DOCUMENTAȚIE

TEMA NUMĂRUL_3

GESTIONAREA COMENZILOR

NUME STUDENT: ARDELEAN RALUCA-NICOLETA

GRUPA: 30227

Cuprins

1. Obiectivul temei

- ## Obiectivul principal

  Obiectivul principal al acestei teme este crearea și implementarea unui aplicații de gestionare a clienților, a produselor și a comenzilor unui depozit cu o interfață grafică interactivă prin intermediul căreia utilizatorul poate insera/actualiza/șterge clienți și produse și poate realiza o comandă selectând un client și un produs.

- ## Obiectivele secundare

  Pentru îndeplinirea obiectivului principal au fost urmați următorii pași:

  - Analiza problemei, modelarea, scenarii, cazuri de utilizare
  - Proiectarea aplicației de gestionare a clienților, a produselor și a comenzilor unui depozit
  - Implementarea propriu-zisa a aplicației
  - Testarea aplicației

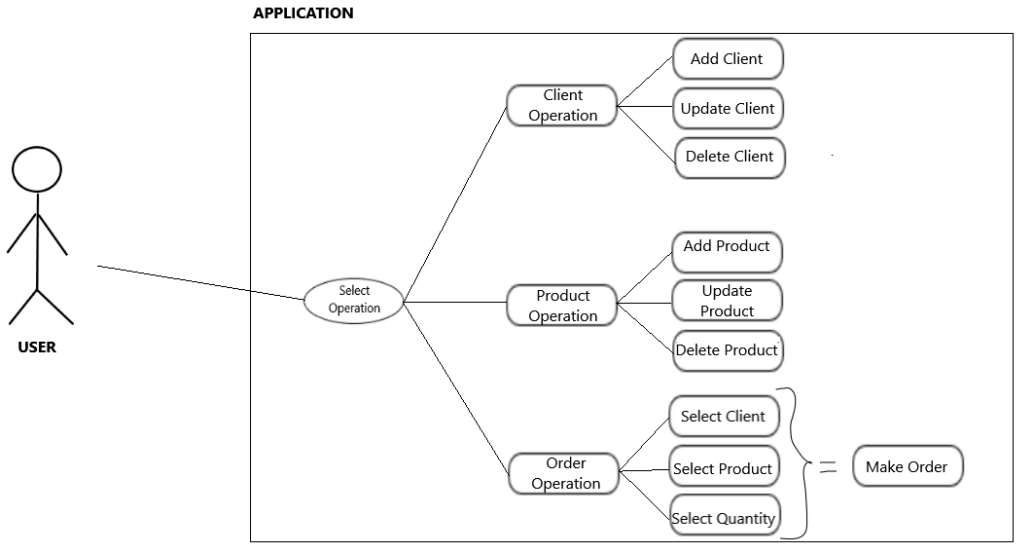
  Aceste aspecte vor fi detaliate în capitolele următoare.

2. Analiza problemei, modelarea, scenarii, cazuri de utilizare

- ## Cerințe funcționale

  - Aplicația trebuie să permită utilizatorului să aleagă operațiunea dorită: Operații pe tabela de clienți, operații pe tabela de produse și operații pe tabela de comenzi.
  - Aplicația trebuie să permită utilizatorului să adauge un client nou.
  - Aplicația trebuie să permită utilizatorului să editeze un client existent.
  - Aplicația trebuie să permită utilizatorului să șteargă un client existent.
  - Aplicația trebuie să permită utilizatorului să adauge un produs nou.
  - Aplicația trebuie să permită utilizatorului să editeze un produs existent.
  - Aplicația trebuie să permită utilizatorului să șteargă un produs existent.
  - Aplicația trebuie să permită utilizatorului să creeze o comandă prin selectarea unui client și a unui produs dintr-un tabel de clienți și un tabel de produse și introducerea unei cantități valide, iar apoi inserarea acestei comenzi în tabela de comenzi.

- Cerințe non-funcționale
  - Aplicația trebuie să fie intuitivă și ușor de utilizat.
  - Aplicația trebuie să aibă o interfață grafică interactivă care să faciliteze utilizarea lui
- Cazuri de utilizare



Cazuri de utilizare: Operații pe tabela de clienți, Operații pe tabela de produse

1. Utilizatorul alege din fereastra principală operațiunea dorită: Operații pe tabela de clienți sau pe tabela de produse.

2.1.Pentru adăugarea unui nou client/produs, utilizatorul introduce datele, iar apoi apasă butonul “INSERT”.

2.2.Pentru actualizarea unui client/produs, utilizatorul selectează clientul/produsul din tabel și introduce noile date, iar apoi apasă butonul „UPDATE”.

2.3.Pentru ștergerea unui client/produs, utilizatorul selectează clientul/produsul ce se dorește a se șterge, iar apoi apasă butonul “DELETE”.

3.Baza de date corespunzătoare clienților/produselor se actualizează în funcție de operația realizată la punctul anterior.

Cazuri de utilizare: Operații pe tabela de comenzi

1.Utilizatorul alege din fereastra principală operațiunea dorită: Operații pe tabela de comenzi.

2.Utilizatorul trebuie să selecteze un client, un produs și să introducă o cantitate validă, iar apoi apasă butonul “Make order”, ceea ce va determina inserarea unei noi comenzi în tabelă.

3. Baza de date corespunzătoare comenzilor se actualizează.

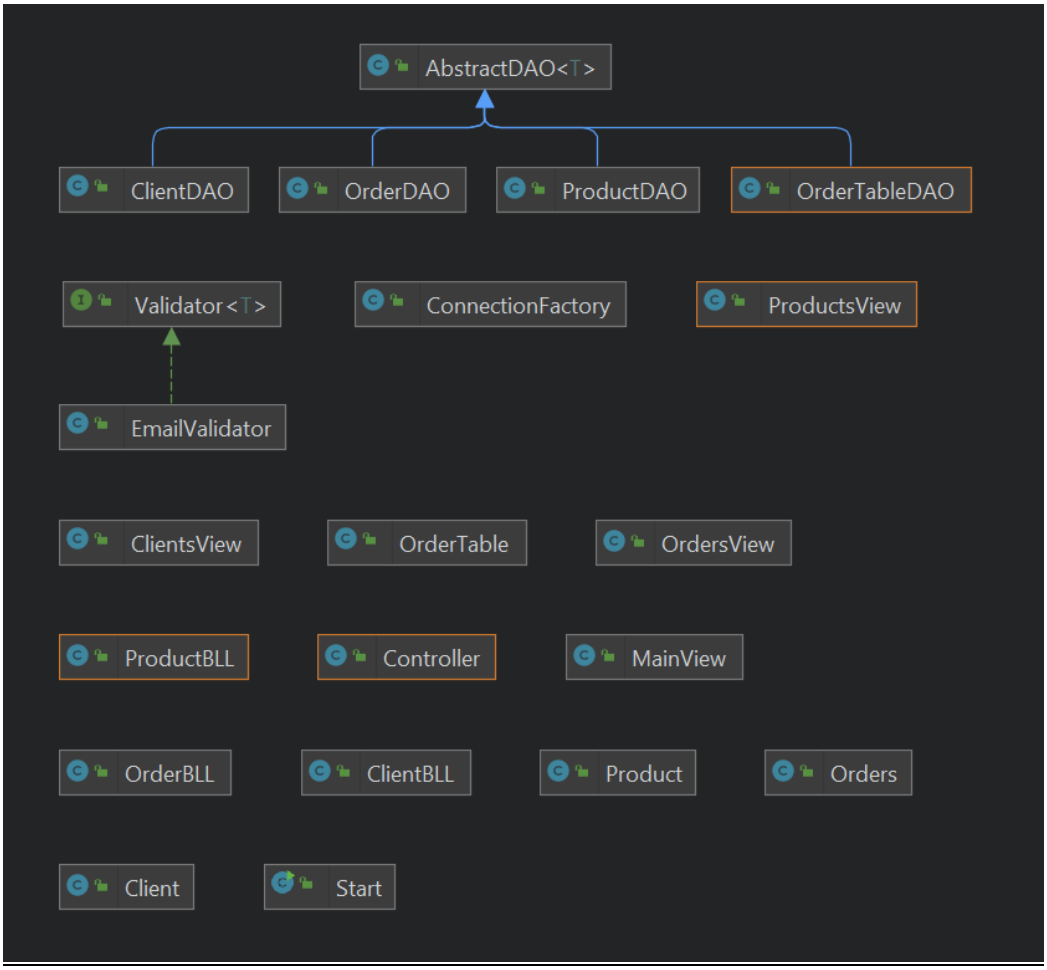
Scenariu alternativ: Email invalid, cantitate invalidă.

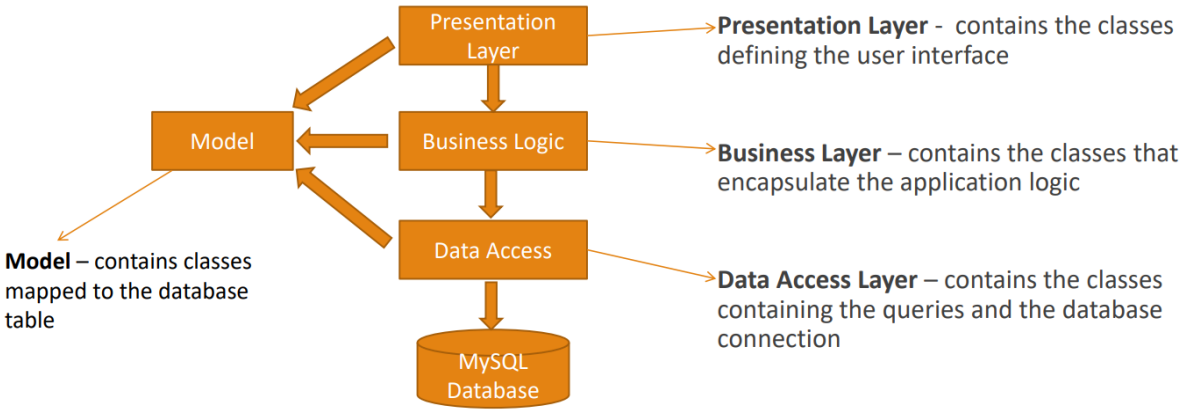
Pentru cazul când utilizatorul dorește introducerea unui nou client în baza de date, dacă email-ul nu respecta pattern-ul stabilit, se va afișa un mesaj corespunzător și utilizatorul va trebui să introducă un alt mail-ul.

Pentru cazul când utilizatorul dorește introducerea unei cantități pentru crearea unei comenzi, dacă această cantitate este negativă sau este mai mare decât stocul existent, se va afișa un mesaj corespunzător și utilizatorul va trebui să introducă o altă cantitate.

3. Proiectarea

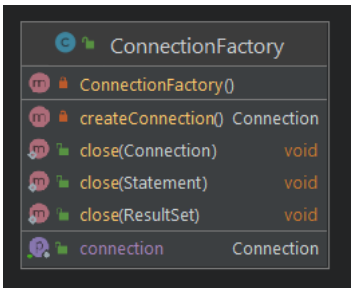
Diagrama UML generată direct din IntelliJ:





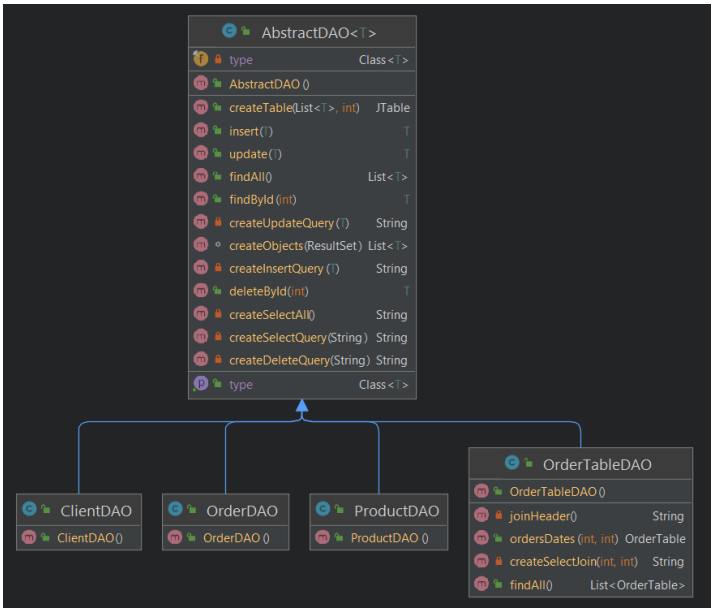
4. Implementarea

- MySQL Database -> Connection



Clasa "ConnectionFactory" face legătura cu baza de date și conține o metodă care deschide baza de date, o metoda care o închide, o metodă pentru închiderea unei operații și una pentru închiderea rezultatului obținut în urma unei operații.

- Data Access -> dao



Clasa "AbstractDAO" conține metode care aplică tehnica reflexiei pentru manipularea eficienta a tabelelor din baza de date.

Conține metode care creează interogări pe tabele pentru insert/update/delete/find și metode care aplică efectiv aceste interogări:

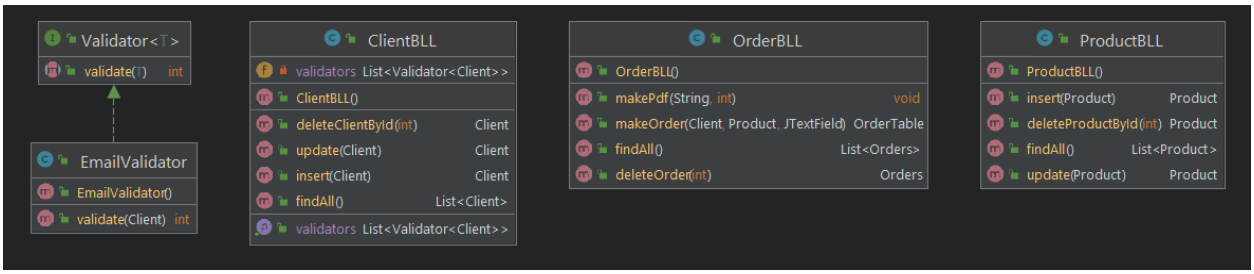
- update() folosește createUpdateQuery() -> realizează actualizarea unei înregistrară din tabel
- insert() folosește createInsertQuery() -> adaugă obiect nou în tabel

- findall() folosește createInsertQuery() -> returnează o listă cu toate obiectele din tabel
- findById() folosește createSelectQuery(field) cu field = “id” -> găsește înregistrarea din tabel care are un anumit id
- deleteById folosește createDeleteQuery(field) cu field = “id” -> sterge o înregistrare din tabel care are id-ul specificat.

Pe lângă acestea mai există o metodă care creează prin reflexie un tabel dintr-o listă de obiecte și o metodă care creează un obiect din ceea ce returnează interogarea asupra tabelului.

Clasele “ClientDAO”, “ProductDAO”, “OrderDAO”, “OrderTableDAO” moștenesc clasa “AbstractDAO”. Primele 3 folosesc metodele nemodificate din clasa părinte, în timp ce clasa “OrderTableDAO” suprascrie metoda „findAll()” folosind interogarea creată de metoda “joinHeader()”.În plus față de clasa părinte, aceasta mai conține metoda “ordersDates” care utilizează interogarea creată de metoda “ createSelectJoin()” și returnează un nou obiect cu datele din înregistrarea ce a rezultat în urma selecției făcute. Această clasă nu are un tabel corespondent în baza de date, ea fiind creată pentru a ușura afișarea tabelului de comenzi care conține doar id-uri.

❖ Business logic -> bll



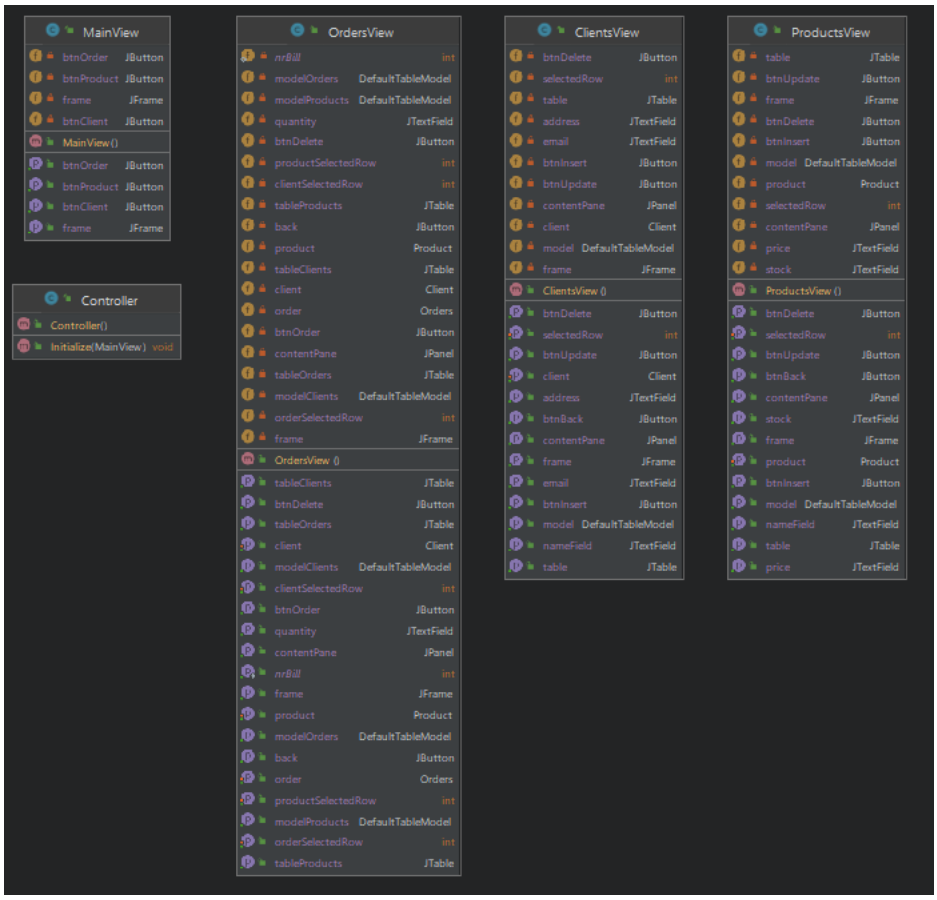
Clasa “EmailValidator” implementează interfața “Validator” și validează dacă email-ul unui client introdus respectă structura de mail folosind un pattern.

Clasa “ClientBLL” implemetează efectiv operațiile de insert/update/delete/find asupra tabelei corespunzătoare clienților.

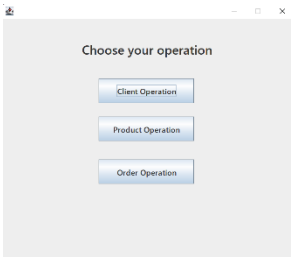
Clasa “ProductBLL” implemetează efectiv operațiile de insert/update/delete/find asupra tabelei corespunzătoare produselor.

Clasa “OrderBLL” implemetează efectiv operațiile de insert/delete/find asupra tabelei corespunzătoare comenzilor. De asemenea, conține și o metodă care creează un pdf ce reprezintă factura pentru o anumită comandă plasată, aceasta fiind apelată în metoda “makeOrder” care realizează de fapt inserarea în tabelă.

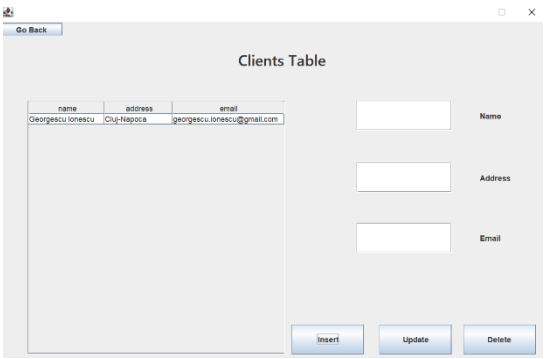
❖ Presentation layer -> presentation



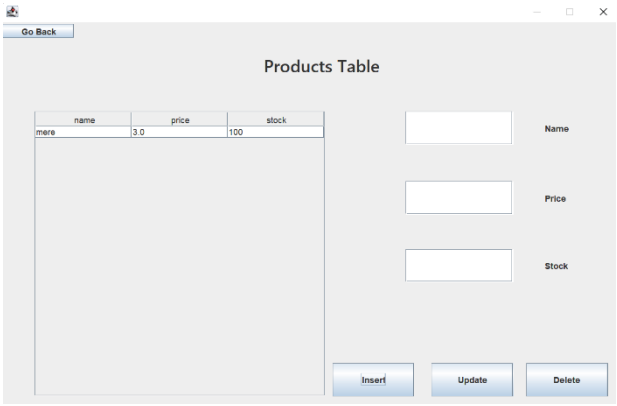
Clasa “MainView” corespunde ferestrei principale din interfață în care utilizatorul alege asupra cărei tabele va realiza operatii.



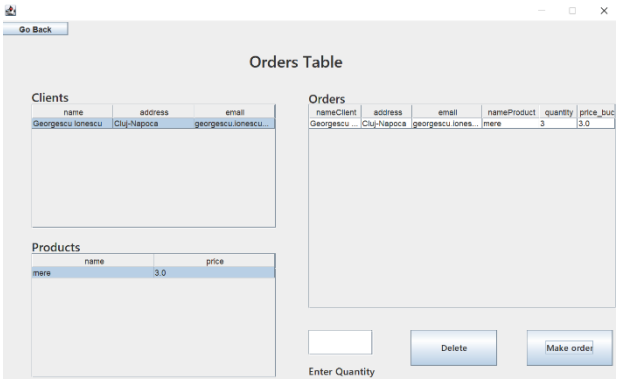
Clasa “CliensView” corespunde ferestrei din interfață în care utilizatorul vede tabela corespunzătoare clienților și poate realizeaza operații asupra acesteia.



Clasa “ProductsView” corespunde ferestrei din interfață în care utilizatorul vede tabela corespunzătoare produselor și poate realizeaza operații asupra acesteia.

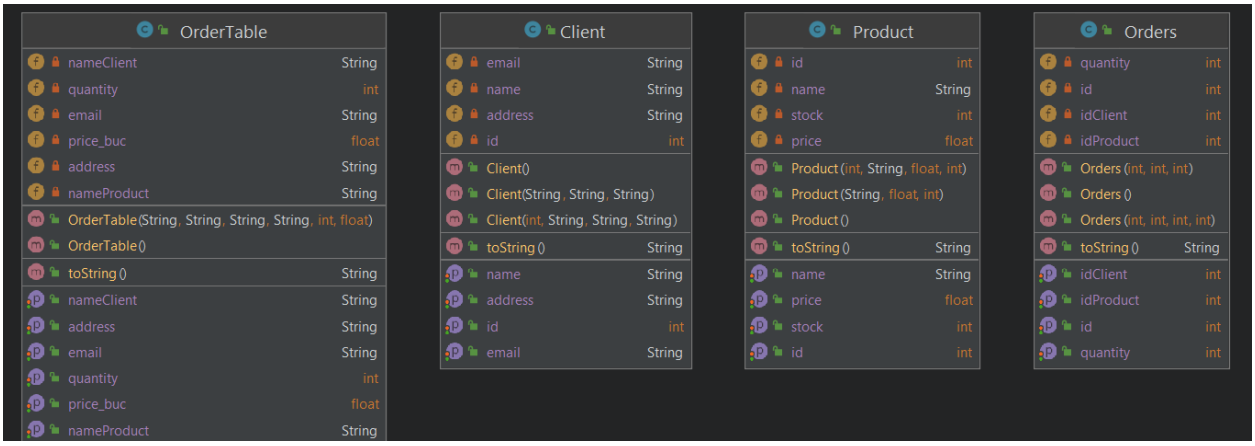


Clasa “OrdersView” corespunde ferestrei din interfață în care utilizatorul vede tabela corespunzătoare comezilor și poate realizeaza operații asupra acesteia.



Clasa “Controller” realizează manipulează butoanele, textField-urile și tabelelor din cele 4 ferestre.

❖ Model -> model



Clasa “Client” e corespunzătoare tabelei clienților care conține înregistrări ce reprezintă de fapt obiecte de această clasa. Cu aceasta clasă se mapează tabelul clienților din interfață.

Clasa “Product” e corespunzătoare tabelei produselor care conține înregistrări ce reprezintă de fapt obiecte de această clasa. Cu aceasta clasă se mapează tabelul clienților din interfață.

Clasa “Orders” e corespunzătoare tabelei comenzilor care conține înregistrări ce reprezintă de fapt obiecte de această clasa.

Clasa “OrderTable” nu corespunde niciunei tabele din baza de date, dar prin intermediul ei se mapează tabelul comenzilor din interfață.

6. Concluzii

Realizând această temă am învățat o tehnică foarte utilă și des folosită în programarea orientata pe obiect, și anume reflexia. Utilizând această tehnică este ușor de realizat o clasa abstractă care poate fi moștenită, evitându-se astfel scrierea de mai multe ori al aceluiași cod care folosește clase diferite.

- ## Posibile dezvoltări ulterioare
  - Realizarea de rapoarte pentru fiecare client, analizând de exemplu cât de des acel client a cumpărat din acel magazin, determinând astfel și clienții fideli.
  - Realizarea de rapoarte pentru fiecare produs, analizând de exemplu cât de des acel produs a fost cumpărat din acel magazin, determinând astfel și produsele cele mai vândute.

7. Bibliografie

1. Stack Overflow
2. stackHowTo
3. Baeldung