**PROIECT 3**

**Interfata Web pentru o baza de date folosind tehnologia**

**Java Server Faces.**

Student: Dumitru Teodor-Mihai

**Cuprins**

3 … 1. Breviar Teoretic

6 … 2. Implementare cod

6 … 2.1 MySQL

9 … 2.2 Java

15 … 2.3 XHTML

17 … 2.4 XML

18 … 3. Interfata Web

21 … 4. Bibliografie

1. **Breviar Teoretic**

**Java Server Faces**

multe scheme API disponibile pentru MySQL ce permit scrierea aplicațiilor în numeroase limbaje de programare pentru accesarea bazelor de date MySQL, cum ar fi: C, C++, C#, Java, [Perl](https://ro.wikipedia.org/wiki/Perl), [PHP](https://ro.wikipedia.org/wiki/PHP), [Python](https://ro.wikipedia.org/wiki/Python" \o "Python), JavaServer Faces (JSF) este o specificație [Java](https://ro.wikipedia.org/wiki/Java_(limbaj_de_programare)) pentru construirea de interfețe utilizator bazate pe componente pentru aplicații web și a fost formalizată ca standard prin intermediul [Java Community Process](https://en.wikipedia.org/wiki/Java_Community_Process), care face parte din Platforma Java Enterprise Edition. Este, de asemenea, un cadru web MVC care simplifică construirea de interfețe utilizator (UI) pentru aplicațiile bazate pe server, utilizând componente UI reutilizabile într-o pagină. [1]

JSF 2 folosește Facelets ca sistem default de template-uri. Pot fi utilizate și alte tehnologii de vizualizare, cum ar fi [XUL](https://en.wikipedia.org/wiki/XUL) sau simplu, Java. În schimb, JSF 1.x utilizează JavaServer Pages(JSP) ca sistem default de template-uri. [1]

Un set de API-uri pentru reprezentarea componentelor UI și gestionarea stării acestora, gestionarea evenimentelor și validarea intrărilor, definirea navigării pe pagini și susținerea internaționalizării și accesibilității. [2]

O bibliotecă de etichete personalizate Pagini JavaServer (JSP) pentru exprimarea unei interfețe JavaServer Faces în cadrul unei pagini JSP. [2]

Proiectat pentru a fi flexibil, tehnologia JavaServer Faces utilizează noțiunile de interfață standard și interfețe standard, fără a limita dezvoltatorii la un anumit limbaj, protocol sau dispozitiv client. Clasele componente ale UI incluse în tehnologia JavaServer Faces încorporează funcționalitatea componentei, nu prezentarea specifică clientului, permițând astfel ca componentele UI JavaServer Faces să fie redate la diverse dispozitive client. Prin combinarea funcționalității componentei UI cu randările personalizate, care definesc atributele de randare pentru o componentă UI specifică, dezvoltatorii pot construi etichete personalizate pentru un anumit dispozitiv client. Tehnologia JavaServer Faces oferă un renderer personalizat și o bibliotecă de etichete personalizate JSP pentru redarea la un client HTML, permițând dezvoltatorilor aplicațiilor Java Platform, Enterprise Edition (Java EE) să utilizeze tehnologia JavaServer Faces în aplicațiile lor. [2]

Ușor de utilizat fiind obiectivul principal, arhitectura JavaServer Faces definește clar o separare între logica aplicației și prezentare, facilitând în același timp conectarea stratului de prezentare la codul aplicației. Acest proiect permite fiecărui membru al unei echipe de dezvoltare a aplicațiilor web să se concentreze asupra procesului de dezvoltare a acestuia și oferă, de asemenea, un model simplu de programare care leagă piesele împreună. De exemplu, dezvoltatorii de pagini web fără expertiză în programare pot utiliza etichete de componente UI ale JavaServer Faces pentru a conecta codul aplicației dintr-o pagină web fără a scrie niciun script. [2]

Dezvoltat prin Procesul comunitar Java în conformitate cu JSR - 314, tehnologia JavaServer Faces stabilește standardul pentru crearea interfețelor de utilizator de pe server. Cu ajutorul contribuțiilor grupului de experți, API-urile JavaServer Faces sunt proiectate astfel încât să poată fi utilizate cu ajutorul unor instrumente care vor face mai ușoară dezvoltarea aplicațiilor web. Mai mulți furnizori de instrumente respectate au fost membri ai grupului de experți JSR-314, care a dezvoltat specificația JavaServer Faces 1.0. Acești furnizori se angajează să sprijine tehnologia JavaServer Faces în instrumentele lor, promovând astfel adoptarea standardului tehnologiei JavaServer Faces. [2]

**MySQL**

MySQL este un [sistem de gestiune a bazelor de date](https://ro.wikipedia.org/wiki/Sistem_de_gestiune_a_bazelor_de_date) [relaționale](https://ro.wikipedia.org/w/index.php?title=Baze_de_date_rela%C8%9Bionale&action=edit&redlink=1), produs de compania suedeza [MySQL AB](https://ro.wikipedia.org/wiki/MySQL_AB" \o "MySQL AB) și distribuit sub [Licența Publică Generală GNU](https://ro.wikipedia.org/wiki/Licen%C8%9Ba_Public%C4%83_General%C4%83_GNU). Este cel mai popular [SGBD](https://ro.wikipedia.org/wiki/SGBD) [open-source](https://ro.wikipedia.org/wiki/Open-source) la ora actuală, fiind o componentă cheie a stivei [LAMP](https://ro.wikipedia.org/wiki/LAMP) ([Linux](https://ro.wikipedia.org/wiki/Linux), [Apache](https://ro.wikipedia.org/wiki/Apache), MySQL, [PHP](https://ro.wikipedia.org/wiki/PHP)). [3]

Deși este folosit foarte des împreună cu [limbajul de programare](https://ro.wikipedia.org/wiki/Limbaj_de_programare) [PHP](https://ro.wikipedia.org/wiki/PHP), cu MySQL se pot construi aplicații în orice limbaj major. Există FreeBasic, etc., fiecare dintre acestea folosind un tip specific API. O interfață de tip ODBC denumită MyODBC permite altor limbaje de programare ce folosesc această interfață, să interacționeze cu bazele de date MySQL cum ar fi [ASP](https://ro.wikipedia.org/wiki/ASP) sau [Visual Basic](https://ro.wikipedia.org/wiki/Visual_Basic). În sprijinul acestor limbaje de programare, unele companii produc componente de tip COM/COM+ sau .NET (pentru Windows) prin intermediul cărora respectivele limbaje să poată folosi acest [SGBD](https://ro.wikipedia.org/wiki/SGBD) mult mai ușor decât prin intermediul sistemului ODBC. Aceste componente pot fi gratuite (ca de exemplu MyVBQL) sau comerciale. [3]

[LAMP](https://ro.wikipedia.org/wiki/LAMP) cuprinde și MySQL

Licența GNU GPL nu permite încorporarea MySQL în softuri comerciale; cei care doresc să facă acest lucru pot achiziționa, contra cost, o licență comercială de la compania producătoare, MySQL AB. [3]

MySQL este componentă integrată a platformelor [LAMP](https://ro.wikipedia.org/wiki/LAMP) sau WAMP (Linux/Windows-Apache-MySQL-PHP/Perl/Python). Popularitatea sa ca aplicație web este strâns legată de cea a PHP-ului care este adesea combinat cu MySQL și denumit Duo-ul Dinamic. În multe cărți de specialitate este precizat faptul ca MySQL este mult mai ușor de învățat și folosit decât multe din aplicațiile de gestiune a bazelor de date, ca exemplu comanda de ieșire fiind una simplă și evidentă: „exit” sau „quit”. [3]

Pentru a administra bazele de date MySQL se poate folosi modul linie de comandă sau, prin descărcare de pe internet, o interfață grafică: [MySQL Administrator](https://ro.wikipedia.org/w/index.php?title=MySQL_Administrator&action=edit&redlink=1" \o "MySQL Administrator — pagină inexistentă) și [MySQL Query Browser](https://ro.wikipedia.org/w/index.php?title=MySQL_Query_Browser&action=edit&redlink=1" \o "MySQL Query Browser — pagină inexistentă). Un alt instrument de management al acestor baze de date este aplicația gratuită, scrisă în PHP, [phpMyAdmin](https://ro.wikipedia.org/wiki/PhpMyAdmin" \o "PhpMyAdmin). [3]

MySQL poate fi rulat pe multe dintre platformele software existente: AIX, FreeBSD, GNU/Linux, Mac OS X, [NetBSD](https://ro.wikipedia.org/wiki/NetBSD" \o "NetBSD), [Solaris](https://ro.wikipedia.org/wiki/Solaris), SunOS, Windows 9x/NT/2000/XP/Vista. [3]

**SHA-2**

**SHA-2** (**Secure Hash Algoritm 2**) este un set de [funcții hash criptografice](https://ro.wikipedia.org/wiki/Func%C8%9Bie_hash) conceput de [Agenția de Securitate Națională](https://ro.wikipedia.org/wiki/National_Security_Agency) (NSA) a Statelor Unite ale Americii. Aceasta are la bază structurile Merkle–Damgård, care la rândul ei este o funcție de compresie unidirecțională făcută cu structuri Davies–Meyer cu un [cifru pe blocuri](https://ro.wikipedia.org/wiki/Cifru_pe_blocuri) specializat. [4]

Funcțiile criptografice hash sunt [operații matematice](https://ro.wikipedia.org/wiki/Opera%C8%9Bii_matematice) care se aplică pe date digitale; prin compararea unui „hash” (valoarea finală a funcției) cu un alt hash cunoscut, se poate determina integritatea. De exemplu, calcularea hash-ului unui [fișier](https://ro.wikipedia.org/wiki/Fi%C8%99ier) descărcat și compararea acestuia cu un hash publicat anterior se poate determina dacă în procesul de descărcare, fișierul a fost modificat sau alterat.[[2]](https://ro.wikipedia.org/wiki/SHA-2#cite_note-2) Un aspect cheie a funcțiilor criptografice hash este rezistența la coliziune: nimeni nu ar trebui să fie poate să găsească două de valori de intrare diferite (de exemplu, două fișiere distincte) care ar duce la unul și același hash de ieșire. [4]

SHA-2 include schimbări semnificative de la predecesorul său, [SHA-1](https://ro.wikipedia.org/w/index.php?title=SHA-1&action=edit&redlink=1). SHA-2 familia este format din șase funcții hash cu digesturi (valori hash) care pot fi de 224, 256, 384 sau 512 biți: **SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256**.[4]

1. **Implementare si cod**

**2.1) MySQL**

Baza de date a fost facuta folosind MySQL.

create database proiect3; // creare baza de date numit proiect3

use proiect3;

Trebuie realizate 3 tabele Clienti, ProdusAlimentar si Producator, intre Clienti si ProdusAlimentar este relatie M:N, la fel este si intre ProdusAlimentar si Producatori, din acest motiv trebuie create inca 2 tabele intermediare care sa faca legatura de comunicare intre tabel, adica tabelele IntermediarCP si IntermediarPP. In total 5 tabele.

create table Clienti (

ClientID int not null auto\_increment primary key,

Nume varchar(45),

Prenume varchar(45),

Adresa varchar(45) ) ;

create table IntermediarCP (

IntermediarCP int not null auto\_increment primary key,

ClientID int not null,

ProdusID int not null );

create table ProdusAlimentar (

ProdusID int not null auto\_increment primary key,

Denumire varchar(45),

DataProducere datetime,

DataExpirare datetime );

create table IntermediarPP (

IntermediarPP int not null auto\_increment primary key,

ProdusID int not null,

ProducatorID int not null );

create table Producatori (

ProducatorID int not null auto\_increment primary key,

Denumire varchar(45),

TaraOrigine varchar(45),

Adresa varchar(45) );

Se creeaza tabela Users deoarece folosim Log-In in interfata noastra web, vom face username-ul primary key pentru ca il dorim sa fie unic. Iar parola (pass) o vom face de tipul binary(32) pentru ca o sa avem nevoie la hash. Vom acorda si prioritati user-ilor.

create table Users(

username varchar(50) not null primary key,

pass binary(32) not null,

priviledged bool default false );

Vom introduce in Users un user cu parola criptata folosind Secure Hash Algoritm.

Inroducem si un admin care are privilegii mai multe decat user-ul. Un admin are dreptul sa si stearga din tabela, in schimb un user poate doar sa adauge, sa vizualizeze sau sa modifice tabela.

insert into Users(username, pass) values('user', sha2('user', 256));

insert into Users(username, pass, priviledged) values('admin', sha2('admin', 256), true);

select pass from users where username = 'user';

alter table users modify pass char(64);

Alteram cele 2 tabele pentru ca asiguram legatura intre cele 5 tabele, pentru acest lucru vom folosi foreign key-uri.

alter table IntermediarCP

add constraint `fk1` foreign key (ClientID)

references Clienti(ClientID);

alter table IntermediarCP

add constraint `fk2` foreign key (ProdusID)

references ProdusAlimentar(ProdusID);

alter table IntermediarPP

add constraint `fk3` foreign key (ProdusID)

references ProdusAlimentar(ProdusID);

alter table IntermediarPP

add constraint `fk4` foreign key (ProducatorID)

references Producatori(ProducatorID);

O vizualizare de ansamblu:

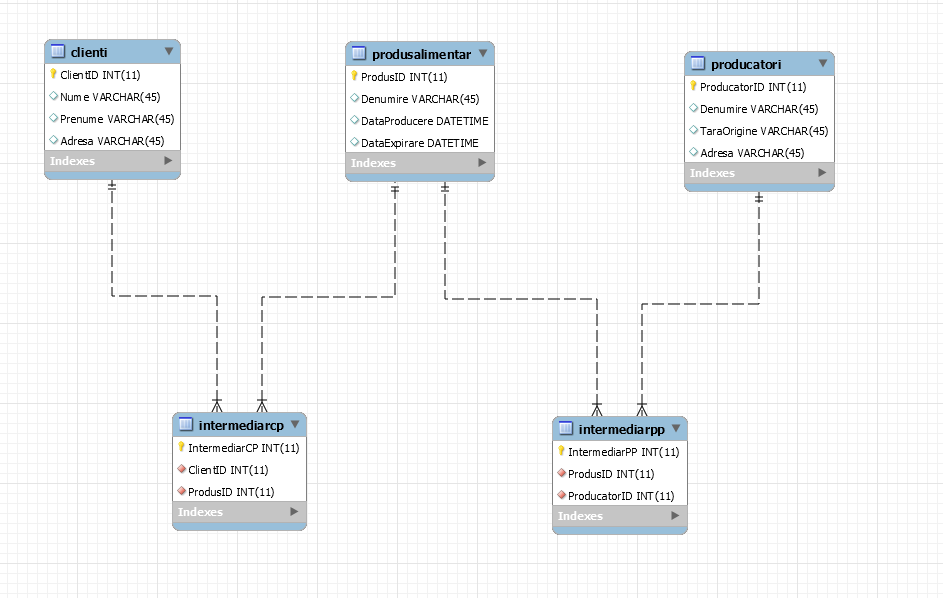
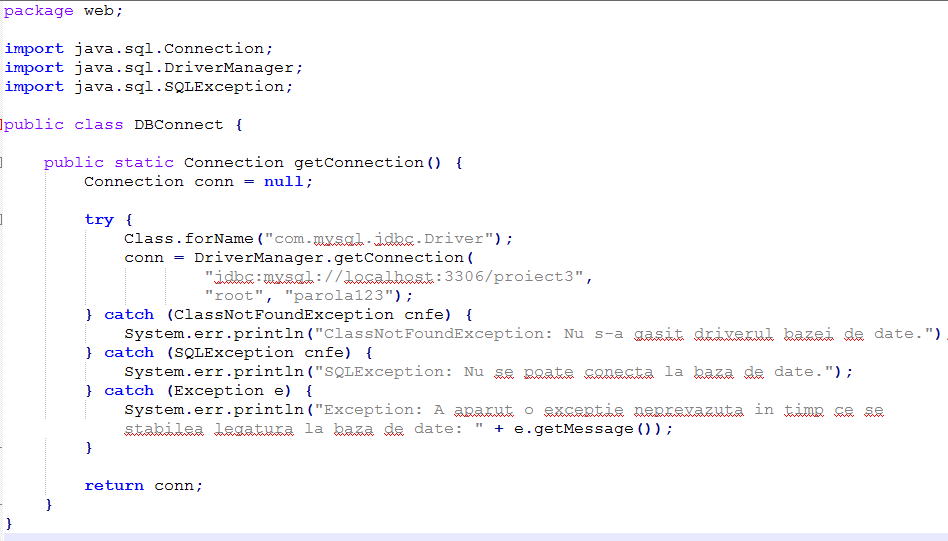


FIG. Baza de date proiect3, cu cele 3 tabele principale si cele 2 intermediare

**2.2) Java**

Vom face in total 11 clase: DBConnect, AuthorizationFilter, Client, EditClienti, Login, MenuView, SessionUtils, Produs, EditProdus, Producator, EditProducator.

Incepem cu clasa DBConnect.

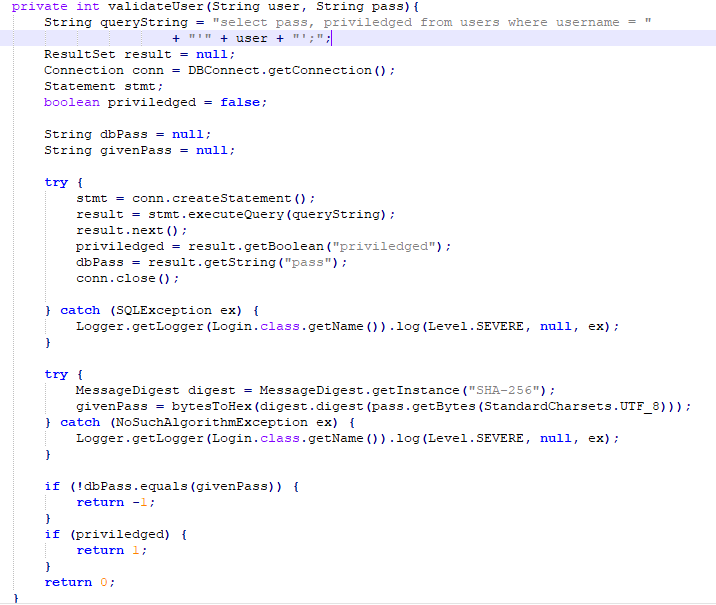


Am creat clasa DBConnect in care ne ocupam de conectarea la baza noastra de date(mysql).

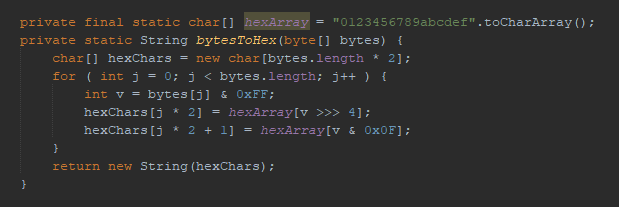
Urmatoarea clasa este cea de Log-IN, ea se numeste login.

In aceasta vom avea 4 atribute, pwd pt parola, user , admin(are dreptul de a sterge din baza de date) si msg deoarece vom intoarce un mesaj la o introducere de cont esuat. Pentru aceasta vom folosi setteri si getteri.

Vom avea o metoda pentru validarea logarii unde se face si un mesaj de tip pop up la introducerea credentialelor gresit, o metoda de logout, o metoda validateUser Compara parola pass cu parola din baza de date pt userul use, daca userul este admin returneaza 1 altfel returneaza 0 pentru user normal si -1 pt autentificare esuata:



Si o metoda pentru criptarea parolei:



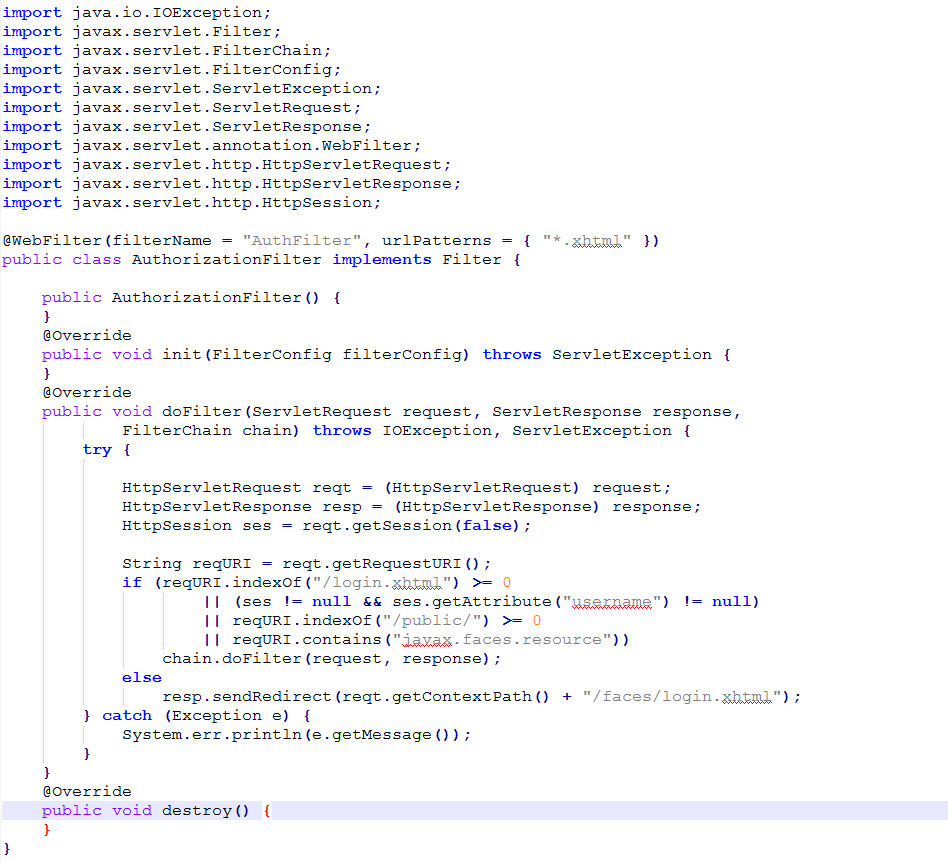
In java s-au folosit adnotari precum:

@ManagedBean(name = "editClienti")

@ViewScoped

Ele au fost necesare deoarece am folosit JSF.

Facem o clasa AuthorizationFilter, scopul acesteia este de a fi redirectati in pagina de LogIN atunci cand o persoana introduce un user invalid in pagina de Log-IN.



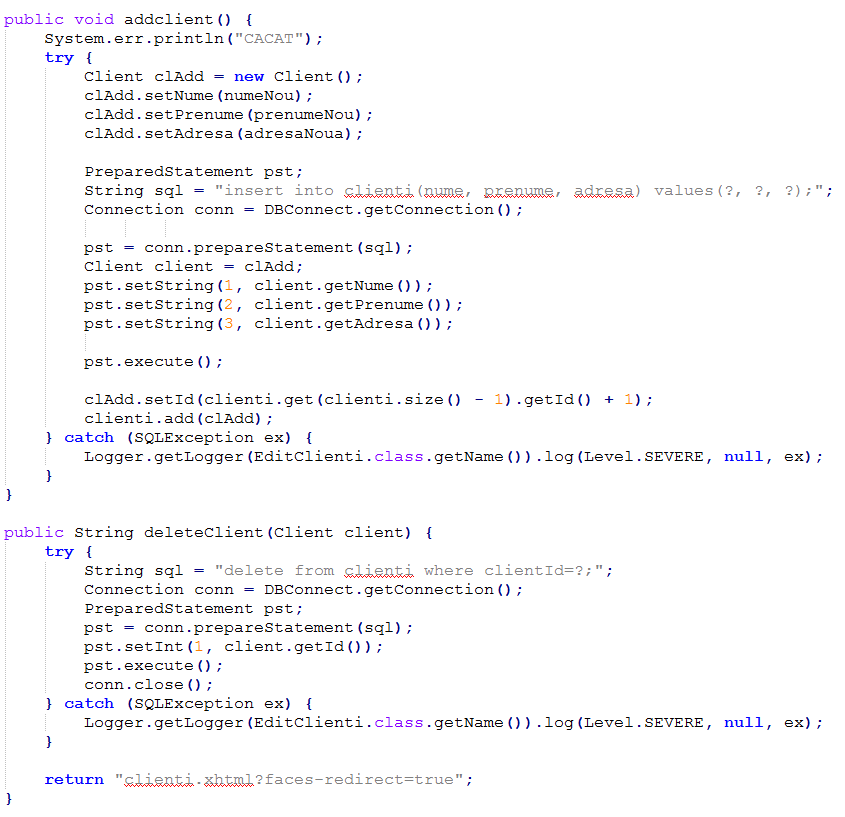
Clasa SessionUtils este pentru a mentine userul logat pe to parcursul sesiunii.

Clasa MenuView este pentru introducerea meniului si componentele sale.

Clasa Client are atributele id, nume, prenume si adresa, adica cele din tabela Clienti. In aceasta vom pune setteri si getteri pt fiecare.

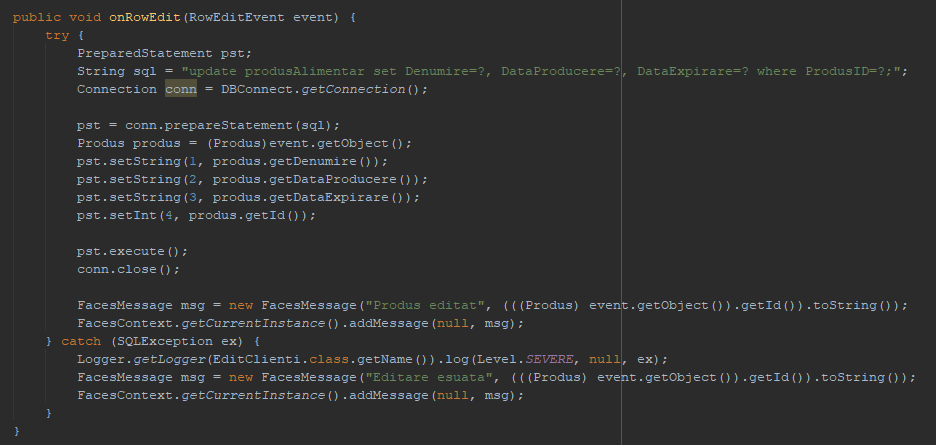
Clasa EditClient contine metodele cele mai importante, pentru vizualizarea tabelelor, modificarea si stergerea unei linii. Se va facea o updatare la baza de date.

Aici avem metoda de adaugare a unui client si metoda de stergere.



Mai avem clasele Produs, EditProdus si Proucator, EditProducatori, acestea sunt similare cu clasele Client si EditClienti, logica fiind aceeasi.

Metoda de editare a unei linii din tabela Produse:



Vom implementa comanda specifica MySQL si o salvam intr-un string.

Vom folosi setteri pentru a seta valorile in linia noastra, dupa vom executa si inchide conexiunea.

Cand produsul va fi editat (se apasa in interfata pe bifa) se va primi o notificare de tip pop-up cu mesajul „Produs editat” (folosim FacesMessage specific jsf), de asemenea daca editarea esueaza va aparea o notificare „Editare esuata”.

**2.3) XHTML**

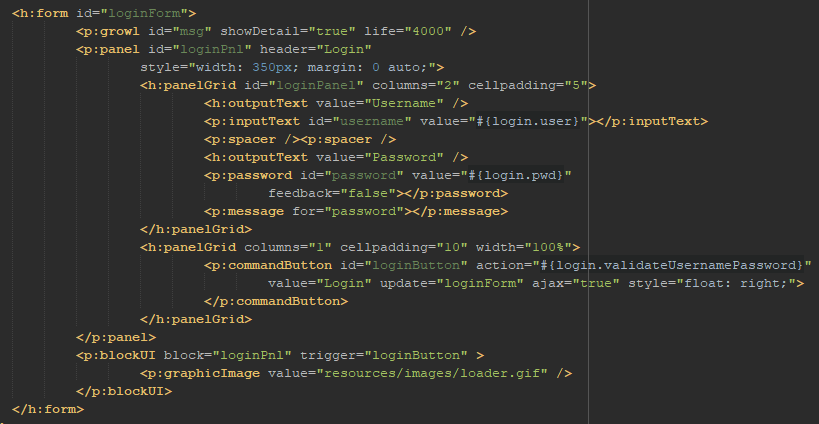
Specifice java server faces:



Avem 5 pagini xhtml.

Prima este cea de login unde avem cele 2 label-uri de completat si un buton de Log In.

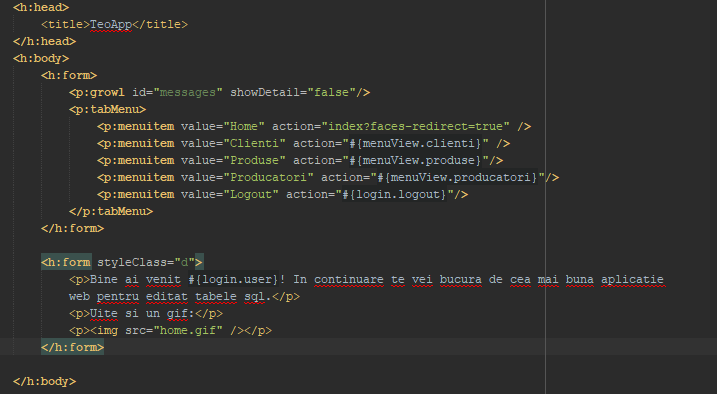
Formularul de login:



Se remarca folosirea primefaces-urilor si celorlalte componente jsf.

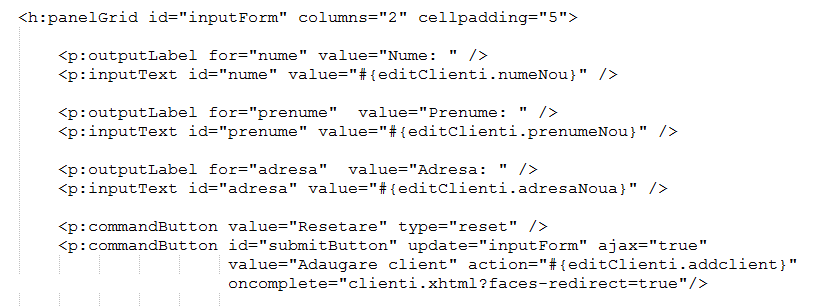
Se foloseste un grid in care vom avea Username si password si buton de login.

A 2-a pagina este index-ul unde avem meniul cu cele 5 tab-uri si un gif in tab-ul de home+text.



La apasarea tab-ului din meniu se va deschide pagina clienti, unde avem tabela Clienti (bagat in tip datatable), aici se pot vizualiza clientii (ClientID, Nume, Prenume, Adresa). Avem optiunea de a modifica clientii si de a adauga noi clienti. Doar adminul poate sterge clienti.

Panel Grid-ul pentru introducerea unui nou client:



Mai sunt paginile xhtml(tabs) de produse si producatori care au acelasi lucru, decat ca adaptat pentru fiecare. Si tab-ul de logout, care ne redirectioneaza catre pagina de login.

**2.4) XML**

Au fost editate xml-urile pt a fi posibila folosirea la java server faces.

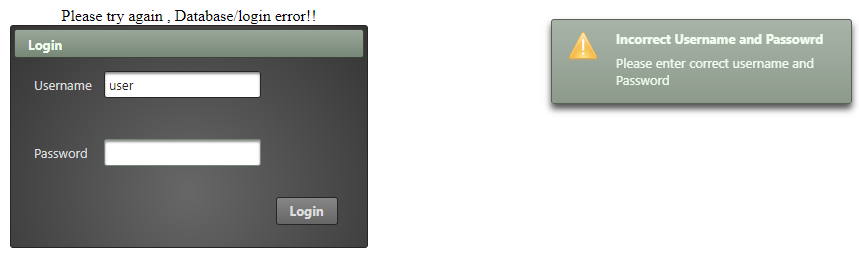
Si vom crea un xml numit faces-config pt a trece de la pagina de login la pagina index.



1. **Interfata Web**

La rularea aplicatiei se va deschide pagina login.xhtml in care trebuiesc introduse credentialele. Avem user user si admin admin, adminul are avantajul ca poate sa si stearga linii din baza noastra de date.

La introducerea credentialelor gresit se va afisa un mesaj de tip pop-up:

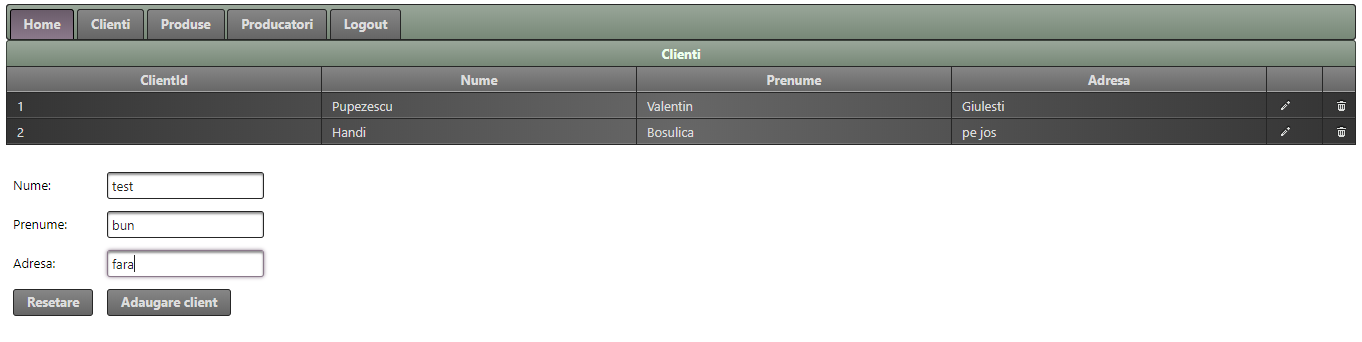


Daca introducem totul corect login-ul va functiona si pagina index.xhtml va fi deschisa:

// in acest moment suntem logati cu admin – admin



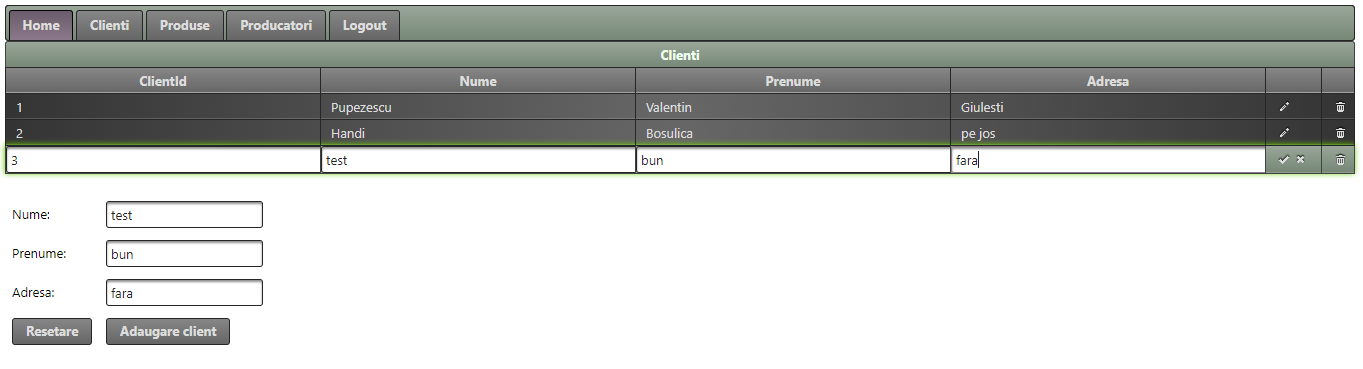
Din meniul de sus vom selecta tab-ul Clienti, acesta va arata asa:



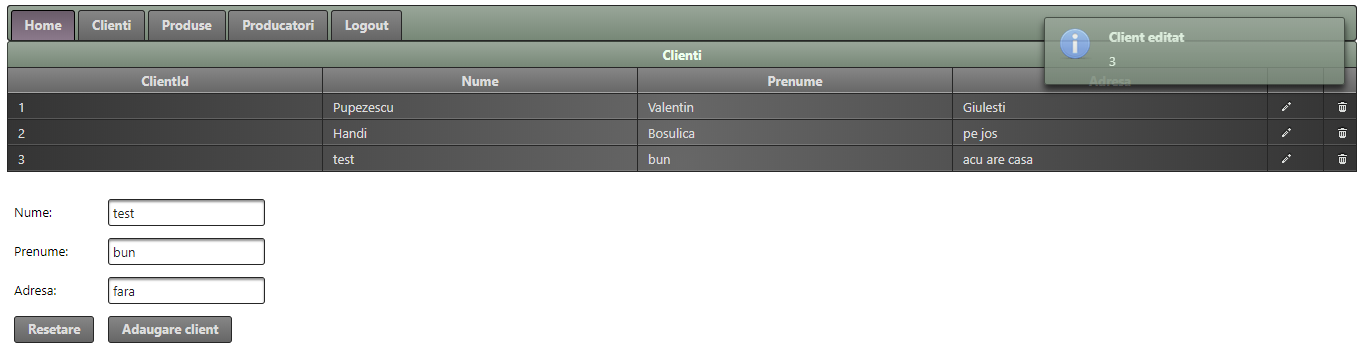
In acest moment dorim sa introducem un client, deci va trebui sa introducem nume, prenume si adresa, precum se vede in poza precedenta si vom apasa pe buton de Adaugare client. Butonul de resetare sterge elementele introduse in cele 3 campuri.

In acest moment a fost adaugat un client in baza de date.

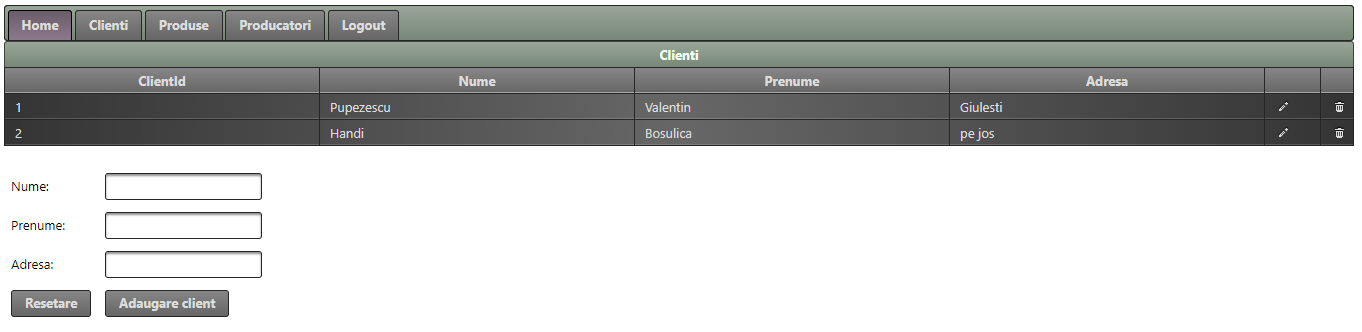
Daca dorim sa editam atributele acestui client, apasam pe buton din dreapta (cel sub forma de creion).



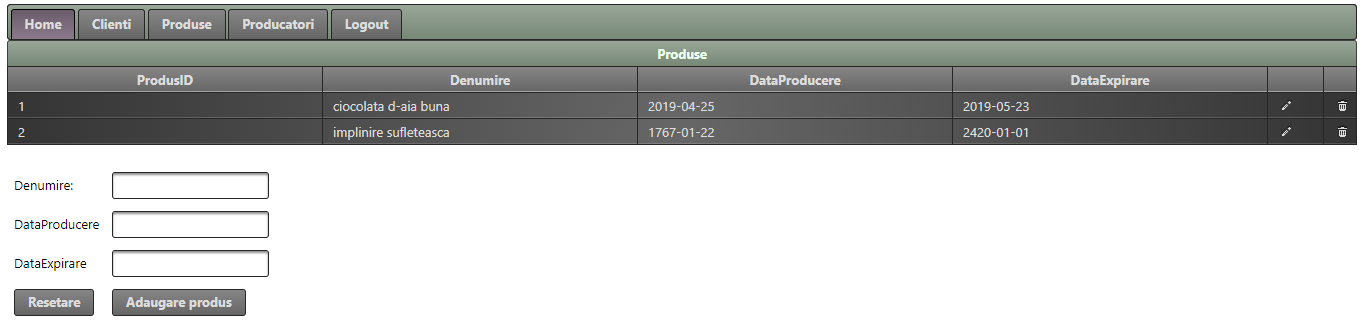
Nu putem modifica ClientId, doar Nume, Prenume si Adresa. Si daca apasam pe bifa si pe x, vom avea o notificare in functie de selectia facuta.



Daca dorim sa stergem, pur si simplu apasam pe butonul cu pictograma unui cos de gunoi:



Daca apasam pe tab-ul de produse vom vizualiza tabela Produse si la fel putem modifica, adauga sau sterge (doar adminul).



Daca apasam pe tab-ul de produse vom vizualiza tabela Producatori si la fel putem modifica, adauga sau sterge (doar adminul).



Daca apasam pe tab-ul de logout, vom fi redirectionati catre login.xhtml si foarte important este de stiut, ca si daca vom da Back, nu vom putea reveni, deoarece sesiunea a fost incheiata.

1. **Bibliografie**

[1] <https://ro.wikipedia.org/wiki/JavaServer_Faces>

[2] <https://www.oracle.com/technetwork/java/javaee/overview-140548.html>

[3] <https://ro.wikipedia.org/wiki/MySQL>

[4] https://ro.wikipedia.org/wiki/SHA-2

https://www.primefaces.org/showcase/ui/data/dataView.xhtml

https://www.journaldev.com/7252/jsf-authentication-login-logout-database-example

https://docs.oracle.com/javaee/5/tutorial/doc/bnapl.html