**Informe laboratorio 1, Introducción a Java EE**

**Arquitectura de software**

**Astrid Carolina Díaz Gómez**

**Felipe Cadavid Rincón**

**Profesor: Diego Iván Oliveros Acosta**

**Universidad de Antioquia**

**Ingeniería de sistemas**

**2019/2**

**Aplicación Web Concesionario**

Se creará una simple aplicación de tipo CRUD utilizando JSP, Servlets y EJB3 por medio de la plataforma JEE. Esta contendrá 4 operaciones dentro del JSP.

**● Insert:** Inserta datos en las tablas vehículo, cliente, ventas y vendedores.

**● List:** Lista los datos de los contactos generados y de los vehículos agregados.

**● Delete**: Elimina datos en la tabla vehiculo, cliente y vendedores.

**● Login:** Realiza acceso por medio de usuario y contraseña.

**OBJETIVOS**

1. ***Objetivo General***

Crear una aplicación CRUD para la gestión de cuentas de usuarios utilizando un pool de conexiones, recursos JDBC, anotaciones JPA, EJB, unidades de persistencia, Servlets y JSP.

1. ***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.

1. ***Herramientas Empleadas***

* Apache Netbeans IDE 11.0.
* Web Browser (Firefox, Chrome).
* Java EE 7 Web, plataforma de desarrollo.
* Java Servlets.
* Java Server Pages.
* Enterprise Javabeans.
* Bootstrap.
* GlassFish Server 4.1.2 (servidor de aplicaciones).
* Mysql 8.0.

1. ***Arquitectura Utilizada***

La aplicación está dividida en 6 capas bien definidas, las cuales están representadas en la figura 1.

**Backend:**

El Backend está compuesta de 3 capas las cuales son:

* *Capa Datos:* Corresponde al motor de base de datos MySQL, que tendrá la instancia VEHÍCULO, CLIENTE, VENTAS Y VENDEDORES, previamente creadas. Acá se almacenarán los respectivos datos que lleguen por medio del patrón DAO (Data Access Object), generado en la capa de la lógica del negocio.

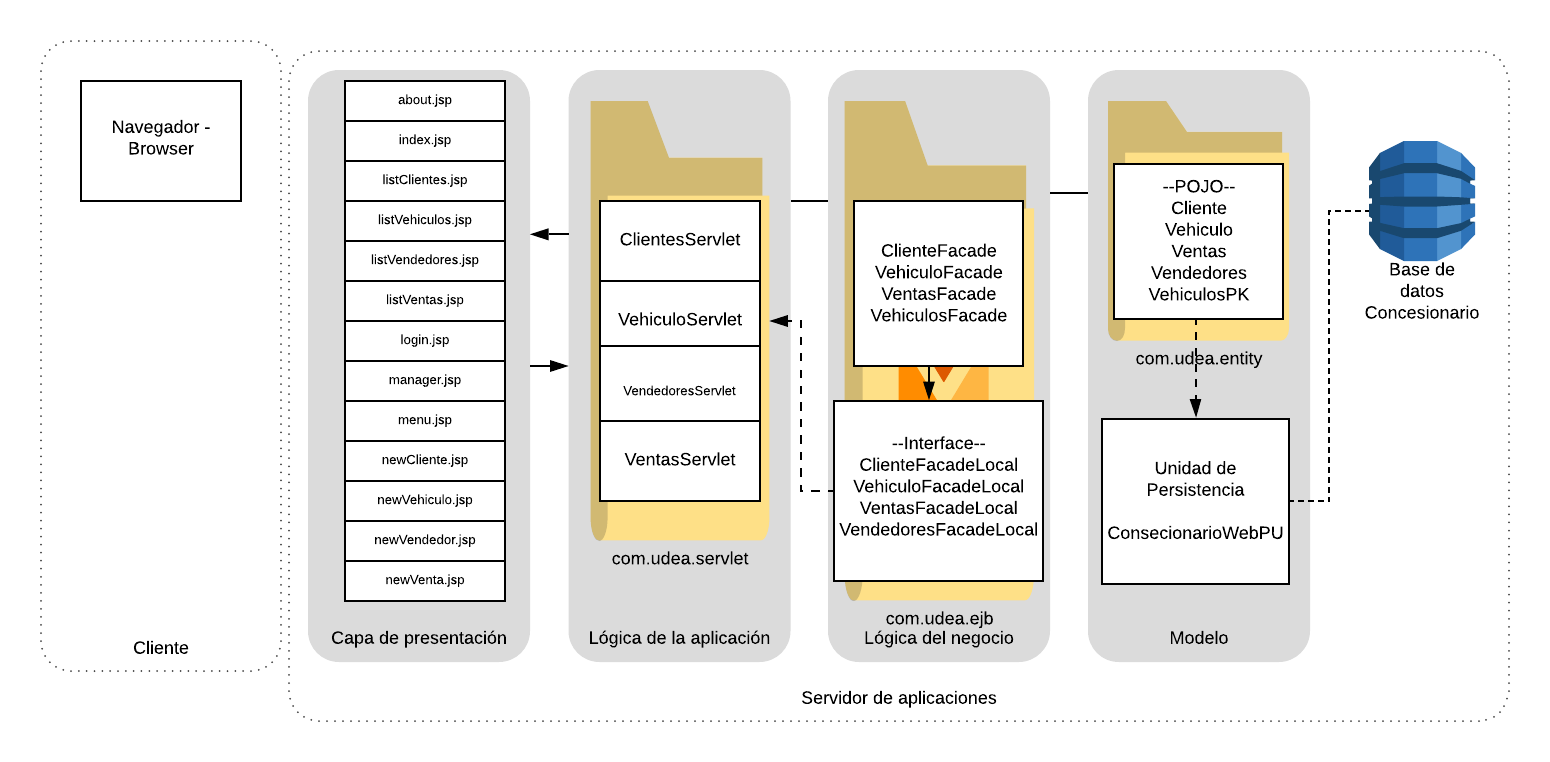
* *Capa del Modelo:* Contiene un Entity Bean que en esencia es un simple POJO (Plain Old Java Object) con anotaciones JPA para facilitar el acceso al modelo del dominio y poder enlazar los datos serializados usando el patrón DTO (Data Transfer Object). Hay que tener en cuenta que al momento de generar el Entity Bean se realiza un mapeo objeto relacional (ORM), de forma transparente al desarrollador, lo que facilita el acceso a los datos. El otro componente es la llamada Unidad de Persistencia (P.U), que permite a través de la clase Entity Manager, realizar la conexión y transaccionalidad hacia la base de datos usando los API JPA y JTA respectivamente. Esto es importante porque facilita al desarrollador acceder al modelo del dominio a través de métodos predefinidos sin preocuparse de la complejidad que implica emplear llamadas por medio de JDBC.

* *Capa de Lógica del Negocio:* En esta capa se usará el patrón DAO, por medio de un Stateless Session Bean, el cual es una clase que contendrá la implementación de los principales métodos CRUD que serán expuestos a través del Facade. La implementación se realizará en una clase genérica llamada Abstract Facade.

**FrontEnd:**

Está compuesta de 3 capas las cuales son:

* *Capa Cliente:* Corresponde a donde estará el web browser elegido por el usuario para interactuar con el sistema.
* *Capa Presentación:* Corresponde a las diferentes páginas generadas en JSP y usando etiquetas de HTML5 y Twitter Bootstrap que contienen los formularios y las vistas principales del sistema.
* *Capa Lógica de la Aplicación:* Acá se tendrá las clases Servlet que actuarán como un controlador de las vistas y genera una comunicación con los métodos del DAO por medio de la inyección de dependencias usando la anotación @EJB.

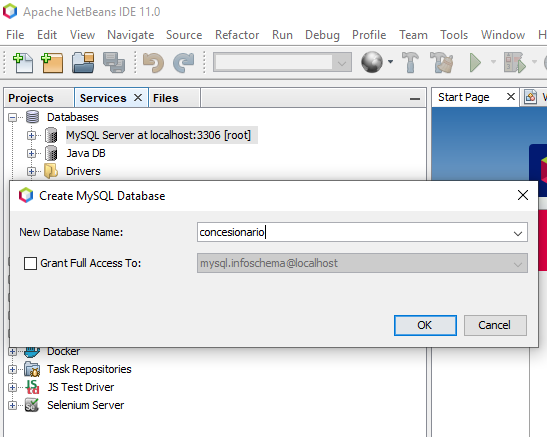
****

**Figura 1: arquitectura de la aplicación.**

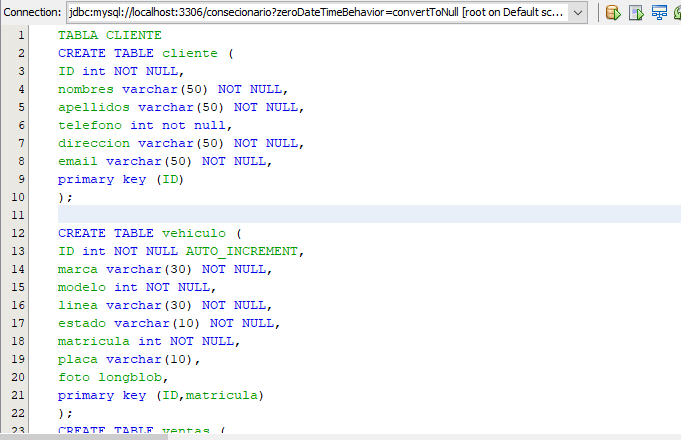
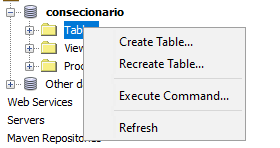
1. **Procedimiento**

**Configurar la Base de Datos.**

Se crea la base de datos utilizando el motor MySQL.

****

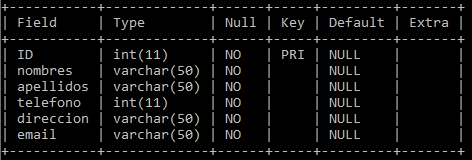
Luego de conectar la base de datos creada anteriormente, se procede a crear las tablas, para ello desplegamos el nodo de nuestra base de datos y le damos click derecho sobre **Tables** para luego elegir **Execute Command**, y a continuación pegamos el código para crear nuestras tablas.



Con esto queda creada nuestra bases datos que tiene la siguiente estructura:

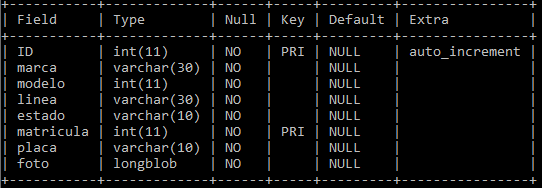
**Tabla Clientes:**

En esta tabla están almacenados todos los clientes del concesionario, los cuales poseen un Id, nombres, apellidos, teléfono, dirección y email.



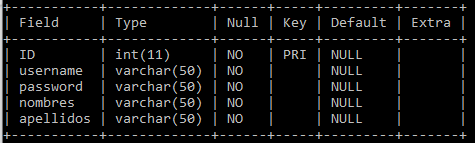
**Tabla Vehículo:**

Esta tabla contiene los vehículos que pasan por el concesionario, tanto los que están disponibles para la venta, como los que ya se han vendido, cada vehículo cuenta con un ID, la marca y línea del vehículo, el modelo, la matrícula (número único ante el organismo de tránsito), placa (también única), el estado (Disponible o vendido) y la foto del vehículo.

****

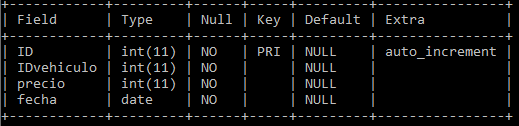
**Tabla Vendedores:**

En esta tabla se almacenan los vendedores, que este caso son los usuarios con acceso a la aplicación, cada uno cuenta con un ID, username (único), password, nombres y apellidos.



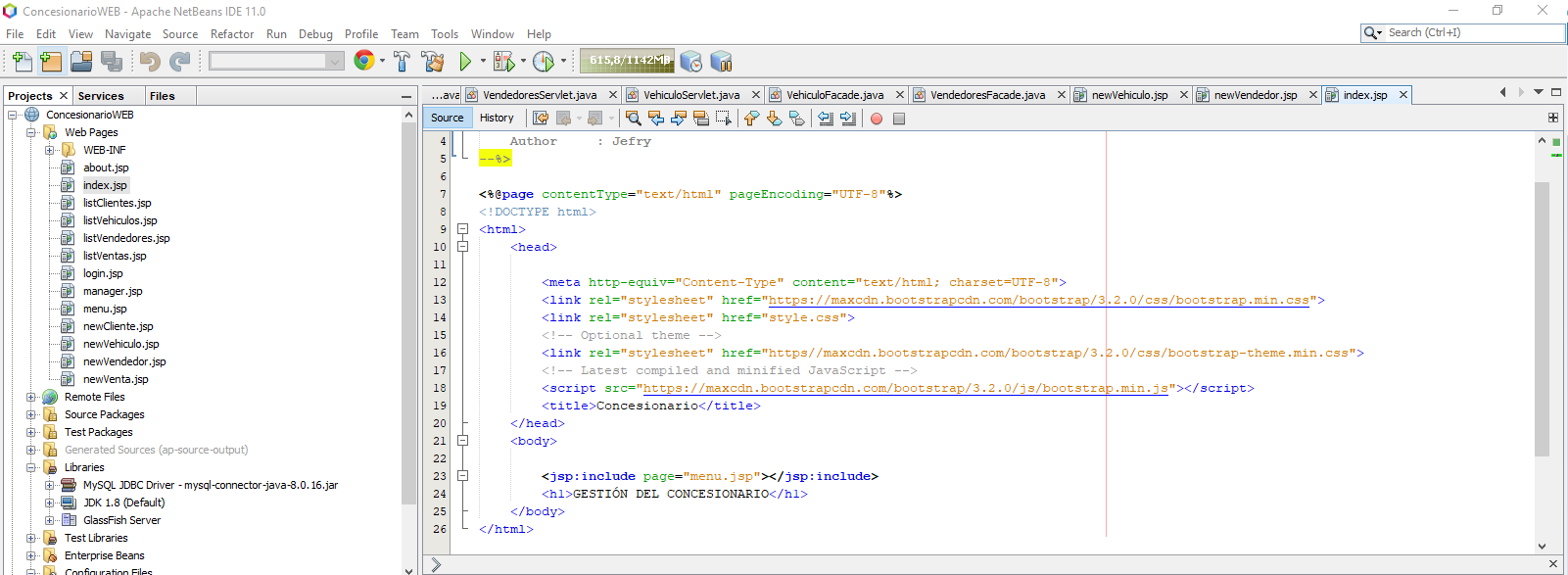
**Tabla Ventas:**

Esta tabla contiene todas las ventas realizadas por el concesionario y cuenta con el ID de la venta, el ID del vehículo vendido, el precio y la fecha de la venta.



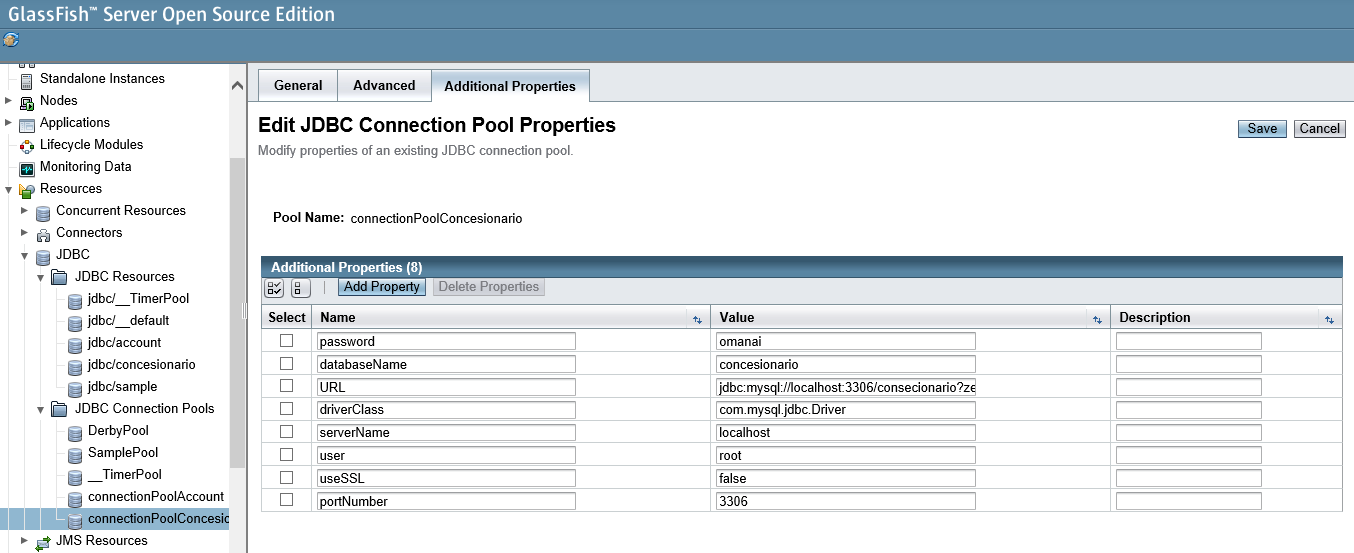
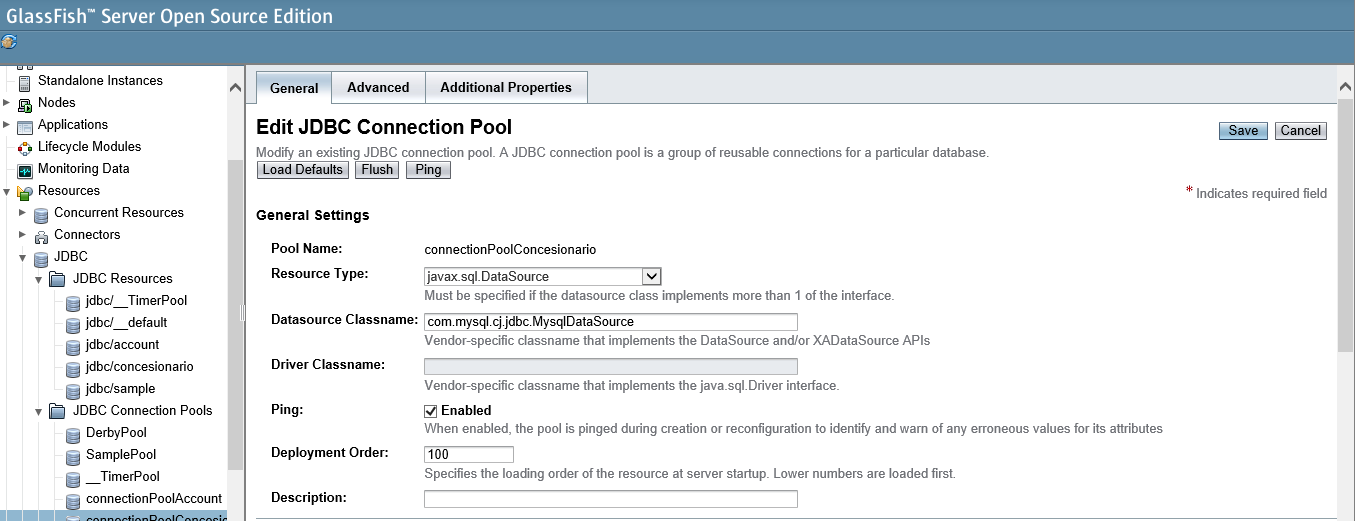
**Creando la aplicación.**

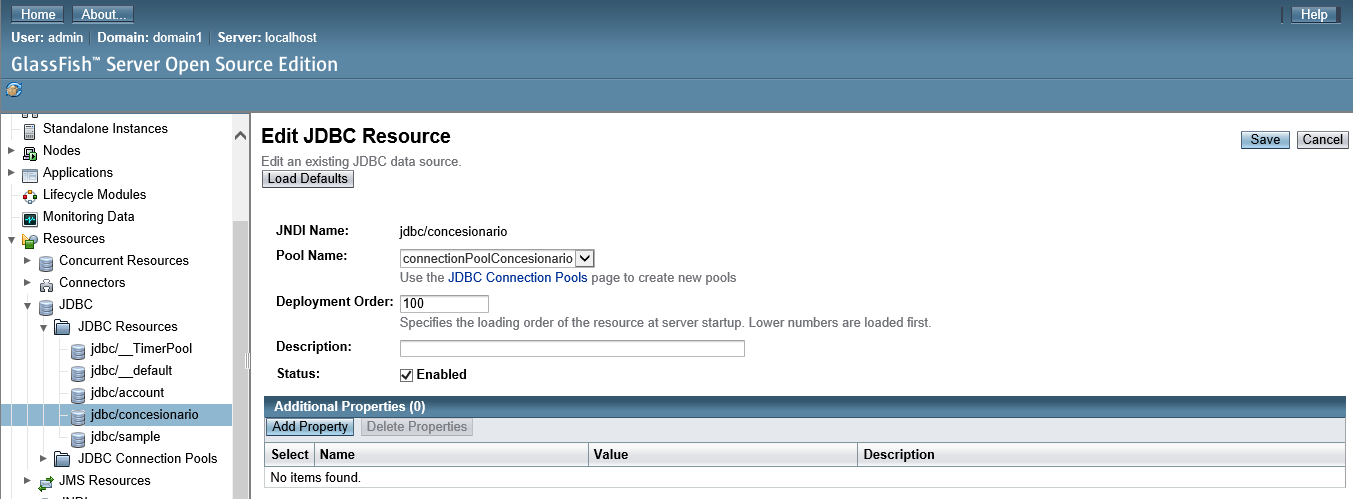
Se siguen los pasos enunciados en la guía y se crea el proyecto tipo Web Application y se nombra ConcesionarioWeb.



**Creando un pool de conexiones a la BD.**

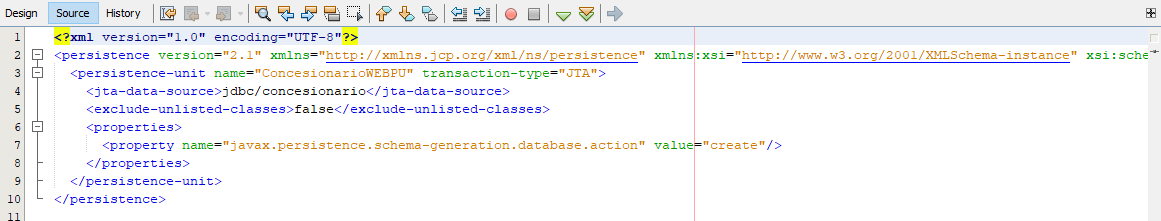
Se siguen los paso descritos en al guía y se crea el siguiente pool de conexiones así como su respectivo resource:





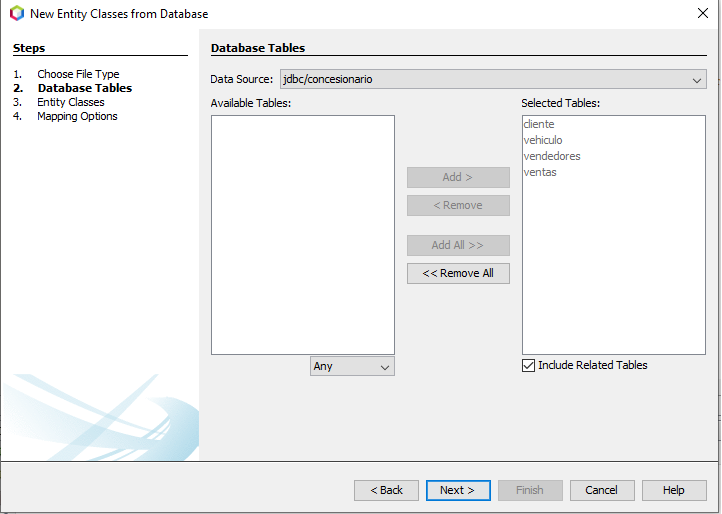
**Creación de la Unidad de Persistencia**

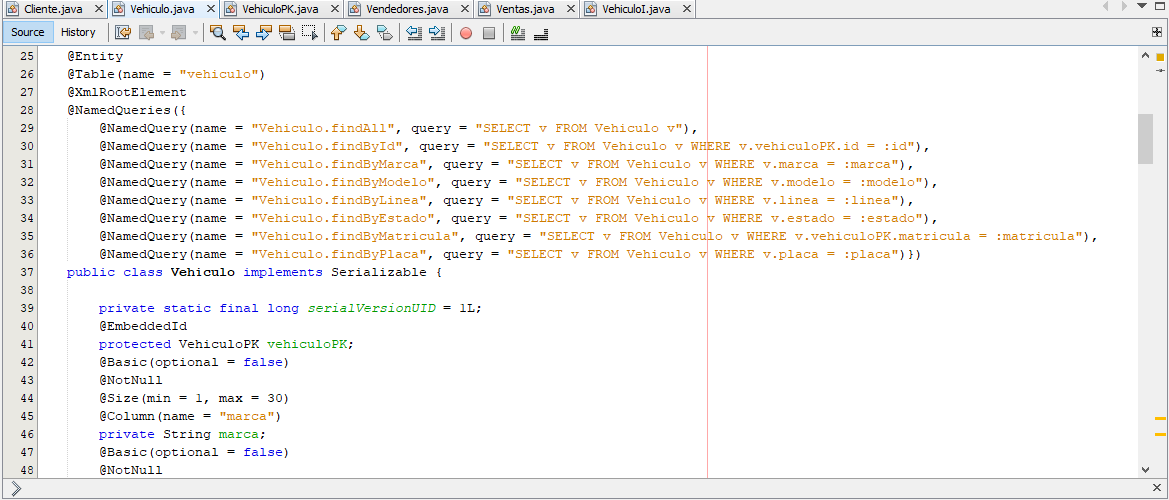
Se siguieron los pasos de la guía y se creó la siguiente unidad de persistencia:



**Creación del Entity Bean.**

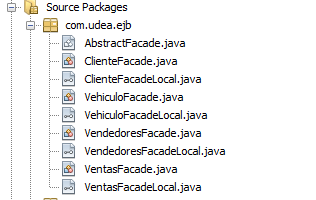
Se crearon los entity bean de cada una de las tablas, tal y como se especifica en la guía, obteniendo las siguientes entidades:



****

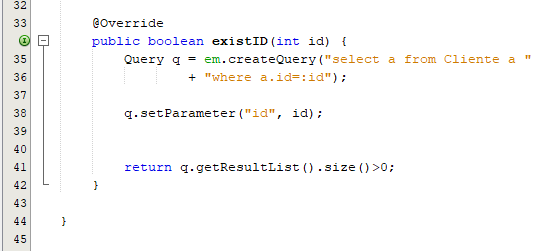
**NOTA:** La clase VehiculoI se agregó para poder implementar el método que muestra las fotografías de los vehículos en el formulario.

Lugo se crean los Session Bean correspondientes a las anteriores Entity:

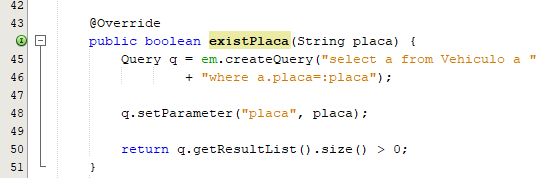


Y siguiendo los mismo pasos que en la guía se crearon varios métodos de negocio como enunciamos a continuación:

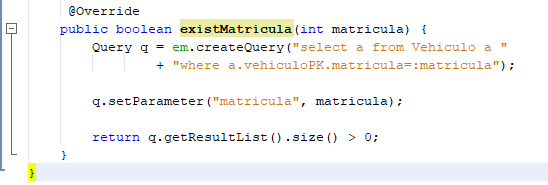
**exitID:** Este método nos permite verificar si el Id ingresado ya existe en la base de datos**.** Este método fue implementado en lasSession Bean de todas la entidades, exceptuando la ve ventas, ya que en eta el Id es generado automáticamente.



**existPlaca:** Este método permite verificar si la placa ingresada para un nuevo vehículo ya exite, visto que cada placa debe ser única, este método fue implementado en la case vehículo.

****

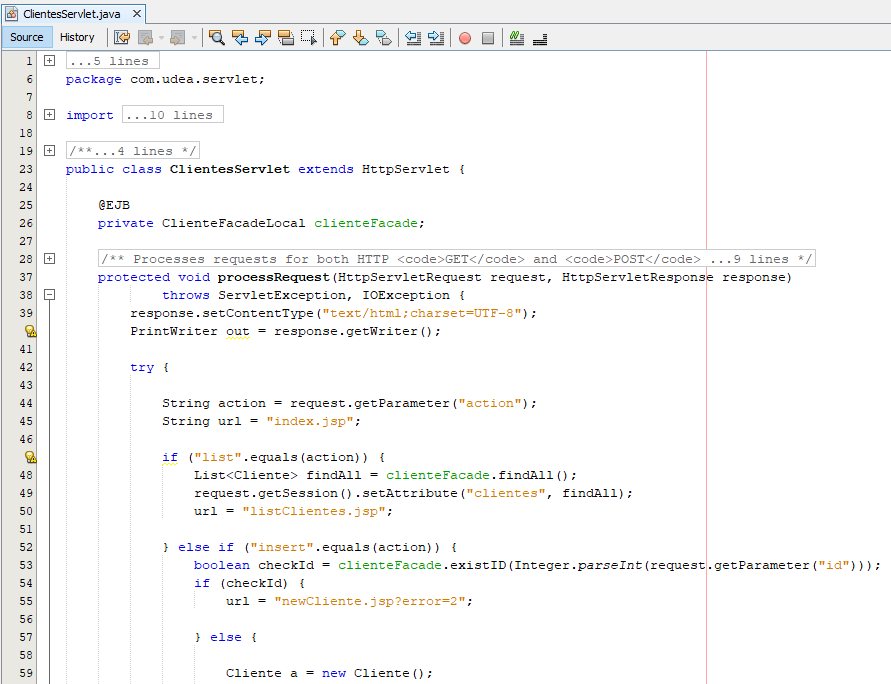
**existMatricula:** Este método permite verificar si la matrícula ingresada para un nuevo vehículo ya exite, visto que cada matrícula debe ser única, este método fue implementado en la case vehículo.

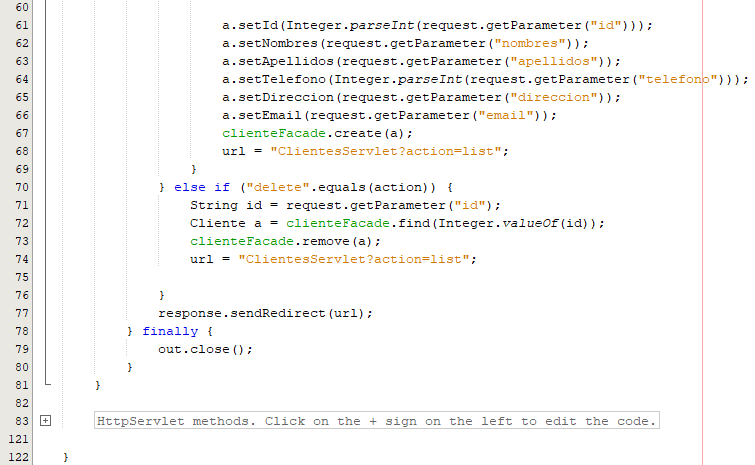
****

**Creación de los Servlet.**

Siguiendo los pasos de la guía se crearon los servlets correspondientes a cada vista de la aplicación:

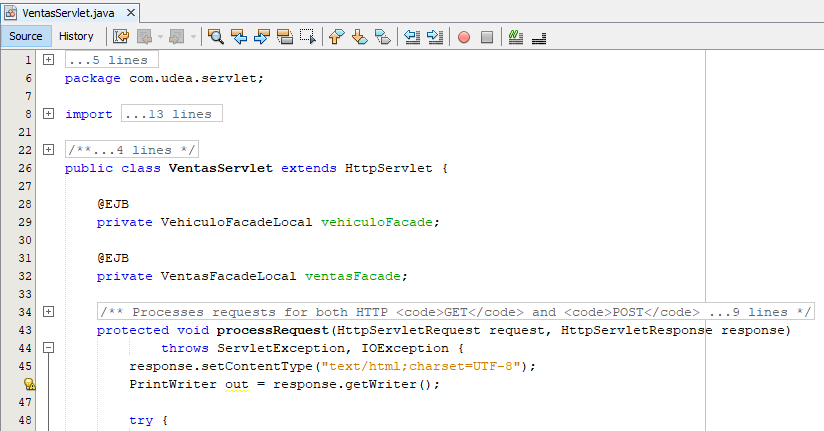
*Servlet para Clientes:*

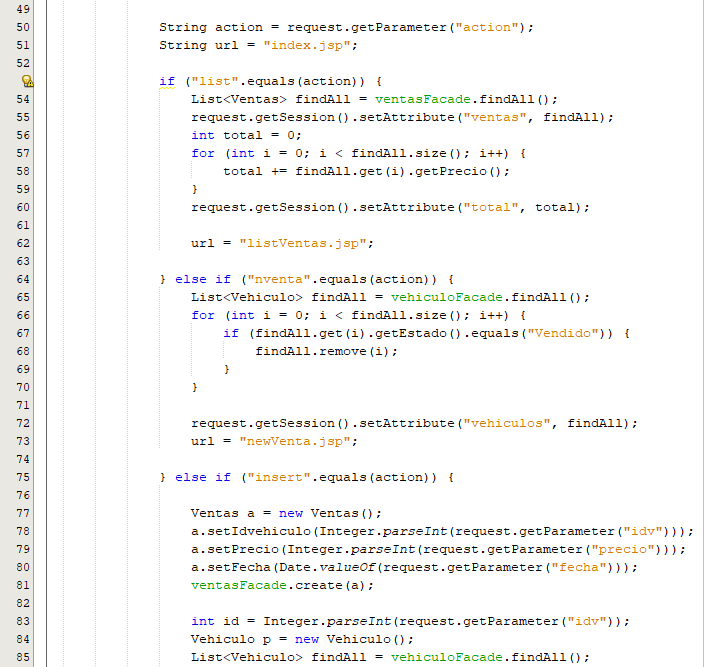




Este servlet se encarga de procesar lo referenciado a los métodos de negocio referentes a los clientes, como listar, insertar y borrar clientes. Cuando se intenta crear un nuevo cliente, primero se hace el chequeo para ver si el Id ingresado ya existe.

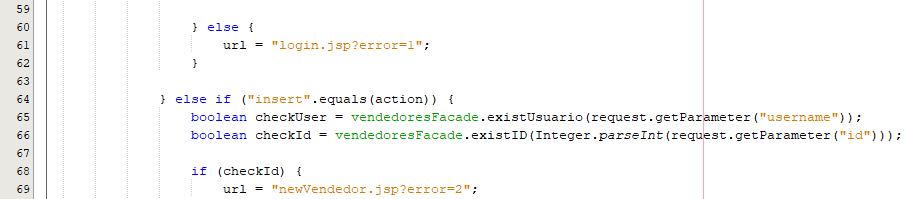
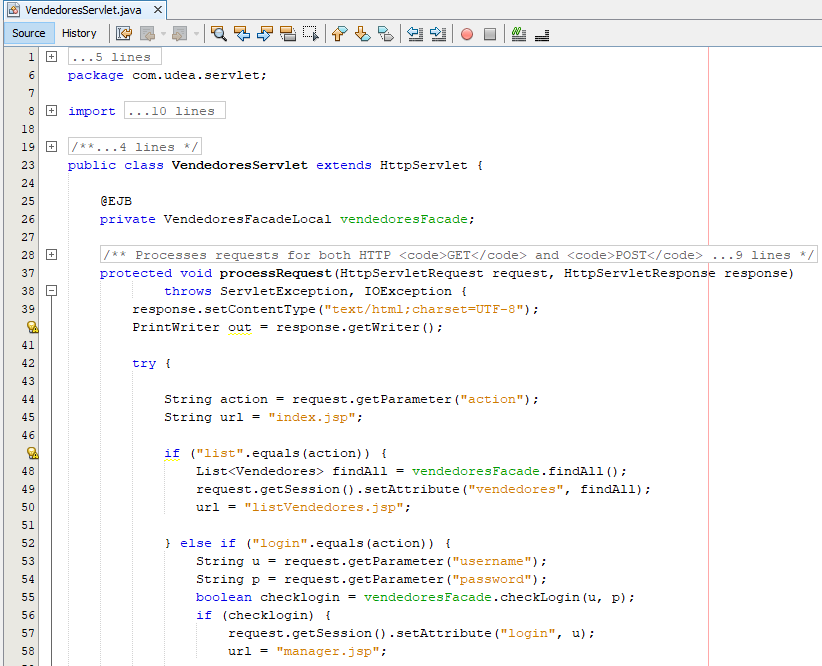
*Servlet para Ventas:*

**

**

Este servlet se encarga de procesar lo referenciado a los métodos de negocio referentes a las ventas, como listar e insertar. Cuando se intenta registrar una nueva venta, primero se buscan los vehículos que estén disponibles para la venta y estos son pasados a la vista, si no existen vehículos con el campo **estado = Disponible** no se puede ingresar una nueva venta. En caso de que hayan vehículos disponibles para la venta y está se efectúe, el estado de dicho vehículo pasará a Vendido.

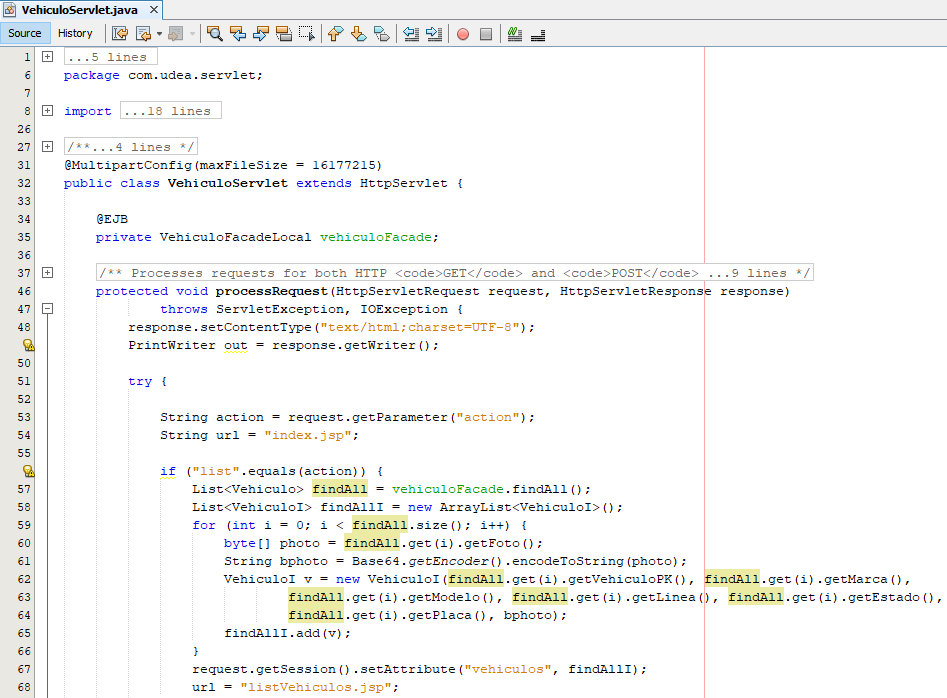
*Servlet para Vendedores:*

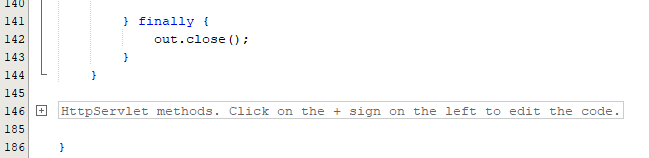
**

**

Este servlet se encarga de procesar lo referenciado a los métodos de negocio referentes a los vendedores, como login, listar, insertar, borrar y logout. Cuando se intenta crear un nuevo vendedor, primero se hace el chequeo para ver si el Id ingresado ya existe, luego se chequea que el usuario tampoco exista. Para realizar el login se utiliza el método **checkLogin** para verificar que el usuario y password sean correctos.

*Servlet para Vehículos:*

**

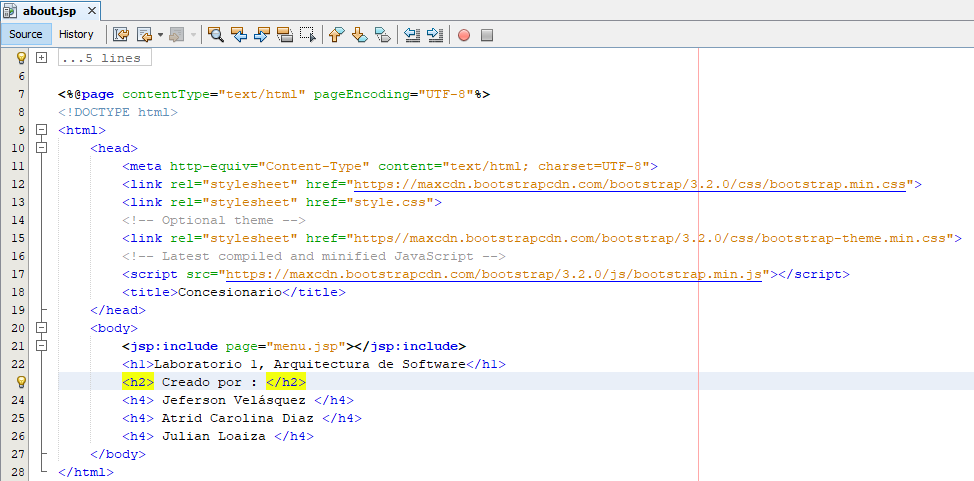
**

Este servlet se encarga de procesar lo referenciado a los métodos de negocio referentes a los vehículos, como listar, insertar, buscar y borrar vehículos. Cuando se intenta crear un nuevo vehículo, primero se hace el chequeo para ver si el Id ingresado ya existe igual que con la placa y la matrícula. Luego se crea el vehículo con su estado en **Disponible** y se almacena en la base de datos su respectiva fotografía. Cuando se quieren visualizar vehículos tanto en el apartado de listar como en buscar, se hace uso de clase Vehículo, la cual tiene un atributo de tipo String llamado bfoto, el cual utilizamos para poder visualizar la foto almacenada en la base de datos que es de tipo longblob.

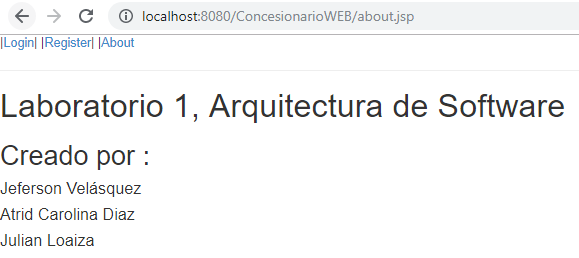
**Creación de Vistas JSP.**

Se crearon las siguientes vistas para la aplicación:

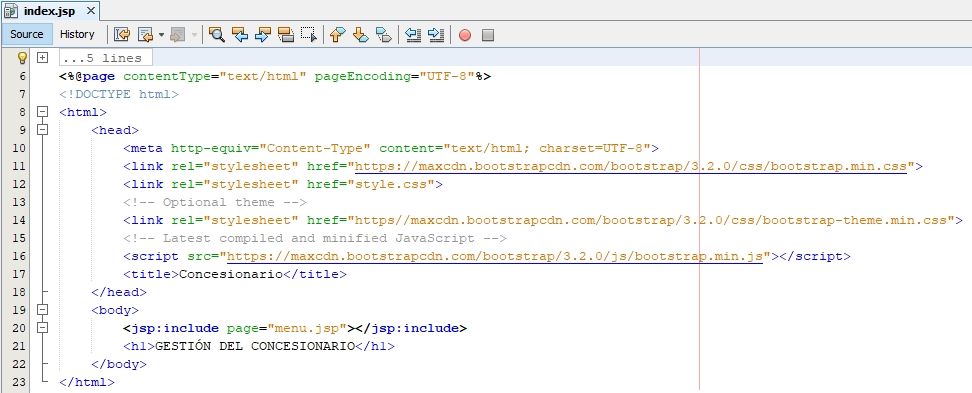
*about.jsp:*

**

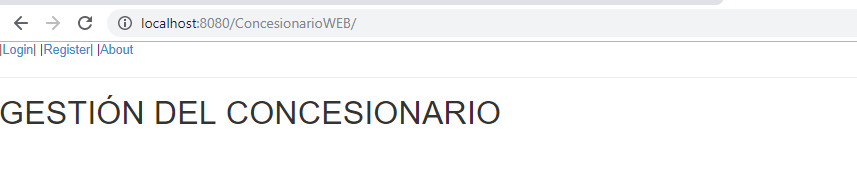
Vista que muestra la información acerca de los integrantes del grupo:



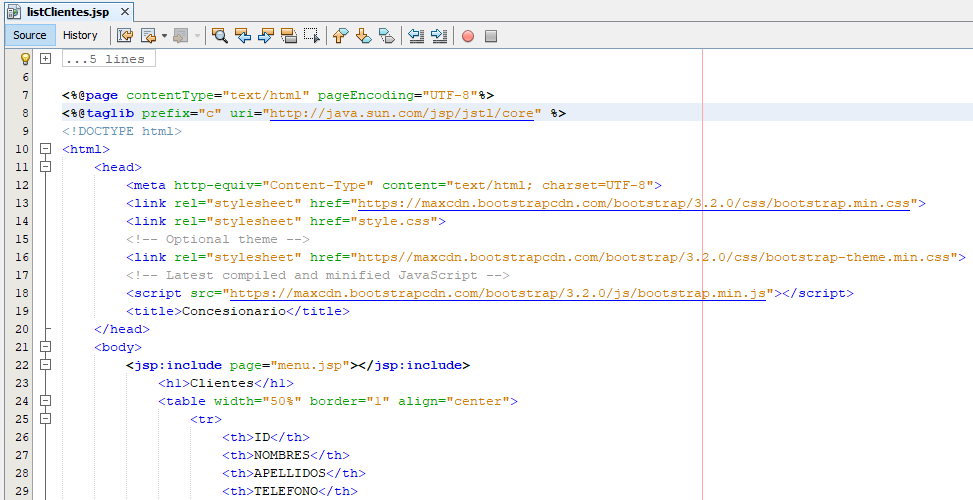
*index.jsp:*

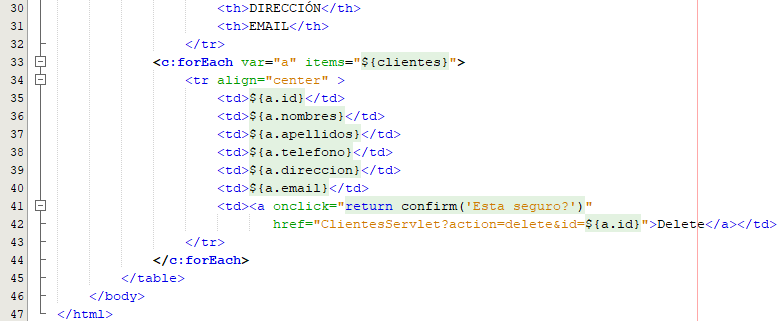
**

Vista principal de aplicación:

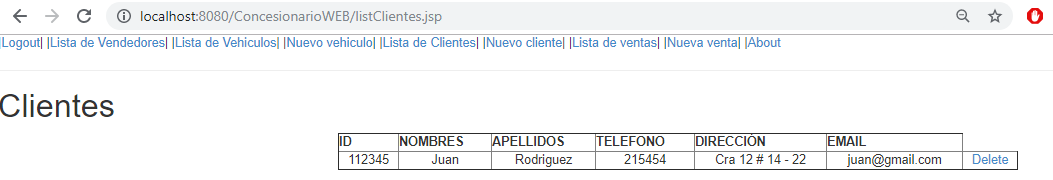


*listClientes.jsp:*

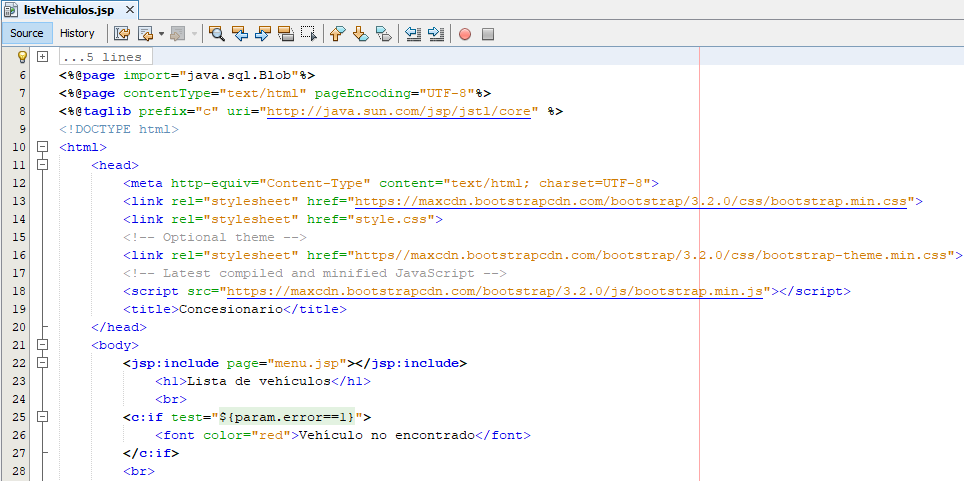
**

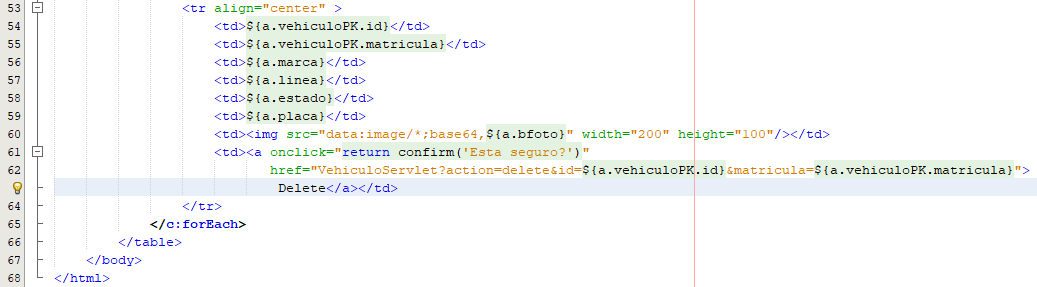
**

Esta vista se encarga de mostrar el listado de los clientes actuales en el concesionario, dándonos la oportunidad de borrar alguno:

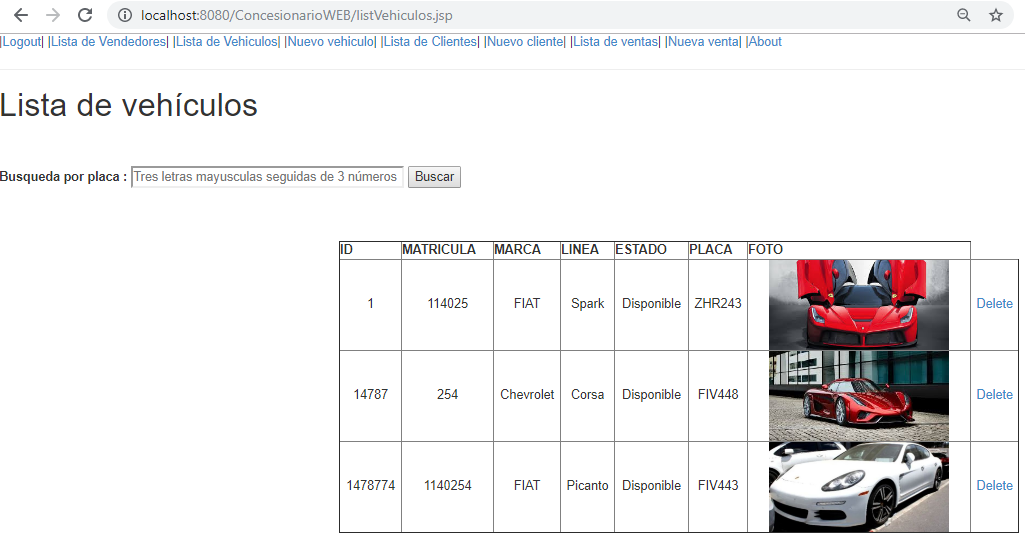


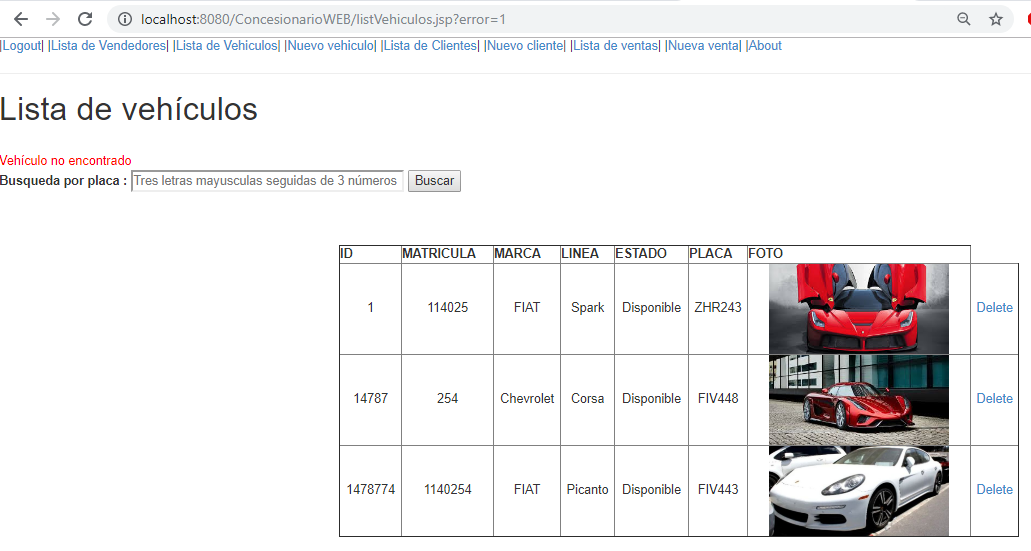
*listVehiculos.jsp:*



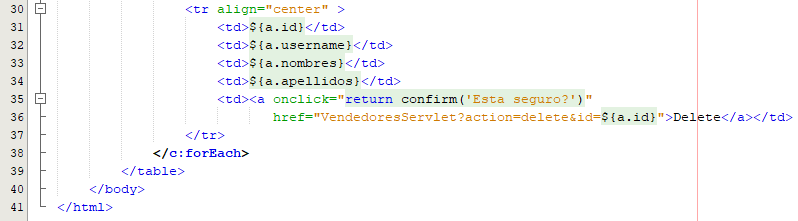
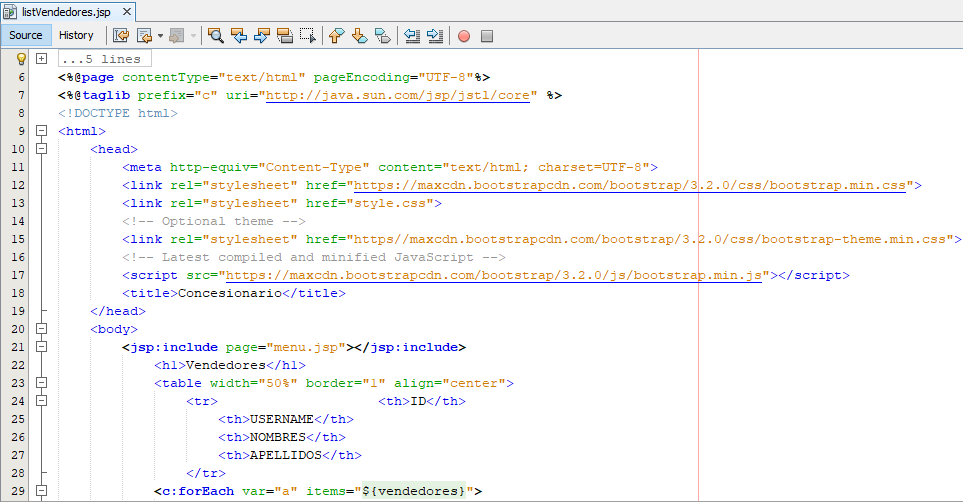


Vista que nos permite ver el listado de vehículos en el concesionario, tanto disponibles como vendidos, en esta vista podemos buscar a través del campo de texto alguna placa en específico, en caso de que dicho vehículo no exista, la vista nos lo indicará y nos mostrará todos los vehículos disponibles:

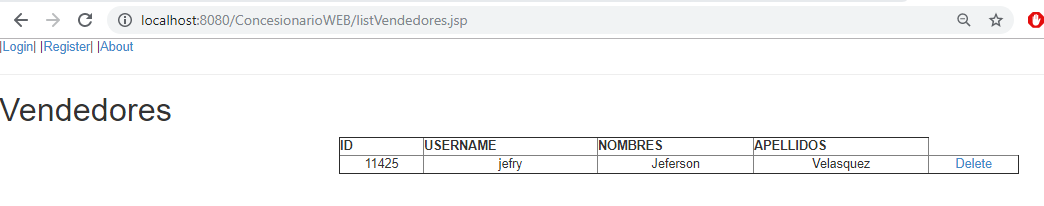




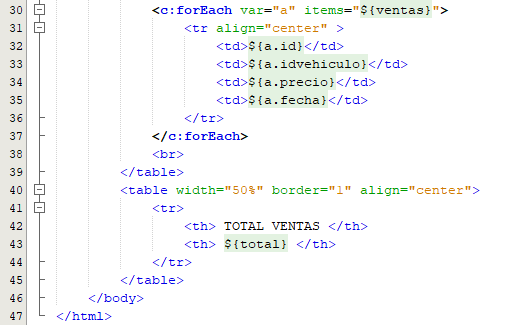
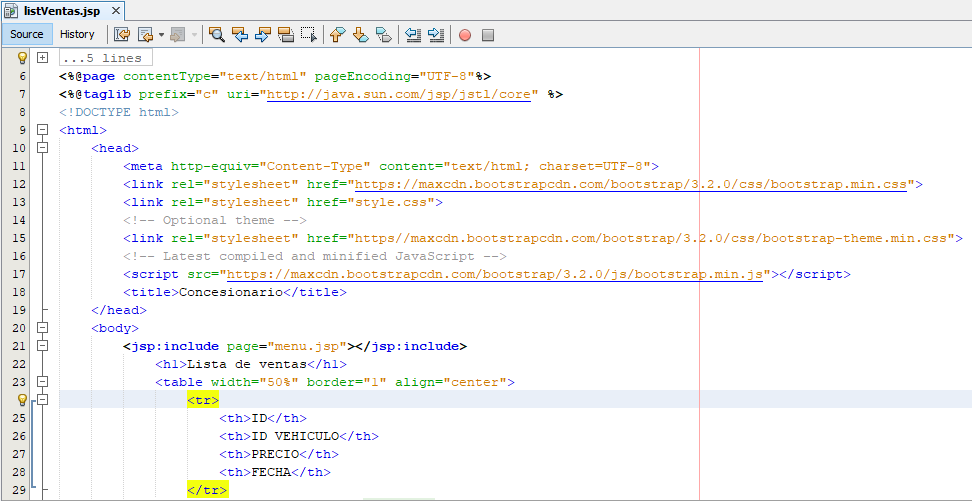
*listVendedores.jsp:*

**

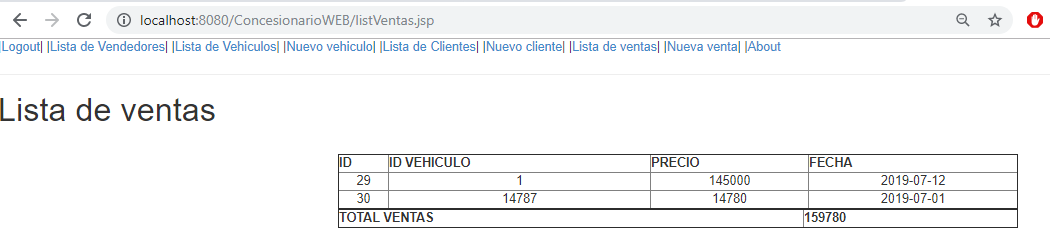
Vista que muestra la lista de vendedores disponibles, los cuales con lo que pueden hacer login en la app:

**

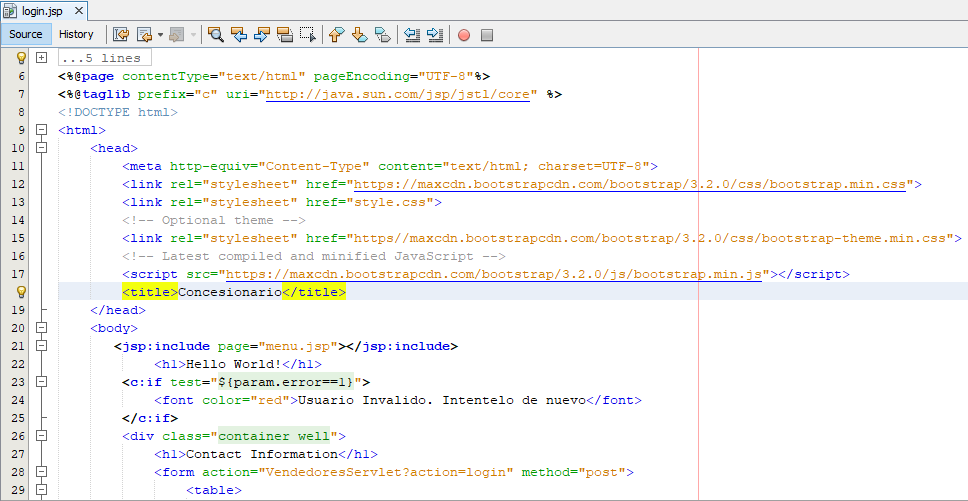
*listVentas.jsp:*

**

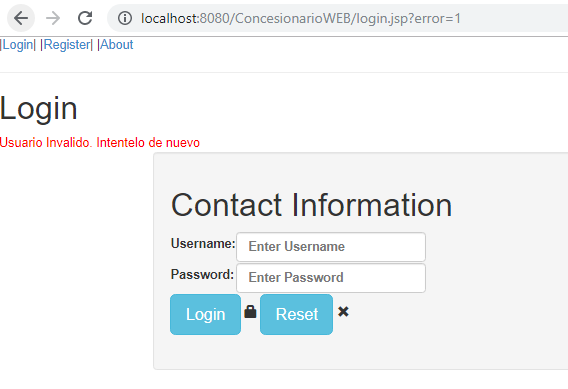
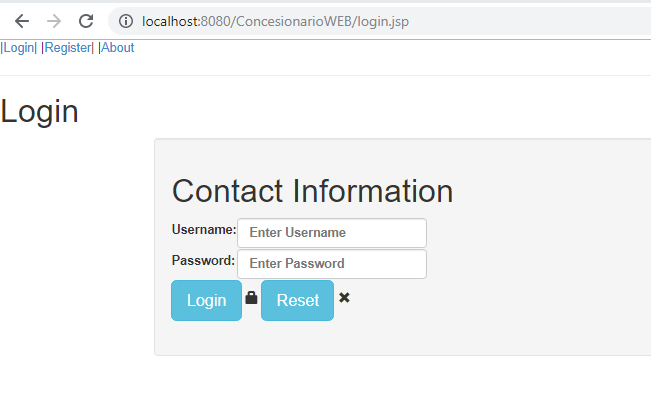
Vista que muestra la lista de ventas realizadas, las cuales no se pueden eliminar, al final de la vista no muestra el total de ventas:



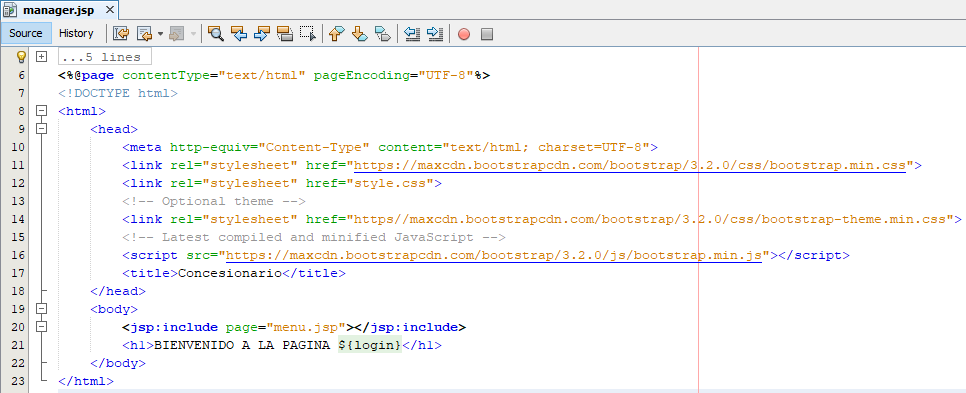
*login.jsp:*



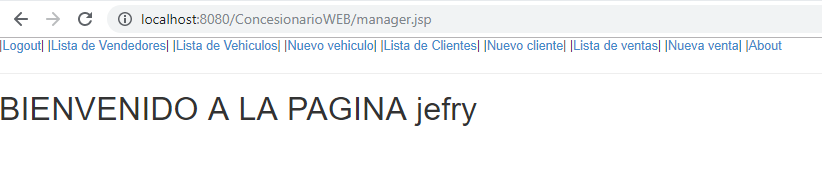
Esta vista es la que nos permite realizar el login en la app y así poder hacer uso de ella si ingresamos datos erróneos nos lo indicará:



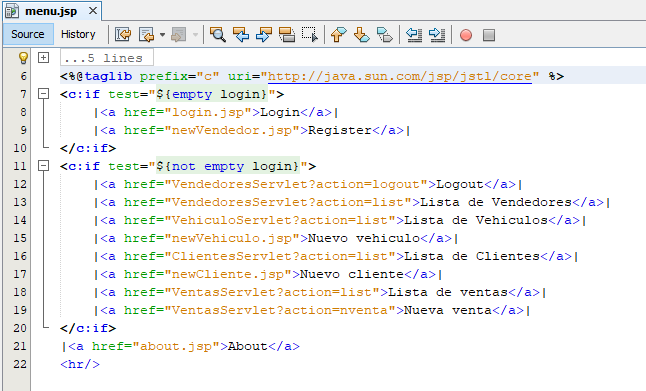
*manager.jsp:*

**

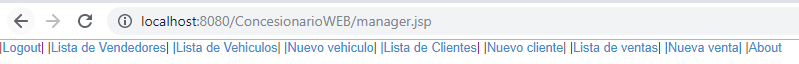
Vista que nos da la bienvenida a la app, después de loguearnos:

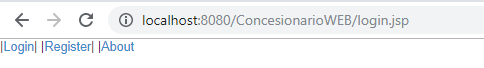


*menu.jsp:*

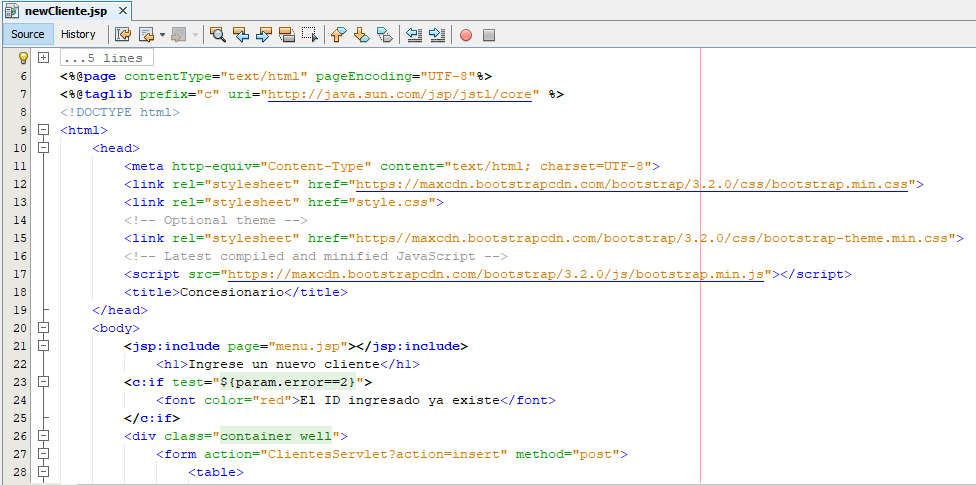


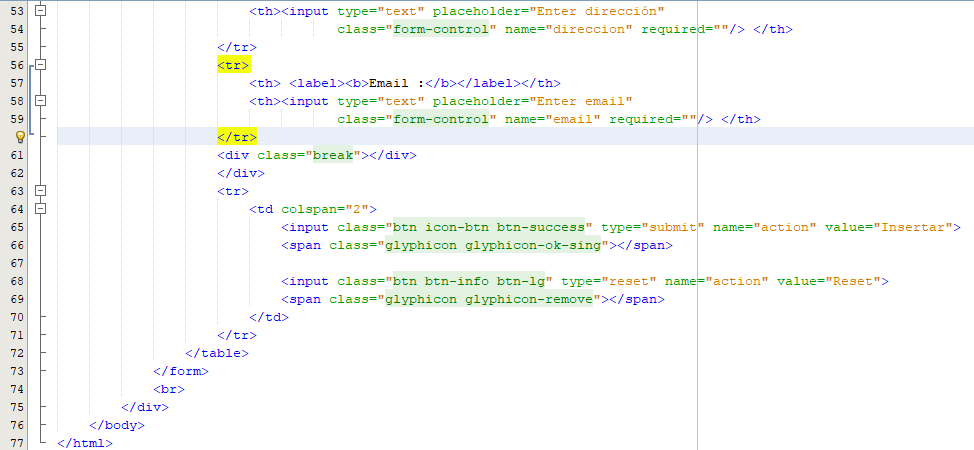
Vista incrustada en todas las vista, la cual nos permite ver el menú de acciones en la parte superior de la página, muestra un menú dependiendo si estamos logueados o no:





*newCliente.jsp:*

**

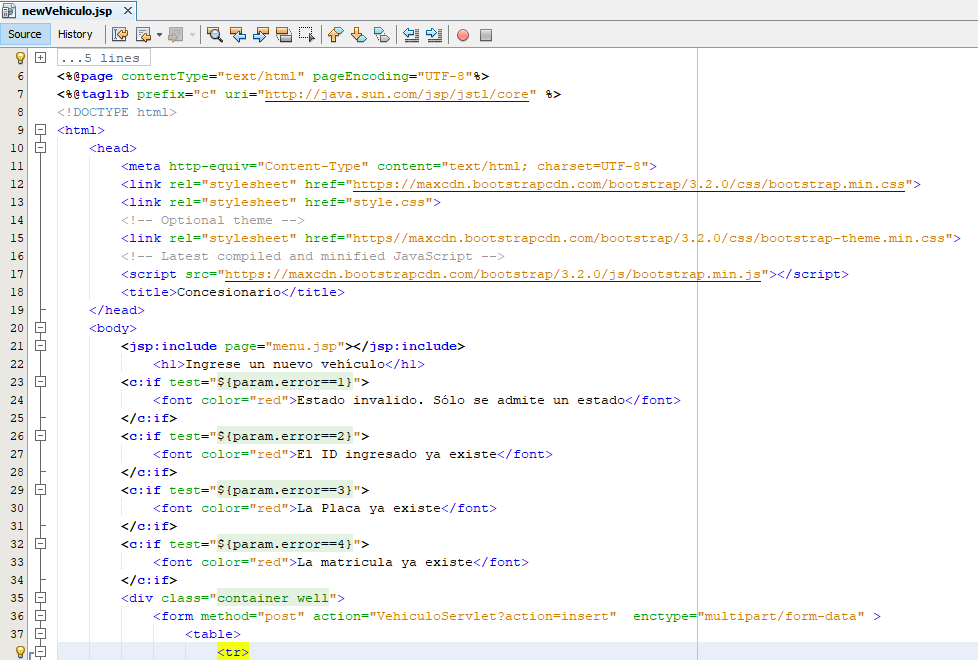
**

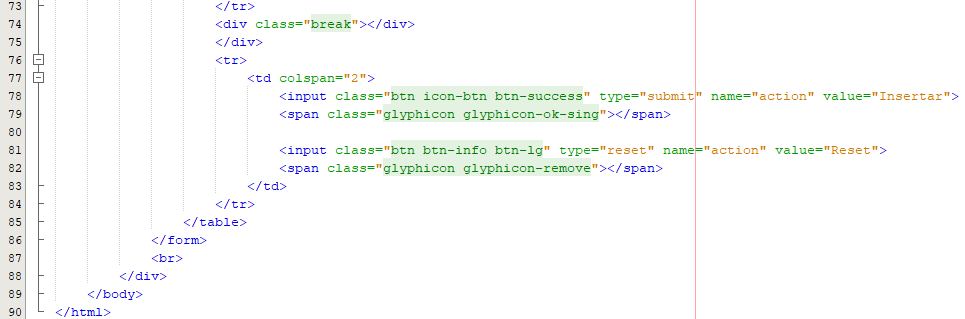
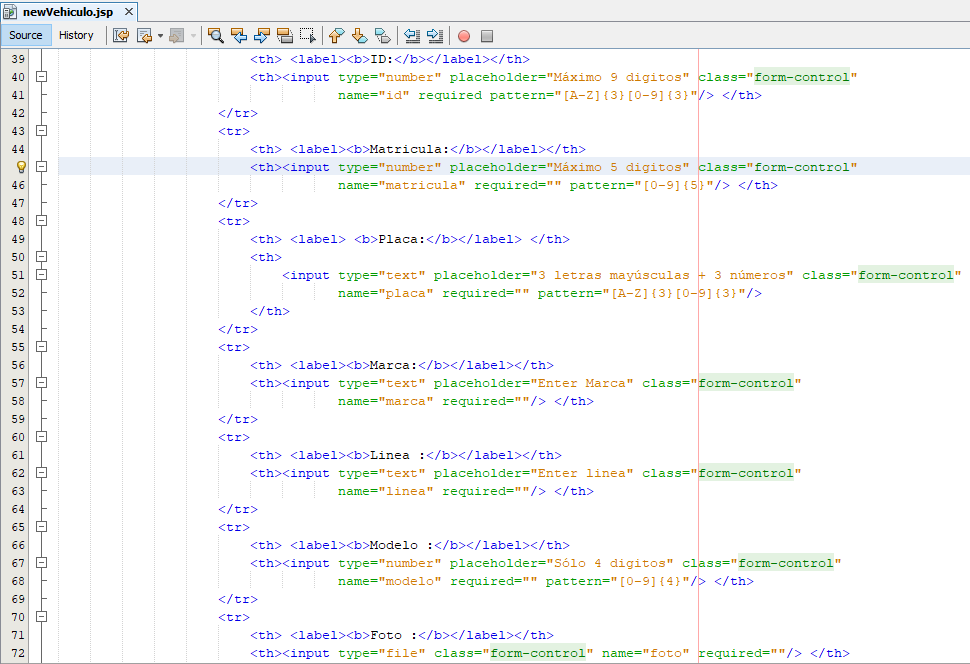
Vista que nos permite ingresar un nuevo cliente, la pagina realiza las siguientes validaciones: el id no puede ser un número mayor a 9 dígitos, el teléfono no puede tener más de 8 dígitos, todos los campos son obligatorios. Si el id que ingresamos ya existe en la base de datos, nos lo hace saber con un mensaje:





*newVehiculo.jsp:*

**

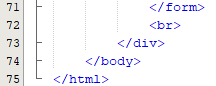
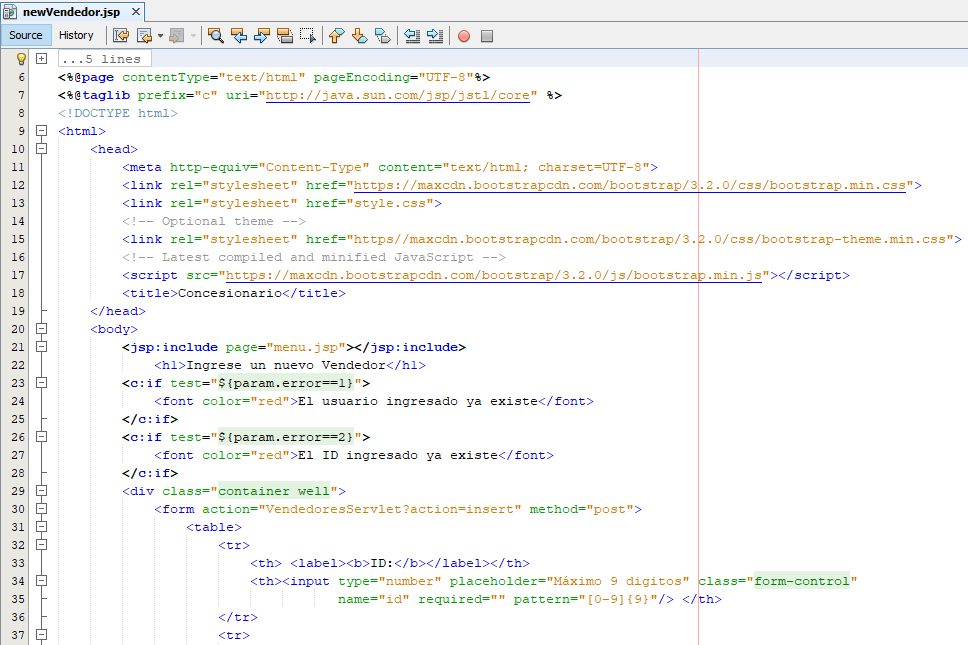
**

Vista que nos permite ingresar un nuevo vehículo, la pagina realiza las siguientes validaciones: el id no puede ser un número mayor a 9 dígitos, la matrícula no puede ser un número de más de 5 dígitos, la placa debe estar conformada por 3 letras mayúsculas seguidas de 3 números, el modelo no puede ser de más de 4 dígitos, todos los campos son obligatorios. Si el id, la matrícula o la placa que ingresamos ya existe en la base de datos, nos lo hace saber con un mensaje, para la foto seleccionamos un archivo tipo imagen de nuestro ordenador:





*newVendedor.jsp:*

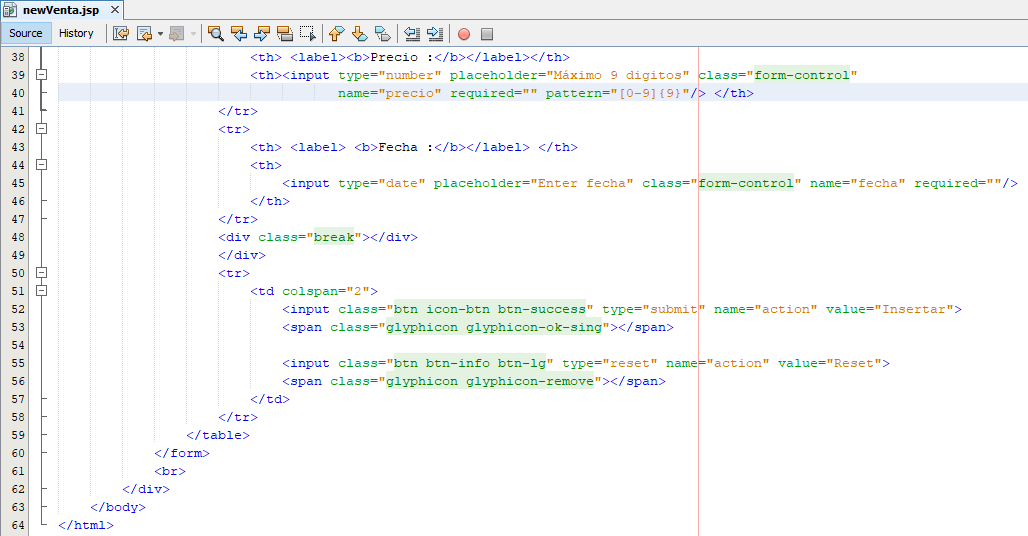
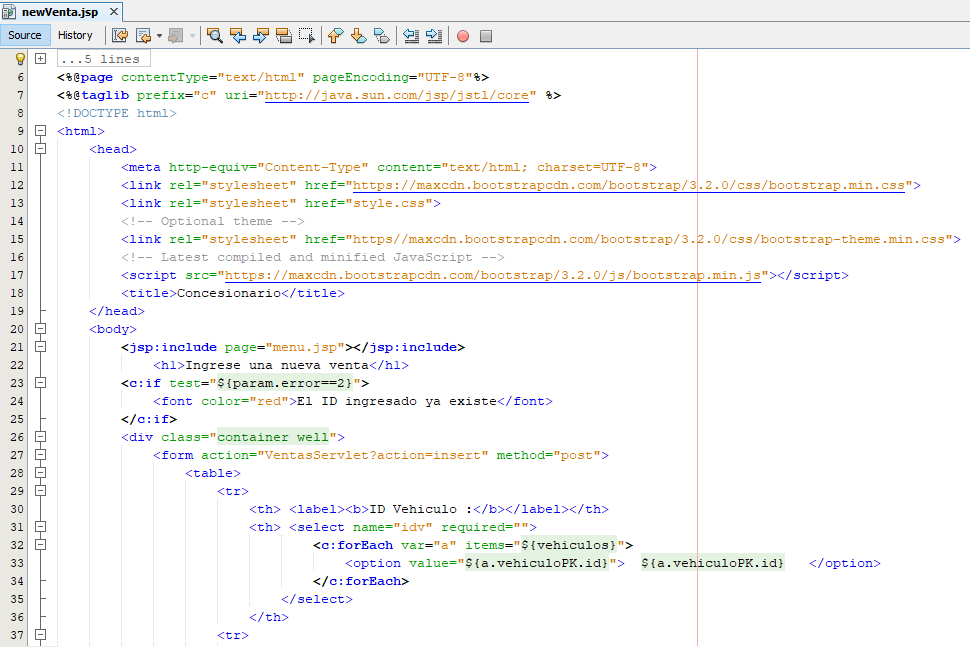


Esta vista nos permite ingresar un nuevo vendedor, es el formulario que se abre al darle en el menú superior en **register** (sin estar logueado)**,**  la página valida que el id no sea de más de 9 dígitos, además nos indica si el id o el username ya existen:





*newVenta.jsp:*



Esta vista nos permite realizar una nueva venta, solo si hay vehículos con estado=disponible, el formulario verifica el el precio de la venta sea de máximo 9 dígitos

: