## Descriere

Aplicația web de e-business este o platformă online complexă, multifuncțională, ce permite utilizatorilor să participe atât în rol de cumpărători cât și de vânzători, asemănătoare cu platforme consacrate precum eMag sau eBay. unde utilizatorii pot naviga, cumpăra și vinde produse.

Utilizatorii își pot crea propriile conturi, oferind datele necesare pentru înregistrare și autentificare. Contul le va permite să administreze produsele pe care doresc să le vândă, să urmărească comenzi, să primească notificări și să se aboneze la magazinele sau vânzătorii preferați.După înregistrare, utilizatorii pot lista produsele pe care doresc să le vândă, completând informații despre produs, cum ar fi descrierea, prețul, fotografiile și categoriile relevante.Aplicația va permite utilizarea cupoanelor de discount, care pot fi aplicate la cumpărături pentru a obține reduceri. Acest lucru stimulează vânzările și ajută la fidelizarea clienților.

Interfața trebuie să fie intuitivă și ușor de navigat, oferind o experiență plăcută și fără probleme pentru toate categoriile de utilizatori,această aplicație de e-business reprezintă o piață digitală dinamică și interactivă, concepută pentru a facilita o experiență de cumpărare și vânzare eficientă, sigură și plăcută pentru o gamă largă de produse și servicii.

## Diagrame

O imagine care conține text, captură de ecran

Descriere generată automat

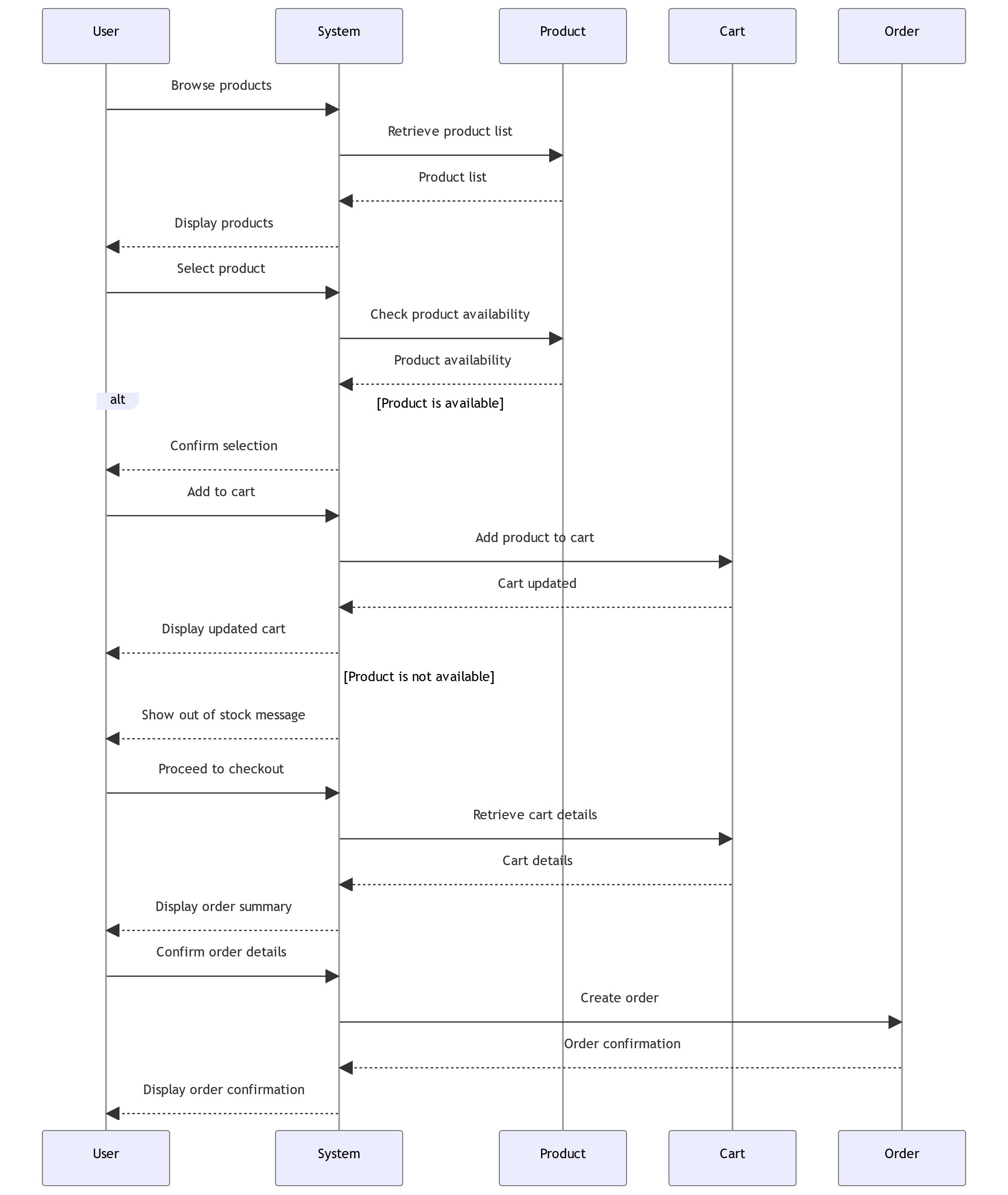
Această diagramă reprezinta modelul bazei de date pentru sistem. În mod specific, diagrama ilustrează tabelele din baza de date și legăturile (sau cheile străine) dintre ele.

**Users**: Acest tabel conține informațiile despre utilizatori. Fiecare utilizator are un ID unic și atribute precum email, prenume, nume de familie, parolă și nume de utilizator,acestia sunt persoanele ce interactioneaza cu aplicatia.**Admins**: Similar cu tabelul utilizatorilor, include un cod de admin, email, parolă, rol și nume de utilizator,admini au rolul de a administra useri prin operatii crud pe acestia pentru a mentie o aplicatie calitativa.

**Products**: Acest tabel conține lista de produse disponibile, cu atribute precum preț, cantitate, ID-ul categoriei la care aparține produsul, ID-ul utilizatorului care a adăugat produsul sau mai specific al vanzatorului.**Product\_category**: Tabelul de categorii de produse, cu fiecare categorie cu descriere și un nume fiind o relatie one to many asupra tabelei de produse

**Orders**: Aici sunt stocate detaliile comenzilor efectuate de utilizatori, inclusiv totalul de plată, ID-ul utilizatorului care a efectuat comanda, data livrării, data comenzii, metoda de plată și statusul comenzii,**cart**: reprezintă coșul de cumpărături al utilizatorilor, înregistrând cantitatea de produse și ID-ul utilizatorului pentru fiecare coș de cumpărături.**Cart\_products**: Un tabel asociativ care leagă coșurile de cumpărături de produsele pe care le conțin, prin stocarea ID-urilor pentru fiecare coș și pentru fiecare produs.

**Discounts**: conține un procentaj de discount, un status care indică dacă discountul este activ sau nu, un cod de discount, data de expirare a discountului și un ID unic.

Urmatoarea diagramă secvențială ilustrează procesul de cumpărături online din perspectiva interacțiunilor dintre utilizator și sistem, precum și, coșul de cumpărături și comanda,conform pasiilor urmatori

Căutarea produselor: Utilizatorul navighează prin lista de produse, iar sistemul preia și afișează lista de produse disponibile.Selectarea produsului: Utilizatorul alege un produs pentru a-l adăuga în coș. Sistemul verifică disponibilitatea produsului.Dacă produsul este disponibil, îl adaugă în coș. Sistemul adaugă produsul în coș și actualizează coșul de cumpărături, afișând detaliile actualizate utilizatorului.Finalizarea cumpărăturilor: Utilizatorul procedează la checkout. Sistemul preia detaliile coșului și afișează un rezumat al comenziiDiagrama folosește notația standard pentru a reprezenta interacțiunile sincrone și asincrone, folosind linii continue pentru apeluri sincrone și linii întrerupte pentru răspunsuri sau mesaje asincrone. De asemenea, utilizează o structură "alt" pentru a arăta decizii condiționale în fluxul procesului – în acest caz, pentru a gestiona scenariul în care produsul este sau nu este disponibil.

O imagine care conține text, linie, diagramă, Font

Descriere generată automat

Ultima diagrama prezinta implementarea si functiile utilizate pentru a realiza procesul mentionat anterior.

## Design Pattern

Pentru acest proiect sa utilizat design patternul **Builder** pe partea de backend pentru a contrui o serie de obiecte precum user,produse,comenzi. Builder Design Pattern este un pattern de proiectare creational care se folosește pentru a construi obiecte complexe pas cu pas. Acest model este folosit pentru a separa construcția unui obiect complex de reprezentarea sa, astfel încât același proces de construcție să poată crea reprezentări diferite.

Odată ce un obiect este construit, acesta poate fi făcut imutabil fără a fi nevoie să expui setteri care pot compromite starea obiectului. Constructorii în Java pot deveni confuzi atunci când un obiect are multe atribute, mai ales dacă multe dintre aceste atribute sunt de același tip. Folosind Builder Pattern, fiecare pas de inițializare este clar definit, ceea ce face codul mai lizibil.De asemenea, poate facilita construcția obiectelor cu diverse variante fără a polua codul cu multiple construcții.Builder-ul poate fi folosit pentru a construi diverse reprezentări ale aceluiași obiect bazic, reutilizând aceeași rutină de construcție.Atributele opționale sunt mai ușor de gestionat cu Builder Pattern. Fără acesta, ar putea fi necesari multipli constructori, fiecare cu diferite combinații de parametri.

In exemplul urmator se alfa un snippet din class Product se incorporeaza ca clasa statica clasa de builder utilizata pentru a realiza produse mult mai simplu.De asemenza acest design pattern este util cand dorim sa folosim dto precum in cadrul clasei utilizator ce poate combina o serie de atribute de la alte class precum comenzi sau produse sau daca dorim sa incapsulam anumite date sensibile.

public static class Builder {  
 private final Product product;  
  
 public Builder() {  
 product = new Product();  
 }  
  
 public Builder withId(Long id) {  
 product.id = id;  
 return this;  
 }  
  
 public Builder withName(String name) {  
 product.name = name;  
 return this;  
 }  
  
 public Builder withDescription(String description) {  
 product.description = description;  
 return this;  
 }

public Product build() {

return new Product(this);

}