



DOCUMENTATIE TEMA 3

ORDERS MANAGEMENT

Sarkozi Stefan

Grupa: 30223



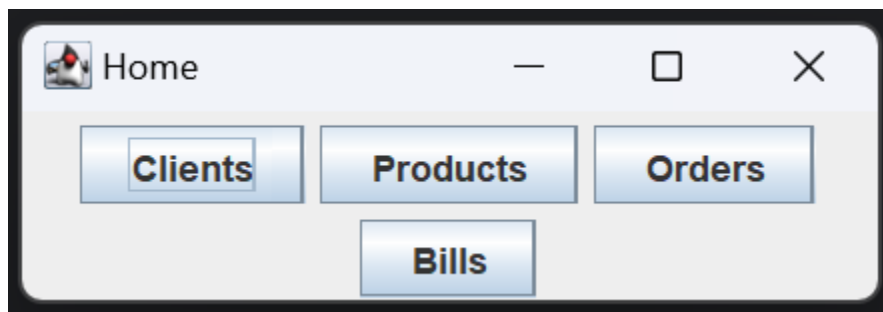
Cuprins

Cerinte Functionale	3
Obiective.....	4
Analiza Problemei	5
Proiectare:.....	9
Implementare.....	10
Concluzii si Dezvoltari Ulterioare.....	14
Bibliografie	14



Cerinte Functionale

Aplicatie pentru gestionarea si procesarea comenzilor clientilor intr-un depozit. Baza de date relationala folosita pentru a stoca clienti, produse si comenzi.





Obiective

Obiectiv principal:

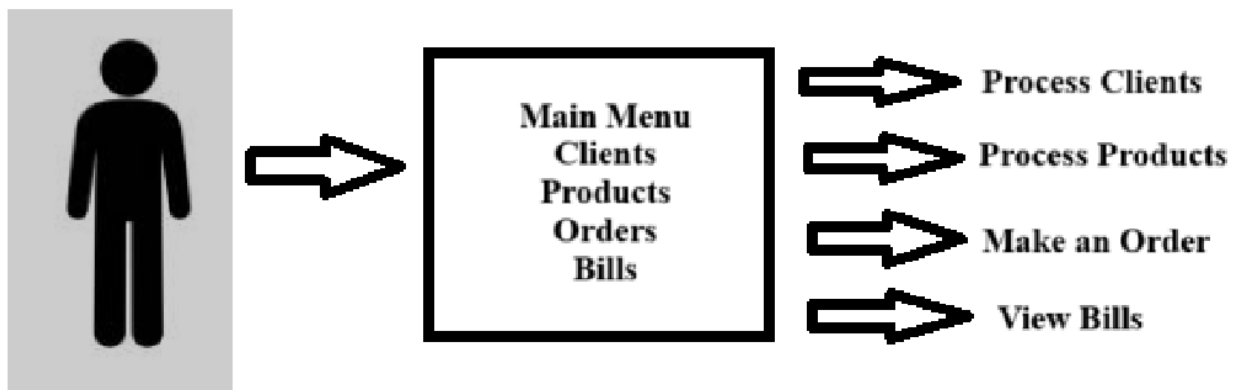
Realizarea unui program in ajutorul angajatilor de la un depozit care vor sa tina evidenta clientilor, produselor si a comenzilor printr-o baza de date care stocheaza aceste informatii, baza de date fiind accesata prin intermediul unor interfete simple si intuitive de folosit.

Obiective Secundare:

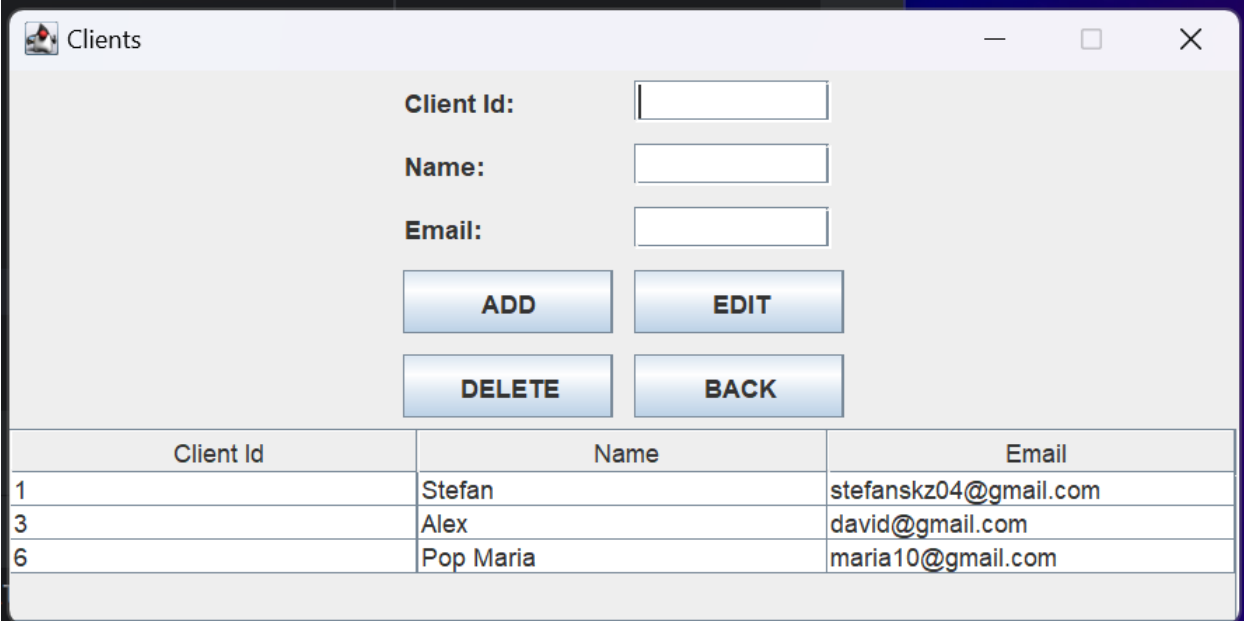
Aplicatie proiectata conform modelului de arhitectura stratificata, care permite realizarea unor modificari ulterioare asupra programului in functie de cerintele clientului.

Analiza Problemei

Use Case-uri:



Interfata Procesare Clienti:



Client Id	Name	Email
1	Stefan	stefanskz04@gmail.com
3	Alex	david@gmail.com
6	Pop Maria	maria10@gmail.com



Interfata realizeaza operatiile de adaugare, editare si stergere a unui client.

Interfata Procesare Produse:

The screenshot shows a window titled "Products" with the following elements:

- Input fields for: Product Id, Product Name, Price, and Quantity.
- Buttons: ADD, EDIT, DELETE, and BACK.
- A table with the following data:

Product Id	Product Name	Price	Quantity
1	Adidasi	150.5	2
2	Minge	99.9	1
4	Jucarie	29.5	1

Interfata realizeaza operatiile de adaugare, editare si stergere a unui produs.



Interfata Plasarii unei comenzi:

The screenshot shows a window titled "Orders" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains several input fields and buttons:

- Client Id:** A text input field followed by a blue button labeled "Search!".
- Client Email:** A text input field followed by a blue button labeled "Search!".
- A long, empty text input field.
- Product Id:** A text input field followed by a blue button labeled "Search".
- Product Name:** A text input field followed by a blue button labeled "Search!".
- A long, empty text input field.
- Quantity:** A text input field followed by a blue button labeled "Order!".
- A long, empty text input field.
- A blue button labeled "BACK" at the bottom center.

Se cauta un client ori dupa id ori dupa email, un produs dupa id sau dupa nume, se introduce cantitatea din produsul rtespectiv si se plaseaza comanda.



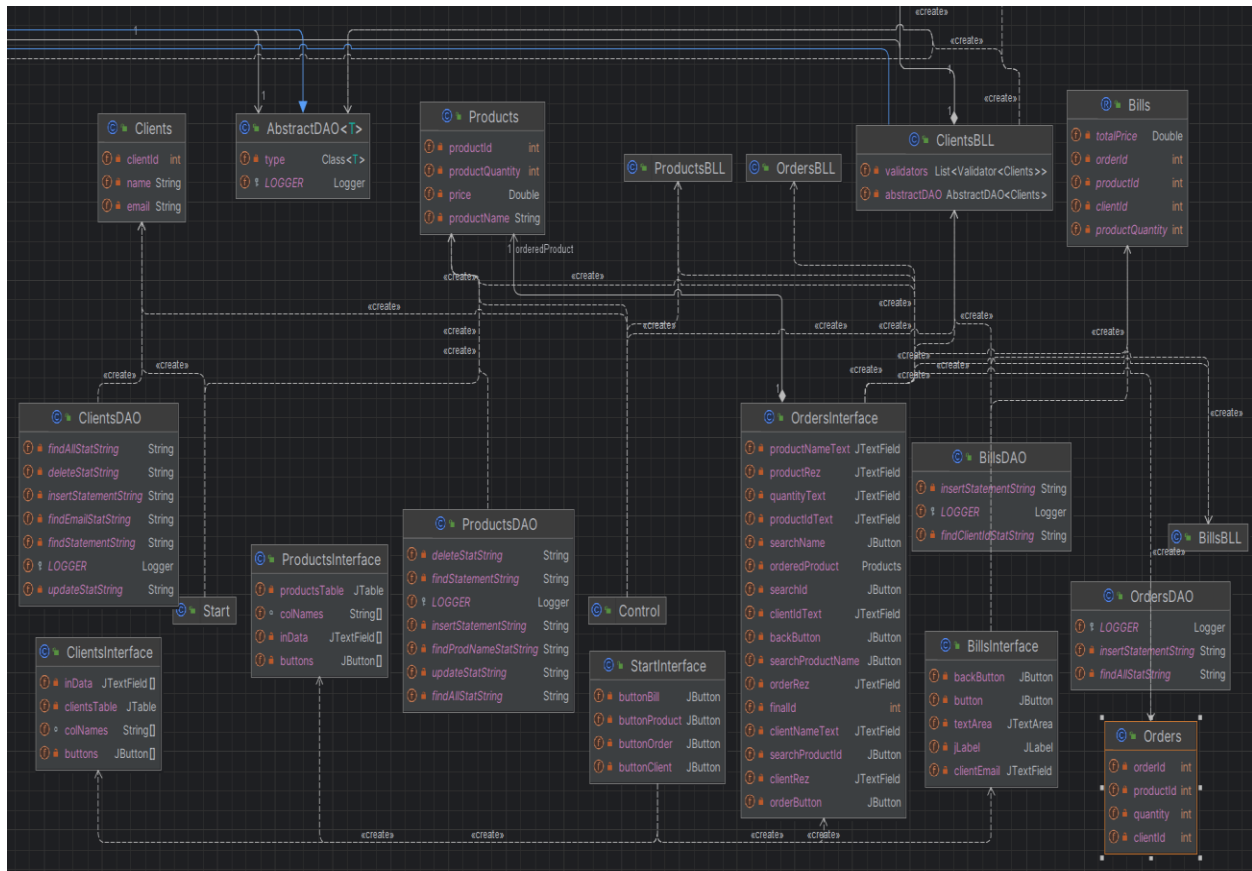
Interfata de Cautare Bonuri

The screenshot shows a web application window titled "Bills". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Below the title bar, there is a search bar with the label "Client Email:" and an input field. To the right of the input field are two buttons: "Search!" and "BACK". The main content area of the window is empty.

Se introduce un email si in functie de acest email se afiseaza toate bonurile clientului respectiv.



Proiectare:





Structuri de Date:

În general am folosit structuri de date precum List<>, în care am stocat obiecte de tipul Clients, Products sau Orders, care reprezintă o mască a fiecărui tabel din baza de date, am mai folosit o matrice de String-uri în care am extras toate informațiile dintr-un tabel, inclusiv numele coloanelor din acel tabel.

Implementare

```

public static String[][] retrieveProp(List<Object> objects) {
    String[][] outData = new String[objects.size() + 1][objects.get(0).getClass().getDeclaredFields().length];
    int k = 0, n = 0;
    for (Field fieldName : objects.get(0).getClass().getDeclaredFields()) {
        outData[0][k++] = fieldName.getName();
    }
    for (Object index : objects) {
        k = 0;
        n++;
        for (Field field : index.getClass().getDeclaredFields()) {
            field.setAccessible(true);
            Object values;
            try {
                values = field.get(index);
                outData[n][k++] = String.valueOf(values);
            } catch (IllegalArgumentException e) {
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                e.printStackTrace();
            }
        }
    }
    return outData;
}

```



Metoda `retrieveProp` furnizeaza toate informatiile unui tabel impreuna cu numele coloanelor, utilizand tehnici de reflexie.

```
2 usages  👤 Your Name
public void deleteById(int idToDelete) {
    PreparedStatement preparedStatement = null;
    Connection dbConnection = null;
    String toExecute = deleteString( toDelete: "id");
    try {
        dbConnection = ConnectionFactory.getConnection();
        preparedStatement = dbConnection.prepareStatement(toExecute);
        preparedStatement.setInt( parameterIndex: 1, idToDelete);
        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        LOGGER.log(Level.WARNING, msg: type.getName() + "DAO:absDelete " + e.getMessage());
    } finally {
        ConnectionFactory.close(preparedStatement);
        ConnectionFactory.close(dbConnection);
    }
}
```

Metoda abstracta `deleteById` din cadrul clasei `AbstractDAO`, are rolul de a sterge o tupla din orice tabel.

Metoda care furnizeaza query-ul de executat in cazul de delete este:



```
private String deleteString(String toDelete) {
    StringBuilder string = new StringBuilder();
    string.append("DELETE ");
    string.append("FROM ");
    string.append(type.getSimpleName());
    string.append(" WHERE ").append(toDelete).append(" = ?");
    return string.toString();
}
```

```
2 usages  Your Name
public void absUpdate(T toUpdate) {
    List<Object> objectList = new ArrayList<>();
    objectList.add(toUpdate);
    String[][] updateData = ReflexiveSelection.retrieveProp(objectList);
    List<String> strings = new ArrayList<>();
    int n = type.getDeclaredFields().length;
    for (int i = 0; i < n; i++) {
        strings.add(updateData[0][i]);
    }
    String toExecute = updateString(strings);
    Connection dbConnection = ConnectionFactory.getConnection();
    PreparedStatement preparedStatement = null;
    try {
        preparedStatement = dbConnection.prepareStatement(toExecute);
        for (int i = 1; i < n; i++) {
            preparedStatement.setString(i, updateData[1][i]);
        }
        preparedStatement.setInt(n, Integer.parseInt(updateData[1][0]));
        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        LOGGER.log(Level.WARNING, msg: type.getName() + "DAO:absDelete " + e.getMessage());
    } finally {
        ConnectionFactory.close(preparedStatement);
        ConnectionFactory.close(dbConnection);
    }
}
```

Metoda absUpdate din cadrul clasei AbstractDAO se ocupa cu editarea unei tuple din orice tabel.



Query-ul executat de metoda `absUpdate` este furnizat de:

```
private String updateString(List<String> toUpdate) {
    StringBuilder string = new StringBuilder();
    string.append("UPDATE ");
    string.append(type.getSimpleName());
    string.append(" SET ");
    boolean ok = false, next = false;
    for (String index : toUpdate) {
        if (ok) {
            if (next)
                string.append(", ");
            string.append(index).append(" = ?");
            next = true;
        }
        ok = true;
    }
    string.append(" WHERE ");
    string.append(toUpdate.get(0));
    string.append(" = ?");
    return string.toString();
}
```



Concluzii si Dezvoltari Ulterioare

In urma realizarii acestui proiect am invatat structura stratificata in cadrul gestionarii bazelor de date, precum si metodele abstracte care sunt universale si pot sa intervina asupra tuturor tabelelor, logica care sta la baza Hibernate-ului. Ca si dezvoltari ulterioare am putea implementa o logica care calculeaza profitul pe fiecare luna al depozitului, precum si cati clienti noi au facut comenzi intr-o luna.

Bibliografie

https://gitlab.com/utcn_dsrl/pt-layered-architecture

https://gitlab.com/utcn_dsrl/pt-reflection-example

<https://www.baeldung.com/javadoc>

<https://jenkov.com/tutorials/java-reflection/index.html>

<https://stackoverflow.com/>