**Temă de casă**

**Aplicație E-commerce pentru Magazin de Mobilă**

*Autori: Ivan Iustin, Clipicioiu Ciprian, Ignat Tudor*

***Temă in echipă:***

## Surse de informatii

* 1. **Articole de Specialitate:** Articole din domeniul e-commerce și al aplicațiilor mobile pentru a înțelege cerințele și tendințele actuale din industrie.
  2. **Studii de Caz:** Analiza altor aplicații e-commerce pentru a identifica bune practici și a evita greșelile comune.

## Tehnici de analiză

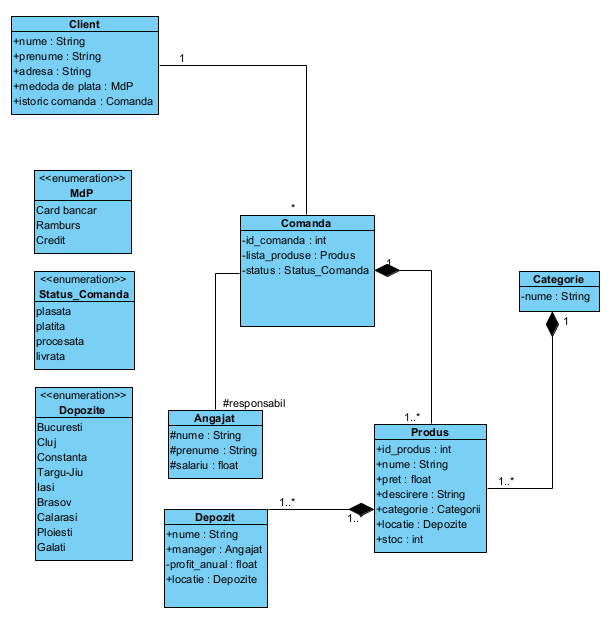
**Am folosit tehnicile de analiza:** Analiza Cerințelor si Analiza Competiției

Am definit clar obiectivele, inclusiv funcționalitățile cheie ale magazinului online, experiența utilizatorului și securitatea datelor. Am stabilit cerințele funcționale și non-funcționale și am implicat părțile interesate, precum utilizatorii, clienții și echipa de dezvoltare, pentru a înțelege nevoile și așteptările acestora.

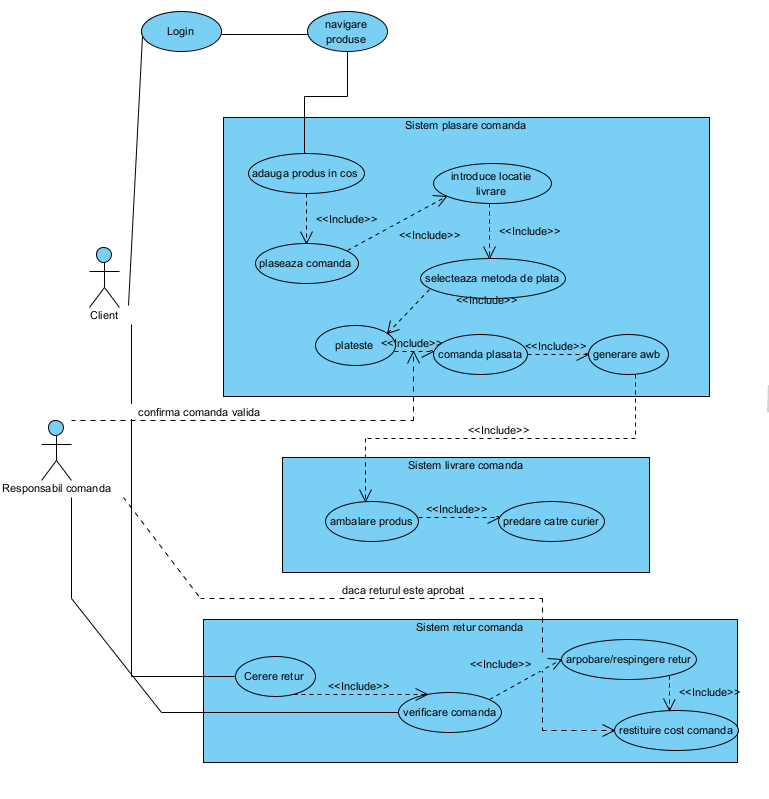
Documentarea cerințelor a fost realizată într-un mod clar, concis și detaliat pentru a ghida dezvoltarea proiectului. De asemenea, am efectuat o analiză a competiției, identificând concurența din piața magazinelor online de mobilă, evaluând punctele lor forte și slabe, strategiile de marketing și vânzare. Am comparat funcționalitățile oferite de concurenți pentru a identifica aspecte ce pot fi îmbunătățite și caracteristici care ne pot diferenția. Am studiat feedback-ul clienților despre magazinele concurente pentru a înțelege nevoile acestora și pentru a oferi o experiență superioară.

Prin combinarea Analizei Cerințelor și Analizei Competiției, am reușit să redactăm cerințele proiectului "Magazin ecommerce de mobilă" într-un mod detaliat, orientat către nevoile utilizatorilor și adaptat pentru a ne diferenția de concurență.

## Diagrama de clase UML



## Diagrama cazurilor de utilizare UML



***Temă individuală:***

## Cerinte functionașe si nefunctionale

**Cerință Funcțională**: Implementarea unei funcționalități de filtrare avansată;

**Cerință Nefuncțională**: Asigurarea unei performanțe optime a platformei;

## Detalierea cerinței funcționale

Permiterea utilizatorilor să sorteze produsele după multiple criterii simultan, cum ar fi prețul, marca, dimensiunea. Această funcționalitate va îmbunătăți experiența utilizatorilor și va facilita procesul de căutare a produselor dorite.

Pentru a implementa această funcționalitate, este necesară dezvoltarea unei interfețe intuitive și ușor de utilizat, care să ofere utilizatorilor posibilitatea de a selecta și combina diferite criterii de filtrare. Aceste criterii ar trebui să fie flexibile și personalizabile, permițând utilizatorilor să-și ajusteze căutarea în funcție de preferințele lor.

De asemenea, este important ca sistemul de filtrare să fie rapid și eficient, oferind rezultate relevante în timp real pe măsură ce utilizatorii aplică filtrele. Funcționalitatea ar trebui să fie integrată în mod coeziunat cu restul platformei, asigurându-se că utilizatorii pot naviga ușor și intuitiv prin rezultatele filtrate.

## Descriere cerință funcțională

***Funcție:*** Funcționalitatea de Recomandare a Produselor

***Descriere:*** Sistemul de recomandare a produselor va utiliza algoritmi de învățare automată pentru a sugera utilizatorilor produse relevante, bazate pe istoricul de navigare, preferințele anterioare și comportamentul de cumpărare.

***Intrări:*** Istoricul de navigare al utilizatorului, preferințele anterioare, informații despre produsele vizualizate și achiziționate.

***Ieșiri:*** Recomandări personalizate de produse, listate în funcție de gradul de potrivire cu preferințele și comportamentul de cumpărare al utilizatorului.

## Scenariului de plasare comandă

Procesul de plasare a comenzilor pe etape:

1. Clientul se loghează în sistem.
2. Clientul navighează prin produsele disponibile și le adaugă în coșul său.
3. Clientul introduce locația sa de livrare.
4. Clientul selectează o metodă de plată.
5. Clientul plateste comanda (daca a selectat optiunea de plata online/credit) si plasează comanda.
6. Responsabilul de comanda verifică comanda si lista de produse pentru a se asigura că este valabilă.
7. Dacă comanda este valabilă, sistemul de livrare a comenzilor este notificat.
8. Sistemul de livrare a comenzilor ambalează produsele și le predă curierului.
9. Curierul livrează produsele clientului.
10. Clientul confirmă primirea comenzii.
11. Sistemul de plasare a comenzilor marchează comanda ca fiind finalizată.

## Implementare 2 clase in C++

class **Comanda** {

private:

int idComanda;

vector<Produs> listaProduse;

string statusComanda;

string metodaPlata;

string adresaLivrare;

public:

// Constructor

Comanda(int idComanda, const vector<Produs>& listaProduse, const string& statusComanda, const string& metodaPlata, const string& adresaLivrare) {

this->idComanda = idComanda;

this->listaProduse = listaProduse;

this->statusComanda = statusComanda;

this->metodaPlata = metodaPlata;

this->adresaLivrare = adresaLivrare;

}

// doar un setter pentru toate datele

void setData(const vector<pair<string, string>>& data) {

for (const auto& pereche : data) {

if (pereche.first == "idComanda") {

idComanda = stoi(pereche.second);

} else if (pereche.first == "listaProduse") {

// preia datele din string

vector<Produs> produseTemporare;

string produseString = pereche.second;

istringstream iss(produseString);

string produsData;

while (getline(iss, produsData, ';')) {

vector<string> detaliiProdus;

istringstream issProdus(produsData);

string detaliuProdus;

while (getline(issProdus, detaliuProdus, ':')) {

detaliiProdus.push\_back(detaliuProdus);

}

if (detaliiProdus.size() == 5) { // Asigura date complete

produseTemporare.emplace\_back(Produs(stoi(detaliiProdus[0]), detaliiProdus[1], stof(detaliiProdus[2]), detaliiProdus[3], detaliiProdus[4]));

} else {

cerr << "Warning: Incomplete product data found in '" << produsData << "'" << endl;

}

}

listaProduse = produseTemporare;

} else if (pereche.first == "statusComanda") {

statusComanda = pereche.second;

} else if (pereche.first == "metodaPlata") {

metodaPlata = pereche.second;

} else if (pereche.first == "adresaLivrare") {

adresaLivrare = pereche.second;

}

}

}

// Functie pentru afisarea datelor

void afiseazaDate() const {

cout << "Id comanda: " << idComanda << endl;

cout << "Lista produse:" << endl;

for (const Produs& produs : listaProduse) {

produs.afiseazaDate();

}

cout << "Status comanda: " << statusComanda << endl;

cout << "Metoda de plata: " << metodaPlata << endl;

cout << "Adresa livrare: " << adresaLivrare << endl;

}

};

class **Produs** {

private:

int idProdus;

string nume;

float pret;

string descriere;

string categorie;

public:

// Constructor for creating a product

Produs(int idProdus, const string& nume, float pret, const string& descriere, const string& categorie) {

this->idProdus = idProdus;

this->nume = nume;

this->pret = pret;

this->descriere = descriere;

this->categorie = categorie;

}

// Function to display product details

void afiseazaDate() const {

cout << "Id produs: " << idProdus << endl;

cout << "Nume: " << nume << endl;

cout << "Pret: " << pret << endl;

cout << "Descriere: " << descriere << endl;

cout << "Categorie: " << categorie << endl;

}

};

## Implementarea relatiei de asociere dintre clasa Comanda si Produs

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class Produs {

private:

int idProdus;

string nume;

float pret;

string descriere;

string categorie;

public:

// Constructor

Produs(int idProdus, string nume, float pret, string descriere, string categorie) {

this->idProdus = idProdus;

this->nume = nume;

this->pret = pret;

this->descriere = descriere;

this->categorie = categorie;

}

float getPret() const { return pret; }

string getNume() const { return nume; }

string getDescriere() const { return descriere; }

// Afișează detaliile produsului (ID, nume, preț, descriere, categorie)

void afiseazaDate() const {

cout << "Id produs: " << idProdus << endl;

cout << "Nume: " << nume << endl;

cout << "Pret: " << pret << endl;

cout << "Descriere: " << descriere << endl;

cout << "Categorie: " << categorie << endl;

}

};

class Comanda {

private:

int idComanda;

vector<Produs\*> listaProduse;

string statusComanda;

string metodaPlata;

string adresaLivrare;

public:

// Constructor

Comanda(int idComanda, vector<Produs\*> listaProduse = {}, string statusComanda = "", string metodaPlata = "", string adresaLivrare = "") {

this->idComanda = idComanda;

this->listaProduse = listaProduse;

this->statusComanda = statusComanda;

this->metodaPlata = metodaPlata;

this->adresaLivrare = adresaLivrare;

}

// Afișează detaliile comenzii (ID comandă, lista de produse, statusul comenzii, metoda de plată, adresa de livrare, prețul total și descrierea comenzii)

void afiseazaDate() const {

cout << "Id comanda: " << idComanda << endl;

cout << "Lista produse:" << endl;

for (const Produs\* produs : listaProduse) {

produs->afiseazaDate();

}

cout << "Status comanda: " << statusComanda << endl;

cout << "Metoda de plata: " << metodaPlata << endl;

cout << "Adresa livrare: " << adresaLivrare << endl;

cout << "Pret total: " << getPret() << endl;

cout << "Descriere: " << getDescriere() << endl << endl;

}

// Calculează și returnează prețul total al comenzii prin adunarea prețurilor tuturor produselor din listă.

float getPret() const {

float pretTotal = 0;

for (const Produs\* produs : listaProduse) {

pretTotal += produs->getPret();

}

return pretTotal;

}

// Generează și returnează o descriere a comenzii care include ID-ul comenzii și o listă de produse cu numele și descrierea fiecărui produs.

string getDescriere() const {

string descriereComanda = "Comanda " + to\_string(idComanda) + ":\n";

for (const Produs\* produs : listaProduse) {

descriereComanda += "- " + produs->getNume() + " (" + produs->getDescriere() + ")\n";

}

return descriereComanda;

}

// Adaugă un produs la lista de produse din comandă

void adaugaProdus(Produs\* produs) {

listaProduse.push\_back(produs); // Adaugarea pointerului la produs in lista

}

};

int main() {

// Creez instante de produse

Produs produs1(1, "Laptop", 2500, "Laptop i7 cu 16GB RAM", "Electronice");

Produs produs2(2, "Telefon", 500, "Telefon mobil Android", "Electronice");

cout<<"\n";

// Creez instanta de comanda

Comanda comanda1(101, {},"procesata", "card", "Calarasi, prel bucuresti"); // Constructor cu parametrii optionali

// Adaugă produsul "Laptop" la lista de produse din comandă comanda1.adaugaProdus(&produs1);

comanda1.adaugaProdus(&produs2);

// Afișează toate detaliile comenzii, inclusiv lista de produse, prețul total și descriereas comanda1.afiseazaDate();

return 0;

}