**Shop Online**

Întocmit de: **Balog David Alexandru**

Pentru disciplina: **Proiectarea Aplicaților Web**

Cuprins

[Capitolul 1. Introducere 1](#_Toc384994105)

[Capitolul 2. Arhitectura Aplicației 2](#_Toc384994105)

[2.1. Arhitectura generală 2](#_Toc384994106)

[2.2. Diagrama bazei de date 2](#_Toc384994106)

[Capitolul 3. Backend 3](#_Toc384994105)

[Capitolul 4. Frontend 4](#_Toc384994105)-5

[Capitolul 5. Configurare și rulare 6](#_Toc384994105)

[Capitolul 6. Concluzie 7](#_Toc384994105)

[Bibliografie 8](#_Toc384994115)

# Introducere

Proiectul reprezintă un magazin online de haine second-hand, creat pentru a oferi o soluție modernă și eficientă de cumpărături în mediul digital. Dezvoltat utilizând tehnologiile Spring Boot pentru backend și React pentru frontend, aplicația combină performanța și securitatea cu o interfață prietenoasă și intuitivă.

Platforma permite utilizatorilor să navigheze cu ușurință printre produse, să-și creeze un cont pentru a beneficia de funcționalități personalizate, să adauge articole în coșul de cumpărături și să finalizeze comenzi într-un mod rapid și sigur. Gestionarea produselor, autentificarea utilizatorilor și procesarea comenzilor sunt realizate printr-o integrare eficientă între componentele backend și frontend, susținute de o bază de date robustă MySQL.

Aplicația își propune să răspundă cerințelor unui magazin online modern, punând accent pe flexibilitate, modularitate și extensibilitate, oferind totodată un exemplu clar de integrare între tehnologiile utilizate.

# Arhitectura aplicației

## Arhitectura generală

Aplicația este organizată pe arhitectura client-server, unde:

* Frontend-ul comunică cu backend-ul prin API-uri REST.
* Backend-ul gestionează logica aplicației, persistă datele în baza de date și oferă API-uri securizate.

## Diagrama bazei de date

Baza de date a proiectului este implementată utilizând MySQL și este proiectată pentru a susține funcționalitățile specifice unui magazin online. Structura bazei de date include șase tabele principale:

Include tabelele:

* Utilizator – id, email, nume, parola
* Produs – id, nume, pret, imagine
* CosCumparaturi – id, utilizator\_id
* DetaliiCos – id, cos\_id, produs\_id, cantitate
* Comanda – id, data\_comanda, utilizator\_id
* DetaliiComanda – id, cantitate, pret, comanda\_id, produs\_id

# Backend

Backend-ul aplicației este construit utilizând framework-ul Spring Boot, oferind o arhitectură bine organizată și modulară pentru gestionarea datelor și implementarea funcționalităților esențiale. Structura este împărțită în patru componente principale pentru fiecare entitate din sistem: controller, service, repository și entity.

* Entități: Reprezintă tabelele din baza de date și definesc structura datelor. Printre principalele entități se numără:
* Utilizator: gestionează informațiile despre utilizatori, inclusiv numele, adresa de e-mail și coșul de cumpărături asociat.
* Produs: definește produsele disponibile în magazin, cu detalii precum prețul, descrierea și stocul.
* CosCumparaturi: stochează produsele adăugate de utilizator înainte de plasarea comenzii.
* Comanda: înregistrează detaliile comenzilor finalizate.
* Repository: Interacționează cu baza de date folosind Spring Data JPA. Fiecare entitate are un repository dedicat, care oferă metode standard pentru operațiuni CRUD (create, read, update, delete) și permite definirea de metode personalizate pentru interogări complexe.
* Servicii: Conțin logica de afaceri a aplicației, implementând funcționalități precum gestionarea produselor, procesarea comenzilor sau autentificarea utilizatorilor. Fiecare entitate are un serviciu asociat care intermediază între controller și repository.
* Controlere: Expun endpoint-urile RESTful prin care frontend-ul interacționează cu backend-ul. Acestea gestionează cererile HTTP (GET, POST, PUT, DELETE) și mapelează răspunsurile adecvate. De exemplu:
* UtilizatorController: gestionează operațiunile legate de utilizatori, cum ar fi autentificarea, înregistrarea și actualizarea datelor.
* ProdusController: oferă endpoint-uri pentru afișarea, adăugarea, actualizarea și ștergerea produselor.
* CosCumparaturiController și ComandaController: gestionează funcționalitățile specifice coșului de cumpărături și comenzilor.

Această structură modulară asigură scalabilitate, claritate și separarea responsabilităților, facilitând extinderea funcționalităților sau depanarea sistemului.

# Frontend

Frontend-ul aplicației este construit utilizând React, un framework JavaScript, care permite crearea unei interfețe de utilizare interactive și moderne. Structura proiectului este organizată pentru a asigura modularitate, reutilizare a codului și o navigare fluidă între pagini.

* Componente: Acestea sunt blocurile de bază ale interfeței de utilizare și includ:
* Navbar: asigură navigarea între diferitele secțiuni ale aplicației, precum pagina principală, autentificare și coșul de cumpărături.
* Footer: afișează informații suplimentare sau legături utile în partea de jos a paginii.
* ProductCard: reprezintă un produs individual, incluzând imaginea, prețul și adăugare în coș.
* ProductList: afișează o listă de produse sub formă de grilă, utilizând componentele ProductCard.
* Pagini: Structurate pentru a oferi funcționalități și informații specifice:
* Home: afișează lista produselor disponibile și reprezintă punctul de pornire al aplicației.
* Login: permite utilizatorilor să se autentifice în sistem pentru a accesa funcționalitățile personalizate.
* Signup: oferă un formular pentru înregistrarea de noi utilizatori.
* Cart: afișează conținutul coșului de cumpărături, permițând utilizatorilor să vizualizeze, să actualizeze sau să finalizeze comanda.
* Rute: Configurate pentru a permite navigarea între pagini, utilizând React Router. Aplicația include rute protejate (ProtectedRoutes), care restricționează accesul la anumite secțiuni, cum ar fi coșul de cumpărături, doar pentru utilizatorii autentificați.
* Servicii: Gestionate prin module separate pentru a centraliza logica aplicației și interacțiunile cu backend-ul:
* AuthService: gestionează autentificarea și înregistrarea utilizatorilor, inclusiv stocarea și verificarea token-urilor.
* CartService: oferă funcționalități pentru gestionarea produselor din coșul de cumpărături, cum ar fi adăugarea, eliminarea sau actualizarea cantităților.
* ProductService: comunică cu backend-ul pentru a obține lista produselor disponibile.
* axios: configurat pentru a realiza cereri HTTP către API-ul backend-ului, incluzând gestionarea automatizată a autentificării prin token-uri.

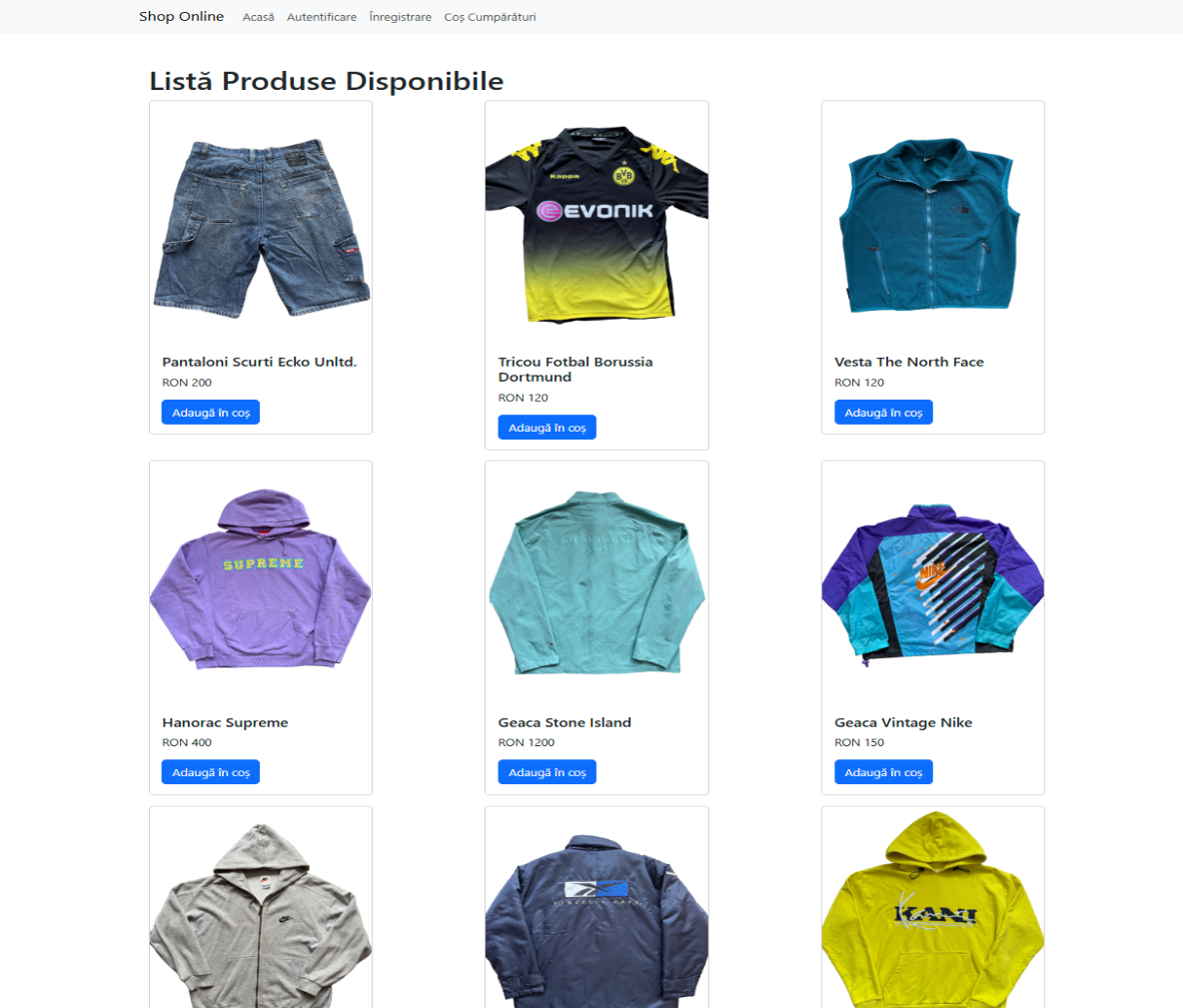


Figura 1.1 Pagina principală a aplicației - Home

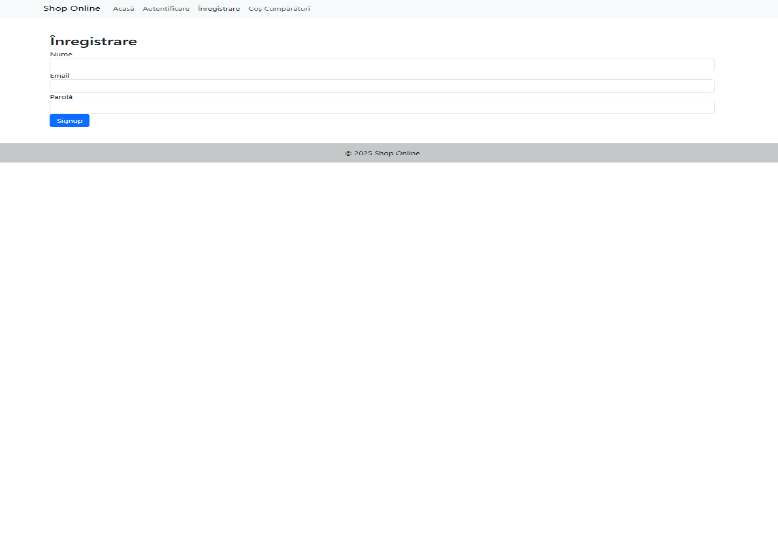
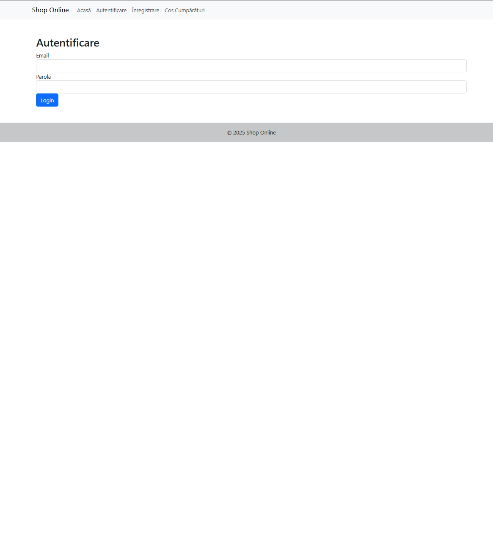


Figura 1.2 Paginile de Înregistrare și Autentificare – Login și Signup

# Configurare și rulare

Pentru a rula aplicația, atât partea de backend, cât și cea de frontend, trebuie configurate și pornite separat. Baza de date se va porni automat cu partea de backend.

* Backend (Spring Boot):

Backend-ul aplicației este construit utilizând Spring Boot și poate fi pornit direct din IDE-ul IntelliJ IDEA. Acest lucru se realizează executând clasa principală ShopOnlineApplication. Odată pornită aplicația, serverul backend va asculta cererile pe portul specificat (implicit, 8080).

* Frontend (React):

Partea de frontend este construită utilizând React. Pentru a o porni, se accesează directorul proiectului frontend în terminal, de exemplu:

cd shop-online-frontend

npm start

Această comandă va inițializa serverul de dezvoltare și va deschide automat aplicația în browser pe adresa http://localhost:3000.

# Concluzie

Proiectul „Shop Online” demonstrează integrarea eficientă a tehnologiilor moderne, precum Spring Boot pentru backend și React pentru frontend, într-o aplicație web complet funcțională. Printr-o bază de date MySQL bine structurată, sistemul gestionează utilizatorii, produsele, coșurile de cumpărături și comenzile, oferind o experiență intuitivă utilizatorilor.

# Bibliografie

<https://docs.spring.io/spring-boot/index.html>

<https://react.dev/>

Curs Proiectare Aplicațiilor WEB - https://kb.cunbm.utcluj.ro/course/view.php?id=1977

http://cti.ubm.ro/paw/