DOCUMENTATIE

TEMA 3

GHICA MADALINA

GRUPA 30229

# CUPRINS

[1. Obiectivul temei 3](file:///C:\Users\ghica\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///C:\Users\ghica\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297886)

[3. Proiectare 4](file:///C:\Users\ghica\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297887)

[4. Implementare 5](file:///C:\Users\ghica\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297888)

[5. Rezultate 12](file:///C:\Users\ghica\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297889)

[6. Concluzii 13](file:///C:\Users\ghica\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297890)

[7. Bibliografie 13](file:///C:\Users\ghica\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297891)

# Obiectivul temei

Obiectivul principal al proiectului este să dezvoltăm o aplicație care să faciliteze gestionarea comenzilor clienților într-un depozit. Pentru a atinge acest obiectiv, vom analiza în detaliu problema și vom identifica cerințele specifice ale aplicației. Urmează să proiectăm o interfață intuitivă și eficientă pentru gestionarea comenzilor, luând în considerare toate aspectele relevante.

Apoi, vom implementa aplicația, asigurându-ne că funcționalitățile sunt implementate corespunzător și că aceasta poate fi utilizată într-un mod fluid și eficient de către utilizatori. În cele din urmă, vom efectua teste riguroase pentru a verifica funcționalitatea și performanța aplicației, asigurându-ne că aceasta funcționează corect și îndeplinește toate cerințele stabilite.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

**Cerințe funcționale:**

Aplicația de management trebuie să ofere posibilitatea adăugării unui nou client în sistem.

Aplicația de management trebuie să permită editarea informațiilor unui client existent.

Aplicația de management trebuie să permită ștergerea unui client existent din sistem.

Aplicația de management trebuie să afișeze lista tuturor clienților existenți în sistem.

Aplicația de management trebuie să ofere posibilitatea adăugării unui nou produs în inventar.

Aplicația de management trebuie să permită editarea informațiilor unui produs existent.

Aplicația de management trebuie să permită ștergerea unui produs existent din inventar.

Aplicația de management trebuie să afișeze lista tuturor produselor existente în inventar.

Aplicația de management trebuie să ofere posibilitatea de a crea comenzi de produse prin selectarea unui client existent și a unui produs existent, precum și specificarea cantității dorite.

Aplicația de management trebuie să afișeze un mesaj de eroare în cazul în care nu există suficiente produse pentru a completa comanda.

Aplicația de management trebuie să actualizeze stocul de produse în urma finalizării unei comenzi.

Aplicația de management trebuie să genereze o factură pentru fiecare comandă și să o stocheze într-un tabel de log-uri.

**Cazuri de utilizare:**

* Funcționalitatea de adăugare a unui client nou trebuie implementată în aplicație.
* Utilizatorul ar trebui să poată edita informațiile unui client existent în aplicație.
* Funcționalitatea de ștergere a unui client existent trebuie să fie disponibilă în aplicație. Utilizatorul ar trebui să poată vizualiza o listă a tuturor clienților într-un tabel.
* Implementarea adăugării unui produs nou trebuie să fie inclusă în aplicație.
* Utilizatorul trebuie să aibă posibilitatea de a edita informațiile unui produs existent în aplicație.
* Funcționalitatea de ștergere a unui produs existent trebuie să fie implementată în aplicație.
* Utilizatorul ar trebui să poată vizualiza o listă a tuturor produselor într-un tabel.
* Aplicația trebuie să ofere posibilitatea de a crea o comandă de produse.
* În cazul în care nu există suficiente produse pentru a completa comanda, aplicația ar trebui să afișeze un mesaj de eroare corespunzător.
* Stocul de produse trebuie să fie actualizat în urma finalizării unei comenzi.
* Pentru fiecare comandă, aplicația trebuie să genereze o factură și să o stocheze într-un tabel de log-uri.
* Aplicația trebuie să permită accesarea bazei de date pentru a crea, edita, șterge și căuta obiecte.

A diagram of a person with text

Description automatically generated with low confidence

# Proiectare

A picture containing diagram, text, line, plan

Description automatically generated

**Diagrame UML**

**A picture containing text, diagram, parallel, plan

Description automatically generated**

# Implementare

Pachetul Business Logic contine:

* interfata Vaidators, validatori pentru varsta si email (clasele AgeValidator si EmailValidator)
* clasa ClientBLL prezintă o componentă a logică a aplicației care se ocupă de operațiile referitoare la clienți.
  + De asemenea, clasa ClientBLL depinde de clasa ClientDAO, care gestionează operațiunile de acces la date pentru obiectele Client.
* clasa OrderBLL oferă metode pentru căutarea și inserarea comenzilor în sistem, asigurându-se că comenzile sunt valide înainte de a fi salvate în baza de date.
* clasa ProductBLL oferă metode pentru căutarea și inserarea produselor în sistem, asigurându-se că produsele sunt valide înainte de a fi salvate în baza de date.

Pachetul Connection contine:

* clasa ConnectionFactory oferă metode pentru crearea și gestionarea conexiunilor la baza de date folosind JDBC. Aceasta asigură înregistrarea driver-ului JDBC, crearea conexiunilor, precum și închiderea conexiunilor, a statement-urilor SQL și a ResultSet-urilor pentru a evita scurgerile de resurse și a asigura o gestionare adecvată a conexiunilor la baza de date.

Pachetul Data Access contine:

* clasa ClientDAO oferă metode pentru efectuarea operațiilor de bază ale bazei de date (cum ar fi căutarea și inserarea) pentru entitatea Client. Aceasta utilizează ConnectionFactory pentru gestionarea conexiunii la baza de date și asigură închiderea corectă a resurselor pentru a evita scurgerile de resurse și problemele de performanță.
* clasa OrderDAO oferă metode pentru accesarea și manipularea datelor de comenzi în baza de date. Aceasta utilizează ConnectionFactory pentru gestionarea conexiunii la baza de date și asigură închiderea corectă a resurselor pentru a evita scurgerile de resurse și problemele de performanță.
* clasa ProductDAO oferă metode pentru accesarea și manipularea datelor de produse în baza de date. Aceasta utilizează ConnectionFactory pentru gestionarea conexiunii la baza de date și asigură închiderea corectă a resurselor pentru a evita scurgerile de resurse și problemele de performanță.

Pachetul Model contine:

* clasa Client cu getterele si setterele specifice
* clasa Product cu getterele si setterele specifice
* clasa Order cu getterele si setterele specifice

Pachetul Presentation contine:

* Clasa GUIInterface reprezintă interfața grafică a utilizatorului pentru gestionarea clienților și produselor. Aceasta oferă funcționalități pentru adăugarea, editarea și ștergerea clienților și produselor, precum și crearea de comenzi pentru produse.Metoda main(String[] args) reprezintă punctul de intrare principal al aplicației și creează o instanță a clasei GUIInterface. De asemenea, se obțin și se afișează clienții și produsele din baza de date prin apelarea metodelor getAllClients() și getAllProducts() din clasele ClientDAO și ProductDAO.

# Rezultate

A screenshot of a computer

Description automatically generated

Cand deschidem interfata prima data se vor afisa in cele doua tabele ce avem in tabelul Client din MySQL (cel mare) si in tablul Product(cel mic).

Atunci cand adaugam, editam sau stergem un client, respective un produs, si tabelul din baza de date se actualizeaza.

Pentru a crea o comanda introducem in casetele din partea de jos a interfetei ce dorim pentru acea comanda si apasam butonul Create Order. La apasarea acestui buton se va deschide o noua fereastra in care se vor afisa comenzile deja existente in baza de date, dar si noua comanda introdusa de noi, tot atunci actualizandu-se si baza de date pentru comenzi.

# Concluzii

Realizând proiectul "Sistem de management al cozilor", am dobândit cunoștințe și abilități valoroase pe care le pot aplica în viitoarele proiecte. Printre acestea se numără:

* utilizarea bazelor de date relaționale: Am învățat să utilizez eficient bazele de date pentru stocarea și gestionarea diferitelor obiecte, cum ar fi clienții, produsele și comenzile. Aceasta facilitează accesul și manipularea datelor într-un mod structurat și eficient.
* arhitectură stratificată (layered architecture pattern): Am înțeles importanța unei arhitecturi bine structurate și modularizate, care separă componentele sistemului în straturi diferite. Aceasta permite o dezvoltare mai ușoară, întreținere și extensibilitate a proiectului.
* utilizarea tehnicilor de reflexie: Am explorat tehnici de reflexie, care permit examinarea și manipularea structurii interne a codului în timpul execuției. Aceasta oferă flexibilitate și capacitatea de a face operațiuni dinamice asupra obiectelor și metodelor.

# Bibliografie

* [*https://stackoverflow.com*](https://stackoverflow.com)
* *https://dev.java/learn/records/*
* [*https://app.diagrams.net/*](https://app.diagrams.net/)