**Laboratorio 1 – Arquitectura de Software**



**Autores:**

María Isabel Martínez Rendón

Raúl Antonio Martínez Silgado

Neyder Leoncio Daza Cardona

Jean Carlos Herrera Meza

Sebastián Montoya

**Profesor:**

Diego Jose Luis Botia

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS

FACULTAD DE INGENIERÍA

2016

**Tabla de contenido**

1. Introducción………………………………………………………………………… 3
2. Objetivos…………………………………………………………………………….. 4
   1. Objetivo general…………………………………………………………… 4
   2. Objetivos específicos…………………………………………………….. 4
3. Marco teórico……………………………………………………………………….. 5
   1. Anotaciones utilizadas…………………………………………………... 6
4. Herramientas empleadas…………………………………………………………. 9
5. Modelo propuesto…………………………………………………………………..10
6. Arquitectura propuesta …………………………………………………………… 11
7. Procedimiento…………………………………………………………………….... 12

6.1 Clases del Modelo…………………………………………………………….. 16

6.1.1 Cliente.Java…………………………………………………………..16

6.1.2 Vehiculo.Java………………………………………………………...18

6.1.3 Vendedor.Java……………………………………………………….20

6.1.4 Ventas\_generales.Java……………………………………………. 22

6.2 Clases del DAO………………………………………………………………... 29

6.2.1 ClienteDAO.Java……………………………………………………. 29

6.2.2 VehiculoDAO.Java………………………………………………….. 30

6.2.3 VendedorDAO.Java………………………………………………… 31

6.2.4 VentasDAO.Java…………………………………………...……….. 32

6.3 Clases del Controlador……………………………………………………….. 34

6.3.1 ClienteServlet.Java…………………………………………………. 34

6.3.2 VehiculoServlet.Java………………………………………….…… 37

6.3.3 VendedorServlet.Java……………………………………………... 40

6.3.4 VentasServlet.Java…………………………………………….…… 42

6.4 Vistas…………………………………………………………………………….. 45

6.4.1 Cliente.jsp……………………………………………………………. 45

6.4.2 Vehiculo.jsp………………………………………………………….. 48

* + 1. Vendedor.jsp……………………………………………………… 51

6.4.4 Ventas.jsp………………………………………………………….. 53

1. Imágenes de la aplicación……………………………………………………….. 56

7.1 Cliente………………………………………………………………………… 56

7.2 Vehículo…………………………………………………………………………57

7.3 Vendedor……………………………………………………………………….. 57

7.4 Ventas………………………………………………………………………....…58

1. Conclusiones………………………………………………………………………. 59
2. Bibliografía………………………………………………………………………….. 60

**INTRODUCCIÓN**

Se ha propuesto implementar una aplicación de tipo CRUD (Create, Read, Update, Delete), que permita manejar información de un concesionario de vehículos; para ello, se tiene pensado montar una base de datos (desarrollada en MySQL), que con la ayuda de Servlets, nos permitan realizar estas operaciones.

A partir de los conocimientos adquiridos, nos soportaremos en ayudas como la de JDBC, que permitirá crear una conexión a la base de datos.

Se busca que un controlador permita registrar o mostrar datos (correspondientes al programa) y que mediante vistas (JSP) el usuario pueda interactuar con la aplicación de una manera fácil (Modelo request-response). Con la ayuda del EJB, se buscará añadir servicios y funcionalidades que mejoren la calidad de la aplicación. De igual manera, con el pool de conexiones se pretende establecer una mejor relación entre la base de datos y el servidor.

Con todas estas herramientas (pool de conexiones, EJB, JSP, JDBC, anotaciones, etc). El objetivo será encontrar una solución óptima para el concesionario, el cual permitirá registrar vehículos, vendedores, clientes; buscar, eliminar y modificar tanto clientes, como vehículos e inclusive realizar ventas y agregar foto de los vehículos.

**OBJETIVOS**

**Objetivo General**

Crear una aplicación CRUD para un concesionario utilizando pool de conexiones, recurso JDBC, anotaciones JPA, EJB, unidad de persistencia, Servlets y JSP.

**Objetivos específicos**

− Comprender el uso de los recursos JDBC y pool de conexiones para el manejo de conexiones y concurrencia en una base de datos.

− Conocer y comprender las anotaciones JPA para simplificar la persistencia y el mapeo de una base de datos objeto-relacional.

− Comprender el uso de EJB-Session Beans: Stateful, Stateless y Singleton para la seguridad de una aplicación transaccional.

− Utilizar Servlets para la lógica del negocio y JSP para la vista del cliente.

− Conocer la utilidad de los objetos request y response que nos proporcionan los Servlets.

− Comprender el manejo de JSP para la creación de una aplicación web.

− Aplicar una arquitectura básica de 3 o N capas para el desarrollo de una aplicación web.

− Aplicar y entender el uso de los patrones de diseño DTO y DAO.

**MARCO TEÓRICO**

* **Servlet:** Componente Web que se ejecuta dentro de un contenedor web y genera contenido dinámico. Los Servlets son módulos escritos en Java que se utilizan en un servidor, que puede ser o no ser servidor web, para extender sus capacidades de respuesta a los clientes al utilizar las potencialidades de Java.
* **CRUD:** Es un acrónimo de: Create(crear), Read(obtener), Uptade(actualizar), Delete(eliminar). Esto nos indica, que es una aplicación que permite estas cuatro operaciones.
* **JSP:** (JavaServer Pages) es una tecnología Java que permite generar contenido dinámico para servidores web que lo soporten, en forma de documentos normalmente HTML ó XML.

JSP permite la utilización de código Java mediante scripts. Además, es posible utilizar algunas acciones JSP predefinidas mediante etiquetas. Estas etiquetas pueden ser enriquecidas mediante la utilización de Bibliotecas de Etiquetas (TagLibs o Tag Libraries) externas e incluso personalizadas.

* **Pool de conexiones:** Un pool de conexiones es un conjunto limitado de conexiones a una base de datos, que es manejado por un servidor de aplicaciones de forma tal, que dichas conexiones pueden ser reutilizadas por los diferentes usuarios. Este pool es administrado por un servidor de aplicaciones que va asignando las conexiones a medida que los clientes van solicitando consultas o actualizaciones de datos.
* **JDBC:** (Java Database Connectivity) es una API de Java para ejecutar sentencias SQL. Consta de un conjunto de clases e interfaces escrito en lenguaje de programación Java. Usando esta API es fácil enviar sentencias SQL a virtualmente cualquier base de datos relacional. En otras palabras, no es necesario escribir un programa para acceder a una base de datos tipo Access, otro programa para acceder a una base de datos tipo Oracle y así para cada tipo de base de datos. Uno puede escribir un solo programa usando la API JDBC y el programa será capaz de enviar sentencias SQL a la base de datos apropiada.
* **JPA:** (Java Persistence API) es un conjunto de clases y métodos que persistentemente almacenar la gran cantidad de datos a una base de datos que es proporcionada por Oracle Corporation.
* **POJO (Plain Old Java Object):** Un POJO es una instancia de una clase que no extiende ni implementa nada en especial. Por ejemplo, un Servlet. Un Servlet tiene que extender de HttpServlet, por lo tanto no es un POJO. En cambio, si se define una clase JugadorDTO con atributos y unas cuantas operaciones, se tiene un POJO.
* **Anotaciones:** Es una forma de añadir metadatos al código fuente Java que están disponibles para la aplicación en tiempo de ejecución. Muchas veces se usa como una alternativa a la tecnología XML.

Las Anotaciones pueden añadirse a los elementos de programa tales como clases, métodos, campos, parámetros, variables locales, y paquetes.

* **EJB:** (Enterprise Java Bean) es un componente que debe ejecutarse de un contenedor de EJBs y se diferencia bastante de un JavaBean normal. Un JavaBean es un objeto Java al cual accedemos de forma directa desde nuestro programa.
* **DTO:** (Objeto de transferencia de datos) se utiliza para encapsular los datos, y enviarlo desde un subsistema de una aplicación a otra**.**

DTO’s son los más utilizados por la capa de servicios en una aplicación de n niveles para transferir datos entre sí y la capa de interfaz de usuario. El beneficio principal aquí es que reduce la cantidad de datos que necesita ser enviada a través del cable en aplicaciones distribuidas.

* **DAO:** (Objeto de Acceso a Datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.

Anotaciones utilizadas:

* **(At-rules):** Son un conjunto de reglas especiales que comienzan por el carácter @. Su función es englobar o llamar a una serie de reglas que actuarán conjuntamente en una determinada circunstancia. Unos ejemplos: *@font-face, @import, @page, @keyframes.*
* **@WebServlet:** Anotación utiliza para declarar un servlet.

Esta anotación es procesada por el contenedor durante el despliegue, y el servlet correspondiente que se disponga en los patrones de URL especificados.

* **@Entity:** Especifica que voy a crear una entidad. Se coloca al inicio de la definición de la clase. Es usada, junto con un “EntityManager”, para crear, persistir y combinar datos de una base de datos.
* **@EJB:** Con esta anotación se pueden inyectar los EJB construidos, en otros objetos gestionados por el servidor, como por ejemplo los servlets. El objeto en el que se inyecta el EJB se llama cliente. En nuestro caso, por ejemplo, los DAOs se inyectan en los servlets por medio de ésta.
* **@Override:** Indica que un método está sobrecargado. La sobrecarga de métodos es la creación de varios métodos con el mismo nombre pero con diferentes firmas (que es la combinación del tipo de dato que regresa, su nombre y su lista de argumentos) y definiciones. Java utiliza el número y tipo de argumentos para seleccionar cuál definición de método ejecutar. Java diferencia los métodos sobrecargados con base en el número y tipo de argumentos que tiene el método y no por el tipo que devuelve.
* **@MultipartConfig:** Es necesario usar la anotación *@MultipartConfig(maxFileSize = 16177215)* en los Servlet que trabajen con imágenes, esto para indicar que se va a trabajar con datos de tipo Medium Blob, dándole por medio del parámetro maxFileSize, el tamaño máximo del archivo.
* **@Stateless**: Indica que el Bean de sesión carece de estado. Un bean de sesión sin estado no mantiene un estado de conversación con el cliente. Cuando un cliente invoca los métodos de un bean sin estado, variables de instancia del bean pueden contener un estado específico a ese cliente, pero sólo durante la duración de la invocación. Cuando se termina el procedimiento, el estado específico del cliente no debe ser retenido. Los clientes pueden, sin embargo, cambiar el estado de las variables de instancia en los granos sin estado agrupados, y este estado se aplaza hasta la siguiente invocación del bean sin estado agrupado. Excepto durante la invocación del método, todas las instancias de un bean sin estado son equivalentes, permitiendo que el contenedor EJB para asignar una instancia de cualquier cliente. Es decir, el estado de un bean de sesión sin estado debe aplicarse en todos los clientes.
* **@PersistenceContext:** El @PersistenceContext es la anotación para la inyección del contexto de persistencia actual. Las entidades, pueden ser creadas, eliminadas o actualizadas por medio de un objeto “EntityManager”. Este objeto “EntityManager” es configurado por medio del archivo persistence.xml. El EntityManager es creado “automáticamente” utilizando la información que hay en el persistence.xml, por lo tanto, para utilizarlo, simplemente tenemos que solicitar que se inyecte en uno de nuestros componentes, y es aquí donde entra a ser útil esta anotación. Se escribe dentro de la clase y anterior a los atributos de la misma. Además instanciado, pero no inicializando (pues ya se inicializó con la información que toma del persistence.xml) un objeto de la clase “EntityManager”, cobra su funcionalidad.
* **@Local:** Indica que la interfaz es local. Es laa interfaz utilizada por los objetos de un módulo que acceden a los EJB, que están en la misma máquina Java.
* **@Table:** Especifica el nombre de la tabla principal relacionada con la entidad.
* **@NamedQueries:** Son consultas con nombres, para la organización de la aplicación. Donde el parámetro name es el nombre de la consulta, y el query es la consulta a la base de datos. Ej: @NamedQueries(@NamedQuery(name="Cliente.getAll",query="SELECT c FROM Cliente c"))
* **@Id:** Indica cual es la clave primaria y se anota antes de la creación de este atributo. Ej: private String cedula;
* **@GeneratedValue:** Asociado con la clave primaria, indica que ésta se debe generar por ejemplo con una secuencia de la base de datos.

strategy – estrategia a seguir para la generación de la clave: AUTO (valor por defecto, el contenedor decide la estrategia en función de la base de datos), IDENTITY (utiliza un contador, ej: MySQL), SEQUENCE (utiliza una secuencia, ej: Oracle, PostgreSQL) y TABLE (utiliza una tabla de identificadores).

* **@Column:** Especifica una columna de la tabla a mapear con un campo de la entidad.

**HERRAMIENTAS EMPLEADAS**

● NetBeans 8.0.1.

● Web Browser (Firefox, Chrome).

● Java EE 7 Web, plataforma de desarrollo.

● Java Servlets.

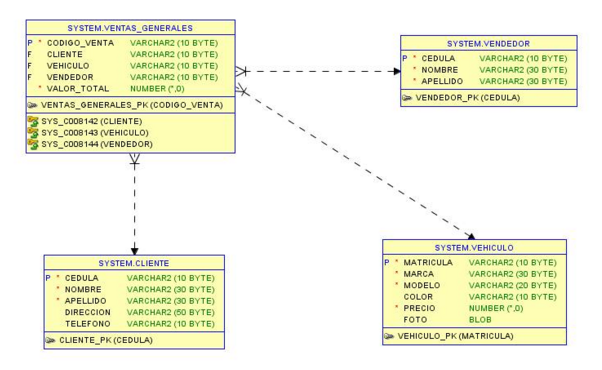
● JavaServer Pages.

● Enterprise Javabeans.

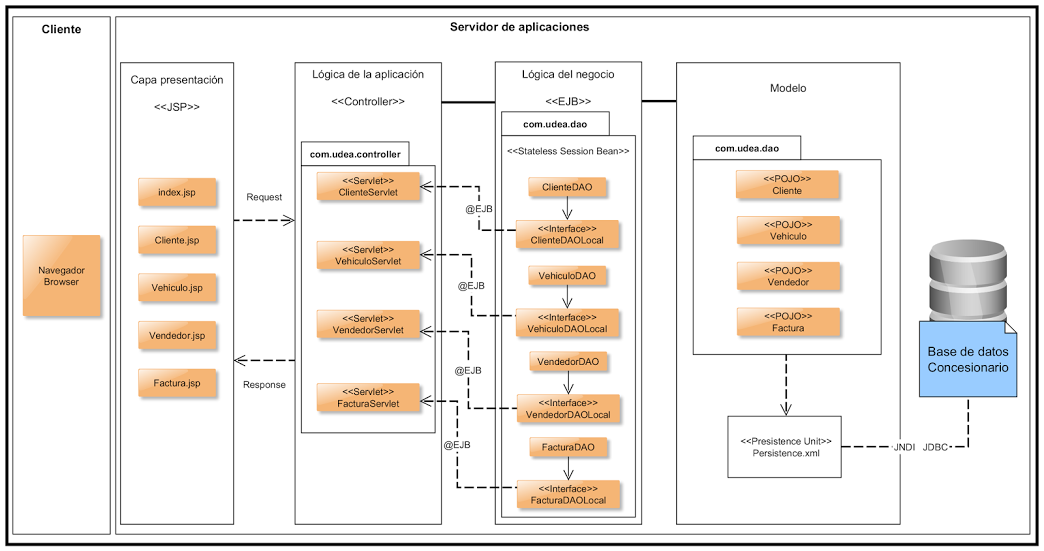
● GlassFish Server 4, servidor de aplicaciones.

● Mysql 5.0.

**MODELO PROPUESTO**



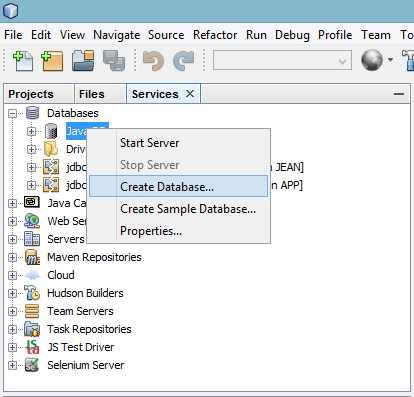
**ARQUITECTURA PROPUESTA**



**PROCEDIMIENTO**

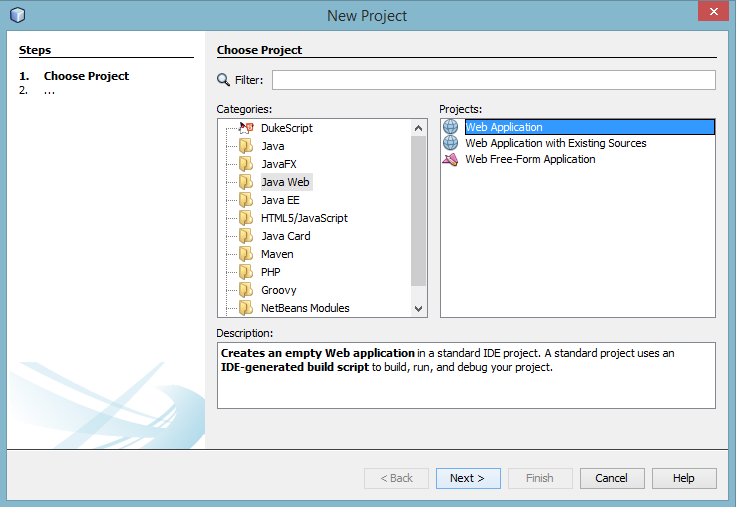
Lo primero es crear la base de datos. Para esto cargamos el netbeans, nos vamos a la pestaña de servicios, Database, desplegamos Java DB y procedemos a dar click derecho.

Creamos la base de datos y posteriormente la conectamos, dando click derecho en el nodo con el nombre de la base de datos creada. En nuestro caso la llamaremos concesionario.

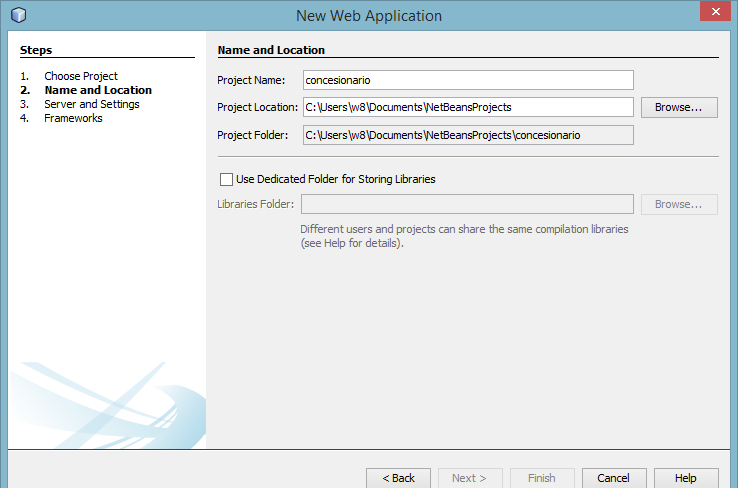


Se crean los respectivos campos relacionados a la tabla y se le asignan los nombres correspondientes.

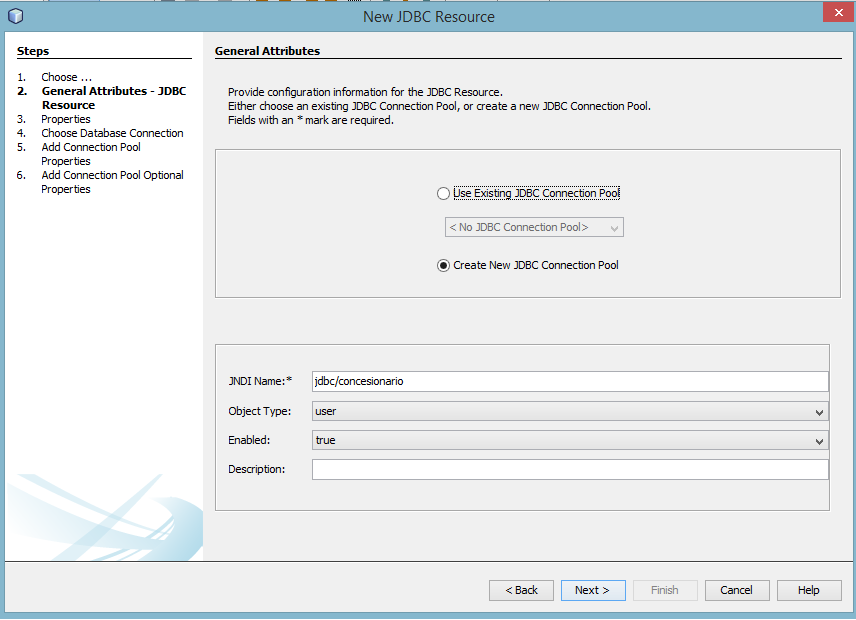
Para crear la aplicación web: Nos dirigimos hacia File, new Project, Java web y seleccionamos web aplication.



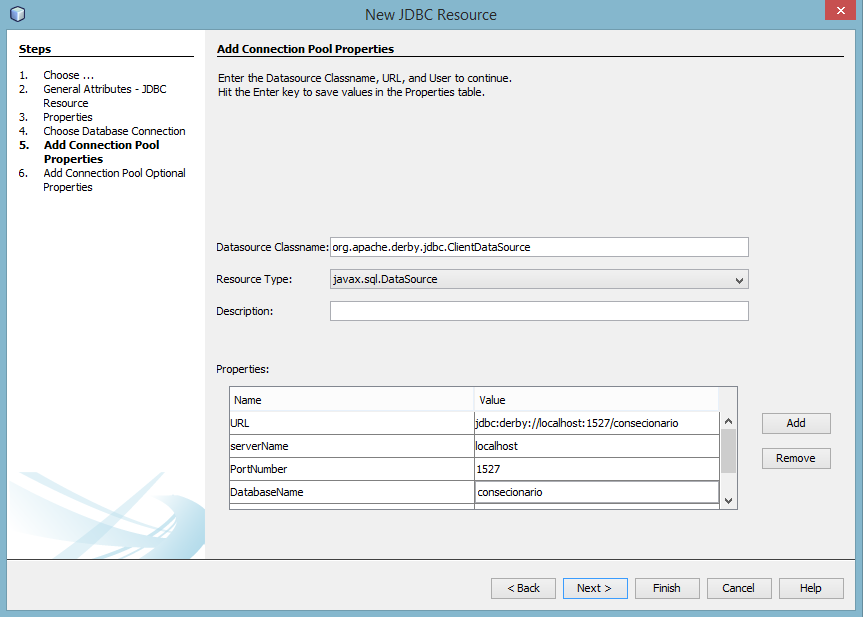
Introducimos el nombre que le asignaremos, en nuestro caso ‘concesionario’ y también la ruta de destino, escogemos el server (GlassFish Server) y Java EE version (Java EE 7 web), y para este proyecto no seleccionaremos ningún Framework.



Creamos el pool de conexiones: Le asignamos un nombre al JNDI

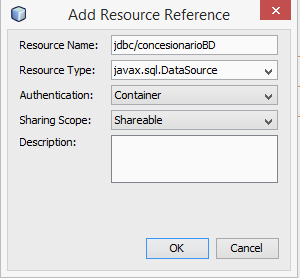


Se selecciona la conexión correspondiente y presionamos next



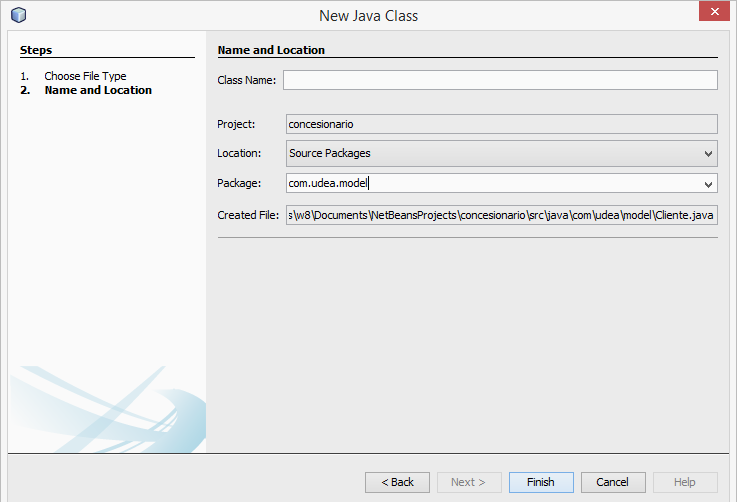
Se crea el archivo web.xml, click derecho sobre web pages, new, other, web y seleccionamos Standard Deployment Descriptor (web.xml).

En el archivo web.xml, nos vamos a la pestaña de Reference; en la parte de Resource Reference presionamos el botón Add.



A continuación procederemos a crear un Java Class, que permite crear los POJO’s. Con el click derecho sobre Source Packages, new, Java Class…

El nombre del paquete que utilizaremos será: com.udea.model



**Cliente.Java**

package com.udea.model;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.Table;

@Entity

@Table(name="cliente")

@NamedQueries(@NamedQuery(name="Cliente.getAll",query="SELECT c FROM Cliente c"))

public class Cliente implements Serializable{

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column

private String cedula;

@Column

private String Nombre;

@Column

private String Apellido;

@Column

private String Direccion;

@Column

private String Telefono;

public Cliente(String Cedula, String Nombre, String Apellido, String Direccion, String Telefono) {

this.cedula = Cedula;

this.Nombre = Nombre;

this.Apellido = Apellido;

this.Direccion = Direccion;

this.Telefono = Telefono;

}

public Cliente() {

}

public String getCedula() {

return cedula;

}

public void setCedula(String Cedula) {

this.cedula = Cedula;

}

public String getNombre() {

return Nombre;

}

public void setNombre(String Nombre) {

this.Nombre = Nombre;

}

public String getApellido() {

return Apellido;

}

public void setApellido(String Apellido) {

this.Apellido = Apellido;

}

public String getDireccion() {

return Direccion;

}

public void setDireccion(String Direccion) {

this.Direccion = Direccion;

}

public String getTelefono() {

return Telefono;

}

public void setTelefono(String Telefono) {

this.Telefono = Telefono;

}

}

**Vehiculo.Java**

package com.udea.model;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.Table;

@Entity

@Table(name="vehiculo")

@NamedQueries(@NamedQuery(name="Vehiculo.getAll",query="SELECT veh FROM Vehiculo veh"))

public class Vehiculo implements Serializable{

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column

private String matricula;

@Column

private String marca;

@Column

private String Modelo;

@Column

private String Color;

@Column

private float Precio;

@Column

private byte[] foto;

public Vehiculo(String Matricula, String Marca, String Modelo, String Color, float Precio, byte[] Foto) {

this.matricula = Matricula;

this.marca = Marca;

this.Modelo = Modelo;

this.Color = Color;

this.Precio = Precio;

this.foto = Foto;

}

public Vehiculo() {

}

public String getMatricula() {

return matricula;

}

public void setMatricula(String Matricula) {

this.matricula = Matricula;

}

public String getMarca() {

return marca;

}

public void setMarca(String Marca) {

this.marca = Marca;

}

public String getModelo() {

return Modelo;

}

public void setModelo(String Modelo) {

this.Modelo = Modelo;

}

public String getColor() {

return Color;

}

public void setColor(String Color) {

this.Color = Color;

}

public float getPrecio() {

return Precio;

}

public void setPrecio(float Precio) {

this.Precio = Precio;

}

public byte[] getFoto() {

return foto;

}

public void setFoto(byte[] Foto) {

this.foto = Foto;

}

}

**Vendedor.Java**

package com.udea.model;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.Table;

@Entity

@Table(name="vendedor")

@NamedQueries(@NamedQuery(name="Vendedor.getAll",query="SELECT v FROM Vendedor v"))

public class Vendedor implements Serializable{

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column

private String cedula;

@Column

private String Nombre;

@Column

private String Apellido;

public Vendedor(String Cedula, String Nombre, String Apellido) {

this.cedula = Cedula;

this.Nombre = Nombre;

this.Apellido = Apellido;

}

public Vendedor() {

}

public String getCedula() {

return cedula;

}

public void setCedula(String Cedula) {

this.cedula = Cedula;

}

public String getNombre() {

return Nombre;

}

public void setNombre(String Nombre) {

this.Nombre = Nombre;

}

public String getApellido() {

return Apellido;

}

public void setApellido(String Apellido) {

this.Apellido = Apellido;

}

}

**Ventas\_generales.Java**

package com.udea.model;

import java.io.Serializable;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.Table;

@Entity

@Table

@NamedQueries(@NamedQuery(name="Ventas.getAll",query="SELECT ven FROM Ventas\_generales ven"))

public class Ventas\_generales implements Serializable{

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column

private String codigo\_venta;

@Column

private String Cliente;

@Column

private String Vehiculo;

@Column

private String Vendedor;

@Column

private float valor\_total;

public Ventas\_generales(String Codigo\_venta, String Cliente, String Vehiculo, String Vendedor, float Valor\_total) {

this.codigo\_venta = Codigo\_venta;

this.Cliente = Cliente;

this.Vehiculo = Vehiculo;

this.Vendedor = Vendedor;

this.valor\_total = Valor\_total;

}

public Ventas\_generales() {

}

public String getCodigo\_venta() {

return codigo\_venta;

}

public void setCodigo\_venta(String Codigo\_venta) {

this.codigo\_venta = Codigo\_venta;

}

public String getCliente() {

return Cliente;

}

public void setCliente(String Cliente) {

this.Cliente = Cliente;

}

public String getVehiculo() {

return Vehiculo;

}

public void setVehiculo(String Vehiculo) {

this.Vehiculo = Vehiculo;

}

public String getVendedor() {

return Vendedor;

}

public void setVendedor(String Vendedor) {

this.Vendedor = Vendedor;

}

public float getValor\_total() {

return valor\_total;

}

public void setValor\_total(float Valor\_total) {

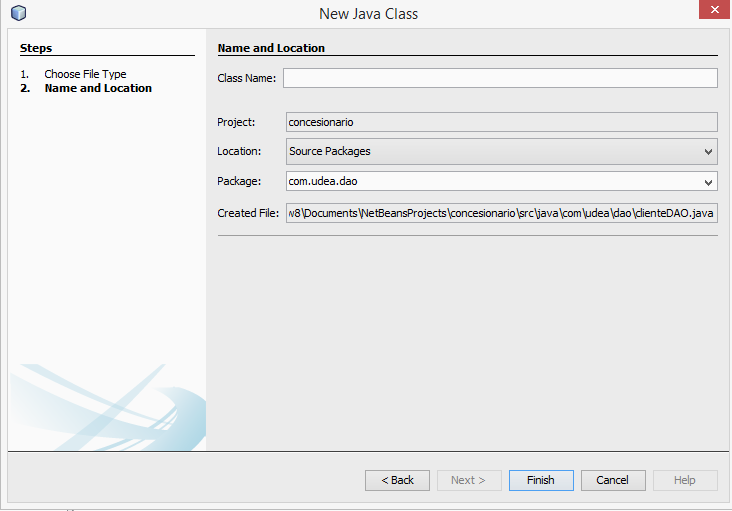
this.valor\_total = Valor\_total;

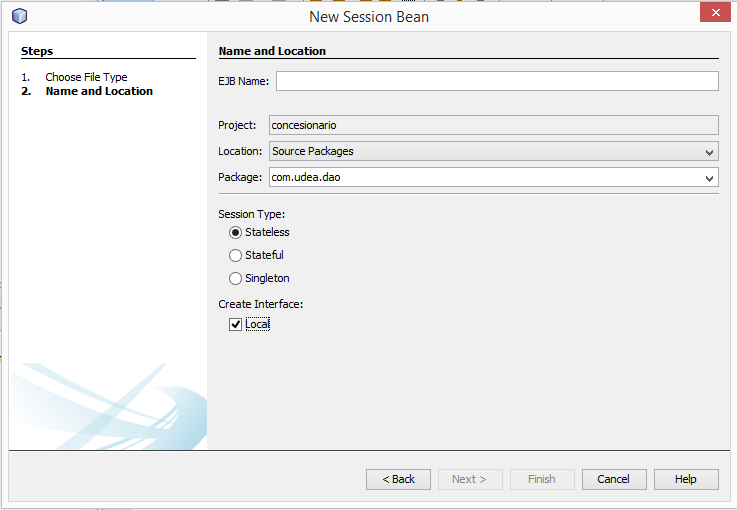
}

}

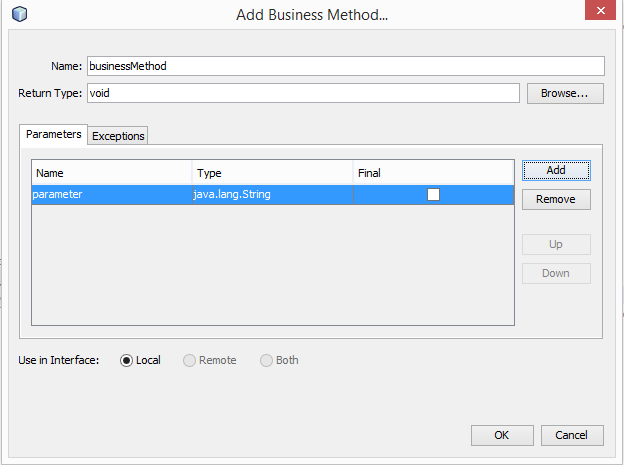
Se agregará un EJB de tipo Session Bean, que representará el patrón DAO de la capa lógica del negocio. Serán de tipo Stateless y con interfaz Local. Se ubicará en el paquete com.udea.dao

Para crearlo, le damos click derecho al proyecto, new, other, Enterprise JavaBean y luego Session Bean.





Se agregan los métodos del negocio a la clase



**ClienteDAO.Java**

@Stateless

public class ClienteDAO implements ClienteDAOLocal {

@Override

public void addCliente(Cliente cliente) {

}

@Override

public void editCliente(Cliente cliente) {

}

@Override

public void deleteCliente(String clienteID) {

}

@Override

public Cliente getCliente(String clienteID) {

return null;

}

@Override

public List<Cliente> getAllClientes() {

return null;

}

}

**VehiculoDAO.Java**

@Stateless

public class VehiculoDAO implements VehiculoDAOLocal {

@Override

public void addVehiculo(Vehiculo veh) {

}

@Override

public void editVehiculo(Vehiculo veh) {

}

@Override

public void deleteVehiculo(String vehID) {

}

@Override

public Vehiculo getVehiculo(String vehID) {

return null;

}

@Override

public List<Vehiculo> getAllVehiculos() {

return null;

}

}

**VendedorDAO.Java**

@Stateless

public class VendedorDAO implements VendedorDAOLocal {

@PersistenceContext

@Override

public void addVendedor(Vendedor ven) {

}

@Override

public void editVendedor(Vendedor ven) {

}

@Override

public void deleteVendedor(String venID) {

}

@Override

public Vendedor getVendedor(String venID) {

return null;

}

@Override

public List<Vendedor> getAllVendedores() {

return null;

}

}

**VentasDAO.Java**

@Stateless

public class VentasDAO implements VentasDAOLocal {

@Override

public void addVenta(Ventas\_generales venta) {

}

@Override

public void editVenta(Ventas\_generales venta) {

}

@Override

public void deleteVenta(String ventaID) {

}

@Override

public Ventas\_generales getVenta(String ventaID) {

return null;

}

@Override

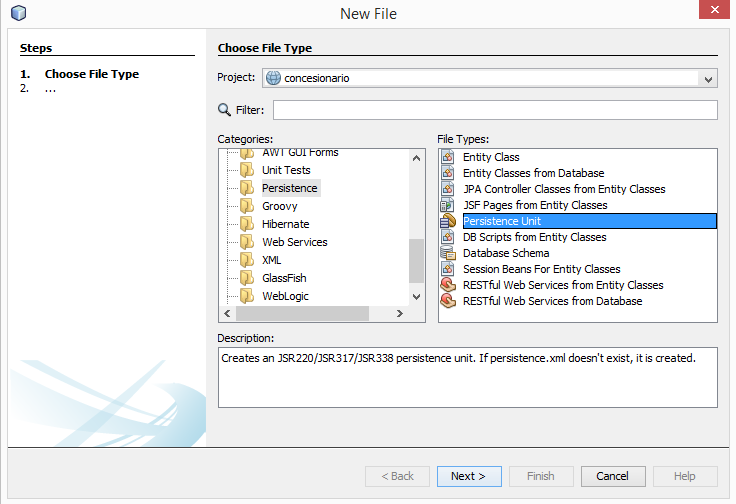
public List<Ventas\_generales> getAllVentas() {

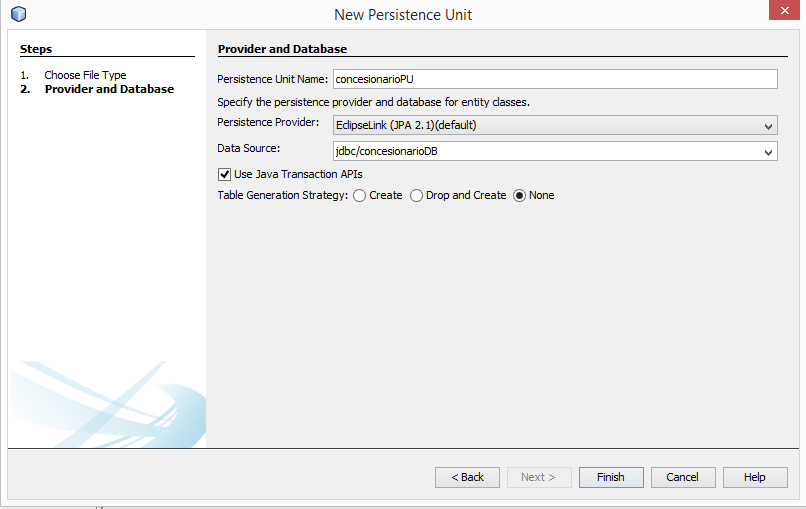
return null;

}

}

Posteriormente se agrega la unidad de persistencia. Click derecho en el proyecto, new, other, Persistence y Persistence Unit.





Para la Table Generation Strategy se debe seleccionar **None**, ya que la tabla en la base de datos ha sido creada previamente.

Se modifican las clases para que se agregue la funcionalidad de cada método del negocio de la clase Enitity Manager.

**ClienteDAO.Java**

package com.udea.dao;

import com.udea.model.Cliente;

import java.util.List;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

@Stateless

public class ClienteDAO implements ClienteDAOLocal {

@PersistenceContext

private EntityManager em;

@Override

public void addCliente(Cliente cliente) {

em.persist(cliente);

}

@Override

public void editCliente(Cliente cliente) {

em.merge(cliente);

}

@Override

public void deleteCliente(String clienteID) {

em.remove(getCliente(clienteID));

}

@Override

public Cliente getCliente(String clienteID) {

return em.find(Cliente.class, clienteID);

}

@Override

public List<Cliente> getAllClientes() {

return em.createNamedQuery("Cliente.getAll").getResultList();

}

}

**VehiculoDAO.Java**

package com.udea.dao;

import com.udea.model.Vehiculo;

import java.util.List;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

@Stateless

public class VehiculoDAO implements VehiculoDAOLocal {

@PersistenceContext

private EntityManager em;

@Override

public void addVehiculo(Vehiculo veh) {

em.persist(veh);

}

@Override

public void editVehiculo(Vehiculo veh) {

em.merge(veh);

}

@Override

public void deleteVehiculo(String vehID) {

em.remove(getVehiculo(vehID));

}

@Override

public Vehiculo getVehiculo(String vehID) {

return em.find(Vehiculo.class, vehID);

}

@Override

public List<Vehiculo> getAllVehiculos() {

return em.createNamedQuery("Vehiculo.getAll").getResultList();

}

}

**VendedorDAO.Java**

package com.udea.dao;

import com.udea.model.Vendedor;

import java.util.List;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

@Stateless

public class VendedorDAO implements VendedorDAOLocal {

@PersistenceContext

private EntityManager em;

@Override

public void addVendedor(Vendedor ven) {

em.persist(ven);

}

@Override

public void editVendedor(Vendedor ven) {

em.merge(ven);

}

@Override

public void deleteVendedor(String venID) {

em.remove(getVendedor(venID));

}

@Override

public Vendedor getVendedor(String venID) {

return em.find(Vendedor.class, venID);

}

@Override

public List<Vendedor> getAllVendedores() {

return em.createNamedQuery("Vendedor.getAll").getResultList();

}

}

**VentasDAO.Java**

package com.udea.dao;

import com.udea.model.Ventas\_generales;

import java.util.List;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

@Stateless

public class VentasDAO implements VentasDAOLocal {

@PersistenceContext

private EntityManager em;

@Override

public void addVenta(Ventas\_generales venta) {

em.persist(venta);

}

@Override

public void editVenta(Ventas\_generales venta) {

em.merge(venta);

}

@Override

public void deleteVenta(String ventaID) {

em.remove(getVenta(ventaID));

}

@Override

public Ventas\_generales getVenta(String ventaID) {

return em.find(Ventas\_generales.class, ventaID);

}

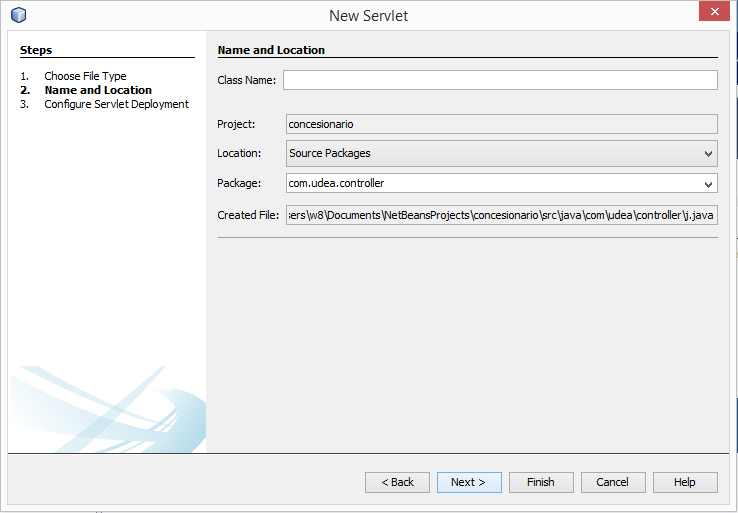
@Override

public List<Ventas\_generales> getAllVentas() {

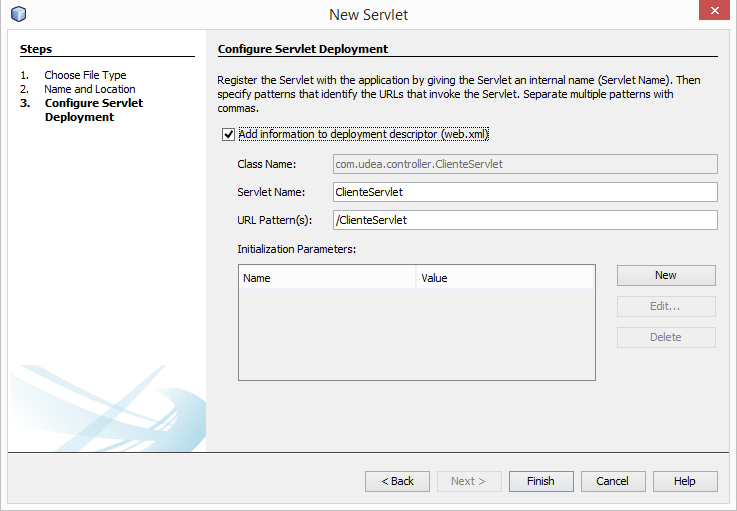
return em.createNamedQuery("Ventas.getAll").getResultList();

}

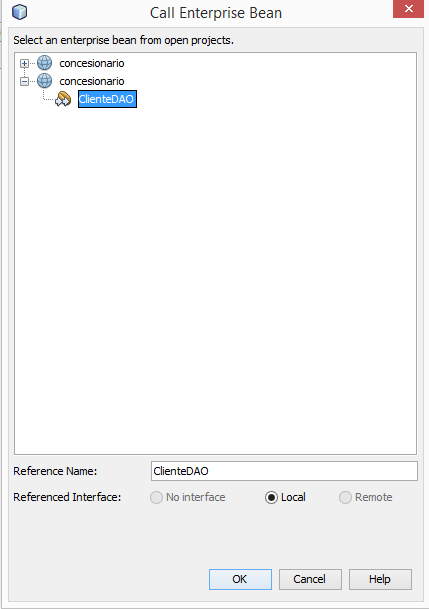
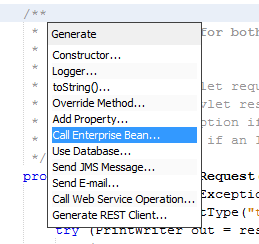
}

Ahora se agregarán los Servlets, que actuarán como controladores de las vistas.

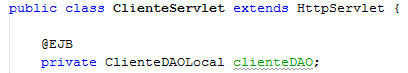
Se verifica que esté activada la opción de usar el descriptor de despliegue.



Sobre el código, se agrega en **Insert Code**  la opción **Call Enterprise Bean** y se agrega la Interfaz Local de la fachada de ClienteDAO.



Nos aseguramos que esté referenciado el acceso a los métodos del negocio con la interfaz a través de la llamada con la anotación @EJB



**clienteServlet.Java**

package com.udea.controller;

import com.udea.dao.ClienteDAOLocal;

import com.udea.model.Cliente;

import java.io.IOException;

import java.util.List;

import javax.ejb.EJB;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "ClienteServlet", urlPatterns = {"/ClienteServlet"})

public class clienteServlet extends HttpServlet {

    @EJB

    private ClienteDAOLocal clienteDAO;

    /\*\*

     \* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

     \* methods.

     \*

     \* @param request servlet request

     \* @param response servlet response

     \* @throws ServletException if a servlet-specific error occurs

     \* @throws IOException if an I/O error occurs

     \*/

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

        String mensaje="";

        boolean buscarTodos= false;

        boolean controlMensaje = false;

        //Se toman las acciones de los botones del formulario cliente

        String action = request.getParameter("action");

        //se capturan los datos del formulario cliente

         String cedulaStr = request.getParameter("cedula");

         String cedula = "";

            //Valida que el campo tenga algun dato

            if (cedulaStr != null && !cedulaStr.equals(""))

            {

                cedula = cedulaStr;

            }

        String nombre = request.getParameter("Nombre");

        String apellido = request.getParameter("Apellido");

        String direccion = request.getParameter("Direccion");

        String telefono = request.getParameter("Telefono");

        //se invoca el constructor del POJO

        Cliente cliente = new Cliente(cedula, nombre, apellido, direccion,telefono);

        List<Cliente> c=null;

        //se invoca la accion de cada boton

        if ("Agregar".equalsIgnoreCase(action)) {

            try{

                clienteDAO.addCliente(cliente);

                mensaje= "El cliente se ha agregado exitosamente";

                controlMensaje = true;

            }catch(Exception e){

                mensaje="Error: Cédula inválida o ya existe en la base de datos";

                controlMensaje = true;

            };

        } else if ("Editar".equalsIgnoreCase(action)) {

            try{

            clienteDAO.editCliente(cliente);

             mensaje= "El cliente se ha editado exitosamente";

             controlMensaje = true;

            }catch(Exception e){

                mensaje="Error: No se pudo editar, el cliente no existe en la base de datos";

                controlMensaje = true;

            };

        } else if ("Borrar".equalsIgnoreCase(action)) {

            try{

            clienteDAO.deleteCliente(cedula);

             mensaje= "El cliente se ha borrado exitosamente";

             controlMensaje = true;

            }catch(Exception e){

                 mensaje="Error: No se pudo borrar, el cliente no existe en la base de datos";

                 controlMensaje = true;

            };

        } else if ("Buscar".equalsIgnoreCase(action)) {

             try{

            cliente=clienteDAO.getCliente(cedula);

             //Se cargan los datos directamente del formulario

        request.setAttribute("message", cliente.getCedula()+"  ");

        request.setAttribute("message1", cliente.getNombre()+"  ");

        request.setAttribute("message2", cliente.getApellido()+"  ");

        request.setAttribute("message3", cliente.getDireccion()+"  ");

        request.setAttribute("message4", cliente.getTelefono()+"  ");

        buscarTodos= false;

         //se redirecciona a la vista del cliente

        request.getRequestDispatcher("cliente.jsp").forward(request, response);

            }catch(Exception e){

                mensaje="ERROR, No se pudo mostrar la información, el cliente no existe en la base de datos.";

                controlMensaje = true;

            };

        }else if("BuscarTodos".equalsIgnoreCase(action)){

                    buscarTodos= true;

                    c =clienteDAO.getAllClientes();

        }

        //se definen los atributos para la carga de datos

        request.setAttribute("cliente", cliente);// solo se llama un objeto

        //se llama a clienteDAO.getAllClientes());

        request.setAttribute("allClientes", clienteDAO.getAllClientes());

        request.setAttribute("buscarTodo", buscarTodos);

        request.setAttribute("mensaje", mensaje);

        request.setAttribute("controlMensaje", controlMensaje);

        //redirecciona

        request.getRequestDispatcher("/cliente.jsp").forward(request,response);

    }

**vehiculoServlet.Java**

package com.udea.controller;

import com.udea.dao.VehiculoDAOLocal;

import com.udea.model.Vehiculo;

import java.io.ByteArrayOutputStream;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStream;

import static java.lang.System.out;

import java.util.List;

import javax.ejb.EJB;

import javax.servlet.ServletException;

import javax.servlet.ServletOutputStream;

import javax.servlet.annotation.MultipartConfig;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import javax.servlet.http.Part;

@MultipartConfig(maxFileSize = 16177215)

@WebServlet(name = "VehiculoServlet", urlPatterns = {"/VehiculoServlet"})

public class vehiculoServlet extends HttpServlet {

    @EJB

    private VehiculoDAOLocal vehiculoDAO;

    /\*\*

     \* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

     \* methods.

     \*

     \* @param request servlet request

     \* @param response servlet response

     \* @throws ServletException if a servlet-specific error occurs

     \* @throws IOException if an I/O error occurs

     \*/

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

        boolean buscarTodo=false;

        boolean controlMensaje = false;

        String mensaje = "";

        String codigo = request.getParameter("code"); //permite saber si se requiere texto !=1 o imagen ==1

        String vehmatricula = request.getParameter("matricula"); //Buscar el vehiculo para obtener su foto

        byte[] data = null;

        List<Vehiculo> listaVehiculos = vehiculoDAO.getAllVehiculos(); //Obtiene todos los vehiculos en la BD

        //Busca el vehículo requerido

        for (Vehiculo car : listaVehiculos) {

            String mat = car.getMatricula();

            if (mat.equalsIgnoreCase(vehmatricula)) {

                data = car.getFoto();

            }

        }

        //Envía la imagen a la vista del vehículo

        if (codigo != null && "1".equalsIgnoreCase(codigo)) {

            response.setContentType("image/jpeg");

            response.getOutputStream().write(data);

            response.getOutputStream().close();

        } else {

            response.setContentType("text/html;charset=UTF-8");

        }

        //Se toman las acciones de los botones del formulario Vehículo

        String action = request.getParameter("action");

        //se captura la matricula del formulario

        String matriculaStr = request.getParameter("matricula");

        String matricula = "";

        //Valida que el campo tenga algún dato

        if (matriculaStr != null && !matriculaStr.equals("")) {

            matricula = matriculaStr;

        }

        String marca = request.getParameter("marca");

        String modelo = request.getParameter("modelo");

        String color = request.getParameter("color");

        String preciostr = request.getParameter("precio");

        float precio = 0;

        //Valida que el campo precio tenga algun dato

        if (preciostr != null && !preciostr.equals("")) {

            precio = Float.parseFloat(preciostr);//convierte cadena de caracteres a float

        }

       InputStream inputStream;

        //Obtiene la parte del archivo a cargar en la peticion (multipart)

        Part filePart = request.getPart("foto");

        byte[] foto = null;

        //valida que no esté vacio el archivo

        if (filePart != null) {

            //Obtiene el input stream del archivo cargado y lo almacena como un arreglo de bytes

            inputStream = filePart.getInputStream();

            ByteArrayOutputStream output = new ByteArrayOutputStream();

            byte[] buffer = new byte[1024];

            for (int longitud = 0; (longitud = inputStream.read(buffer)) > 0;) {

                output.write(buffer, 0, longitud);

            }

            foto = output.toByteArray();

        }

        //se invoca el constructor del POJO

       Vehiculo vehiculo = new Vehiculo(matricula, marca, modelo, color, precio, foto);

        List<Vehiculo> v;

        //se invoca la acción de cada botón

        if ("Agregar".equalsIgnoreCase(action)) {

            try {

                vehiculoDAO.addVehiculo(vehiculo);

                 mensaje= "El vehículo se ha agregado exitosamente";

            } catch (Exception e) {

                mensaje = "Error: Matrícula inválida o ya existe en la base de datos";

                controlMensaje = true;

            };

        } else if ("Editar".equalsIgnoreCase(action)) {

            try {

                vehiculoDAO.editVehiculo(vehiculo);

                mensaje= "El vehículo se ha editado exitosamente";

                controlMensaje = true;

            } catch (Exception e) {

                mensaje = "Error: No se pudo editar, la matricula no existe en la base de datos";

                controlMensaje = true;

            };

        } else if ("Borrar".equalsIgnoreCase(action)) {

            try {

                vehiculoDAO.deleteVehiculo(matricula);

                mensaje= "El vehículo se ha borrado exitosamente";

                controlMensaje = true;

            } catch (Exception e) {

                mensaje = "Error: No se pudo borrar, la matricula no existe en la base de datos";

                controlMensaje = true;

            };

        } else if ("Buscar".equalsIgnoreCase(action)) {

            try {

                vehiculo = vehiculoDAO.getVehiculo(matricula);

                //Se cargan los datos directamente del formulario

                request.setAttribute("message", vehiculo.getMatricula() + "  ");

                request.setAttribute("message1", vehiculo.getMarca() + "  ");

                request.setAttribute("message2", vehiculo.getModelo() + "  ");

                request.setAttribute("message3", vehiculo.getColor() + "  ");

                request.setAttribute("message4", vehiculo.getPrecio() + "  ");

                buscarTodo=false;

                //se redirecciona a la vista del vehiculo

                request.getRequestDispatcher("vehiculo.jsp").forward(request, response);

            } catch (Exception e) {

                mensaje = "ERROR, No se pudo mostrar la información, el vehículo no existe en la base de datos";

                controlMensaje = true;

            };

        } else if ("BuscarTodos".equalsIgnoreCase(action)) {

            buscarTodo= true;

            v = vehiculoDAO.getAllVehiculos();

        }

        //se definen los atributos para la carga de datos

        request.setAttribute("vehiculo", vehiculo);// solo se llama un objeto

        //se llama todos los objetos retornados para la tabla HTML

        request.setAttribute("allVehiculos", vehiculoDAO.getAllVehiculos());

        request.setAttribute("mensaje", mensaje);

        request.setAttribute("buscarTodo", buscarTodo);

        request.setAttribute("controlMensaje", controlMensaje);

        //redirecciona

        request.getRequestDispatcher("/vehiculo.jsp").forward(request, response);

    }

**vendedorServlet.Java**

package com.udea.controller;

import com.udea.dao.VendedorDAOLocal;

import com.udea.model.Vendedor;

import java.io.IOException;

import java.util.List;

import javax.ejb.EJB;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "VendedorServlet", urlPatterns = {"/VendedorServlet"})

public class vendedorServlet extends HttpServlet {

    @EJB

    private VendedorDAOLocal vendedorDAO;

    /\*\*

     \* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

     \* methods.

     \*

     \* @param request servlet request

     \* @param response servlet response

     \* @throws ServletException if a servlet-specific error occurs

     \* @throws IOException if an I/O error occurs

     \*/

   protected void processRequest(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

        boolean buscarTodo=false;

        boolean controlMensaje = false;

        String mensaje="";

        //Se toman las acciones de los botones del formulario vendedor

        String action = request.getParameter("action");

        //se capturan los datos del formulario vendedor

         String cedulaStr = request.getParameter("cedula");

         String cedula = "";

            //Valida que el campo tenga algun dato

            if (cedulaStr != null && !cedulaStr.equals(""))

            {

                cedula = cedulaStr;

            }

        String nombre = request.getParameter("nombre");

        String apellido = request.getParameter("apellido");

        //se invoca el constructor del POJO

        Vendedor vendedor= new Vendedor(cedula, nombre, apellido);

        List<Vendedor> v;

        //se invoca la accion de cada boton

        if ("Agregar".equalsIgnoreCase(action)) {

             try{

                 vendedorDAO.addVendedor(vendedor);

                 mensaje= "El vendedor se ha agregado exitosamente";

                 controlMensaje = true;

            }catch(Exception e){

                mensaje="Error, Cédula inválida o ya existe en la base de datos";

                controlMensaje = true;

            };

        } else if ("Editar".equalsIgnoreCase(action)) {

             try{

                vendedorDAO.editVendedor(vendedor);

                mensaje= "El vendedor se ha editado exitosamente";

                controlMensaje = true;

            }catch(Exception e){

                mensaje="Error, No se pudo editar, el vendedor no existe en la base de datos";

                controlMensaje = true;

            };

        } else if ("Borrar".equalsIgnoreCase(action)) {

            try{

            vendedorDAO.deleteVendedor(cedula);

            mensaje= "El vendedor se ha borrado exitosamente";

            controlMensaje = true;

            }catch(Exception e){

                 mensaje="Error, No se pudo borrar, el vendedor no existe en la base de datos";

            };

        } else if ("Buscar".equalsIgnoreCase(action)) {

              try{

             vendedor=vendedorDAO.getVendedor(cedula);

             //Se cargan los datos directamente del formulario

        request.setAttribute("message", vendedor.getCedula()+"  ");

        request.setAttribute("message1", vendedor.getNombre()+"  ");

        request.setAttribute("message2", vendedor.getApellido()+"  ");

        buscarTodo= false;

        //se redirecciona a la vista del vendedor

        request.getRequestDispatcher("vendedor.jsp").forward(request, response);

            }catch(Exception e){

                mensaje="Error, No se pudo mostrar la información, el vendedor no existe en la base de datos";

                controlMensaje = true;

            };

        }else if("BuscarTodos".equalsIgnoreCase(action)){

            buscarTodo= true;

            v =vendedorDAO.getAllVendedores();

        }

        //se definen los atributos para la carga de datos

        request.setAttribute("vendedor", vendedor);// solo se llama un objeto

        //se llama todos los objetos retornados para la tabla HTML

        request.setAttribute("allVendedor", vendedorDAO.getAllVendedores());

        request.setAttribute("mensaje", mensaje);

        request.setAttribute("buscarTodo", buscarTodo);

        request.setAttribute("controlMensaje", controlMensaje);

        //redirecciona

        request.getRequestDispatcher("/vendedor.jsp").forward(request,response);

    }

**ventasServlet.Java**

package com.udea.controller;

import com.udea.dao.VentasDAOLocal;

import com.udea.model.Ventas\_generales;

import java.io.IOException;

import java.util.List;

import javax.ejb.EJB;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "VentasServlet", urlPatterns = {"/VentasServlet"})

public class ventasServlet extends HttpServlet {

    @EJB

    private VentasDAOLocal ventasDAO;

    /\*\*

     \* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

     \* methods.

     \*

     \* @param request servlet request

     \* @param response servlet response

     \* @throws ServletException if a servlet-specific error occurs

     \* @throws IOException if an I/O error occurs

     \*/

     protected void processRequest(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

        boolean buscarTodo=false;

        boolean controlMensaje = false;

        String mensaje="";

        //Se toman las acciones de los botones del formulario ventas

        String action = request.getParameter("action");

        //se capturan los datos del formulario ventas

         String codStr = request.getParameter("codigo\_venta");

         String cod = "";

            //Valida que el campo tenga algun dato

            if (codStr != null && !codStr.equals(""))

            {

                cod = codStr;

            }

        String cliente = request.getParameter("cliente");

        String vehiculo = request.getParameter("vehiculo");

        String vendedor = request.getParameter("vendedor");

        String preciostr = request.getParameter("valor\_total");

         float precio = 0;

            //Valida que el campo precio tenga algun dato

            if (preciostr != null && !preciostr.equals(""))

            {

                precio = Float.parseFloat(preciostr);//convierte cadena de caracteres a float

            }

        //se invoca el constructor del POJO

        Ventas\_generales ventas= new Ventas\_generales(cod, cliente, vehiculo,vendedor,precio);

        List<Ventas\_generales> v;

        //se invoca la accion de cada boton

        if ("Agregar".equalsIgnoreCase(action)) {

             try{

                ventasDAO.addVenta(ventas);

                mensaje= "La venta se ha agregado exitosamente";

                controlMensaje = true;

            }catch(Exception e){

                mensaje="Error, Código de venta inválida o ya existe en la base de datos";

                controlMensaje = true;

            };

        } else if ("Editar".equalsIgnoreCase(action)) {

            try{

            ventasDAO.editVenta(ventas);

            mensaje= "La venta se ha agregado exitosamente";

            controlMensaje = true;

            }catch(Exception e){

                mensaje="Error, No se pudo editar, la venta no existe en la base de datos";

                controlMensaje = true;

            };

        } else if ("Borrar".equalsIgnoreCase(action)) {

             try{

            ventasDAO.deleteVenta(cod);

            mensaje= "La venta se ha agregado exitosamente";

            controlMensaje = true;

            }catch(Exception e){

                 mensaje="Error, No se pudo borrar, la venta no existe en la base de datos";

                 controlMensaje = true;

            };

        } else if ("Buscar".equalsIgnoreCase(action)) {

             try{

                ventas=ventasDAO.getVenta(cod);

             //Se cargan los datos directamente del formulario

        request.setAttribute("message", ventas.getCodigo\_venta()+"  ");

        request.setAttribute("message1", ventas.getCliente()+"  ");

        request.setAttribute("message2", ventas.getVehiculo()+"  ");

        request.setAttribute("message3", ventas.getVendedor()+"  ");

        request.setAttribute("message4", ventas.getValor\_total()+"  ");

        buscarTodo=false;

        //se redirecciona a la vista del vendedor

        request.getRequestDispatcher("venta.jsp").forward(request, response);

            }catch(Exception e){

                mensaje="Error, No se pudo mostrar la información, la venta no existe en la base de datos";

                controlMensaje = true;

            };

        }else if("BuscarTodos".equalsIgnoreCase(action)){

            buscarTodo = true;

            v = ventasDAO.getAllVentas();

        }

        //se definen los atributos para la carga de datos

        request.setAttribute("ventas", ventas);// solo se llama un objeto

        //se llama todos los objetos retornados para la tabla HTML

        request.setAttribute("allVentas", ventasDAO.getAllVentas());

        request.setAttribute("mensaje", mensaje);

        request.setAttribute("buscarTodo", buscarTodo);

        request.setAttribute("controlMensaje", controlMensaje);

        //redirecciona

        request.getRequestDispatcher("/venta.jsp").forward(request,response);

    }

Se crea un CSS llamado style.css y que por contenido lleva:

td{

height: 50px;

vertical-aling: bottom;

}

**Cliente.jsp**

<%@ taglib prefix="c" uri="<http://java.sun.com/jsp/jstl/core>" %>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <link rel="stylesheet"

              href="<https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css>">

        <link rel="stylesheet" href="style.css">

        <!-- Optional theme -->

        <link rel="stylesheet"

              href="<https://bootswatch.com/darkly/bootstrap.min.css>">

        <!-- Latest compiled and minified JavaScript -->

        <script

        src="<https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js>"></script>

        <title>Información del Cliente</title>

         <nav role="navigation" class="navbar navbar-default">

        <div id="navbarCollapse" class="collapse navbar-collapse">

            <ul class="nav navbar-nav">

                <li class="active"><a href="cliente.jsp">Clientes</a></li>

                <li><a href="vehiculo.jsp">Vehículos</a></li>

                <li><a href="vendedor.jsp">Vendedor</a></li>

                <li><a href="venta.jsp">Ventas</a></li>

            </ul>

        </div>

        </nav>

    </head>

    <body>

        <div class="container well">

            <h1>INFORMACIÓN DEL CLIENTE</h1>

            <form action="./clienteServlet" method="POST">

                <table>

                    <tr>

                        <th>Cédula</th>

                        <th><br><input type="text" name="cedula" class="form-control" required="true" placeholder="Cédula" value="${cliente.cedula}"/> </th>

                    </tr>

                    <tr>

                        <th> Nombre</th>

                        <th><br><input type="text" name="Nombre" class="form-control" placeholder="Nombre" value="${cliente.nombre}"/></th>

                    </tr>

                    <tr>

                        <th>Apellido</th>

                        <th><br><input type="text" name="Apellido" class="form-control" placeholder="Apellido" value="${cliente.apellido}"/> </th>

                    </tr>

                    <tr>

                        <th>Dirección</th>

                        <th><br><input type="text" name="Direccion" class="form-control" placeholder="Dirección" value="${cliente.direccion}"/></th>

                    </tr>

                    <tr>

                        <th>Teléfono</th>

                        <th><br><input type="text" name="Telefono" class="form-control" placeholder="Teléfono" value="${cliente.telefono}"/></th>

                    </tr>

                    <tr>

                        <td  colspan="2"><br>

                            <input class="btn btn-primary" type="submit" name="action" value="Agregar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-plus img-circle text-success"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Editar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-share img-circle text-info"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Borrar"/>

                             <span class="glyphicon btn-glyphicon glyphicon-trash img-circle text-danger"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Buscar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-search img-circle text-primary"/>

                            <input class="btn btn-primary" type="submit" name="action" value="BuscarTodos"/>

                            <span class="glyphicon btn-glyphicon glyphicon-search img-circle text-primary"/>

                        </td>

                    </tr>

                </table>

            </form>

            <br>

            <table class="table table-inverse">

                <thead>

                    <tr class="info">

                        <th>Cédula</th>

                        <th>Nombre</th>

                        <th>Apellido</th>

                        <th>Dirección</th>

                        <th>Teléfono</th>

                    </tr>

                </thead>

                <thead>

                <tr>

                    <c:choose>

                        <c:when test="${buscarTodo}">

                            <c:forEach items="${allClientes}" var="cliente">

                                <td>${cliente.cedula}</td>

                                <td>${cliente.nombre}</td>

                                <td>${cliente.apellido}</td>

                                <td>${cliente.direccion}</td>

                                <td>${cliente.telefono}</td>

                            </tr>

                </thead>

                        </c:forEach>

                    </c:when>

                    <c:otherwise>

                <thead>

                <tr>

                    <td>${message}</td>

                    <td>${message1}</td>

                    <td>${message2}</td>

                    <td>${message3}</td>

                    <td>${message4}</td>

                </tr>

                </thead>

                    </table>

                <c:choose>

                        <c:when test="${controlMensaje}">

                            <div class="alert alert-warning">

                                <strong>MENSAJE: </strong> ${mensaje}

                            </div>

                        </c:when>

                    </c:choose>​

                </c:otherwise>

                    </c:choose>

        </div>

    </body>

</html>

**Vehiculo.jsp**

<%@ taglib prefix="c" uri="<http://java.sun.com/jsp/jstl/core>" %>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <link rel="stylesheet"

              href="<https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css>">

        <link rel="stylesheet" href="style.css">

        <!-- Optional theme -->

        <link rel="stylesheet"

              href="<https://bootswatch.com/darkly/bootstrap.min.css>">

        <!-- Latest compiled and minified JavaScript -->

        <script

        src="<https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js>"></script>

        <title>Información del Vehículo</title>

         <nav role="navigation" class="navbar navbar-default">

        <div id="navbarCollapse" class="collapse navbar-collapse">

            <ul class="nav navbar-nav">

                <li><a href="cliente.jsp">Clientes</a></li>

                <li class="active"><a href="vehiculo.jsp">Vehículos</a></li>

                <li><a href="vendedor.jsp">Vendedor</a></li>

                <li><a href="venta.jsp">Ventas</a></li>

            </ul>

        </div>

        </nav>

    </head>

    <body>

        <div class="container well">

            <h1>INFORMACIÓN DEL VEHÍCULO</h1>

            <form action="./vehiculoServlet" method="POST" enctype="multipart/form-data">

                <table>

                    <tr>

                        <th>Matrícula</th>

                        <th><br><input type="text" name="matricula" class="form-control" required="true" placeholder="Matrícula" value="${vehiculo.matricula}"/> </th>

                    </tr>

                    <tr>

                        <th>Marca</th>

                        <th><br><input type="text" name="marca" class="form-control" placeholder="Marca" value="${vehiculo.marca}"/></th>

                    </tr>

                    <tr>

                        <th>Modelo</th>

                        <th><br><input type="text" name="modelo" class="form-control" placeholder="Modelo" value="${vehiculo.modelo}"/> </th>

                    </tr>

                    <tr>

                        <th>Color</th>

                        <th><br><input type="text" name="color" class="form-control" placeholder="Color" value="${vehiculo.color}"/></th>

                    </tr>

                    <tr>

                        <th>Precio en millones:   </th>

                        <th><br><input type="text" name="precio" class="form-control" placeholder="Precio" value="${vehiculo.precio}"/></th>

                    </tr>

                     <tr>

                        <th>Foto</th>

                        <th><br><input type="file" name="foto" size="50" </th>

                    </tr>

                        <td colspan="2"><br>

                            <input class="btn btn-primary" type="submit" name="action" value="Agregar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-plus img-circle text-success"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Editar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-share img-circle text-info"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Borrar"/>

                             <span class="glyphicon btn-glyphicon glyphicon-trash img-circle text-danger"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Buscar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-search img-circle text-primary"/>

                            <input class="btn btn-primary" type="submit" name="action" value="BuscarTodos"/>

                            <span class="glyphicon btn-glyphicon glyphicon-search img-circle text-primary"/>

                        </td>

                    </tr>

                </table>

            </form>

            <br>

             <table class="table table-inverse">

                <thead>

                 <tr class="info">

                    <th>Matrícula</th>

                    <th>Marca</th>

                    <th>Modelo</th>

                    <th>Color</th>

                    <th>Precio</th>

                    <th>Foto</th>

                </tr>

                </thead>

                <thead>

                <tr>

                    <c:choose>

                        <c:when test="${buscarTodo}">

                            <c:forEach items="${allVehiculos}" var="veh">

                                <td>${veh.matricula}</td>

                                <td>${veh.marca}</td>

                                <td>${veh.modelo}</td>

                                <td>${veh.color}</td>

                                <td>${veh.precio}</td>

                                <td> <img src="/concesionario/vehiculoServlet?code=1&matricula=${veh.matricula}"height="150" width="250"/></td>

                            </tr>

                </thead>

                        </c:forEach>

                    </c:when>

                    <c:otherwise>

                        <c:if test="${message !=null}">

                            <thead>

                            <tr>

                                <td> ${message}</td>

                                <td>${message1}</td>

                                <td>${message2}</td>

                                <td>${message3}</td>

                                <td>${message4}</td>

                                <td> <img src="/concesionario/vehiculoServlet?code=1&matricula=${message}"height="150" width="250"/></td>

                            </tr>

                            </thead>

                        </c:if>

                        <c:choose>

                        <c:when test="${controlMensaje}">

                            <div class="alert alert-warning">

                                <strong>MENSAJE: </strong> ${mensaje}

                            </div>

                        </c:when>

                    </c:choose>​

                         </c:otherwise>

                    </c:choose>

           </div>

    </body>

</html>

**Vendedor.jsp**

<%@ taglib prefix="c" uri="<http://java.sun.com/jsp/jstl/core>" %>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <link rel="stylesheet"

              href="<https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css>">

        <link rel="stylesheet" href="style.css">

        <!-- Optional theme -->

        <link rel="stylesheet"

              href="<https://bootswatch.com/darkly/bootstrap.min.css>">

        <!-- Latest compiled and minified JavaScript -->

        <script

        src="<https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js>"></script>

        <title>Información del Vendedor</title>

         <nav role="navigation" class="navbar navbar-default">

        <div id="navbarCollapse" class="collapse navbar-collapse">

            <ul class="nav navbar-nav">

                <li><a href="cliente.jsp">Clientes</a></li>

                <li ><a href="vehiculo.jsp">Vehiculos</a></li>

                <li class="active"><a href="vendedor.jsp">Vendedor</a></li>

                <li><a href="venta.jsp">Ventas</a></li>

            </ul>

        </div>

        </nav>

    </head>

    <body>

        <div class="container well">

            <h1>INFORMACIÓN DEL VENDEDOR</h1>

            <form action="./vendedorServlet" method="POST">

                <table>

                    <tr>

                        <th>Cédula</th>

                        <th><br><input type="text" name="cedula" class="form-control" required="true" placeholder="Cédula" value="${vendedor.cedula}"/> </th>

                    </tr>

                    <tr>

                        <th>Nombre</th>

                        <th><br><input type="text" name="nombre" class="form-control" placeholder="Nombre" value="${vendedor.nombre}"/></th>

                    </tr>

                    <tr>

                        <th>Apellido</th>

                        <th><br><input type="text" name="apellido" class="form-control" placeholder="Apellido" value="${vendedor.apellido}"/> </th>

                    </tr>

                    <tr>

                        <td colspan="2"><br>

                            <input class="btn btn-primary" type="submit" name="action" value="Agregar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-plus img-circle text-success"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Editar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-share img-circle text-info"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Borrar"/>

                             <span class="glyphicon btn-glyphicon glyphicon-trash img-circle text-danger"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Buscar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-search img-circle text-primary"/>

                            <input class="btn btn-primary" type="submit" name="action" value="BuscarTodos"/>

                            <span class="glyphicon btn-glyphicon glyphicon-search img-circle text-primary"/>

                        </td>

                    </tr>

                </table>

            </form>

            <br>

            <table class="table table-inverse">

                <thead>

                    <tr class="info">

                        <th>Cédula</th>

                        <th>Nombre</th>

                        <th>Apellido</th>

                    </tr>

                </thead>

               <thead>

                <tr>

                    <c:choose>

                        <c:when test="${buscarTodo}">

                            <c:forEach items="${allVendedor}" var="vend">

                                <td>${vend.cedula}</td>

                                <td>${vend.nombre}</td>

                                <td>${vend.apellido}</td>

                            </tr>

               </thead>

                        </c:forEach>

                    </c:when>

                    <c:otherwise>

                        <thead>

                        <tr>

                            <td>${message}</td>

                            <td>${message1}</td>

                            <td>${message2}</td>

                        </tr>

                        </thead>

                        <c:choose>

                        <c:when test="${controlMensaje}">

                            <div class="alert alert-warning">

                                <strong>MENSAJE: </strong> ${mensaje}

                            </div>

                        </c:when>

                    </c:choose>​

                    </c:otherwise>

                </c:choose>

        </div>

    </body

**Venta.jsp**

<%@ taglib prefix="c" uri="<http://java.sun.com/jsp/jstl/core>" %>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <link rel="stylesheet"

              href="<https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css>">

        <link rel="stylesheet" href="style.css">

        <!-- Optional theme -->

        <link rel="stylesheet"

              href="<https://bootswatch.com/darkly/bootstrap.min.css>">

        <!-- Latest compiled and minified JavaScript -->

        <script

        src="<https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js>"></script>

        <title>Información de la venta</title>

        <nav role="navigation" class="navbar navbar-default">

        <div id="navbarCollapse" class="collapse navbar-collapse">

            <ul class="nav navbar-nav">

                <li><a href="cliente.jsp">Clientes</a></li>

                <li ><a href="vehiculo.jsp">Vehiculos</a></li>

                <li ><a href="vendedor.jsp">Vendedor</a></li>

                <li class="active"><a href="venta.jsp">Ventas</a></li>

            </ul>

        </div>

        </nav>

    </head>

    <body>

        <div class="container well">

            <h1>INFORMACIÓN DE LA VENTA</h1>

            <form action="./ventasServlet" method="POST">

                <table>

                    <tr>

                        <th>Código Venta</th>

                        <th><br><input type="text" name="codigo\_venta" class="form-control" required="true" placeholder="Código" value="${venta.codigo\_venta}"/> </th>

                    </tr>

                    <tr>

                        <th>Cliente</th>

                        <th><br><input type="text" name="cliente" class="form-control" placeholder="Cliente" value="${venta.cliente}"/></th>

                    </tr>

                    <tr>

                        <th>Vehículo</th>

                        <th><br><input type="text" name="vehiculo" class="form-control" placeholder="Vehículo" value="${venta.vehiculo}"/> </th>

                    </tr>

                    <tr>

                        <th>Vendedor</th>

                        <th><br><input type="text" name="vendedor" class="form-control" placeholder="Vendedor" value="${venta.vendedor}"/></th>

                    </tr>

                    <tr>

                        <th>Valor Total</th>

                        <th><br><input type="text" name="valor\_total" class="form-control" placeholder="Valor" value="${venta.valor\_total}"/></th>

                    </tr>

                    <tr>

                    <td  colspan="2"><br>

                            <input class="btn btn-primary" type="submit" name="action" value="Agregar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-plus img-circle text-success"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Editar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-share img-circle text-info"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Borrar"/>

                             <span class="glyphicon btn-glyphicon glyphicon-trash img-circle text-danger"/>

                            <input class="btn btn-primary" type="submit" name="action" value="Buscar"/>

                            <span class="glyphicon btn-glyphicon glyphicon-search img-circle text-primary"/>

                            <input class="btn btn-primary" type="submit" name="action" value="BuscarTodos"/>

                            <span class="glyphicon btn-glyphicon glyphicon-search img-circle text-primary"/>

                        </td>

                    </tr>

                </table>

            </form>

            <br>

            <table class="table table-inverse">

                <thead>

                <tr class="info">

                    <th>Código Venta</th>

                    <th>Cliente</th>

                    <th>Vehículo</th>

                    <th>Vendedor</th>

                    <th>Valor Total</th>

                </tr>

                </thead>

                <thead>

                <tr>

                    <c:choose>

                        <c:when test="${buscarTodo}">

                            <c:forEach items="${allVentas}" var="venta">

                                <td>${venta.codigo\_venta}</td>

                                <td>${venta.cliente}</td>

                                <td>${venta.vehiculo}</td>

                                <td>${venta.vendedor}</td>

                                <td>${venta.valor\_total}</td>

                            </tr>

                </thead>

                        </c:forEach>

                    </c:when>

                    <c:otherwise>

                        <thead>

                        <tr>

                            <td>${message}</td>

                            <td>${message1}</td>

                            <td>${message2}</td>

                            <td>${message3}</td>

                            <td>${message4}</td>

                        </tr>

                        </thead>

                        <c:choose>

                        <c:when test="${controlMensaje}">

                            <div class="alert alert-warning">

                                <strong>MENSAJE: </strong> ${mensaje}

                            </div>

                        </c:when>

                    </c:choose>​

                    </c:otherwise>

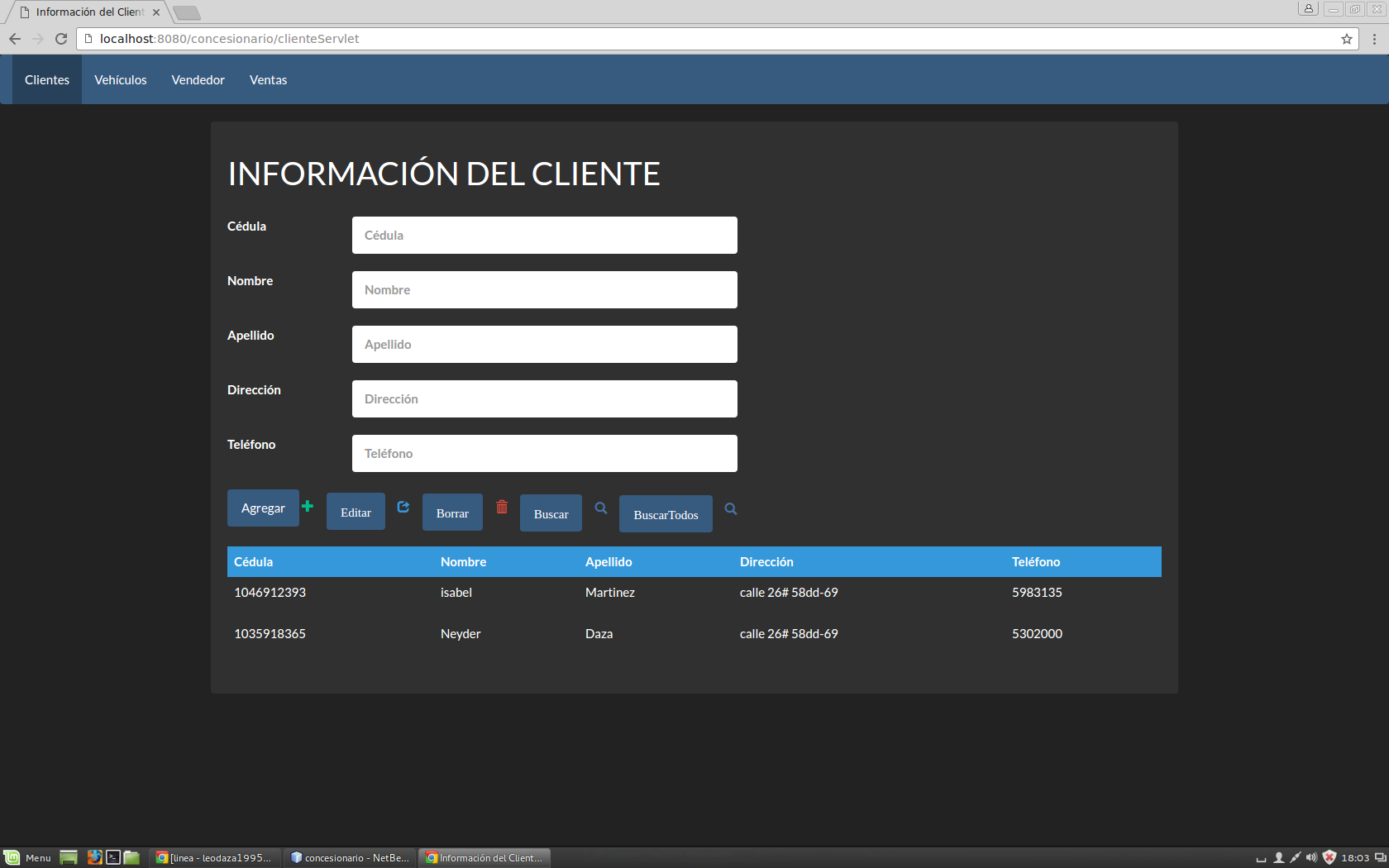
                </c:choose>

        </div>

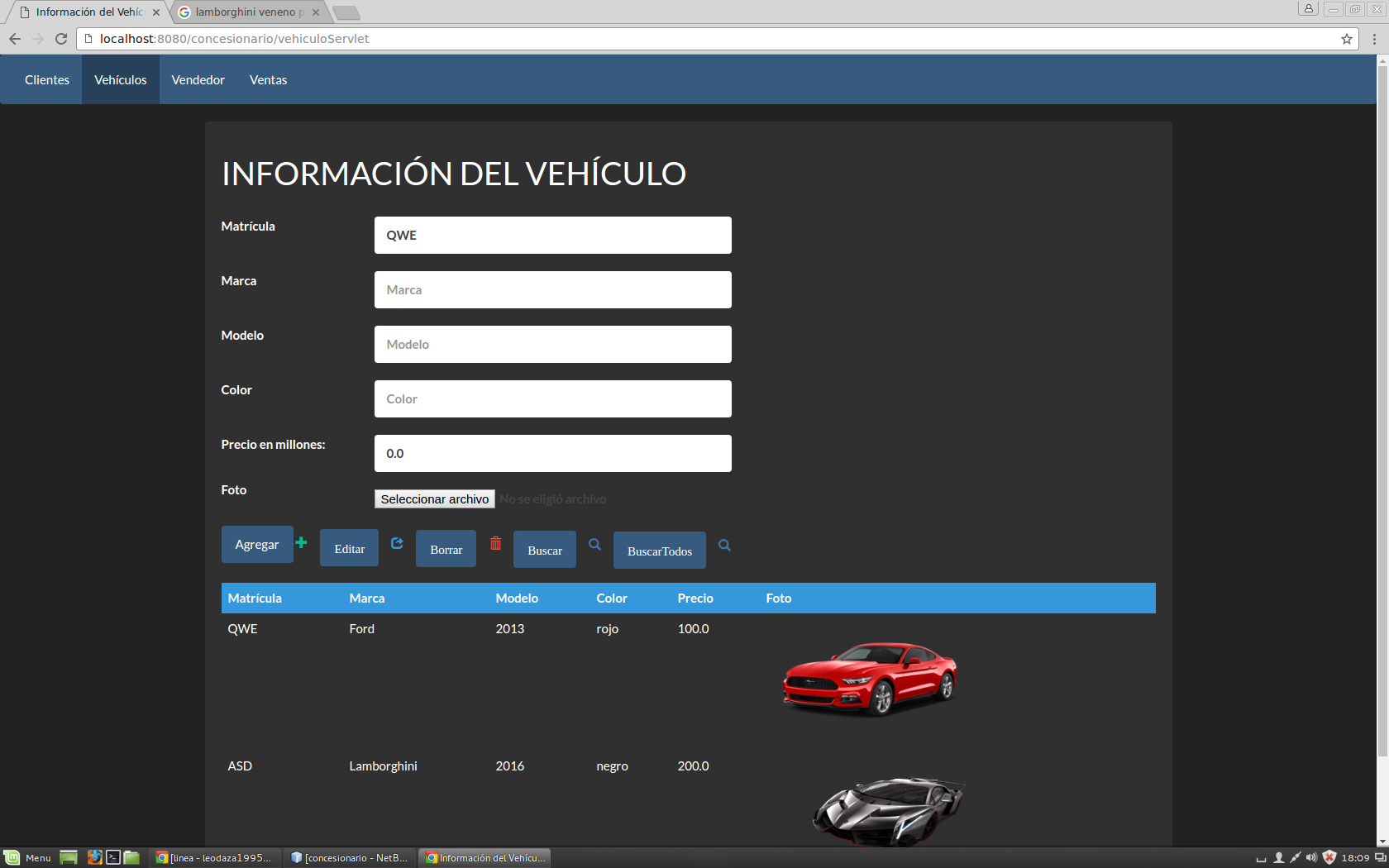
    </body>

**IMÁGENES DE LA APLICACIÓN**

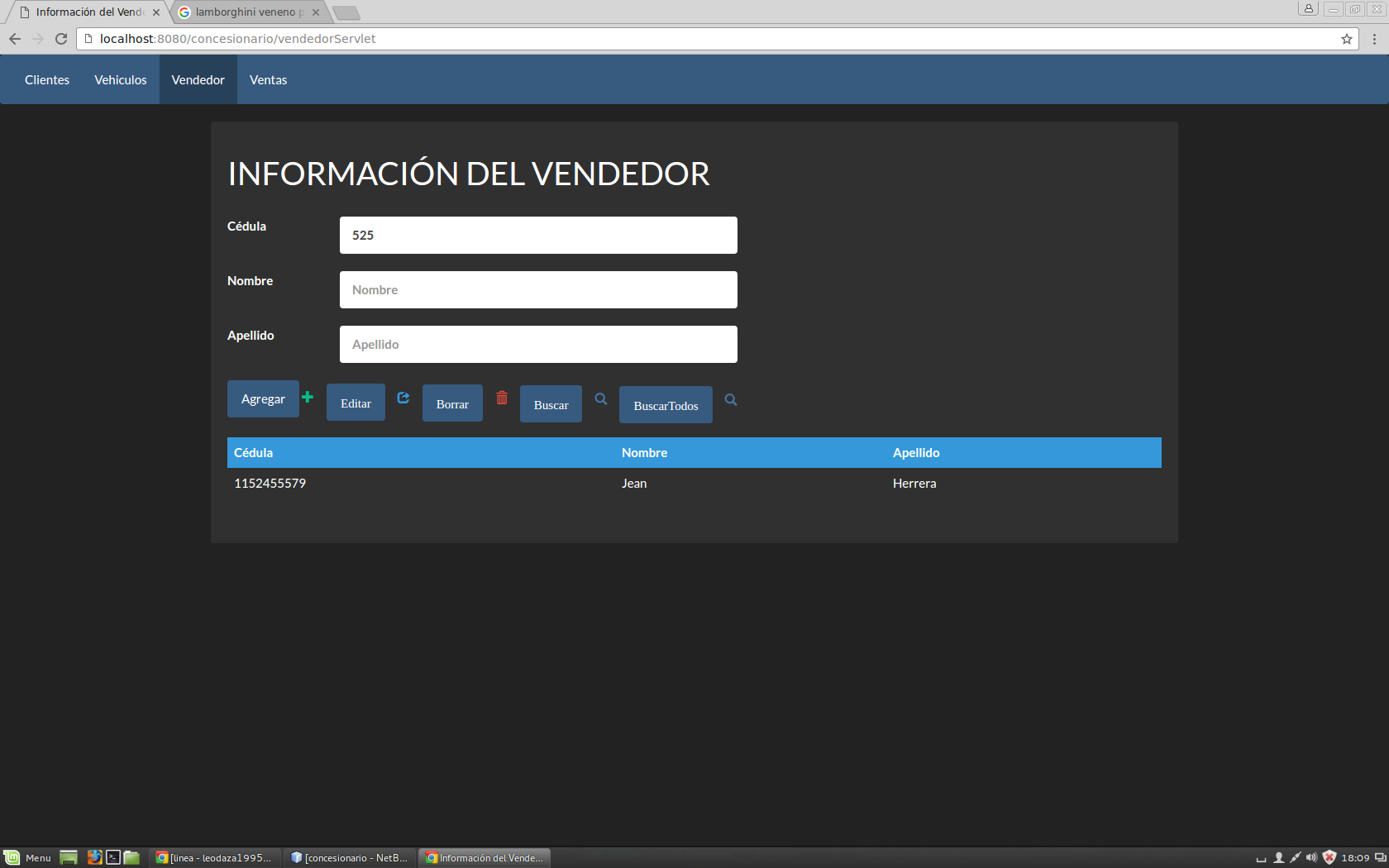
**Vista del Cliente**



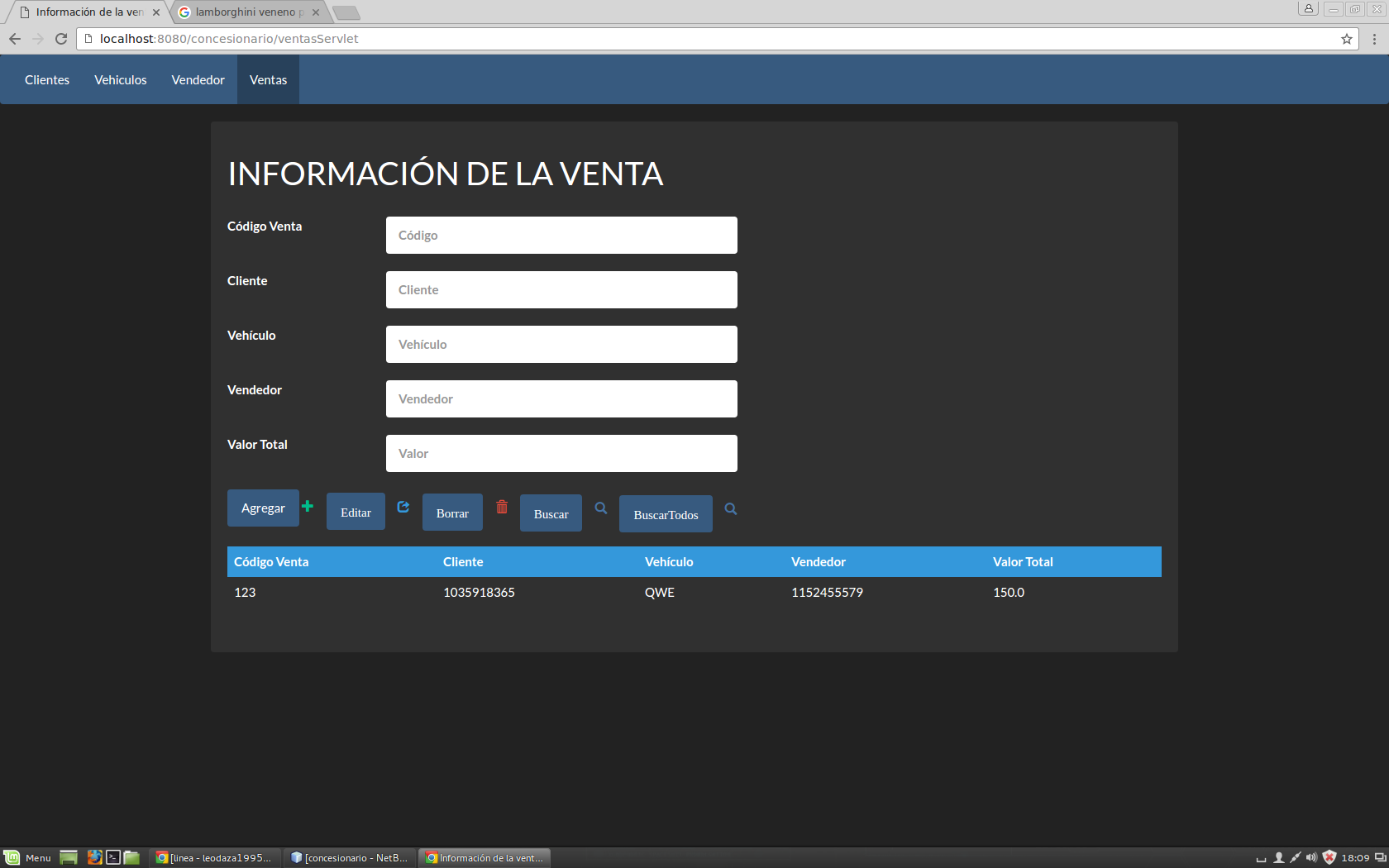
**Vista de Vehículo**



**Vista de Vendedor**



**Vista de la Venta**

****

**CONCLUSIONES**

En la actualidad, podemos ver que el internet tiene aspectos que se pueden mejorar, en materia de diseño de páginas web, conexiones, bases de datos, etc. Muchos de estos avances nos han ayudado en el ahorro de tiempo y dinero, lo cual nos aporta comodidad y facilidad para crear y acceder a los diferentes componentes que el internet nos ofrece.

Para una optimización en la creación de código web, vemos que hay herramientas que nos han colaborado: Pool de conexiones, JSP, anotaciones, JDBC, EJB, entre otros. Son elementos que se han venido transmitiendo en las entidades académicas, como colegios, universidades, tecnológicos y que además son de fácil acceso a ellas, gracias a internet. Este tema es muy importante, ya que en ámbitos de desarrollo web y avances informáticos, hay demasiada tela que cortar, y que se esté inculcando y apoyando desde la escuela, es un gran avance.

Existe una mejora notable por parte de conexiones cliente-servidor, que son de mucha ayuda y tiene que ver con la agilidad y seguridad en que se manejan los procesos. El hecho de mejorar, lo que corresponde al manejo del flujo por parte de los clientes hacía una base de datos o una página web, es admirable

En temas de programación, vemos como EJB nos ayuda a ahorra mucho tiempo en la escritura de código, gracias a sus componentes y funcionalidades que nos brinda.

.

**BIBLIOGRAFÍA**

**[1]:** http://www.lab.inf.uc3m.es/~a0080802/RAI/servlet.html

**[2]:**http://users.dcc.uchile.cl/~jbarrios/servlets/general.html

**[3]:**http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=831

**[4]:**http://library.gxtechnical.com/gxdlsp/pub/genexus/java/docum/manuals/8.0/mjavaf5.htm

**[5]:**http://www.cursohibernate.es/doku.php?id=patrones:pool\_conexiones

**[6]:**http://www.tutorialspoint.com/es/jpa/jpa\_introduction.htm

**[7]:**https://es.wikipedia.org/wiki/Anotaci%C3%B3n\_Java

**[8]:**http://www.davidmarco.es/articulo/introduccion-a-jpa-2-0-i

**[9]:**https://escss.blogspot.com/2012/11/vocabulario-glosario-terminos-css.html

**[10]:**http://www.osmosislatina.com/java/ejb.htm

**[11]:**http://docs.oracle.com/javaee/6/tutorial/doc/gipjg.html

**[12]:**http://stackoverflow.com/questions/1051182/what-is-data-transfer-object

**[13]:**https://docs.oracle.com/javaee/7/tutorial/ejb-intro003.htm#GIPKO

**[14]:**http://docs.oracle.com/javaee/1.2.1/devguide/html/DevGuideTOC.html