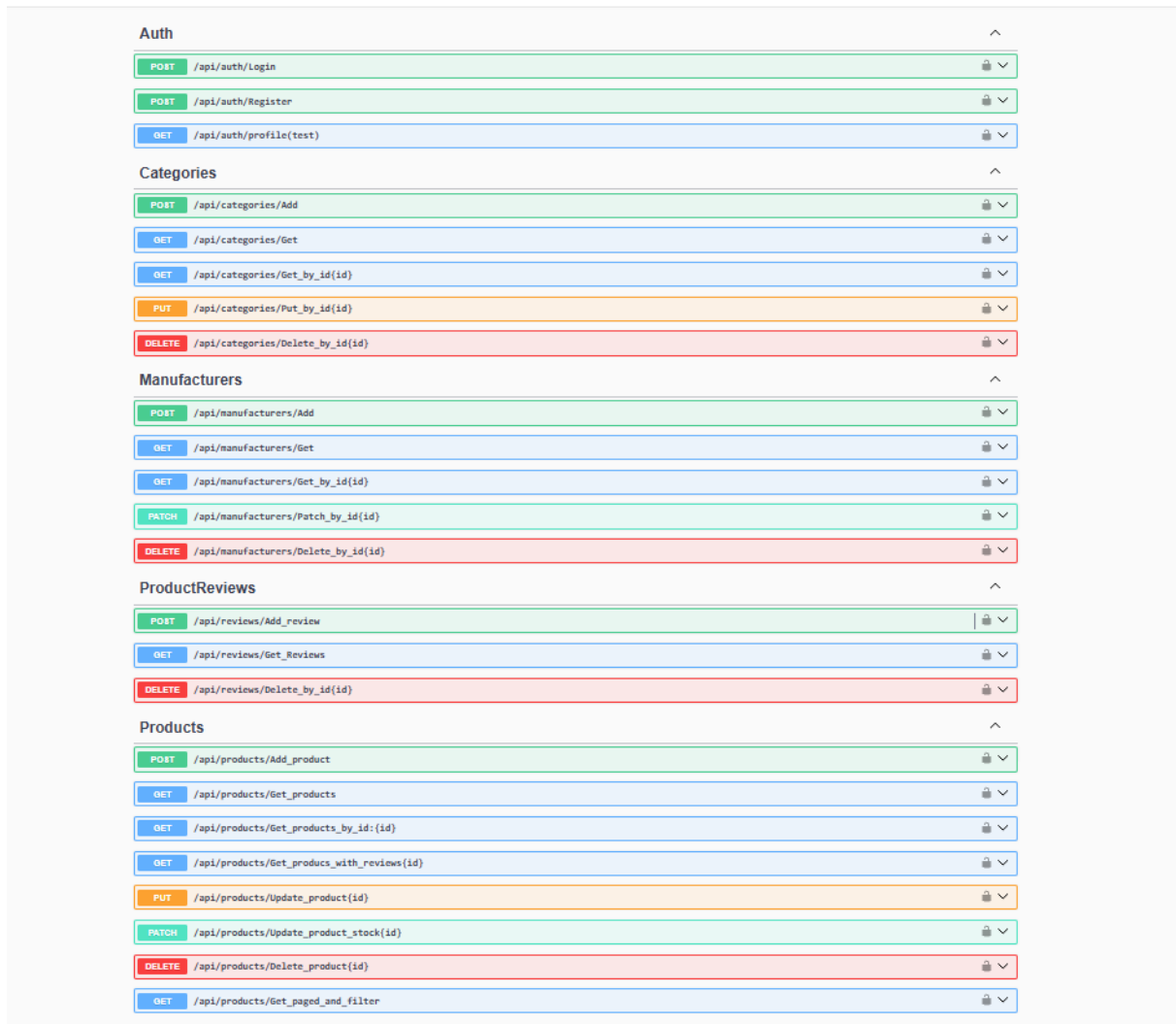


Figura 2: Screenshot al documentării API-ului în Swagger - toate endpoint-urile disponibile

## 5.2 Autentificare

Sistemul foloseşte JWT tokens pentru autentificare. După logare, utilizatorul primeşte un token valid pentru 525600 minute. Toate endpoint-urile (exceptând login/register) necesită autentificare.

## 5.3 Funcţionalităţi avansate

- **Filtrare:** După nume, preţ, categorie, producător, disponibilitate
- **Sortare:** După anumite câmpuri (nume, preţ, stoc, dată creării)
- **Paginare:** Rezultate împărţite pe pagini pentru performanţă optimă
- **Recenzii cu rating:** Sistem de evaluare de la 1 la 5 stele

## 6 Concluzii şi contribuţii

### 6.1 Împărţirea task-urilor

Deşi am împărţit responsabilităţile după cum urmează, am lucrat în echipă pe tot parcursul dezvoltării:

**Porfireanu Constantin Laurenţiu:**

- Implementarea elementelor de bază (entităţi, repository-uri)
- Operaţiunile CRUD pentru toate entităţile
- Configurarea Entity Framework şi migrărilor
- Implementarea pattern-urilor (Repository, DTO mapping)

**Stanga George:**

- Sistem de autentificare şi autorizare JWT
- Implementarea filtrării, sortării şi paginării avansate
- Middleware-urile pentru logging şi exception handling
- Configurarea Swagger şi documentarea API-ului

### 6.2 Ce am învăţat

În urma acestui proiect am dobândit experienţă în:

- Arhitectura Clean Architecture şi separarea responsabilităţilor
- Implementarea sistemelor de autentificare securizate cu JWT
- Lucrul cu Entity Framework Core şi SQL Server
- Dezvoltarea API-urilor RESTful cu documentare Swagger
- Pattern-urile Repository şi Dependency Injection
- Implementarea funcţionalităţilor avansate (filtrare, sortare, paginare)
- Gestionarea excepţiilor şi logging-ul în aplicaţii web

### 6.3 Link GitHub

Codul complet al proiectului este disponibil la: [PcGear](#)