JDelivery

Documentación

**Docente: Ing. Mario Bressano**

**Cátedra: Lenguaje de programación Java**

**UTN - FRRo**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Legajo** | **Email** |
| Patarca, Pablo | 34695 | pablopatarca@gmail.com |
| Boselli, Matias | 34623 | matias.boselli@gmail.com |
| Nari, Juan Pablo | 34568 | juanpy.nari@gmail.com |

[Introducción](#h.htgtao32igj1)

[Descripción del Negocio](#h.ypev6j1hbm9g)

[1- Funcionalidades del sistema](#h.wg7ut0jx6fef)

[2- Pantallas](#h.jkxxy1hvxm2w)

[3- Diagramas y Código](#h.bwh8rw8lcj9f)

[3.1- Diagramas de clases](#h.q94s3zbikh9b)

[3.2- DER](#h.pl2egn8y2xsk)

[3.3- Casos de Uso](#h.nfj4tpfm6xdy)

[3.3.1- “Realizar un pedido”](#h.n8jr6lysi112)

[3.3.2- “Dar de alta un Producto”](#h.ohg3u06d35cu)

[3.3.3- “Editar un Producto”](#h.zgv7j7h57pig)

[3.4- Códigos](#h.j35bdgt012nw)

[3.4.1- Connection\_db.java](#h.4nuifxs6smgu)

[3.4.2- Cliente\_Datos.java](#h.ml4wgmpfql0f)

[3.4.3- Pedidos\_Datos.java](#h.8q4i6gdv3s7v)

[3.4.4- Productos\_Datos.java](#h.mpbvmchre5tk)

[3.4.5- Modelos](#h.d2304w5bt317)

[Final-Java-1\src\java\Classes\Producto.java](#h.81tl9once4ir)

[Final-Java-1\src\java\Classes\Pedido.java](#h.t2276z6usn1v)

[3.4.6- Servlets](#h.6r5y2d1fhsav)

[Final-Java-1\src\java\Extras\ArmarPedido.java](#h.pn019xyd7q4)

[Final-Java-1\src\java\Extras\Carrito.java](#h.rqddiitaf6k9)

[3.4.7- Scripts](#h.xy8chqukk2bk)

[Final-Java-1\web\includes\menu.jsp](#h.olnjd5z6jlsn)

[Final-Java-1\web\actualizarProducto.jsp](#h.sus0kx9qoc6e)

[Final-Java-1\web\eliminarProducto.jsp](#h.ocghqipno3qv)

[3.5- Tecnologías y librerías adicionales](#h.4zllt2d1vvqn)

**PRIMERA PARTE:**

1- Elaborar una aplicación Web en JSP o Servlets que permita visualizar un ABM que se conecte a una DB. Por ejemplo: un carro de compras, librería online, video, etc.

2- Subirlo a un servidor.

3- Enviar al docente responsable, por lo menos con 10 días de anticipación a la mesa de examen un mail, indicando Nombres y Apellidos de los alumnos que van a rendir, mesa y dirección de acceso a la aplicación (Se debe informar además si maneja usuarios, las claves y los usuarios correspondientes).

**SEGUNDA PARTE:**

1- El día del examen, se presentará el grupo con:

a. Un informe escrito indicando:

1. funcionalidades del sistema,
2. pantallas,
3. diagrama de clases y parte del código (lo más importante)

b. En soporte digital la aplicación y el material antes descrito.

c. Rendirán un coloquio defendiendo e informando sobre como está confeccionado el mismo.

# Introducción

A modo de trabajo final para la asignatura Lenguaje de Programación Java, se ha desarrollado una aplicación Web que nos servirá como plataforma de venta de los distintos productos que tiene para ofrecer una casa de comidas que hace delivery.

# Descripción del Negocio

Actualmente la casa de comidas en cuestión, cuenta con un sistema de pedidos clásico, que consiste en la realización del mismo a través de una llamada telefónica, en la cual se solicita información sobre los precios de los productos y luego se realiza un pedido.

Si bien es un método sencillo, carece de determinados aspectos que enriquezcan la información y harían más fáciles las tareas tanto para la empresa como para los usuarios. La posibilidad de centralizar los pedidos y administrarlos desde un solo lugar, le da a la empresa la posibilidad de organizar mejor los mismos, optimizando tiempos, reduciendo errores y confusiones y dejando un registro limpio y ordenado de la cantidad de ventas realizadas. Por la parte del cliente, el mismo podrá ver tanto imágenes, descripción y precios de los productos como comprarlos directamente a través del sistema.

# 1- Funcionalidades del sistema

La aplicación web tiene como objetivo proveer de una manera rápida para hacer un pedido de comidas al usuario, sin la necesidad de utilizar otro medio, pudiendo ver una descripción e imágenes de lo que compra.También podrá ponerse en contacto con la empresa a través de un módulo diseñado con tales fines. La página, cuenta con 3 niveles de acceso: Invitado, Usuario y Administrador.

- **Nivel Invitado**: Cualquier persona que ingrese a la pagina y no realice el registro o el inicio de sesion, sera un Invitado.

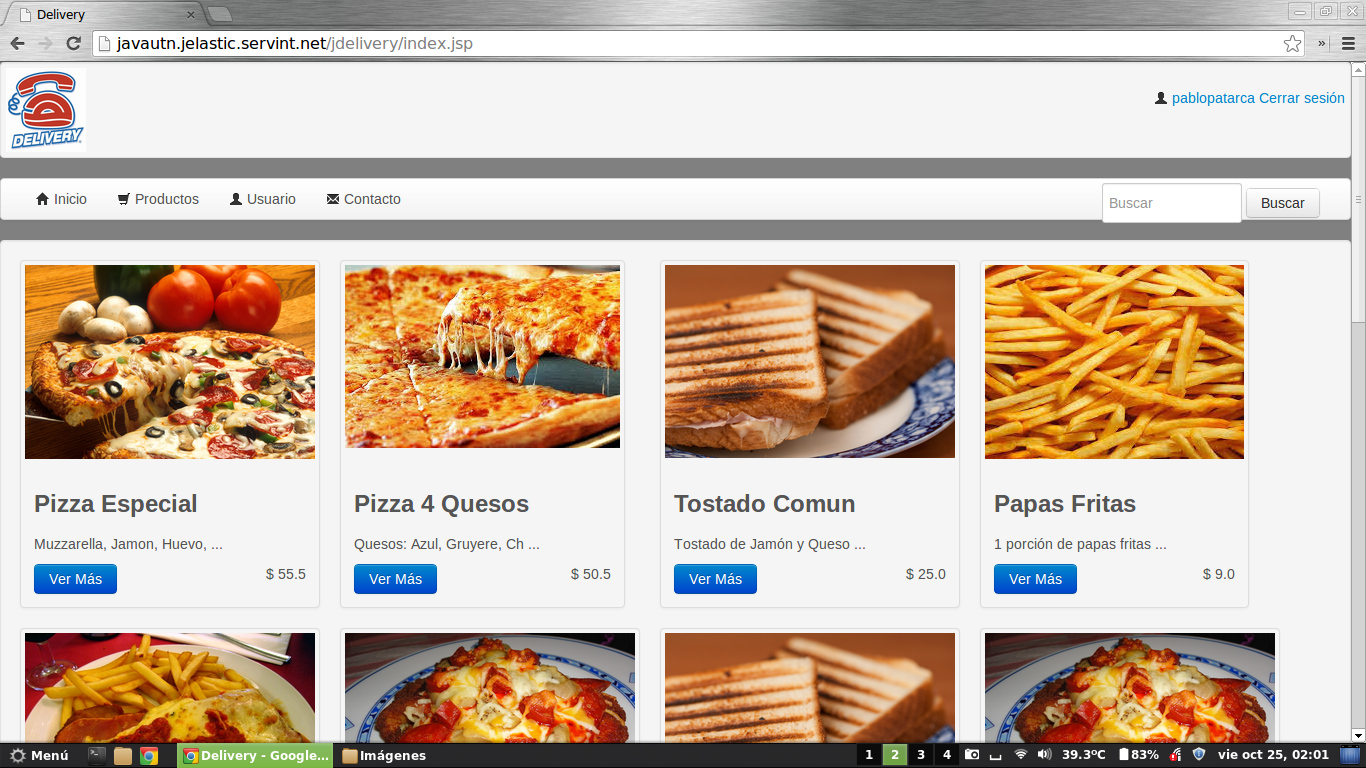
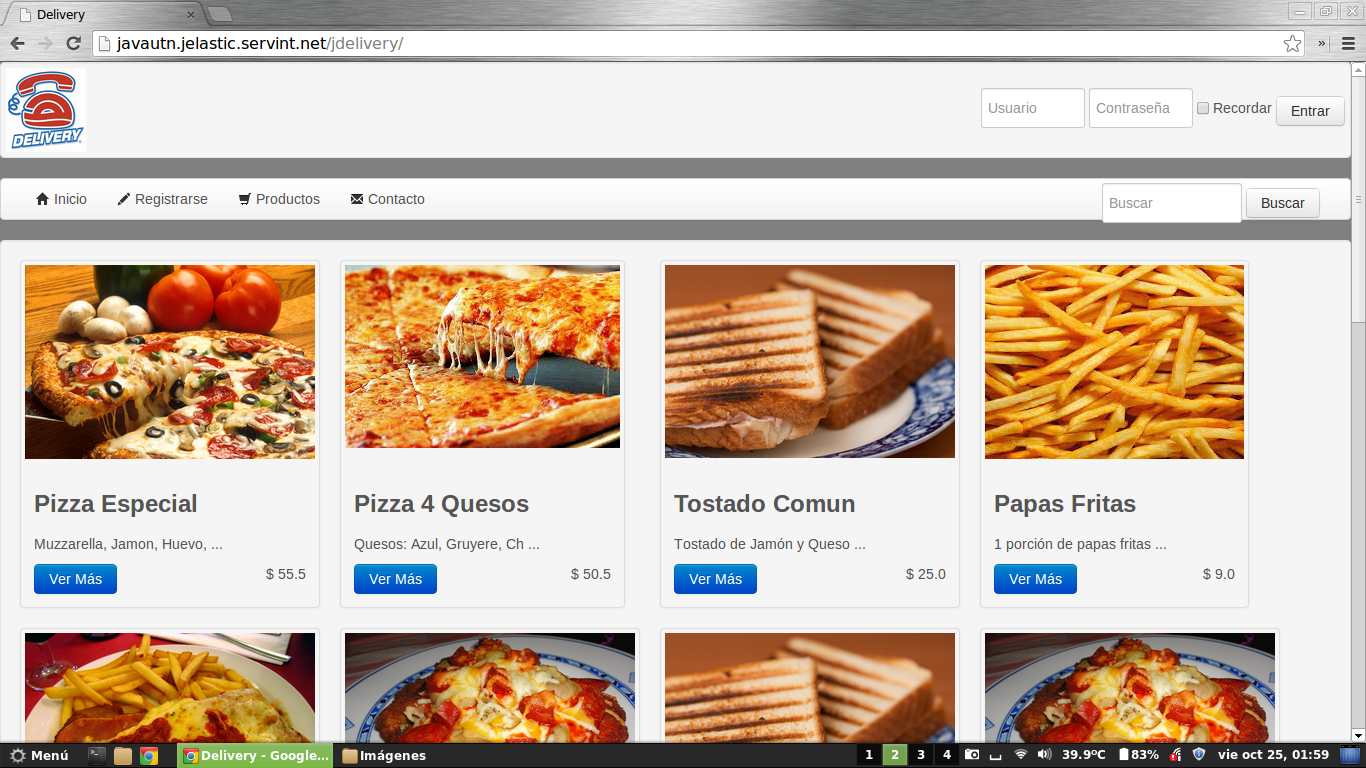
La misma podrá acceder a: la página inicial con un listado de los productos con los que se cuentan, filtrandolos por todos los Productos o aquellos que están en oferta. La la sección de registro donde la persona podrá ingresar sus datos y volverse un usuario. Un módulo de contacto donde podrá ingresar sus datos para comunicarse con la empresa. También se contará con una sección de autenticación, donde ingresando las credenciales de acceso, la persona podrá ingresar como usuario o como administrador. Como invitado, la persona no podrá realizar ningún pedido. En todo momento, sea cual fuere su nivel de acceso, la persona cuenta con la capacidad de realizar una búsqueda general sobre los productos de la página.

- **Nivel Usuario**: a este nivel sólo podrán acceder los clientes registrados a través del formulario de login para autenticarse.

Luego de loguearse podrá configurar sus datos personales, ver el carrito de compra y ver un listado de todos los pedidos realizados. Además, tendrá la posibilidad de agregar los productos que desee a su carrito y realizar un pedido. Si un usuario registrado desea ponerse en contacto con la empresa, podrá hacerlo mediante el formulario de contacto, el cual automáticamente llenará los campos de identidad basándose en el mismo.

-**Nivel Administrador**: a este nivel sólo podrán acceder usuarios definidos como tal, a través del formulario de login para autenticarse. Cuando una persona lo haga, además de todas las funcionalidades de un invitado descritas anteriormente, podrá realizar: Alta, Baja y Modificación de los actuales productos, clientes y pedidos realizados.

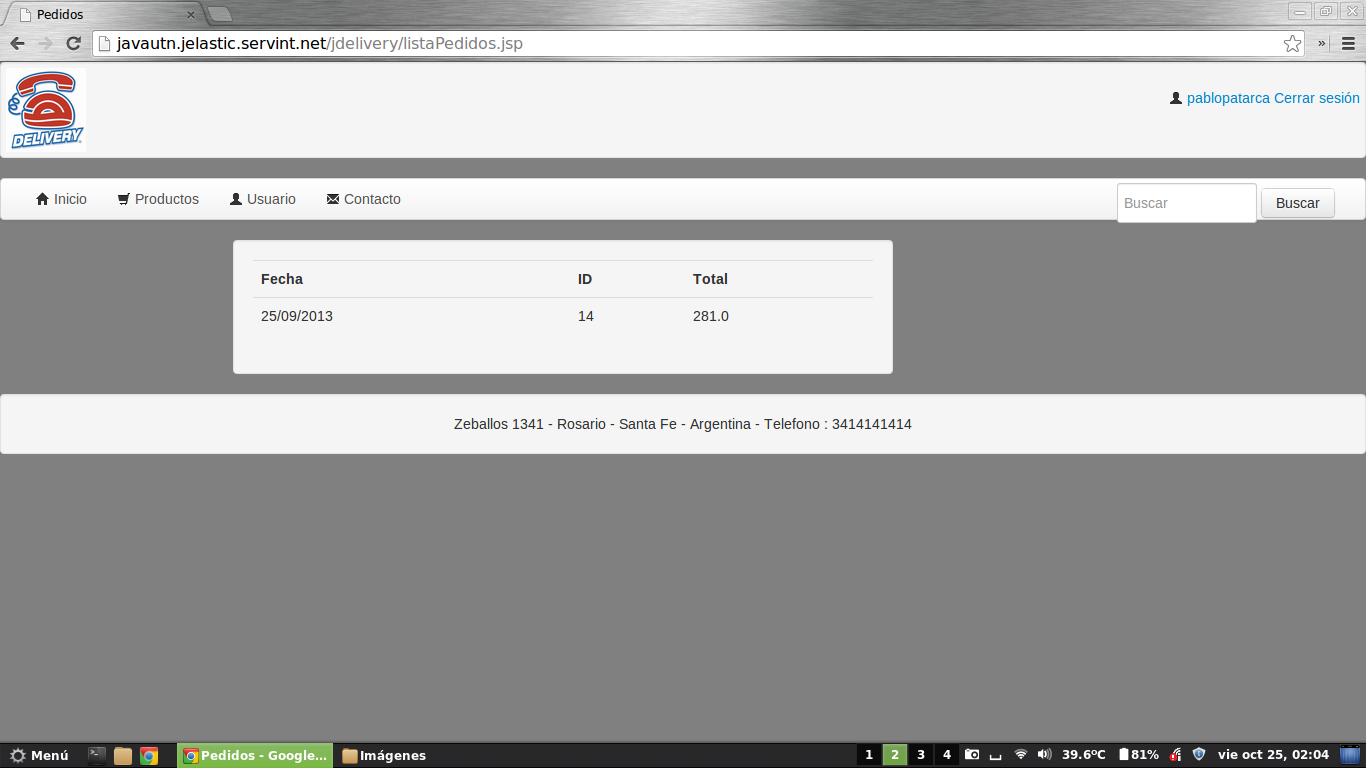
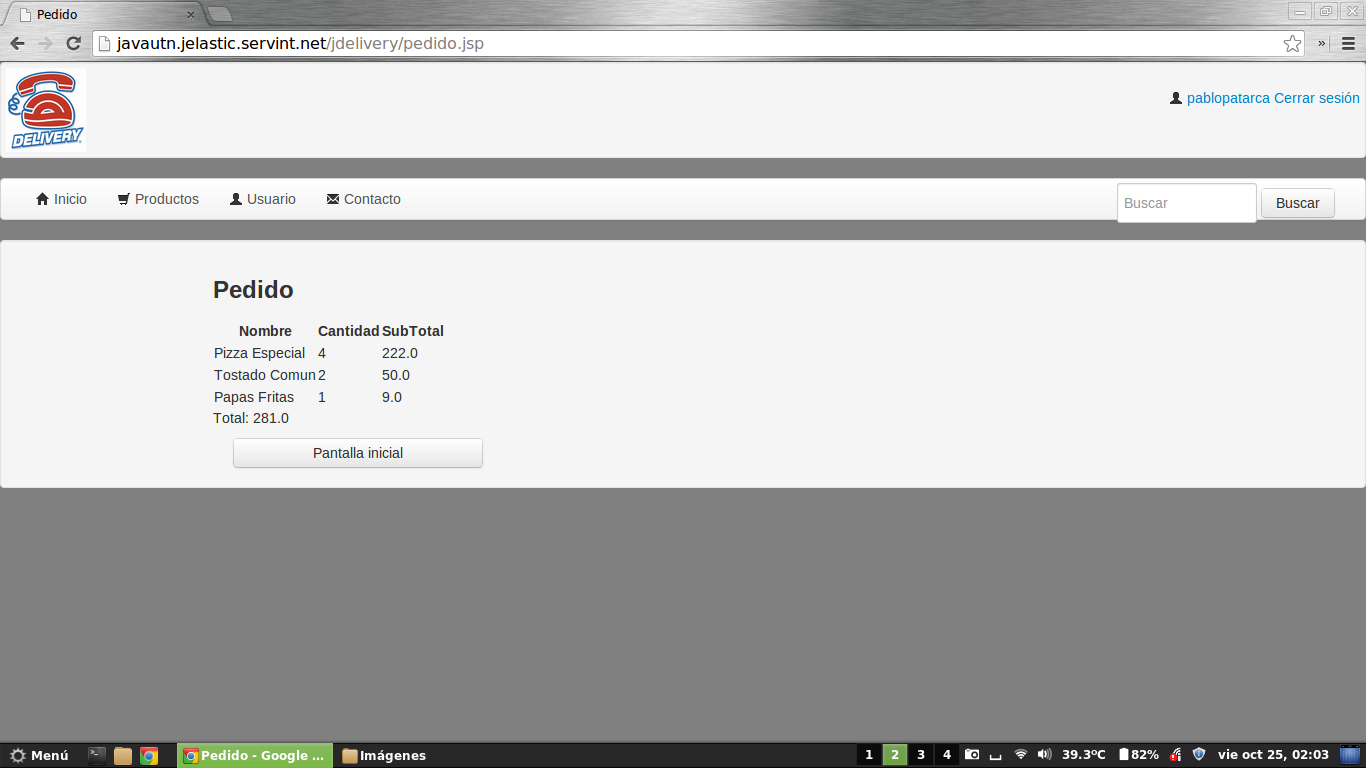
# 2- Pantallas



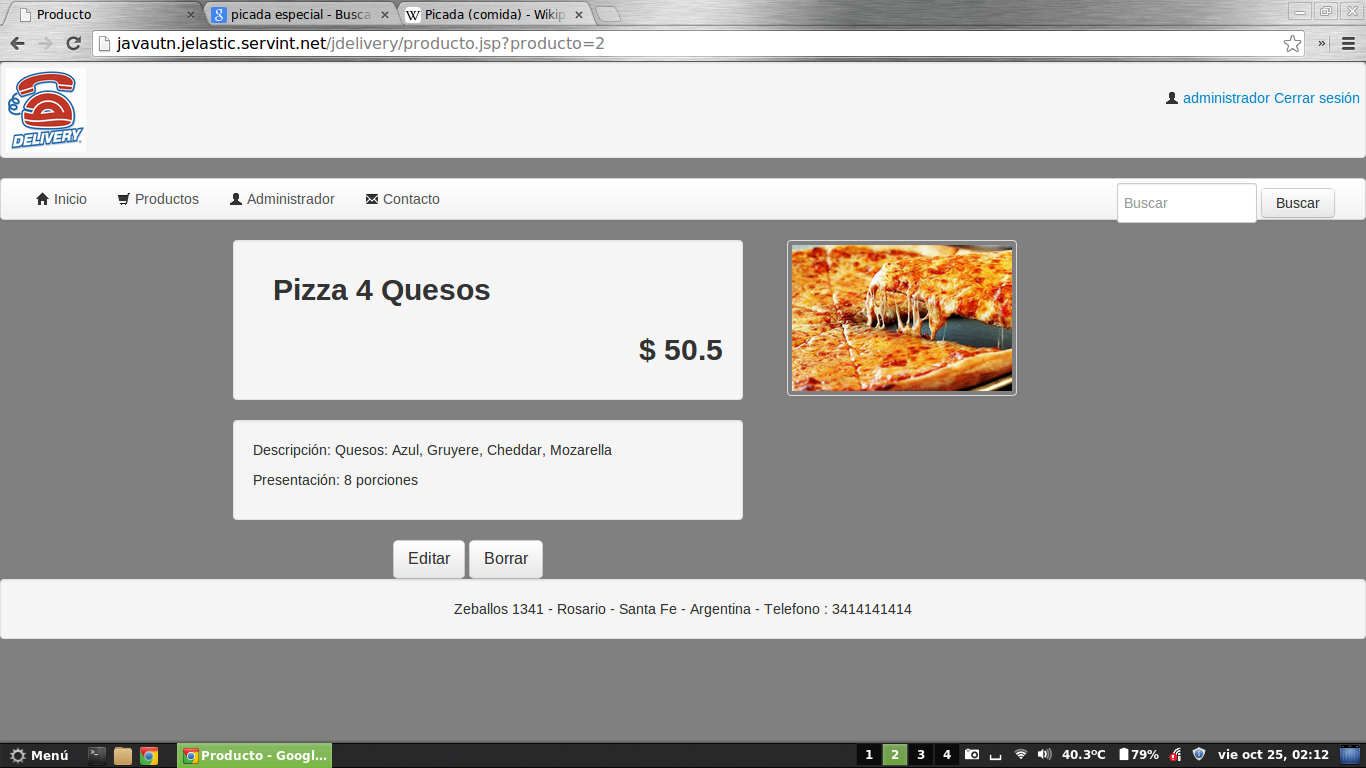
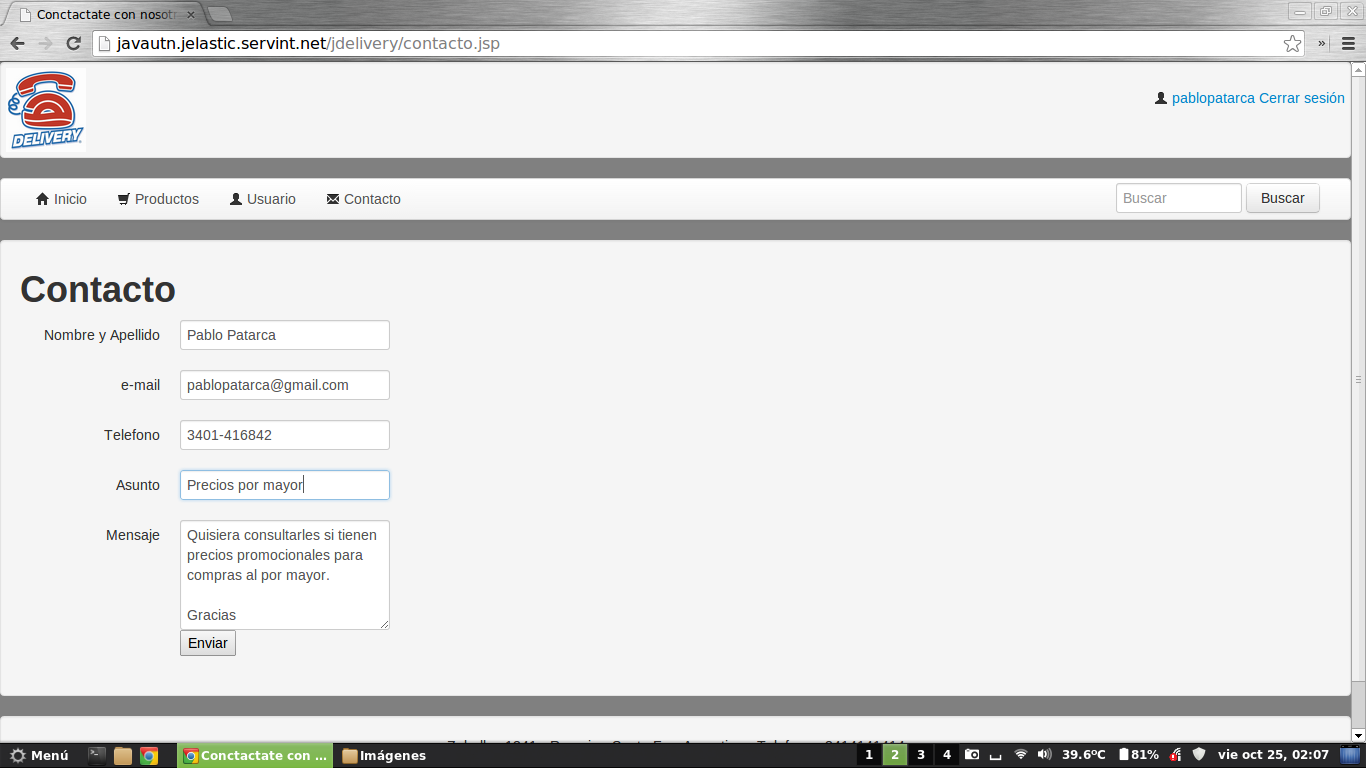
|  |  |
| --- | --- |
| Pantalla #1: Home | Pantalla #2: Usuario logueado |



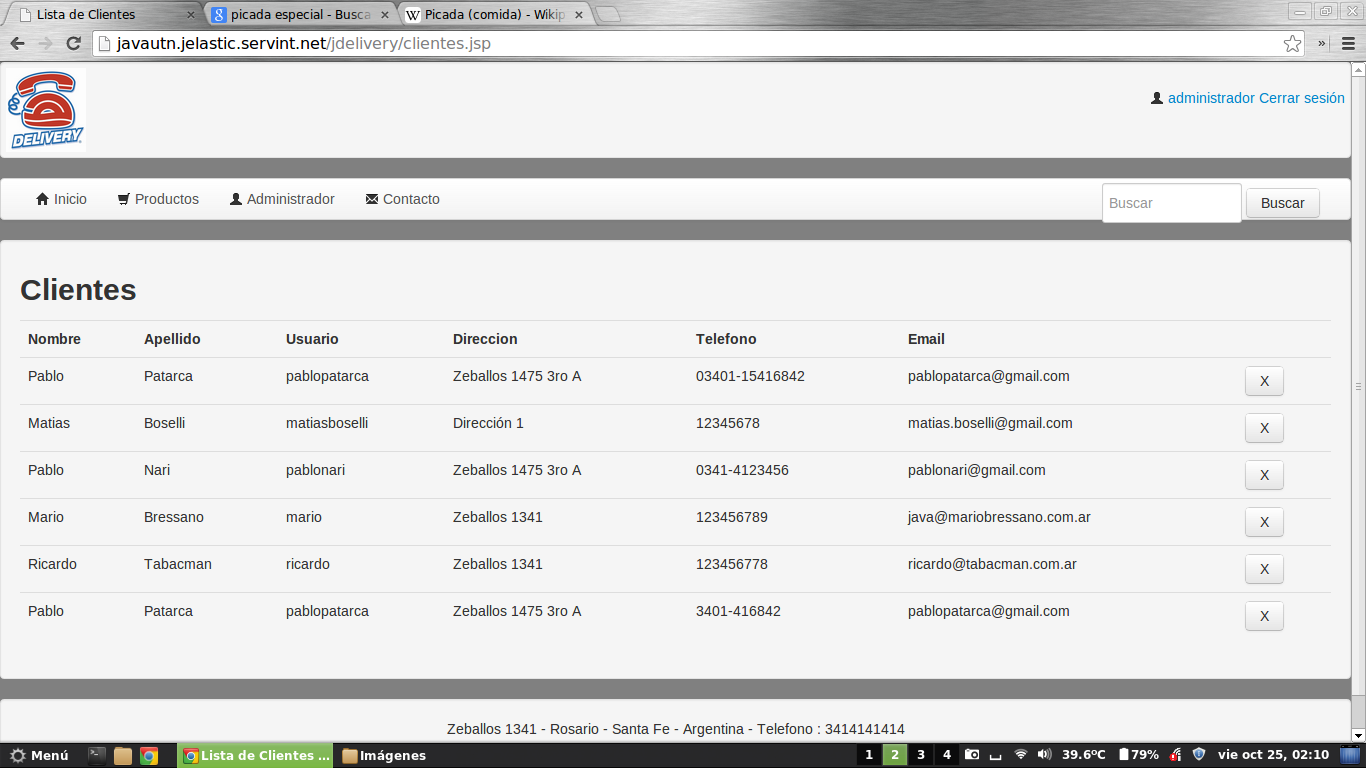
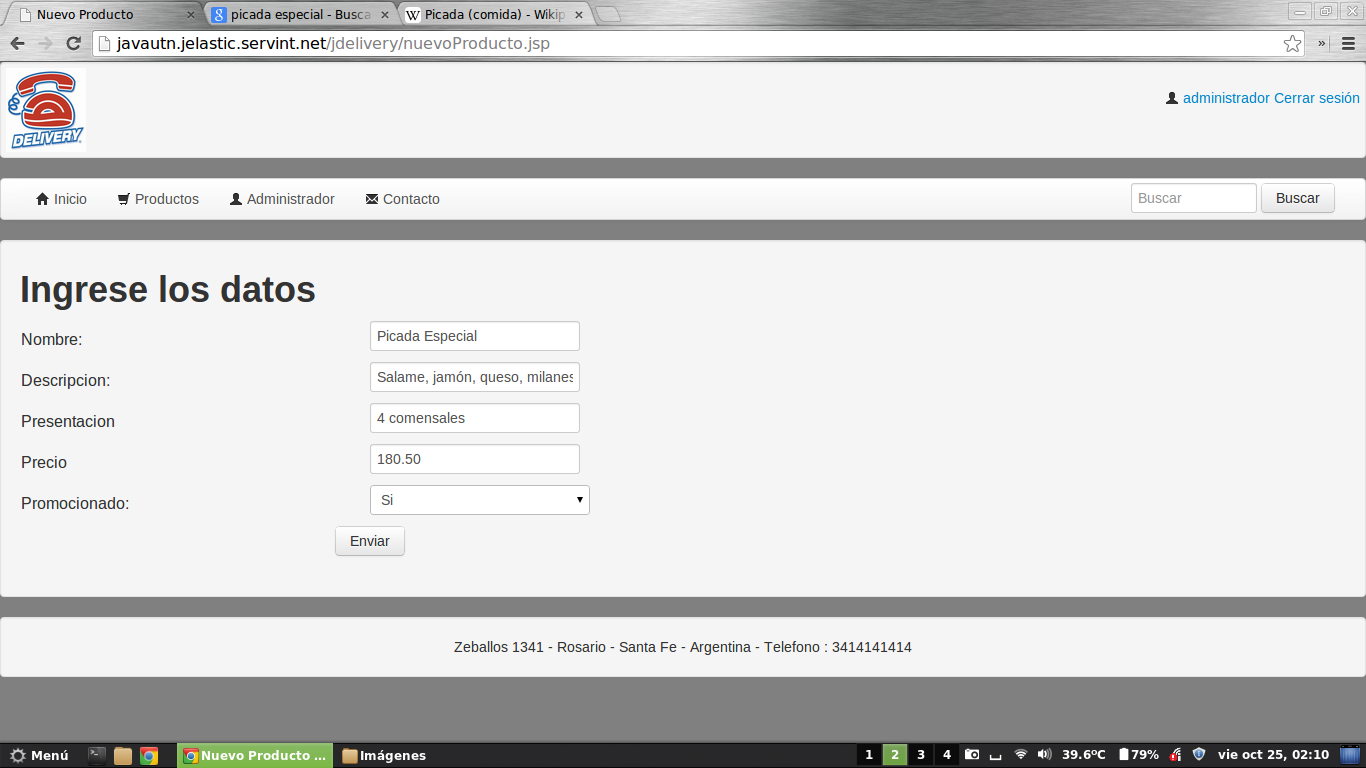
|  |  |
| --- | --- |
| Pantalla #3: Producto | Pantalla #4: Carrito |



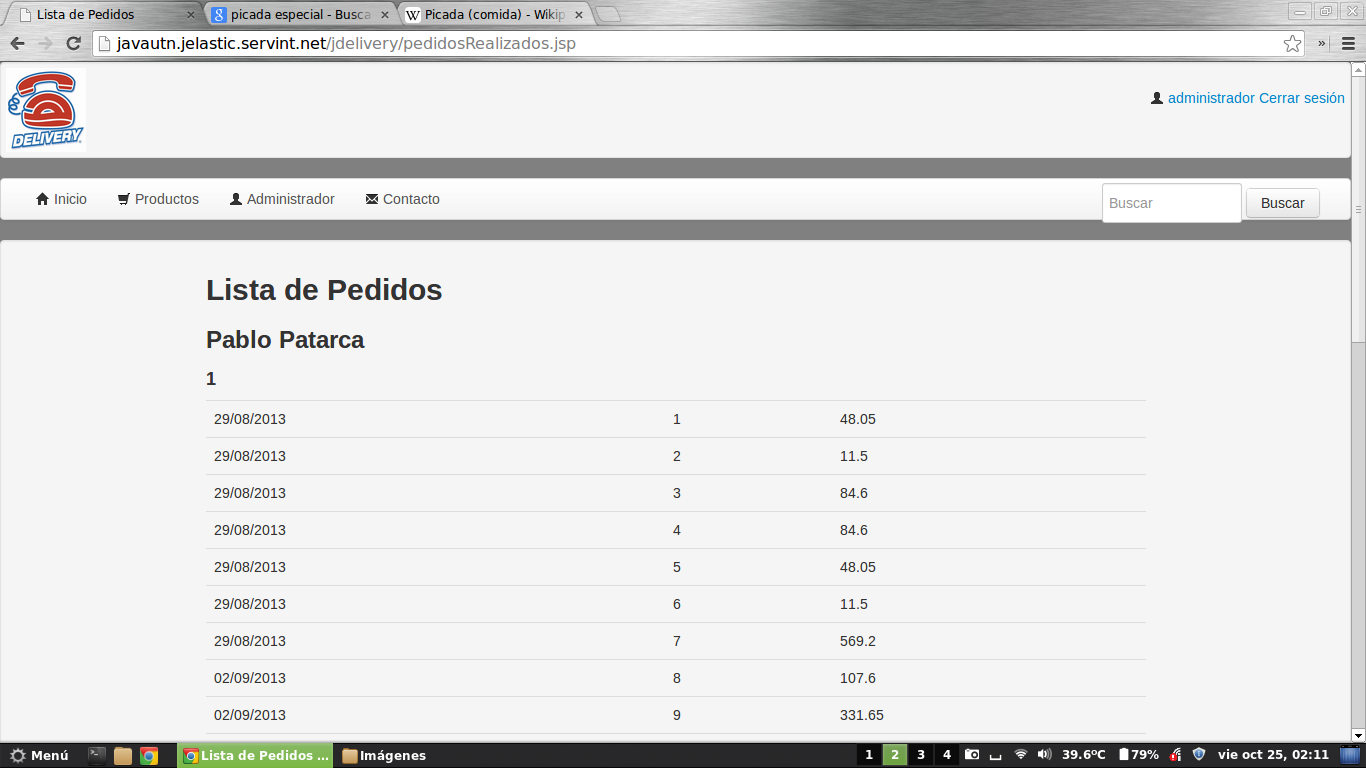
|  |  |
| --- | --- |
| Pantalla #5: Pedido | Pantalla #6: Lista de pedidos de usuario |



|  |  |
| --- | --- |
| Pantalla #7: Contacto | Pantalla #8: Administrador - Producto |



|  |  |
| --- | --- |
| Pantalla #9: Nuevo Producto | Pantalla #10: Listado de clientes |



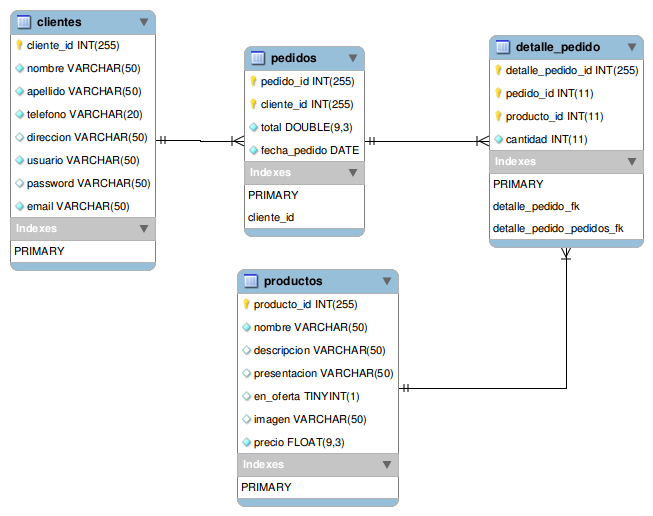
|  |  |
| --- | --- |
| Pantalla #11: Lista de Pedidos de todos los usuarios |  |

# 3- Diagramas y Código

## 3.1- Diagramas de clases



## 3.2- DER



## 3.3- Casos de Uso

### 3.3.1- “Realizar un pedido”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Nivel*** | ***Estructura*** | ***Alcance*** | ***Caja*** | ***Instanciación*** | ***Interacción*** |
| Resumen | Sin-Estructurar | Sistema | Negra | Real | Semántica |

**META**

*Realizar un pedido de los productos deseados, eligiendo la cantidad del mismo.*

**ACTORES**

**Primario:** *Usuario*

**Otros:** -

**PRECONDICIONES (de negocio):** -

**PRECONDICIONES (de sistema)**

**Primarias:** Usuario registrado se encuentra logueado.

**Complementarias:** -

**DISPARADOR:** -

**FLUJO DE SUCESOS**

**CAMINO BÁSICO:**

1. El usuario selecciona el producto que desea que se le envie.

2. El Usuario selecciona una cantidad para el mismo.

3. El Usuario agrega el producto y su cantidad al carrito de compra.

4. El usuario procede al carrito de compras.

5. El usuario realiza el pedido.

6. El Sistema muestra mensaje de confirmación de pedido.

**CAMINOS ALTERNATIVOS:**

**3.a. El usuario desea agregar más productos al pedido.**

3.a.1. Vuelve al paso 1.

**3.b. El Usuario deja sin completar el campo de cantidad.**

3.b. Sistema muestra error informando de los datos faltantes.

**4.a. El usuario desea reiniciar su pedido.**

4.a.1. El usuario presiona limpiar el carrito de compra.

4.a.2. El sistema limpia el carrito de compra.

**4.a. El usuario desea eliminar alguno de los productos de su pedido.**

4.a.1. El usuario presiona eliminar producto del carrito de compra.

4.a.2. El sistema descarta el producto seleccionado del pedido.

**POSTCONDICIONES (de sistema)**

**Éxito:** El sistema registró correctamente el pedido realizado.

**Fracaso:** El pedido fue descartado.

### 3.3.2- “Dar de alta un Producto”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nivel** | **Estructura** | **Alcance** | **Caja** | **Instanciación** | **Interacción** |
| Resumen | Sin-Estructurar | Sistema | Negra | Real | Semántica |

**META**

*Realizar la alta de un producto en el Sistema*

**ACTORES**

**Primario:** *Administrador*

**Otros:** -

**PRECONDICIONES (de negocio):**

**Primarias:** Debe existir el producto en cuestión y estar listo para su comercializacion.

**Complementarias:** -

**PRECONDICIONES (de sistema)**

**Primarias:** Usuario registrado se debe encontrar logueado con una cuenta Administrador.

**Complementarias:** -

**DISPARADOR:** -

**FLUJO DE SUCESOS**

**CAMINO BÁSICO:**

1. El Administrador selecciona la opción Nuevo Producto.

2. El Administrador ingresa el nombre para el Producto.

3. El Administrador ingresa la descripción para el Producto.

4. El Administrador ingresa la presentación del Producto.

5. El Administrador ingresa el precio para el Producto.

6. El Administrador determina si es una producto en oferta.

7. El Administrador guarda la información del Producto.

8. El Sistema muestra mensaje de confirmación de Alta.

**CAMINOS ALTERNATIVOS:**

**7.a. El usuario no completa alguno de los campos del producto.**

7.a.1. El sistema muestra un error informativo.

**POSTCONDICIONES (de sistema)**

**Éxito:** El administrador da de alta correctamente el producto.

**Fracaso:** El producto no se carga.

### 3.3.3- “Editar un Producto”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nivel** | **Estructura** | **Alcance** | **Caja** | **Instanciación** | **Interacción** |
| Resumen | Sin-Estructurar | Sistema | Negra | Real | Semántica |

**META**

*Realizar la edición de un producto en el Sistema*

**ACTORES**

**Primario:** *Administrador*

**Otros:** -

**PRECONDICIONES (de negocio):** -

**PRECONDICIONES (de sistema)**

**Primarias:**

Usuario registrado se debe encontrar logueado con una cuenta Administrador.

Debe existir al menos un producto cargado.

**Complementarias:** -

**DISPARADOR:** -

**FLUJO DE SUCESOS**

**CAMINO BÁSICO:**

1. El Administrador selecciona un producto Producto.

2. El Administrador selecciona la opción de editar el producto.

3. El Administrador cambia algún dato del producto.

4. El Administrador guarda la información del Producto.

5. El Sistema muestra el producto con la información actualizada.

**CAMINOS ALTERNATIVOS:**

**3.a. El usuario desea cargar una foto al producto.**

3.a.1. El Administrador selecciona una foto para el producto.

3.a.2 El Administrador guarda la foto.

3.a.3 El Sistema guarda y asocia la foto al producto.

**4.a. El usuario no completa alguno de los campos del producto.**

4.a.1. El sistema muestra un error informativo.

**POSTCONDICIONES (de sistema)**

**Éxito:** El administrador modifica correctamente el producto.

**Fracaso:** El producto no se edita.

## 3.4- Códigos

### 3.4.1- Connection\_db.java

Final-Java-1\src\java\Extras\Connection\_class.java

1 package Extras;  
 2   
 3 import java.sql.\*;  
 4   
 5 public class Connection\_class{  
 6 public static Connection *mysql\_connect*() throws SQLException {  
 7   
 8 Connection conn = null;  
 9 String url = "jdbc:mysql://mysql-javautn.jelastic.servint.net/";  
10 String dbName = "jdelivery";  
11 String driver = "com.mysql.jdbc.Driver";  
12 String userName = "root";   
13 String password = "6bJwC1aY1Y";  
14 try {  
15 Class.*forName*(driver).newInstance();  
16 conn = DriverManager.*getConnection*(url+dbName,userName,password);  
17 return conn;   
18 } catch (Exception e) {  
19 e.printStackTrace();  
20 return conn;  
21 }  
22 }  
23 }

### 3.4.2- Cliente\_Datos.java

Final-Java-1\src\java\Datos\Cliente\_Datos.java

1 package Datos;  
 2   
 3 import Classes.Cliente;  
 4 import java.sql.\*;  
 5 import java.util.ArrayList;  
 6   
 7 public abstract class Cliente\_Datos   
 8 {  
 9 public static Cliente *ValidarCliente*(String usuario, String pass) throws SQLException  
 10 {   
 11 Cliente cli = null;  
 12 try  
 13 {  
 14 Connection con = Extras.Connection\_class.*mysql\_connect*();  
 15 PreparedStatement pstm = con.prepareStatement(  
 16 "SELECT \* FROM clientes WHERE"  
 17 + " usuario='" + usuario+"'"  
 18 + " AND password='" + pass + "'");  
 19   
 20 ResultSet res= pstm.executeQuery();  
 21   
 22 if(res.next())  
 23 {  
 24 cli = new Cliente();  
 25 cli.setId(Integer.*parseInt*(res.getString("cliente\_id").toString()));  
 26 cli.setNombre(res.getString("nombre"));  
 27 cli.setApellido(res.getString("apellido"));  
 28 cli.setTelefono(res.getString("telefono"));  
 29 cli.setDireccion(res.getString("direccion"));  
 30 cli.setUsuario(res.getString("usuario"));  
 31 cli.setEmail(res.getString("email"));  
 32 }  
 33 pstm.close();  
 34 res.close();  
 35 con.close();  
 36 }  
 37 catch(SQLException e)  
 38 {  
 39 throw e;  
 40 }  
 41   
 42 return cli;  
 43 }  
 44 public static Cliente *buscarCliente*(String usuario) throws SQLException  
 45 {  
 46 Cliente cli = null;  
 47 try  
 48 {  
 49 Connection con = Extras.Connection\_class.*mysql\_connect*();  
 50 PreparedStatement pstm= con.prepareStatement("SELECT \* FROM clientes WHERE usuario='"+usuario+"'");  
 51 ResultSet res= pstm.executeQuery();  
 52 if(res.next())  
 53 {  
 54 cli = new Cliente();  
 55 cli.setNombre(res.getString("nombre"));  
 56 cli.setApellido(res.getString("apellido"));  
 57 cli.setTelefono(res.getString("telefono"));  
 58 cli.setUsuario(res.getString("usuario"));  
 59 cli.setEmail(res.getString("email"));  
 60 }  
 61 pstm.close();  
 62 res.close();  
 63 con.close();  
 64 }  
 65 catch(SQLException e)  
 66 {  
 67 throw e;  
 68 }  
 69   
 70 return cli;  
 71 }  
 72 public static boolean *nuevoCliente*(Cliente cli) throws SQLException  
 73 {  
 74 try{  
 75 Connection con = Extras.Connection\_class.*mysql\_connect*();  
 76 String query = "INSERT INTO clientes"  
 77 + "(usuario, password, apellido, nombre, direccion, telefono, email)"  
 78 + " VALUES"  
 79 + "('"+cli.getUsuario()+"',"  
 80 + "'"+cli.getPassword()+"',"  
 81 + "'"+cli.getApellido()+"',"  
 82 + "'"+cli.getNombre()+"',"  
 83 + "'"+cli.getDireccion()+"',"  
 84 + "'"+cli.getTelefono()+"',"  
 85 + "'"+cli.getEmail()+"')";  
 86   
 87 System.*out*.println(query);  
 88   
 89 Statement statement = con.createStatement();  
 90 statement.executeUpdate(query);  
 91   
 92 con.close();  
 93 return true;  
 94 }  
 95 catch(SQLException e)  
 96 {   
 97 return false;  
 98 //throw e;  
 99 }  
100 }  
101 public static void *actualizarCliente*(Cliente cli) throws SQLException  
102 {  
103 try{  
104 Connection con = Extras.Connection\_class.*mysql\_connect*();   
105 PreparedStatement pstm = con.prepareStatement("UPDATE clientes set nombre=?, apellido=?, email=?, direccion=?, telefono=? where usuario=?");  
106 pstm.setString(1, cli.getNombre());  
107 pstm.setString(2, cli.getApellido());  
108 pstm.setString(3, cli.getEmail());  
109 pstm.setString(4, cli.getDireccion());  
110 pstm.setString(5, cli.getTelefono());  
111 pstm.setString(6, cli.getUsuario());  
112 pstm.executeUpdate();   
113 pstm.close();  
114 con.close();  
115   
116 }  
117 catch(SQLException e)  
118 {  
119 throw e;  
120 }  
121 }  
122   
123   
124 public static ArrayList<Cliente> *getClientes*() throws SQLException  
125 {  
126 ArrayList<Cliente> listaClientes = new ArrayList<Cliente>();  
127 Cliente cli = null;  
128 try  
129 {  
130 Connection con = Extras.Connection\_class.*mysql\_connect*();  
131 PreparedStatement pstm= con.prepareStatement("SELECT \* FROM clientes WHERE usuario not like 'administrador'");  
132 ResultSet res= pstm.executeQuery();  
133 while(res.next())  
134 {  
135 cli = new Cliente();  
136 cli.setId(res.getInt("cliente\_id"));  
137 cli.setNombre(res.getString("nombre"));  
138 cli.setApellido(res.getString("apellido"));  
139 cli.setTelefono(res.getString("telefono"));  
140 cli.setUsuario(res.getString("usuario"));  
141 cli.setEmail(res.getString("email"));  
142 cli.setDireccion(res.getString("direccion"));  
143 listaClientes.add(cli);  
144 }  
145 pstm.close();  
146 res.close();  
147 con.close();  
148 }  
149 catch(SQLException e)  
150 {  
151 throw e;  
152 }  
153   
154 return listaClientes;  
155 }  
156   
157 public static void *eliminarCliente*(int cliente\_id) throws SQLException  
158 {  
159 try  
160 {  
161 Connection con = Extras.Connection\_class.*mysql\_connect*();  
162 PreparedStatement pstm= con.prepareStatement("DELETE FROM clientes WHERE cliente\_id='"+cliente\_id+"'");  
163 System.*out*.println("DELETE FROM clientes WHERE cliente\_id='"+cliente\_id+"'");  
164 pstm.executeUpdate();  
165   
166 pstm.close();  
167 con.close();  
168 }  
169 catch(SQLException e)  
170 {  
171 throw e;  
172 }  
173   
174 }  
175   
176 }  
177

### 3.4.3- Pedidos\_Datos.java

Final-Java-1\src\java\Datos\Pedidos\_Datos.java

1 package Datos;  
 2 import Classes.Pedido;  
 3 import Classes.Producto;  
 4 import java.sql.Connection;  
 5 import java.sql.PreparedStatement;  
 6 import java.sql.ResultSet;  
 7 import java.sql.SQLException;  
 8 import java.sql.Statement;  
 9 import java.util.ArrayList;  
10 /\*\*  
11 \*  
12 \* @author Pablo  
13 \*/  
14 public class Pedido\_Datos {  
15 public static void *nuevoPedido*(Pedido pedido) throws SQLException  
16 {  
17 try {  
18 Class.*forName*("com.mysql.jdbc.Driver");  
19 } catch (ClassNotFoundException e1) {  
20 e1.printStackTrace();  
21 }   
22 Connection con = Extras.Connection\_class.*mysql\_connect*();  
23 int cliente\_id = pedido.getClienteId();  
24 try  
25 {  
26 String query = "INSERT INTO pedidos(cliente\_id, total, fecha\_pedido) VALUES('"+  
27 pedido.getClienteId() +"', '"+  
28 pedido.getTotal()+"', '"+  
29 pedido.getFechaSQL()+"' );";  
30 PreparedStatement pstm = con.prepareStatement(query);  
31   
32 Statement statement = con.createStatement();  
33 String [] columnas = { "pedido\_id" };  
34 statement.executeUpdate(query, columnas);  
35 ResultSet rs = statement.getGeneratedKeys();  
36 if(rs.next()){  
37   
38 //System.out.println("NUMERO DE PEDIDO: "+rs.getInt(1));  
39 pedido.setId(rs.getInt(1));  
40 }  
41   
42 ArrayList<Producto> listaProductos = pedido.getListaProductos();  
43   
44 for (Producto producto : listaProductos){  
45 query = "INSERT INTO detalle\_pedido(pedido\_id, producto\_id, cantidad)"  
46 + "VALUES ("  
47 + "'" + pedido.getId() +"',"  
48 + "'" + producto.getId() +"',"  
49 + "'" + producto.getCanidad() +"'"  
50 + ")";  
51 statement.executeUpdate(query);  
52 }  
53 con.close();  
54 }  
55 catch(SQLException e)  
56 {  
57 throw e;   
58 }  
59 }  
60   
61   
62 public static ArrayList<Pedido> *getPedidos*(int cliente\_id) throws SQLException  
63 {  
64 Pedido pedido;  
65 ArrayList<Pedido> listaPedidos = new ArrayList<Pedido>();  
66 try {  
67 Class.*forName*("com.mysql.jdbc.Driver");  
68 } catch (ClassNotFoundException e1) {  
69   
70 e1.printStackTrace();  
71 }   
72   
73 //Connection con = DriverManager.getConnection(ConexionDB.GetUrlConexion());  
74 Connection con = (Connection) Extras.Connection\_class.*mysql\_connect*();  
75 java.sql.PreparedStatement pstm;  
76 if(cliente\_id!=0){  
77 pstm = con.prepareStatement("SELECT \* FROM pedidos"  
78 + " WHERE pedidos.cliente\_id = '" + cliente\_id + "'");  
79 }else{  
80 pstm = con.prepareStatement("SELECT \* FROM pedidos");   
81 }  
82 ResultSet res = pstm.executeQuery();  
83 while(res.next()){  
84   
85 pedido = new Pedido();  
86 pedido.setClienteId(cliente\_id);  
87 pedido.setId(res.getInt("pedidos.pedido\_id"));  
88 pedido.setTotal(res.getDouble("pedidos.total"));  
89 pedido.setFecha(res.getString("pedidos.fecha\_pedido"));  
90 listaPedidos.add(pedido);  
91 }  
92   
93 return listaPedidos;  
94 }  
95 }  
96

### 3.4.4- Productos\_Datos.java

Final-Java-1\src\java\Datos\Productos\_Datos.java

1 /\*  
 2 \* To change this template, choose Tools | Templates  
 3 \* and open the template in the editor.  
 4 \*/  
 5 package Datos;  
 6 import Classes.Producto;  
 7 import Extras.Carrito;  
 8 import com.mysql.jdbc.Connection;  
 9 import java.sql.PreparedStatement;  
 10 import java.sql.ResultSet;  
 11 import java.sql.SQLException;  
 12 import java.util.ArrayList;  
 13 import java.util.logging.Level;  
 14 import java.util.logging.Logger;  
 15   
 16 /\*\*  
 17 \*  
 18 \* @author Matias  
 19 \*/  
 20 public class Producto\_Datos {  
 21   
 22 String id;  
 23 public static void *nuevoProducto*(Producto p) throws SQLException  
 24 {  
 25 Connection con = (Connection) Extras.Connection\_class.*mysql\_connect*();  
 26 try  
 27 {  
 28 PreparedStatement pstm = (PreparedStatement) con.prepareStatement("INSERT INTO "  
 29 + "productos(nombre, descripcion, presentacion, en\_oferta, precio) "  
 30 + "VALUES(?, ?, ?, ?, ?);");  
 31 pstm.setString(1, p.getNombre());  
 32 pstm.setString(2, p.getDescripcion());  
 33 pstm.setString(3, p.getPresentacion());  
 34 pstm.setInt(4, p.getEnOferta());  
 35 pstm.setString(5, p.getPrecio().toString());  
 36 pstm.executeUpdate();  
 37   
 38 pstm.close();  
 39   
 40 con.close();  
 41 }  
 42 catch(SQLException e)  
 43 {  
 44 throw e;  
 45 }  
 46 }  
 47 public static void *editarProducto*(Producto p) throws SQLException  
 48 {  
 49 Connection con = (Connection) Extras.Connection\_class.*mysql\_connect*();  
 50 try  
 51 {  
 52 PreparedStatement pstm = (PreparedStatement) con.prepareStatement("UPDATE productos SET nombre = ?,descripcion = ?,presentacion = ?,en\_oferta = ?,precio = ? WHERE producto\_id = ?;");  
 53 pstm.setString(1, p.getNombre());  
 54 pstm.setString(2, p.getDescripcion());  
 55 pstm.setString(3, p.getPresentacion());  
 56 pstm.setInt(4, p.getEnOferta());  
 57 pstm.setString(5, p.getPrecio().toString());  
 58 pstm.setString(6, String.*valueOf*(p.getId()));  
 59 pstm.executeUpdate();  
 60   
 61 pstm.close();  
 62   
 63 con.close();  
 64 }  
 65 catch(SQLException e)  
 66 {  
 67 throw e;  
 68 }  
 69 }  
 70 public static void *eliminarProducto*(String id) throws SQLException  
 71 {  
 72 Connection con = (Connection) Extras.Connection\_class.*mysql\_connect*();  
 73 try  
 74 {  
 75 PreparedStatement pstm = (PreparedStatement) con.prepareStatement("DELETE FROM productos WHERE productos.producto\_id = ?");  
 76 pstm.setInt(1, Integer.*parseInt*(id));  
 77   
 78 System.*out*.println(pstm);  
 79   
 80 pstm.executeUpdate();  
 81 pstm.close();  
 82 con.close();  
 83 }  
 84 catch(SQLException e)  
 85 {  
 86 throw e;  
 87 }  
 88 }  
 89   
 90 //Retrona un producto en base al id ingresado  
 91 public static Producto *getProducto*(String id){  
 92 Producto producto = null;  
 93 PreparedStatement pstm;  
 94 ResultSet res;  
 95 try {  
 96   
 97 Class.*forName*("com.mysql.jdbc.Driver");  
 98 java.sql.Connection con = Extras.Connection\_class.*mysql\_connect*();  
 99   
100 String query = "SELECT \* FROM productos WHERE producto\_id='"+id+"';";  
101   
102   
103 System.*out*.println(query);  
104   
105 pstm = con.prepareStatement(query);  
106 res = pstm.executeQuery();  
107   
108 //Si encuentra el producto, crea un nuevo objeto con sus datos  
109 if(res.next()){  
110 producto = new Producto();  
111 producto.setId(res.getInt("producto\_id"));  
112 producto.setNombre(res.getString("nombre"));  
113 producto.setDescripcion(res.getString("descripcion"));  
114 producto.setPresentacion(res.getString("presentacion"));  
115 producto.setImagen(res.getString("imagen"));  
116 producto.setPrecio(res.getDouble("precio"));  
117 producto.setEnOferta(res.getInt("en\_oferta"));  
118   
119 }else{  
120 System.*out*.println("NO ENCONTRO EL PRODUCTO");  
121 }   
122 } catch (Exception ex) {  
123 Logger.*getLogger*(Carrito.class.getName()).log(Level.*SEVERE*, null, ex);  
124 }  
125 return producto;  
126 }  
127   
128 //Retorna un arreglo de productos, si soloOfertas es verdadero  
129 //retorna solo los productos en oferta.  
130 public static ArrayList<Producto> *getListaProductos*(int soloOfertas){  
131   
132 ArrayList<Producto> listaProductos = new ArrayList<Producto>();  
133 Producto producto;  
134 PreparedStatement pstm;  
135 ResultSet res;  
136 try {   
137 Class.*forName*("com.mysql.jdbc.Driver");  
138 java.sql.Connection con = Extras.Connection\_class.*mysql\_connect*();  
139 //Pasar a la capa de datos  
140 if(soloOfertas != 0){  
141 pstm = con.prepareStatement("SELECT \* FROM productos"  
142 + " WHERE productos.en\_oferta = 1 "  
143 );  
144 }else{  
145 pstm = con.prepareStatement("SELECT \* FROM productos");  
146 }  
147   
148 res = pstm.executeQuery();  
149   
150 //Si encuentra el producto, crea un nuevo objeto con sus datos  
151 while(res.next()){  
152 //System.out.println("ENCONTRO EL PRODUCTO");  
153 producto = new Producto();  
154 producto.setId(res.getInt("producto\_id"));  
155 producto.setNombre(res.getString("nombre"));  
156 producto.setDescripcion(res.getString("descripcion"));  
157 producto.setPresentacion(res.getString("presentacion"));  
158 producto.setPrecio(res.getDouble("precio"));  
159 producto.setImagen(res.getString("imagen"));  
160   
161 listaProductos.add(producto);  
162 }   
163   
164 } catch (Exception ex) {  
165   
166 Logger.*getLogger*(Carrito.class.getName()).log(Level.*SEVERE*, null, ex);  
167 }  
168 return listaProductos;  
169 }  
170   
171 //Retorna un arreglo de productos que coinciden con valorCadena  
172 public static ArrayList<Producto> *getListaProductos*(String valorCadena){  
173   
174 ArrayList<Producto> listaProductos = new ArrayList<Producto>();  
175 Producto producto;  
176 PreparedStatement pstm;  
177 ResultSet res;  
178 try {   
179 Class.*forName*("com.mysql.jdbc.Driver");  
180 java.sql.Connection con = Extras.Connection\_class.*mysql\_connect*();  
181 //Pasa a la capa de datos  
182 if(valorCadena != null){  
183 pstm = con.prepareStatement("SELECT \* FROM productos"+  
184 " WHERE (productos.nombre like '%"+valorCadena+"%' OR productos.descripcion like '%"+valorCadena+"%')");  
185 }else{  
186 pstm = con.prepareStatement("SELECT \* FROM productos");  
187 }  
188 res = pstm.executeQuery();  
189 //Si encuentra el producto, crea un nuevo objeto con sus datos  
190 while(res.next()){  
191 producto = new Producto();  
192 producto.setId(res.getInt("producto\_id"));  
193 producto.setNombre(res.getString("nombre"));  
194 producto.setDescripcion(res.getString("descripcion"));  
195 producto.setPresentacion(res.getString("presentacion"));  
196 producto.setPrecio(res.getDouble("precio"));  
197 producto.setImagen(res.getString("imagen"));  
198   
199 listaProductos.add(producto);  
200 }  
201 } catch (Exception ex) {  
202 Logger.*getLogger*(Carrito.class.getName()).log(Level.*SEVERE*, null, ex);  
203 }  
204 return listaProductos;  
205 }  
206 }  
207

### 3.4.5- Modelos

#### Final-Java-1\src\java\Classes\Producto.java

1 /\*  
 2 \* To change this template, choose Tools | Templates  
 3 \* and open the template in the editor.  
 4 \*/  
 5 package Classes;  
 6   
 7 /\*\*  
 8 \*  
 9 \* @author Pablo  
10 \*/  
11 public class Producto {  
12 private int id;  
13 private String nombre;  
14 private String descripcion;  
15 private String presentacion;  
16 private int en\_oferta;  
17 private int cantidad;  
18 private String imagen;  
19 private Double precio = 0.0;  
20   
21 public int getId() {  
22 return id;  
23 }  
24 public void setId(int id) {  
25 this.id = id;  
26 }  
27 public void setPrecio(Double precio) {  
28 this.precio = precio;  
29 }  
30 public String getNombre() {  
31 return nombre;  
32 }  
33 public void setNombre(String nombre) {  
34 this.nombre = nombre;  
35 }  
36 public String getDescripcion() {  
37 return descripcion;  
38 }  
39 public void setDescripcion(String description) {  
40 this.descripcion = description;  
41 }  
42 public String getPresentacion() {  
43 return presentacion;  
44 }  
45 public void setPresentacion(String presentation) {  
46 this.presentacion = presentation;  
47 }  
48 public int getEnOferta() {  
49 return en\_oferta;  
50 }  
51 public Double getPrecio() {  
52 return precio;  
53 }  
54 public void setEnOferta(int promoted) {  
55 this.en\_oferta = promoted;  
56 }  
57   
58 public void setCantidad(int cantidad){  
59 this.cantidad = cantidad;  
60 }  
61   
62 public int getCanidad(){  
63 return cantidad;  
64 }  
65   
66 public void setImagen(String imagen){  
67 this.imagen = imagen;  
68 }  
69   
70 public String getImagen(){  
71 return imagen;  
72 }  
73   
74 }

#### Final-Java-1\src\java\Classes\Pedido.java

1 /\*  
 2 \* To change this template, choose Tools | Templates  
 3 \* and open the template in the editor.  
 4 \*/  
 5 package Classes;  
 6   
 7 import java.sql.Date;  
 8 import java.util.ArrayList;  
 9   
10 /\*\*  
11 \*  
12 \* **@author** Matias  
13 \*/  
14 public class **Pedido** {  
15   
16 private int id;  
17 private int cliente\_id;  
18 private ArrayList<Producto> listaProductos;  
19 double total = 0;  
20 String fecha;  
21   
22 public int **getClienteId**() {  
23 return cliente\_id;  
24 }  
25 public int **getId**() {  
26 return id;  
27 }  
28   
29 public void **setListaProductos**(ArrayList<Producto> listaProductos){  
30 this.listaProductos = listaProductos;  
31 }  
32   
33 public ArrayList<Producto> **getListaProductos**(){  
34 return listaProductos;  
35 }  
36   
37 public void **setClienteId**(int cliente\_id){  
38 this.cliente\_id = cliente\_id;  
39 }  
40   
41 public void **setId**(int id){  
42 this.id = id;  
43 }  
44   
45 public void **setTotal**(double total){  
46 this.total = total;  
47 }  
48   
49 public double **getTotal**(){  
50 return total;  
51 }  
52   
53   
54 public double **calcularTotal**(){  
55 total = 0;  
56 for (Producto producto : listaProductos){  
57 total = total + (producto.getPrecio() \* producto.getCanidad());  
58 }  
59 return total;  
60 }  
61   
62 public void **setFecha**(String fecha){  
63 this.fecha = fecha;  
64 }  
65   
66 public String **getFecha**(){  
67 String year = this.fecha.substring(0, 4);  
68 String month = this.fecha.substring(5, 7);  
69 String day = this.fecha.substring(8, 10);  
70 String formated\_fecha = day + "/" + month + "/" + year;  
71   
72 return formated\_fecha;  
73 }  
74   
75 public String **getFechaSQL**(){  
76 return fecha;  
77 }  
78 }

### 3.4.6- Servlets

#### Final-Java-1\src\java\Extras\ArmarPedido.java

1 /\*  
 2 \* To change this template, choose Tools | Templates  
 3 \* and open the template in the editor.  
 4 \*/  
 5 package Extras;  
 6   
 7 import Classes.Cliente;  
 8 import Classes.Pedido;  
 9 import Classes.Producto;  
10 import Datos.Pedido\_Datos;  
11 import java.io.IOException;  
12 import java.sql.SQLException;  
13 import java.util.ArrayList;  
14 import java.util.Calendar;  
15 import java.util.logging.Level;  
16 import java.util.logging.Logger;  
17 import javax.servlet.ServletException;  
18 import javax.servlet.annotation.WebServlet;  
19 import javax.servlet.http.HttpServlet;  
20 import javax.servlet.http.HttpServletRequest;  
21 import javax.servlet.http.HttpServletResponse;  
22 import javax.servlet.http.HttpSession;  
23   
24 /\*\*  
25 \*  
26 \* @author Pablo  
27 \*/  
28 @WebServlet(name = "ArmarPedido", urlPatterns = {"/ArmarPedido"})  
29 public class ArmarPedido extends HttpServlet {  
30   
31 /\*\*  
32 \* Handles the HTTP  
33 \* <code>POST</code> method.  
34 \*  
35 \* @param request servlet request  
36 \* @param response servlet response  
37 \* @throws ServletException if a servlet-specific error occurs  
38 \* @throws IOException if an I/O error occurs  
39 \*/  
40 @Override  
41 protected void doPost(HttpServletRequest request, HttpServletResponse response)  
42 throws ServletException, IOException {  
43 HttpSession sesion = request.getSession();  
44   
45 Calendar cal1 = Calendar.*getInstance*();  
46 System.*out*.println(""+cal1.get(Calendar.*DATE*)+"/"+cal1.get(Calendar.*MONTH*)  
47 +"/"+cal1.get(Calendar.*YEAR*)+" "+cal1.get(Calendar.*HOUR*)  
48 +":"+cal1.get(Calendar.*MINUTE*)+":"+cal1.get(Calendar.*SECOND*)  
49 +":"+cal1.get(Calendar.*MILLISECOND*));   
50   
51   
52 Pedido pedido = new Pedido();  
53 pedido.setListaProductos((ArrayList<Producto>)sesion.getAttribute("listaProductos"));  
54 Cliente cliente = (Cliente)sesion.getAttribute("cliente");  
55 pedido.setClienteId(cliente.getId());  
56 pedido.setFecha(cal1.get(Calendar.*YEAR*)+"-"+cal1.get(Calendar.*MONTH*)+"-"+cal1.get(Calendar.*DATE*));  
57 double valorTotal = pedido.calcularTotal();  
58 System.*out*.println("Monto total: "+valorTotal);  
59   
60 try {  
61 Pedido\_Datos.*nuevoPedido*(pedido);  
62 sesion.setAttribute("listaProductos", new ArrayList<Producto>());  
63 sesion.setAttribute("pedido", pedido);  
64   
65 } catch (SQLException ex) {  
66 Logger.*getLogger*(ArmarPedido.class.getName()).log(Level.*SEVERE*, null, ex);  
67 }  
68   
69 response.sendRedirect("pedido.jsp");  
70   
71 }  
72   
73 }  
74

#### Final-Java-1\src\java\Extras\Carrito.java

1 /\*  
 2 \* To change this template, choose Tools | Templates  
 3 \* and open the template in the editor.  
 4 \*/  
 5 package Extras;  
 6   
 7 import Classes.Producto;  
 8 import Datos.Producto\_Datos;  
 9 import java.io.IOException;  
10 import java.sql.ResultSet;  
11 import java.util.ArrayList;  
12 import javax.servlet.ServletException;  
13 import javax.servlet.annotation.WebServlet;  
14 import javax.servlet.http.HttpServlet;  
15 import javax.servlet.http.HttpServletRequest;  
16 import javax.servlet.http.HttpServletResponse;  
17 import javax.servlet.http.HttpSession;  
18   
19 /\*\*  
20 \* **Agrega** **un** **producto** **al** **carrito.**  
21 \* **@author** Pablo  
22 \*/  
23 @WebServlet(name = "Carrito", urlPatterns = {"/Carrito"})  
24 public class **Carrito** extends HttpServlet {  
25   
26 /\*\*  
27 \* **Handles** **the** **HTTP**  
28 \* **<code>POST</code>** **method.**  
29 \*  
30 \* **@param** request servlet request  
31 \* **@param** response servlet response  
32 \* **@throws** ServletException if a servlet-specific error occurs  
33 \* **@throws** IOException if an I/O error occurs  
34 \*/  
35 @Override  
36 protected void **doPost**(HttpServletRequest request, HttpServletResponse response)  
37 throws ServletException, IOException {  
38   
39 HttpSession sesion = request.getSession();  
40 String producto\_id = request.getParameter ("producto\_id");  
41 int cantidad = Integer.*parseInt*(request.getParameter ("cantidad"));  
42 ResultSet res = null;  
43 ArrayList<Producto> listaProductos;  
44 Producto producto;  
45   
46 if(sesion.getAttribute("cliente")!=null){  
47   
48 //Obtengo el producto  
49 producto = Producto\_Datos.*getProducto*(producto\_id);  
50 if(producto!=null){  
51 //Establezco la cantidad que quiere el cliente  
52 producto.setCantidad(cantidad);  
53 //busco si esta creada una lista de productos en la sesion  
54 if(sesion.getAttribute("listaProductos") != null ){  
55 //Apunto a la lista de la session  
56 listaProductos = (ArrayList<Producto>) sesion.getAttribute("listaProductos");  
57 }else{  
58 //Genero la lista de productos  
59 listaProductos = new ArrayList<Producto>();  
60 }  
61 //Guardo el producto en la lista, y la lista en la session  
62 listaProductos.add(producto);  
63 sesion.setAttribute("listaProductos", listaProductos);  
64   
65 response.sendRedirect("carrito.jsp");  
66 }else{  
67 response.sendRedirect("index.jsp");  
68 }  
69 }else{  
70 response.sendRedirect("index.jsp");  
71 }  
72 }  
73   
74 }  
75

### 3.4.7- Scripts

#### Final-Java-1\web\includes\menu.jsp

1 <%@page import="Classes.Cliente"%>  
 2 <%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8" session="true"%>  
 3   
 4 <div class="navbar">  
 5 <div class="well" style="padding: 5px;">  
 6 <div class="logo-portada"><img style="width: 80px;float: left;" src="images/logo-portada.jpg"/></div>  
 7 <div style="float: right; margin-top: 20px;">  
 8 <%  
 9 HttpSession sesion = request.getSession(false);  
 10 if(sesion.getAttribute("cliente")!=null){  
 11 %>  
 12 <a href="editar.jsp"><i class="icon-user"></i>  
 13 <%  
 14 out.println(sesion.getAttribute("usuario"));  
 15 %>  
 16 </a><a href="cerrar.jsp">Cerrar sesi&oacute;n</a></h3>  
 17 <%  
 18 }  
 19 else{  
 20 %>  
 21 <form action="login.jsp" class=" form-inline" style="margin: 0 0 0 0">  
 22 <input type="text" class="input-small" name="usuario" size="15" tabindex="1" maxlength="20" placeholder="Usuario" class="control">  
 23 <input type="password" class="input-small" name="password" size="15" tabindex="2" maxlength="20" placeholder="Contraseña" class="control"/>  
 24 <label class="checkbox">  
 25 <input type="checkbox"> Recordar  
 26 </label>  
 27 <button type="submit" name="submit" class="btn">Entrar</button>  
 28 </form>  
 29 <%  
 30 }   
 31 %>  
 32 </div>  
 33 <div style="clear: both"></div>  
 34 </div>  
 35 </div>  
 36   
 37   
 38 <div class="navbar">  
 39 <div class="navbar-inner">  
 40 <a href="index.jsp"><div class="logo"></div></a>  
 41 <div class="menu">  
 42 <ul class="nav">  
 43 <li>  
 44 <a tabindex="-1" href="index.jsp"><i class="icon-home"></i> Inicio</a>  
 45 </li>  
 46 <%  
 47 if(sesion.getAttribute("cliente")==null){  
 48 %>  
 49 <li>  
 50 <a tabindex="-1" href="register.jsp"><i class="icon-pencil"></i> Registrarse</a>  
 51 </li>  
 52 <%  
 53 }   
 54 %>  
 55 <li>  
 56 <a tabindex="-1" href="#" class="dropdown-toggle" data-toggle="dropdown" ><i class="icon-shopping-cart"></i> Productos</a>  
 57 <ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu">  
 58 <li>  
 59 <a href="ofertas.jsp"><div class="titulos\_submenu"> Ofertas</div></a>  
 60 </li>  
 61 <li>  
 62 <a href="catalogo.jsp"><div class="titulos\_submenu"> Todos</div></a>  
 63 </li>  
 64 </ul>  
 65 </li>  
 66 <%  
 67 if(sesion.getAttribute("cliente")!=null){  
 68 if(sesion.getAttribute("usuario").toString().equals("administrador")){  
 69 %>  
 70 <li><a tabindex="-1" href="#" class="dropdown-toggle" data-toggle="dropdown"><i class="icon-user"></i> Administrador</a>  
 71 <ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu">  
 72 <li>  
 73 <a href="nuevoProducto.jsp"><div class="titulos\_submenu">Nuevo Producto</div></a>  
 74 </li>  
 75 <li>  
 76 <a href="pedidosRealizados.jsp"><div class="titulos\_submenu">Pedidos</div></a>  
 77 </li>  
 78 <li>  
 79 <a href="clientes.jsp"><div class="titulos\_submenu">Clientes</div></a>  
 80 </li>  
 81 </ul>  
 82 </li>  
 83 <%  
 84 }else{   
 85 %>  
 86 <li>  
 87 <a tabindex="-1" href="#" class="dropdown-toggle" data-toggle="dropdown"><i class="icon-user"></i> Usuario</a>  
 88 <ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu">  
 89 <li>  
 90 <a href="carrito.jsp"><div class="titulos\_submenu">Mi carrito</div></a>  
 91 </li>  
 92 <li>  
 93 <a href="editar.jsp"><div class="titulos\_submenu">Mi información</div></a>  
 94 </li>  
 95 <li>  
 96 <a href="listaPedidos.jsp"><div class="titulos\_submenu">Mis pedidos</div></a>  
 97 </li>  
 98 </ul>  
 99 </li>  
100 <%  
101 }  
102 }  
103 %>  
104 <li>  
105 <a tabindex="-1" href="contacto.jsp"><i class="icon-envelope"></i> Contacto</a>  
106 </li>  
107 </ul>  
108 <ul class="nav" style="float: right">  
109 <li>  
110 <form action="index.jsp">  
111 <input class="span2" type="text" name="string" placeholder="Buscar" style=" margin: 4px 0px 4px 0px;">  
112 <button class="btn" type="submit" style="margin: 4px 0px 4px 0px;">Buscar</button>  
113 </form>  
114 </li>  
115 </ul>  
116 </div>  
117 </div>  
118 </div>

#### Final-Java-1\web\actualizarProducto.jsp

1 **<%--**   
 2  **Document : actualizarProducto**  
 3  **Created on : 15/06/2013, 20:45:19**  
 4  **Author : Pablo**  
 5 **--%>**  
 6   
 7 **<%@page** import**=**"Classes.Producto"**%>**  
 8 **<%@page** import**=**"Datos.Producto\_Datos"**%>**  
 9 <%   
10 Producto producto = new Producto();  
11 producto.setNombre(request.getParameter("nombre").toString());  
12 producto.setDescripcion(request.getParameter("descripcion").toString());  
13 producto.setPresentacion(request.getParameter("presentacion").toString());  
14 producto.setId(Integer.*parseInt*(request.getParameter("id")));  
15 producto.setPrecio(Double.*parseDouble*(request.getParameter("precio")));  
16 producto.setEnOferta(Integer.*parseInt*(request.getParameter("en\_oferta")));  
17   
18 Producto\_Datos.*editarProducto*(producto);  
19   
20 response.sendRedirect("producto.jsp?producto="+request.getParameter("id").toString());  
21   
22 %>

#### Final-Java-1\web\eliminarProducto.jsp

1 **<%--**   
 2  **Document : eliminarProducto.jsp**  
 3  **Created on : 19-05-2013, 20:12:10**  
 4  **Author : Matias**  
 5 **--%>**  
 6 **<%@page** import**=**"Datos.Producto\_Datos"**%>**  
 7   
 8 <%   
 9 Producto\_Datos.*eliminarProducto*(request.getParameter("producto"));  
10 response.sendRedirect("index.jsp");  
11 %>

## 3.5- Tecnologías y librerías adicionales

Para desarrollo del front end o vistas utilizamos librerías externas las cuales se describen a continuación.

Para el desarrollo de estilos con CSS utilizamos la famosa librería **bootstrap**[[1]](#footnote-1) la cual nos ayudó para acelerar el desarrollo de esta parte del sistema.

Para desarrollo de validación de forms, si bien utilizamos javascript propio, utilizamos la librería **jquery**[[2]](#footnote-2) la cual también nos ayuda de forma notable en el desarrollo de script del lado del cliente.

En los dos casos, para situaciones particulares, generamos nuestros propios archivos con código **CSS** y **javascript**.

Para validación de formularios adicionalmente utilizamos las funcionalidades que ofrece hoy **HTML5**[[3]](#footnote-3) el cual permite definir tipos de datos a ingresar sin necesidad de analizarlos con otro lenguaje.

1. <http://getbootstrap.com/> [↑](#footnote-ref-1)
2. <http://jquery.com/> [↑](#footnote-ref-2)
3. <http://www.w3.org/TR/html5/> [↑](#footnote-ref-3)