DOCUMENTAȚIE

TEMA 3

NUME STUDENT: Ghiorghe Sorana
GRUPA: 30225

Cuprins

1. Obiectivul temei

Obiectiv principal

Obiectivul principal al temei este proiectarea și implementarea unei aplicații pentru gestionarea comenzilor plasate de clienti utilizand o baza de date sql pentru eficiența și accesare usoara. Acest lucru este realizat alaturi de o interfata grafică intuitivă prin care se pot realiza diferite operatii asupra produselor oferite, clienții inregistrați și comenzile plasate de aceștia.

Obiectivele secundare

1.	Determinarea claselor necesare pentru implementarea proiectului
2.	Crearea unei conexiuni la baza de date existenta in MySQL • Crearea unei baze de date cu 3 tabele conectate pentru pastrarea integrității datelor • Adaugarea unei clase responsabile de intretinerea conexiunii cu baza de date
3.	Crearea unei clase pentru accesul la date • Crearea unei clase abstracte pentru accesul general, pentru a fi restrictionat în funcție de nevoile utilizatorului • Adăugarea claselor care extind clasa si functiile sale generice.
4.	Crearea unei protectii asupra accesul la date • Adaugarea unei clase de BusinessLogic pentru validarea accesului la date și restricționarea in accesul bazei de date necorespunzator.
5.	Implementarea unei interfețe grafice pentru utilizare intuitiva • Prezenta ferestrelor diferite în funcție de scopul utilizării (accesul la clienti, accesul la produse, inserarea unei comenzi) • Adaugarea sectiunilor de preluare text de la tastatura pentru a fi manipulat
3.	Generare Javadoc • Descrierea conținutului fiecărei clase pentru claritate și utilizare corespunzatoare.
5.	Documentațe • Elaborarea unei documentații care să cuprindă detalii despre implementarea aplicației pentru o înțelegere corespunzătoare asupra gestiunii comenzilor și modul în care funcționează.

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

Problema abordată constă în managementul ineficient și dificil al comenzilor plasate în cadrul unui depozit care utilizeaza registre scrise de mână care sunt susceptibile erorilor, nu garantează integritatea datelor și consuma mult timp. În acest context, aplicația propune trasarea unei baze de date in MySQL pentru a realiza legaturi ușor de urmarit între produse și clienți, asigurand astfel integritatea datelor. În scopul accesului la aceasta baza de date cu ușurință și intuitiv, este adaugată o interfață grafică cu butoane și zone de text usor de utilizat.

Cerințe Funcționale

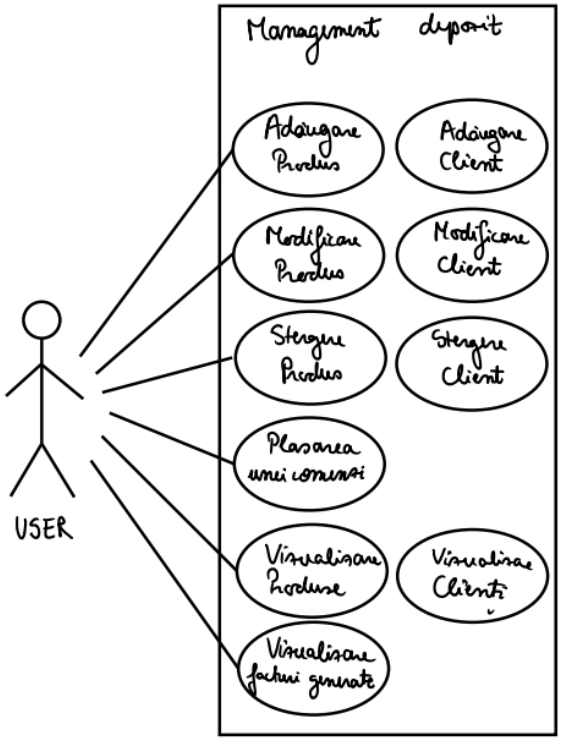
- Utilizatorul poate introduce informații legate de clienți și produse de la tastatură sau să modifice informația existenta despre acestea (incluzand stergerea acestora, realizată prin cascadare).
- Aplicația trebuie să comunice cu o baza de date care afișează conținutul introdus în tabele în funcție de tabelul selectat.
- Datele introduse de la tastatura trebuie validate pentru a pastra integritatea datelor și prevenirea erorilor
- Utilizatorul poate plasa comenzi în limita stocului disponibil, avand posibilitatea de a selecta un client si un produs existent din tabele si generarea unei comenzi, fară memorarea id-urilor care poate conduce la erori.
- Afișarea unui tabel de facturi generate in urma comenzilor plasate, păstrând informațiile indiferent de modificarea datelor ulterioare din celalalte tabele.

Cerințe Non-funcționale

- Interfața prin care utilizatorul interacționează cu aplicația trebuie să fie intuitivă și ușor de utilizat.
- Afișarea continutului tabelelor în timp real (dupa modificari / stergeri / inserari) și alte informații relevante.
- Utilizatorul trebuie să poată folosi aplicația fără erori sau buguri neașteptate.
- Programul trebuie să fie pregătit pentru o utilizare necorespunzătoare și să atenționeze în cazul unui input greșit.

Cazuri de utilizare

Aplicația va fi utilizată pentru a simula inventarul unui depozit cu ajutorul tabelelor MySQl și al operatilor realizate în cadrul acestora (inserari, stergeri, actualizari) în funcție de criterii specificate de utilizator în timp ce acesta poate vedea clar si concis comenzile plasate. Aplicația poate fi utilizată pentru a usura accesul la baza de date SQL avand interfața intuitivă și pentru gestiunea comenzilor plasate.

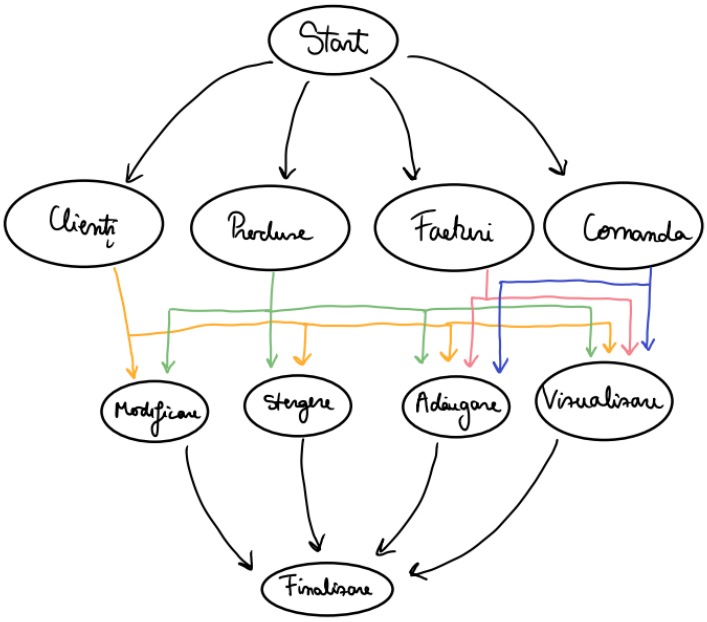


Scenariu principal de succes:

- Utilizare corectă a aplicației
  - Actor primar: utilizatorul
  - Utilizatorul alege să acceseze una dintre ferestrele principale de gestionare a datelor.
  - Utilizatorula adauga un client/produs de la tastatură pentru a-l insera.
  - Utilizatorul alege un client/produs existent din tabel și îl modifică/sterge.
  - Utilizatorul alege clientul și produsul pentru care va plasa comanda și introduce o valoare validă pentru cantitate.
  - Utilizatorul poate accesa istoricul comenzilor prin vizualizarea facturilor generate.

Scenarii alternative:

- Introducerea cantitatii necorespunzatoare
  - În plasarea unei comenzi, utilizatorul introduce o cantitate mai mare decât cantitatea de stoc existentă pentru acel produs. Programul detectează acest lucru și afișează un mesaj de eroare pe ecran, solicitând reintroducerea cantității.
  - Se revine la pasul inițial, utilizatorul fiind încurajat să reintroducă datele.
- Introducerea datelor negative
  - Utilizatorul introduce stoc sau preț negativ pentru un produs, iar aplicația afișează un mesaj de eroare pe ecran.
  - Se revine la pasul inițial, utilizatorul fiind încurajat să reintroducă date corecte.



Comportamentul așteptat al aplicației

3. Proiectare

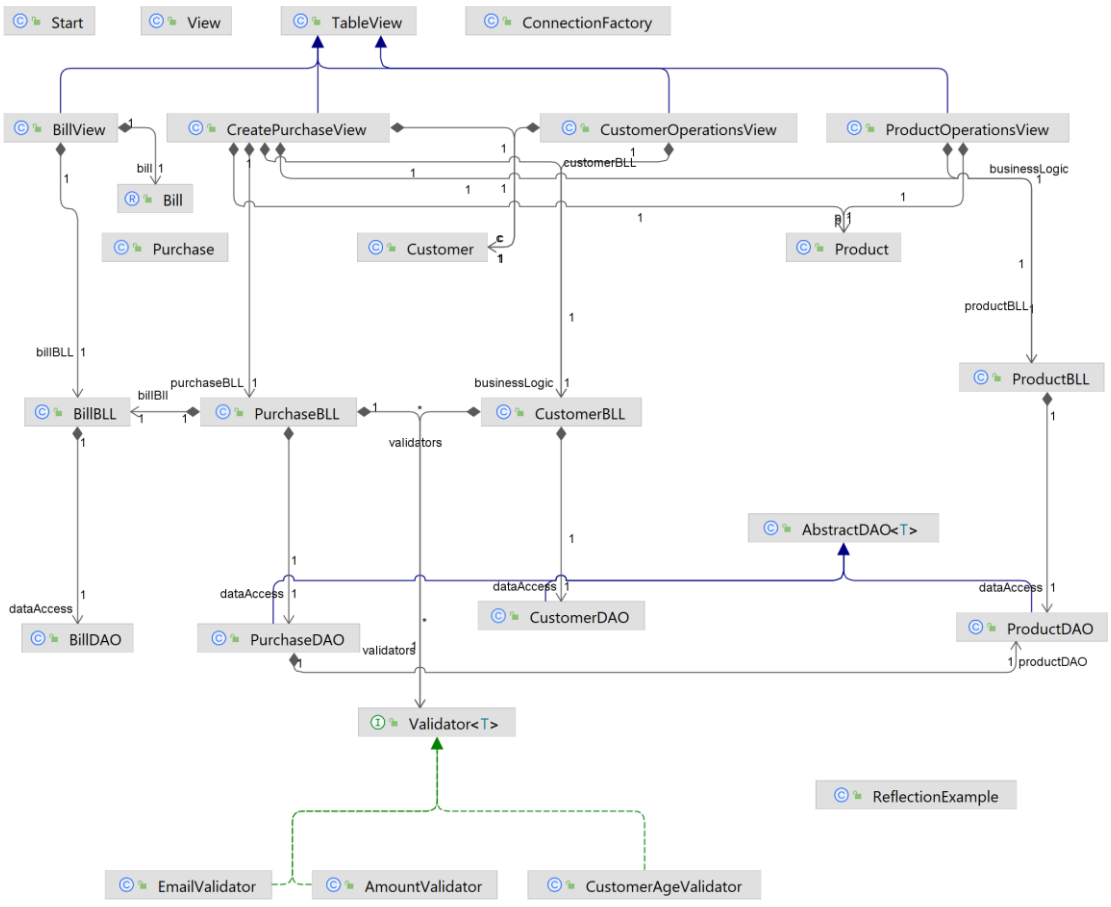
a. Proiectarea OOP

Aplicația este structurată în 5 pachete: Presentation, BLL, DAO, Connection și model. Este utilizată o arhitectură definită pe straturi.

Clasele au fost definite în așa fel încât să se respecte principiul OOP de abstractizare și encapsulare a datelor.

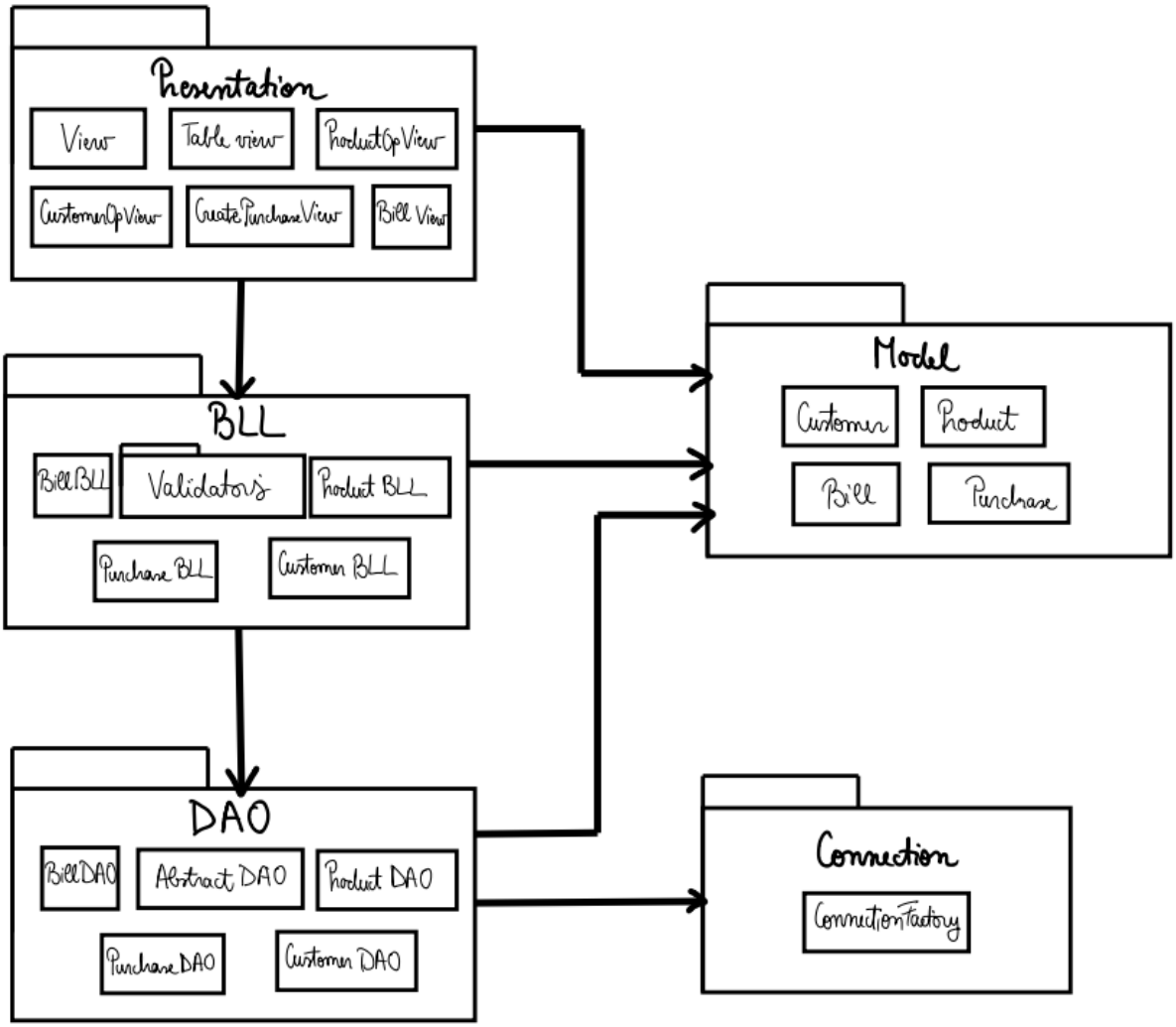
b. Diagrama UML a claselor

Se pot observa legaturile principale intre clase și pastrarea directiei de transmitere a datelor (ex: clasele de tip View nu comunică direct cu DataAccess)



c. Diagrama UML a pachetelor

Se poate observa utilizarea structurii de Layered Architecture.

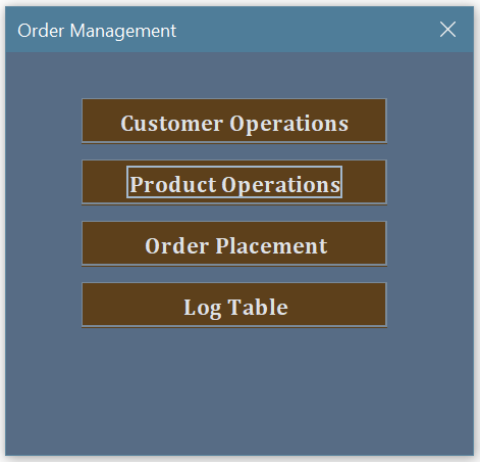


d. Algoritmi folositi

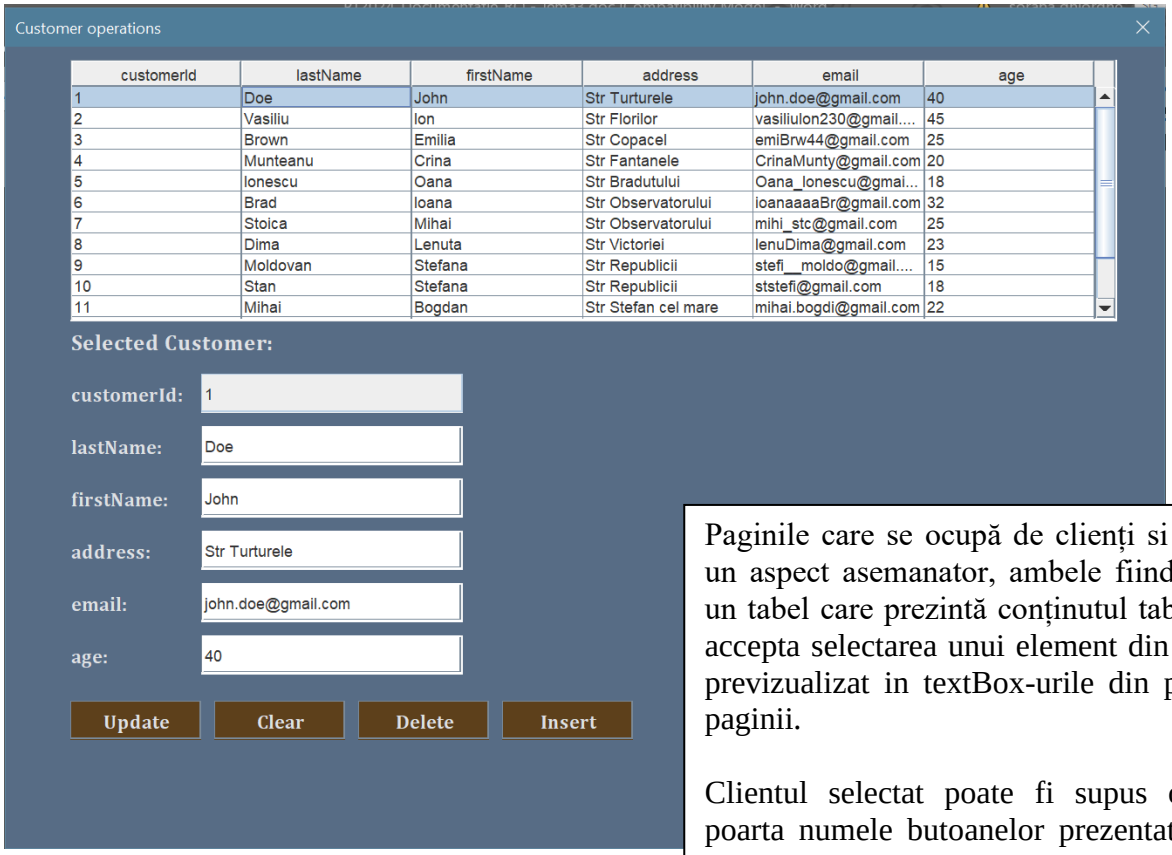
Algoritmii utilizați sunt în principal legați de accesul la baza de date si crearea querry-urilor pentru a accesa si modifica datele din tabele. Aceștia sunt prezenți în pachetul DAO.

4. Implementare

În dezvoltarea acestui proiect, am utilizat 4 module principale: dataAccess, care realizează conexiunea cu baza de date, BusinessLogic, care asigură apelarea corespunzatoare a metodelor din dataAccess, View, care determina interfața grafică pentru urmarirea activității modului BusinessLogic și Model, care modeleaza tipul de obiecte utilizate pentru desfasurarea aplicației. Aspectul interfeței poate fi observat în imaginile de mai jos.

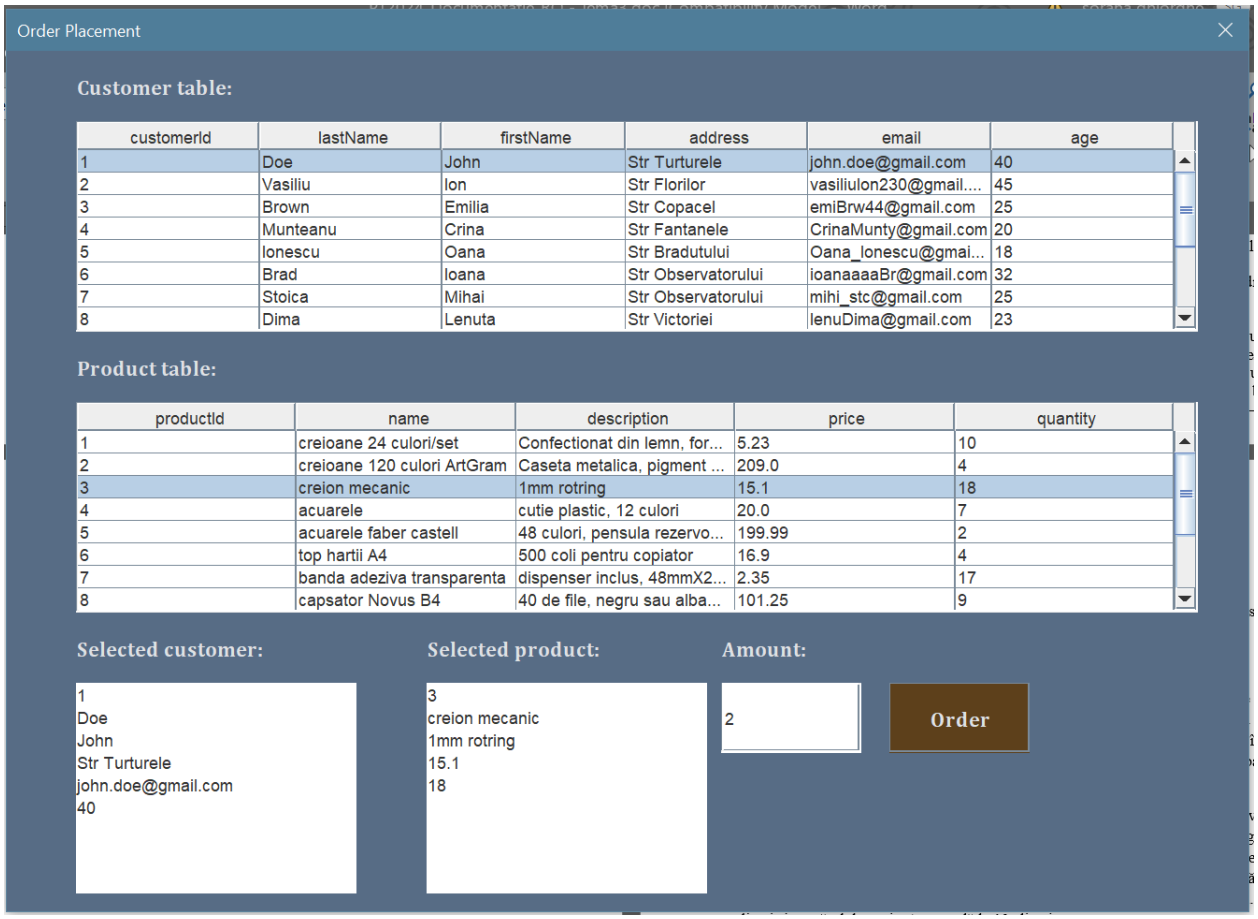


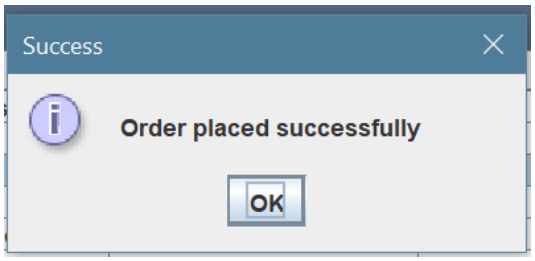
Prima pagină reprezintă un meniu de selecție a diferitelor funcții implementate în aplicație. Fiecare buton va deschide o fereastră nouă care are acces la diferite operații legate de tabelul cu același nume.

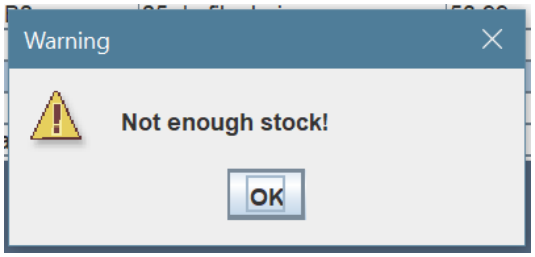


Paginile care se ocupă de clienți si de produse au un aspect asemanator, ambele fiind inițializate cu un tabel care prezintă conținutul tabelelor. Tabelul accepta selectarea unui element din tabel care este previzualizat in textBox-urile din partea stanga a paginii.

Clientul selectat poate fi supus operatilor care poarta numele butoanelor prezentate in partea de jos a paginii. In cazul insertiei unui nou client, se pot goli textBoxurile cu ajutorul butonului clear.



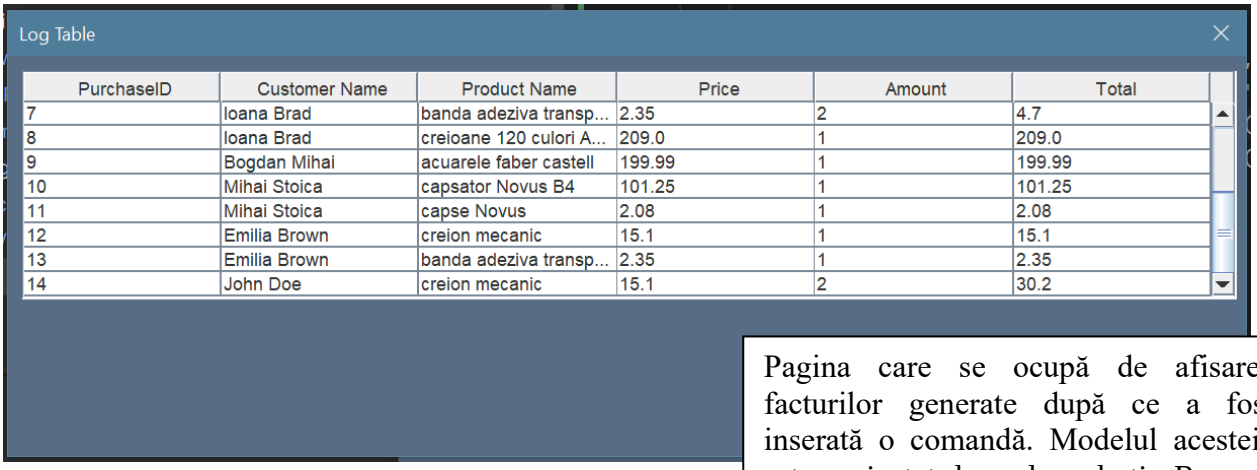




Pagina care se ocupă de comenzi prezintă conținutul ambelor tabele. Tabelul accepta selectarea unui element din fiecare tabel care este previzualizat in textArea-urile din partea de jos a paginii.

Datele din textArea nu pot fi modificate, acest lucru poate fi realizat doar în paginile lor respective.

TextField-ul de “Amount” preia de la tastatura cantitatea de produs de comandat de catre client. Butonul “Order” inserează continutul în tabelul de comenzi dacă datele sunt valide, sau afisează un mesaj de eroare.



Pagina care se ocupă de afisarea facturilor generate după ce a fost inserată o comandă. Modelul acesteia este proiectat de o clasa de tip Record, asigurand faptul că facturile raman și în cazul în care anumite date sunt eliminate sau modificate din tabele.

5. Rezultate

Rezultatul querry-urilor poate fi observat în interfață în timp real, deoarece tabelele sunt updatate după fiecare operație.

În tabelul Bill se poate observa rezultatul comenzii plasate în OrderPlacement în poza atașată anterior.

6. Concluzii

Acest proiect a reprezentat o oportunitate relevantă în dezvoltarea propriilor cunoștințe și abilități și în aprofundarea acestora. Am dobândit abilitatea de a lucra cu clase abstracte și metode generice cât și implementarea acestora într-un mod sigur, folosind alte metode care le accesează în mod sigur. Am învățat cum se utilizează Reflection pentru a prelua campurile claselor model în mod dinamic și cum se implementează un proiect care utilizeaza Layered Architecture.

Mi-am reîmprospătat cunoștințele legate de SQL și baze de date, utilizarea componentelor de tip JTable și diferite elemente de interfață grafică pentru a le updata după modificarile realizate în baza de date. Proiectul mi-a oferit oportunitatea de a aprofunda și repeta diferite elemente și paradigme ale limbajului de programare orientat pe obiect.

Aplicația determină accesul la o bază de date simplă, implementând diferite operații simple cu succes. În ceea ce privește viitoarele dezvoltări ale aplicației, se poate dezvolta baza de date pentru a stoca mai multe informații precum un tabel pentru furnizor, istoric produs în caz că se modifică pretul acestuia, informații despre curieri și livrări, etc. De asemenea, se pot adauga alte querry-uri interesante pentru modificarea tabelelor în alte moduri.

7. Bibliografie

1. *Git Lab* - https://www.jetbrains.com/help/idea/gitlab.html
2. *Interfaces* - https://www.geeksforgeeks.org/interfaces-in-java/
3. *JTables* - https://www.geeksforgeeks.org/java-swing-jtable/
4. *Getting a selected row in JTables* - https://stackoverflow.com/questions/29345792/java-jtable-getting-the-data-of-the-selected-row
5. *Reflection* - https://www.baeldung.com/java-reflection
6. *LayeredArchitecture* - https://medium.com/java-vault/layered-architecture-b2f4ebe8d587
7. *Immutable Class* - https://www.geeksforgeeks.org/create-immutable-class-java/
8. *Javadoc* - https://www.jetbrains.com/help/idea/javadocs.html
9. *Resurse Exemple* - https://gitlab.com/utcn_dsrl/pt-layered-architecture , https://gitlab.com/utcn_dsrl/pt-reflection-example
10. *Resurse Laborator* - https://dsrl.eu/courses/pt/materials/PT2024_A3_S1.pdf , https://dsrl.eu/courses/pt/materials/PT2024_A3_S2.pdf