

DOCUMENTATIE

ORDER MANAGEMENT

STUDENT : TERC RAZVAN DAN

GRUPA : 30221

AN UNIVERSITAR : 2020 – 2021

CUPRINS

1. OBIECTIVUL TEMEI
2. ANALIZA PROBLEMEI , MODELARE, SCENARII , CAZURI DE UTILIZARE
3. PROIECTARE (decizii de proiectare , diagrame UML , structura de date , proiectare clase , interfete , relatii , packages , algoritmi , interfata utilizator)
4. IMPLEMENTARE
5. REZULTATE
6. CONCLUZII
7. BIBLIOGRAFIE

1. OBIECTIVUL TEMEI

Obiectivul acestei teme a fost de a realiza o aplicatie Order Management pentru procesarea comenzilor clientilor pentru un depozit . Pentru a crea aceasta aplicatie, a trebuit sa implementam o interfata grafica in care sa putem introduce noi clienti , noi produse, sa modificam clientii sau produsele , dar si sa le stergem . De asemenea , interfata grafica a trebuit sa aiba si optiunea de cumparaturi , in care un client va putea cumpara un anumit produs , intr-o anumita cantitate . Pentru a putea stoca produsele , clientii si comenzile , a trebuit sa folosim bazele de date . Asadar , am folosit MySQL pentru a crea o baza de date cu tabele pentru client, produs si comanda . In aceste 3 tabele vor fi introduse valori , le vom putea actualiza , dar le vom putea si sterge . Pentru a fi functionala aplicatia , a fost nevoie de a conecta baza de date creata la codul java din IntelliJ .

2. ANALIZA PROBLEMEI , MODELARE , SCENARII , CAZURI DE UTILIZARE

Pentru a putea realiza aceasta tema si a o modela cat mai bine , am folosit structura din exemplul alaturat temei , astfel incat am impartit codul in 6 pachete , in care am adaugat clase specifice pachetului respectiv . Pentru a functiona aceasta aplicatie , am creat o interfata grafica pentru utilizatori , in care , in prima fereasta , putem alege intre a ne loga ca si admin si a face modificari asupra bazei de date , sau putem alege sa facem cumparaturi . Daca vom alege sa ne logam ca si admin , vom putea adauga clienti la baza de date , vom putea sterge anumiti clienti din baza de date , vom putea edita anumite detalii legate de un anumit client din baza de date , dar vom putea si sa listam clientii din baza de date actuala . De asemenea , in ceea ce priveste produsele , optiunile pe care le putem face pentru ele sunt adaugare de produs nou in baza de date, modificarea numelui produsului , a pretului sau a cantitatii , dar vom putea si sa stergem un anumit produs din baza noastra de date . O alta optiune pe care o putem realiza este de a afisa lista de produse care se afla la momentul actual in baza noastra de date din SQL .

3. PROIECTARE (decizii de proiectare , diagrame UML , structura de date , proiectare de clase , interfete , relatii , packages , algoritmi , interfata utilizator)

Pentru realizarea acestei temei , am impartit codul in 6 pachete , care contin fiecare mai multe clase , cu scopul de a structura cat mai bine si mai usor de inteles codul acestui proiect .

Cele 6 pachete in care am impartit codul acestei aplicatii sunt :

- bll – pachetul care contine clasele care incapsuleaza logica aplicatiei
Clasele din acest pachet sunt :
 - CustomerBLL – clasa cu ajutorul careia am implementat metoda “ findCustomerById ” prin care vom cauta clientii dupa id – ul lor , metoda “ insertCustomer ” care va adauga clientii in baza de date , metoda “

deleteCustomer “ care are rolul de a sterge un anumit client din baza de date , metoda “ updateCustomer “ care ne va ajuta sa facem anumite modificati pentru client in baza de date , dar si metoda “ selectAllCustomers “ , cu ajutorul careia vom putea lista toti clientii din baza noastra de date din SQL .

- OrdersBLL – aceasta clasa am folosit-o pentru a crea urmatoarele metode :
“ findProductById ” cu ajutorul careia vom putea gasi un anumit produs ,
dupa id – ul sau unic , dar si metoda “ insertOrder ” pentru a putea insera
noi comenzi in baza de date .
- ProductBLL – in interiorul acestei clase am creat metodele „
findProductByName ” cu ajutorul careia vom gasi produsele pe care le
cautam , „ insertProduct ” pentru a insera noi produse in baza de date , „
deleteProduct ” cu scopul e a sterge un anumit produs din baza noastra de
date , dar si metoda „ updateProduct ” pentru a modifica anumite detalii
ale unui produs din baza de date.
- Validators – pachet cu ajutorul caruia vom valida anumite field-uri
Clasele din interiorul acestui pachet sunt :
 - Interfata Validator – in care am definit metoda validate , care este o
interfata generica
 - QuantityValidator – clasa care mosteneste interfata Validator , si
implementeaza metoda „ validate ” in care verificam daca cantitatea
produsului este una valida .
- Connection
 - ConnectionFactory – clasa cu ajutorul careia ne vom conecta la baza
de date din MySQL
- dataAcces – acest pachet contine clasele care au in interiorul lor interogari si conexiunea
la baza de date
 - AbstractDAO – clasa care ilustreaza „ reflection techniques „
 - CustomerDAO – in aceasta clasa am creat interogari pentru a insera
clienti in baza de date , pentru a afisa clientii din baza de date , pentru
a sterge anumiti clienti , dar si pentru a face modificari asupra
detaliilor unui anumit client. De asemenea exista si metoda „ findById
” cu ajutorul careia vom gasi un anumit client. Metoda „ delete ” din

aceasta clasa are rolul de a sterge un client , metoda „ insert ” are rolul de a adauga un nou client , metoda „ update ” am creat-o cu scopul de a actualiza un anumit client , cum ar fi sa modific numele acestuia , adresa sau numarul de telefon , iar cu ajutorul metodei „ select ” voi afisa toti clientii din baza de date din MySQL .

- OrdersDAO – in aceasta clasa am creat interogarile pentru a putea insera comenzi noi in baza de date , dar si pentru a putea selecta datele din tabela Order . In aceasta clasa avem metodele : „ findById ” , dar si metoda „ insert ” , care va insera noi comenzi in baza de date sub forma (idOrder , quantity , customer_idCustomer , product_idProduct) .
- ProductDAO – in interiorul acestei clase am realizat interogari pentru a insera un produs nou in baza de date , pentru a afisa toate produsele din baza de date , pentru a sterge un produs , dar si pentru a actualiza informatiile referitoare la un anumit produs din baza noastra de date din MySQL . De asemenea , am implementat metodele „ findByName ” pentru a gasi un anumit produs , „ insert ” pentru a insera un anumit produs , „ delete ” care va sterge un produs din baza de date , „ update ” care va actualiza informatiile unui produs , dar si metoda „ select ” care va afisa un anumit produs din baza de date .

- model

C clasele reprezentative din acest pachet sunt urmatoarele :

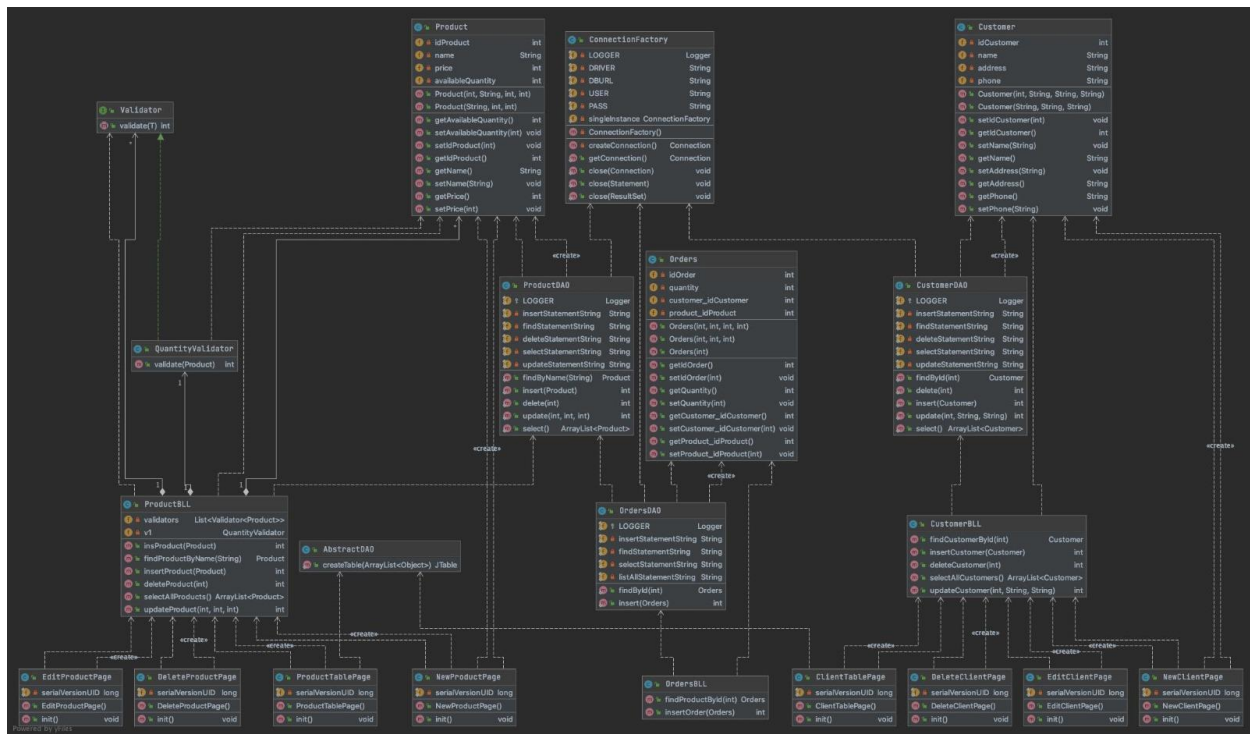
- Customer – in aceasta clasa am definit ca si field-uri caracteristicile unui client, cum ar fi : „ idCustomer ” , „ name ” , „ address ” , dar si „ phone ” . Am creat 2 constructori si am generat setter si getter pentru fiecare field .
- Orders – aceasta clasa contine field-uri reprezentative pentru o comanda din stoc , cum ar fi : „ idOrder ” , „ quantity ” , „ customer_idCustomer ” , „ product_idProduct ” . De asemenea, mai gasim si 2 constructori pentru aceasta clasa, dar si metodele generate de setter si getter .
- Product – in aceasta clasa am definit ca si field-uri urmatoarele : „ idProduct ” , „ name ” , „ price ” , dar si „ availableQuantity ” . Am creat 2 constructori , dar si mai multe metode pentru setter si getter .

- presentation

Clasele din acest pachet sunt urmatoarele :

- AdminOptPage – care reprezinta pagina adminului , de unde poate realiza modificari pentru a adauga un client nou , a sterge un client existent , a actualiza informatii despre un client , dar si pentru a lista toti clientii din baza de date . De asemenea, poate realiza aceste optiuni si asupra produselor din baza de date , cum ar fi : sa introduca un produs nou , sa stearga un produs , sa editeze detaliile unui produs , dar si sa listeze produsele existente . La finalul acestei pagini exista un buton „ Back ” cu ajutorul caruia putem reveni la pagina anterioara .
- AdminPage – aceasta este pagina care se va deschide atunci cand alegem „ connect as admin ” , unde va trebui sa introducem numele de utilizator si parola pentru a ne putea conecta cu succes ca si administrator .
- ClientTablePage – pagina in care vom putea afisa toti clientii existenti in momentul de fata in baza noastra de date din MySQL
- DeleteClientPage – aceasta pagina ne apare atunci cand accesam optiunea de a sterge un client, in care va trebui sa introducem id-ul clientului care dorim sa fie sters din baza de date .
- DeleteProductPage - aceasta pagina ne apare atunci cand accesam optiunea de a sterge un produs, in care va trebui sa introducem id-ul produsului care dorim sa fie sters din baza de date .
- EditClientPage – aceasta pagina ne apare atunci cand accesam optiunea de a actualiza informatiile despre un client, in care va trebui sa introducem id-ul clientului si noua adresa sau noul numar de telefon al clientului .
- EditProductPage – aceasta pagina o putem vizualiza cand accesam optiunea de a actualiza informatiile despre un produs , in care va trebui sa adaugam noile actualizari
- FirstPage – aceasta este pagina principala a aplicatiei, unde va trebui sa alegem daca dorim sa ne autentificam ca si admin sau dorim sa facem cumparaturi
- NewClientPage – pentru a adauga un client nou
- NewProductPage – pentru a adauga un produs nou
- PlaceOrder – pagina cu ajutorul careia vom putea plasa o comanda. Pentru plasarea comenzii va trebui sa introducem id-ul clientului, produsul dorit , dar si cantitatea produsului
- ProductTablePage – pagina in care vom afisa lista tuturor produselor din baza de date din MySQL

Diagrama UML :



4. IMPLEMENTARE

- BLL

➤ CustomerBLL :

```

public class CustomerBLL {

    public Customer findCustomerById(int idCustomer) {
        Customer c = CustomerDAO.findById(idCustomer);
        if (c == null) {
            throw new NoSuchElementException("The customer with id =" + idCustomer + " was not found!");
        }
        return c;
    }

    public int insertCustomer(Customer c) { return CustomerDAO.insert(c); }

    public int deleteCustomer(int id) { return CustomerDAO.delete(id); }

    public ArrayList<Customer> selectAllCustomers() { return CustomerDAO.select(); }

    public int updateCustomer(int id, String address, String phone) { return CustomerDAO.update(id, address, phone); }

}
  
```

➤ OrdersBLL

```
package bll;

import java.util.NoSuchElementException;

import dataAcces.OrdersDAO;
import model.Orders;

public class OrdersBLL {

    public Orders findProductById(int idOrder) {
        Orders o = OrdersDAO.findById(idOrder);
        if (o == null) {
            throw new NoSuchElementException("The order with id =" + idOrder + " was not found!");
        }
        return o;
    }

    public int insertOrder(Orders o) { return OrdersDAO.insert(o); }
}
```

➤ ProductBLL

```
public class ProductBLL {

    private List<Validator<Product>> validators;
    private QuantityValidator v1;

    public int insProduct(Product product) {
        for(Validator<Product> p : validators){
            if(p.validate(product) < 0)
                return -1;
        }
        return 0;
    }

    public Product findProductByName(String name) {
        Product p = ProductDAO.findByName(name);
        if (p == null) {
            throw new NoSuchElementException("The product: " + name + " was not found!");
        }
        return p;
    }
}
```



```

    public Product findProductByName(String name) {
        Product p = ProductDAO.findByName(name);
        if (p == null) {
            throw new NoSuchElementException("The product: " + name + " was not found!");
        }
        return p;
    }

    public int insertProduct(Product p) { return ProductDAO.insert(p); }

    public int deleteProduct(int id) { return ProductDAO.delete(id); }

    public ArrayList<Product> selectAllProducts() { return ProductDAO.select(); }

    public int updateProduct(int id, int price, int availableQuantity){
        return ProductDAO.update(id, price, availableQuantity);
    }
}

```

- dataAccess
 - AbstractDAO

```

public class AbstractDAO {

    public static JTable createTable(ArrayList<Object> objects){
        Object o= objects.get(0);
        Object[] headers= new Object[o.getClass().getDeclaredFields().length];
        int i =0;
        for (Field field : o.getClass().getDeclaredFields()) {
            field.setAccessible(true);
            try {
                headers[i]=field.getName();
                i++;
            } catch (IllegalArgumentException e) {
                e.printStackTrace();
            }
        }

        Object[][] data= new Object[objects.size()][headers.length];

        for(int j=0;j<objects.size();j++){

```

```

Object[][] data= new Object[objects.size()][headers.length];

for(int j=0;j<objects.size();j++){
    int k=0;
    for (Field field : objects.get(j).getClass().getDeclaredFields()) {
        field.setAccessible(true);
        try {
            data[j][k] = field.get(objects.get(j));
            k++;
        } catch (IllegalArgumentException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
    }
}

```

➤ CustomerDAO

```

public class CustomerDAO {

    protected static final Logger LOGGER = Logger.getLogger(CustomerDAO.class.getName());
    private static final String insertStatementString = "INSERT INTO customer (name,phone,address)"
        + " VALUES (?, ?, ?)";
    private final static String findStatementString = "SELECT * FROM customer WHERE idCustomer = ?";
    private final static String deleteStatementString = "DELETE FROM customer WHERE idCustomer = ?";
    private final static String selectStatementString = "SELECT * FROM customer";
    private final static String updateStatementString = "UPDATE customer SET address=?,phone=? WHERE idCustomer=?";

    public static Customer findById(int customerId){
        Customer toReturn = null;

        Connection dbConnection = ConnectionFactory.getConnection();
        PreparedStatement findStatement = null;
        ResultSet rs = null;
        try {
            findStatement = dbConnection.prepareStatement(findStatementString);
            findStatement.setLong( parameterIndex: 1, customerId);
            rs = findStatement.executeQuery();

```

➤ OrdersDAO

```

public class OrdersDAO {

    protected static final Logger LOGGER = Logger.getLogger(ProductDAO.class.getName());
    private static final String insertStatementString = "INSERT INTO orders (quantity,customer_idCustomer,product_idProduct)"
        + " VALUES (?, ?, ?)";
    private final static String findStatementString = "SELECT * FROM orders WHERE idOrder = ?";
    private final static String selectStatementString = "SELECT * FROM orders";
    private final static String listAllStatementString = "SELECT idOrder,customer.name, product.name, product.price";

    public static Orders findById(int orderId){
        Orders toReturn= null;

        Connection dbConnection = ConnectionFactory.getConnection();
        PreparedStatement findStatement = null;
        ResultSet rs = null;

        try {
            findStatement = dbConnection.prepareStatement(findStatementString);

```

➤ ProductDAO

```

public class ProductDAO {
    protected static final Logger LOGGER = Logger.getLogger(ProductDAO.class.getName());
    private static final String insertStatementString = "INSERT INTO product (name,price,availableQuantity)"
        + "VALUES (?, ?, ?)";
    private final static String findStatementString = "SELECT * FROM product where name = ?";
    private final static String deleteStatementString = "DELETE FROM product WHERE idProduct = ?";
    private final static String selectStatementString = "SELECT * FROM product";
    private final static String updateStatementString = "UPDATE product SET price=?,availableQuantity=? WHERE id

    public static Product findByName(String productName){
        Product toReturn= null;

        Connection dbConnection = ConnectionFactory.getConnection();
        PreparedStatement findStatement = null;
        ResultSet rs = null;

        try {
            findStatement = dbConnection.prepareStatement(findStatementString);
            findStatement.setString( parameterIndex: 1, productName);

```

5. REZULTATE

La sfarsitul acestei teme , am reusit sa obtin o interfata grafica care se va sincroniza cu baza mea de date din MySQL . Asadar , atunci cand voi genera interfata grafica , imi va apareea pagina principala :



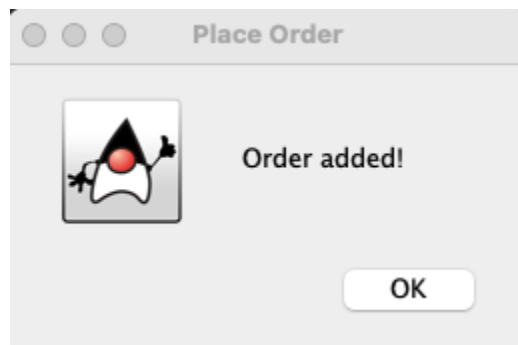
Aici voi putea alege daca doresc sa fac o comanda sau sa ma conectez ca si admin , pentru a avea posibilitatea de a realiza modificari asupra bazei de date din MySQL

Daca voi alege sa fac o comanda , se va afisa urmatoarea pagina :

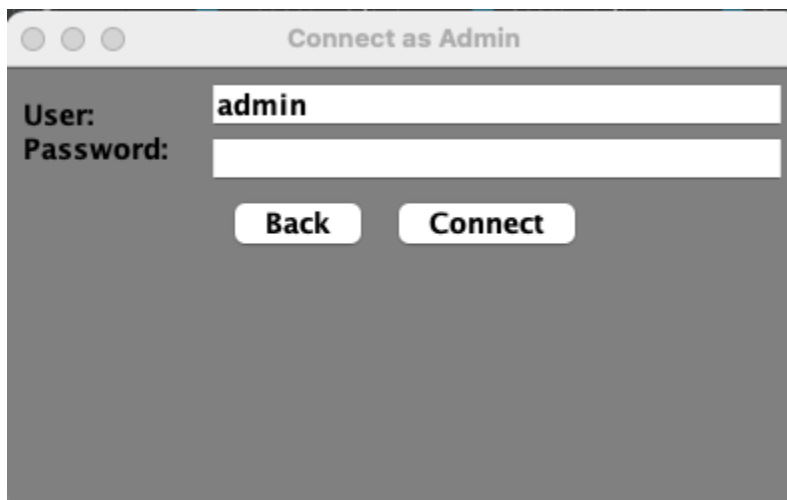


A screenshot of a web application window titled "Place Order". The window has a light gray background and standard macOS window controls (red, yellow, green buttons) in the top-left corner. The form contains three labels on the left: "Customer ID:", "Product", and "Quantity". To the right of "Customer ID:" is a dropdown menu showing the value "4". To the right of "Product" is a dropdown menu showing the value "laptop". To the right of "Quantity" is a text input field containing the value "2". At the bottom of the form are two buttons: "Back" and "Place Order".

Aici va trebui sa aleg ID – ul clientului, produsul pe care doreste sa il cumpere , dar si in ce cantitate doreste sa achizitioneze produsul respectiv . La final, pentru a fi preluata comanda cumparatorul va trebui sa apese „ Place Order ” si se va afisa urmatorul mesaj , care sugereaza faptul ca comanda s-a finalizat cu succes :

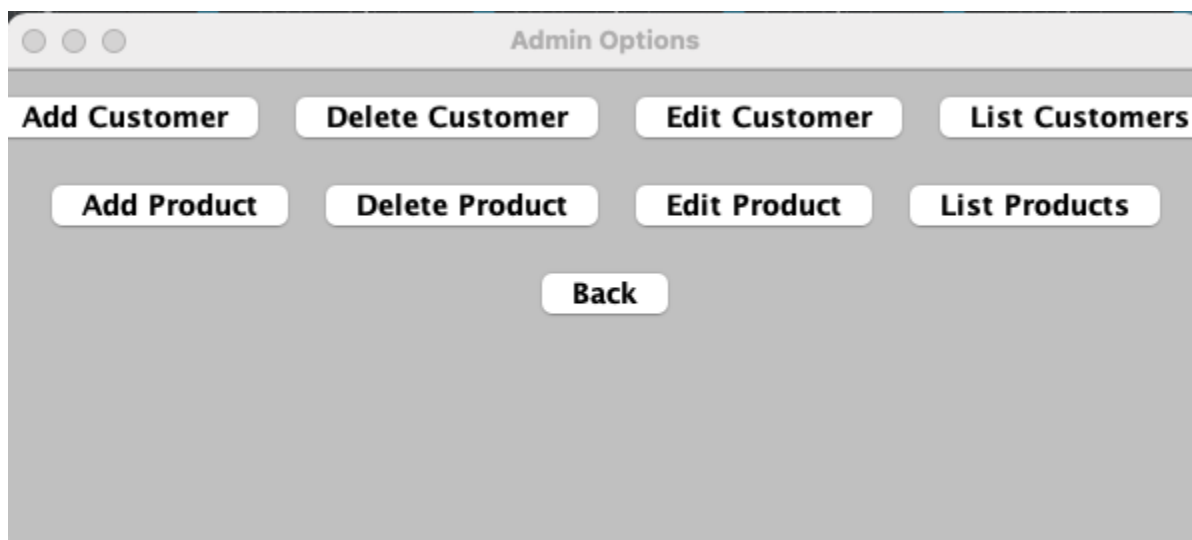


Daca vom alege sa ne conectam ca si administrator , ne va apareea pagina in care va trebui sa ne autentificam cu username -ul si parola noastra :



A screenshot of a web application window titled "Connect as Admin". The window has a light gray background and standard macOS window controls. The form contains two labels on the left: "User:" and "Password:". To the right of "User:" is a text input field containing the value "admin". To the right of "Password:" is an empty text input field. At the bottom of the form are two buttons: "Back" and "Connect".

Dupa ce am inserat username – ul si parola , ne va aparea pagina principala a administratorului



De aici, vom putea alege oricare din optiunile afisate , pentru a face modificari asupra bazei de date. Pagina urmatoare va adauga un nou client :

A screenshot of a web application window titled "New Customer". The window has a light gray header bar with the title. Below the header, there are three input fields. The first field is labeled "Name:" and contains the text "Andrei". The second field is labeled "Address:" and contains the text "Cluj-Napoca". The third field is labeled "Phone number:" and contains the text "0744567543". Below the input fields, there are three buttons: "Back", "Reset", and "Add Customer". The background of the window is a solid light gray.

Fereastra de mai jos va sterge clientul cu id-ul 3 din baza de date :

A screenshot of a web application window titled "Delete Customer". The window has a light gray header bar with the title. Below the header, there is a single input field labeled "Give the customer's ID:" which contains the number "3". Below the input field, there are two buttons: "Back" and "Delete Customer". The background of the window is a solid light gray.

De asemenea, daca accesam optiunea de a afisa lista tuturor clientilor din baza de date, vom avea urmatoarea pagina :



idCustomer	name	address	phone
1	Andrei	Alba Iulia	0745374628
2	Alina	Cluj	0745235356
3	Sergiu	Cluj	0745234543
4	Razvan	Targu Jiu	0755647384
9	Diana	Zalau	0756384735
10	Alexandra	Bacau	0756486583
11	Popescu Victor	Iasi	0785473658
12	Popescu Sonia	Alba Iulia	0785445658
15	Andreea	Braila	0783647563
16	Antonia	Brasov	0745374632
17	Marius	Galati	0746583658

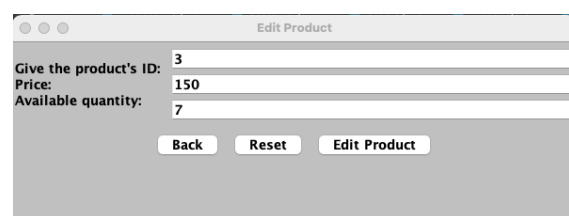


New Product

Name:

Price:

Available quantity:



Edit Product

Give the product's ID:

Price:

Available quantity:

Cu ajutorul ferestrelor de mai sus, vom putea adauga un nou produs sau sa editam detaliile unui produs deja existent in baza de date .

Tabelele din MySQL arata in felul urmator :

Customer :

Product :

idCustomer	name	address	phone
3	Sergiu	Cluj	0745234543
4	Razvan	Targu Jiu	0755647384
9	Diana	Zalau	0756384735
10	Alexandra	Bacau	0756486583
11	Popescu Victor	Iasi	0785473658
12	Popescu Sonia	Alba Iulia	0785445658
15	Andreea	Braila	0783647563
16	Antonia	Brasov	0745374632
17	Marius	Galati	0746583658
NULL	NULL	NULL	NULL

idProduct	name	price	availableQuantity
3	banane	7	12
4	laptop	30000	11
6	penar	10	9
7	caiet	7	50
9	mouse	25	14
10	mere	3	50
11	pere	12	40
12	ghiozdan	60	14
13	microfon	35	12
NULL	NULL	NULL	NULL

Order :

idOrder	quantity	customer_idCustom...	product_idPi
1	5	1	1
2	15	2	3
3	20	3	2
4	5	3	3
5	5	1	3
6	10	3	1
7	15	4	4
► 8	14	4	4
9	5	3	4
10	5	3	3
11	3	1	1
12	5	4	3
13	15	3	6
14	9	11	6
15	1	3	3
16	3	3	3
17	3	2	3
18	2	4	4
NULL	NULL	NULL	NULL

6. CONCLUZII

In concluzie, realizarea acestei teme a fost una benefica pentru mine, deoarece am reusit sa invat destul de multe lucruri in ceea ce privesc interfetele grafice , dar si bazele de date MySQL . Pe viitor , consider ca aceasta aplicatie s-ar putea imbunatati daca am mai adauga cateva functionalitati . O parte din aceste functionalitati ar putea fi : o pagina in care si utilizatorul (clientul) sa se autentifice atunci cand doreste sa plaseze o comanda , pentru a ramane datele lui si istoricul comenzii in contul sau propriu, optiunea de a alege daca doreste sa achite cu cardul sau cash , dar si optiunea de a alege prin ce metoda doreste sa ii fie livrate produsele (curier, posta etc).

7. BIBLIOGRAFIE

Ca si surse de inspiratie am folosit :

<https://stackoverflow.com/>