Shop Platform -REST API

Bîrleanu Teodor Matei - 343C3

January 16, 2026

Descriere

Shop Platform este o aplicatie e-commerce distribuita construita pe arhitectura de microservicii. Sistemul include autentificare SSO (Keycloak), cautare avansata (Elasticsearch), procesare plati online (Stripe) si orchestrare cu Docker Swarm. Platforma ofera functionalitati complete de gestiune produse, comenzi si plati.

Rute Disponibile

Autentificare

- `GET /auth/login` - Initiaza SSO flow cu Keycloak
- `GET /auth/callback` - Callback autentificare, schimb token JWT
- `GET /auth/logout` - Logout si invalidare sesiune

Produse - Toate Rolurile

- `GET /products` - Lista toate produsele (paginare suportata)
- `GET /products/:id` - Detalii produs specific
- `GET /products/search/advanced` - Cautare Elasticsearch cu fuzzy matching

Produse - Admin

- `POST /admin/products` - Creare produs nou
- `DELETE /admin/products/:id` - Stergere produs
- `PUT /admin/products/:id/price` - Actualizare pret
- `PUT /admin/products/:id/stock` - Actualizare stoc

Comenzi - User

- `GET /orders` - Lista comenzi proprii
- `GET /orders/:id` - Detalii comanda specifica
- `POST /orders` - Creare comanda (buy sau sell cu `order_type`)

Comenzi - Admin

- `GET /admin/orders` - Lista toate comenzile sistemului
- `GET /admin/orders/:id` - Detalii orice comanda

De ce este REST API?

1. Resurse Identificabile prin URI

Fiecare resursa (produs, comanda) are un URI unic predictibil:

- `/products/1` identifica produsul cu ID 1
- `/orders/5` identifica comanda cu ID 5
- Structura ierarhica clara: `/admin/products` vs `/products`

2. Metode HTTP Semantice

Operatiile CRUD folosesc metodele HTTP standard:

- `GET` - citire date (idempotent)
- `POST` - creare resurse (non-idempotent)
- `PUT` - actualizare (idempotent)
- `DELETE` - stergere (idempotent)

3. Comunicare Stateless

Serverul nu pastreaza starea sesiunii. Fiecare request contine JWT token in header `Authorization: Bearer <token>` cu toate informatiile necesare (user_id, roles). Token-ul este self-contained si verificat la fiecare request.

4. Reprezentari JSON

Toate request-urile si response-urile folosesc format JSON standard cu header `Content-Type: application/json`. Structurile sunt consistente si documentate.

5. Status Codes Semantice

API-ul foloseste coduri HTTP standard pentru comunicare clara:

- `200 OK` - operatie reusita
- `201 Created` - resursa creata cu succes
- `400 Bad Request` - date invalide
- `401 Unauthorized` - lipseste sau este invalid token-ul
- `403 Forbidden` - utilizator fara permisiuni
- `404 Not Found` - resursa nu exista
- `500 Internal Server Error` - eroare server

6. Interfata Uniforma

API-ul mentine consistenta:

- Toate colectiile returneaza liste: {"products": [...], "count": 10}
- Erorile au format uniform: {"error": "mesaj"}
- Structura URL-urilor este predictibila si logica

7. Idempotenta

Metodele GET, PUT, DELETE sunt idempotente (aceeasi actiune repetata produce acelasi efect). POST este non-idempotent (creeaza resurse noi la fiecare apel).

Exemplu Flow Complet

Plasare comanda cu plata:

1. User obtine token: `POST /auth/login`
2. User creeaza comanda: `POST /orders` cu body {"product_id": 1, "requires_payment": true}
3. API returneaza `201 Created` + `checkout_url`
4. User plateste in browser pe Stripe
5. Stripe trimite webhook: `POST /webhooks/stripe`
6. API actualizeaza comanda automat la status `completed`

Flow-ul demonstreaza comunicare stateless (token in fiecare request), status codes corecte (201), reprezentari JSON si operatii asincrone (webhook).

Arhitectura Microservicii

Trei servicii independente:

- **API Gateway (5000)** - Autentificare, routing, load balancing
- **Product Service (5001)** - CRUD produse, integrare Elasticsearch
- **Order Service (5002)** - Gestiune comenzi, integrare Stripe

Comunicarea inter-servicii se face prin HTTP requests interne (Docker overlay network). Fiecare serviciu poate fi scalat independent (2 replicas pentru high availability).