

EJEMPLO DE REGISTRO E INICIO DE SESIÓN CON JSP, SERVLET Y MYSQL

1) crear la base de datos.

Ver tutorial que hice sobre MySQL WorkBench

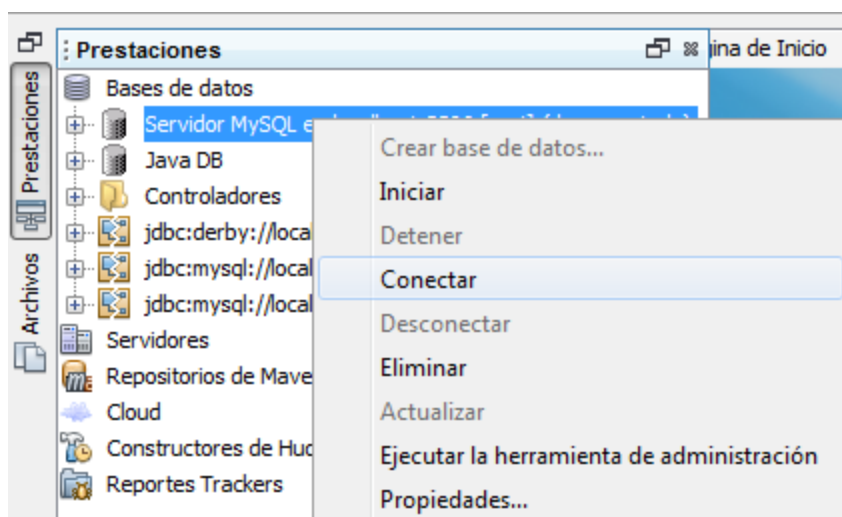
Tabla usuarios (id, password, nombre, apellido, email, tipo)

1). Descargar e instalar el MySQL Server (puede ser por ejemplo mediante XAMPP o WAMP o AppServer)

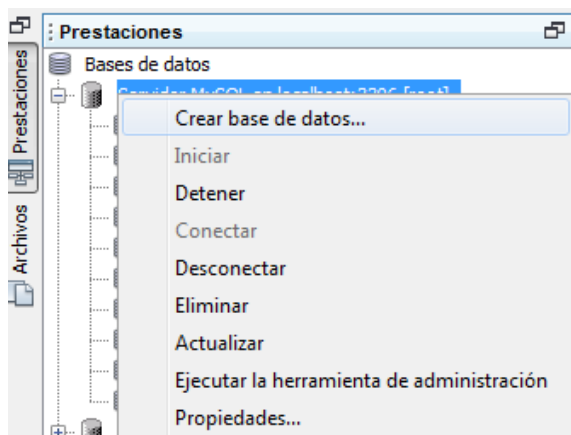
2). Iniciar el servidor MySQL.

3). Descargar, instalar e iniciar el Netbeans

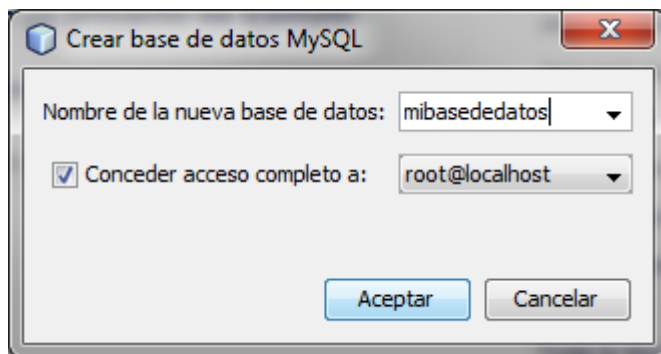
4). ir al panel de servicios (Prestaciones) de Nebeans, seleccionar el servidor MySQL, hacer clic derecho y escoger la opción **Conectar**.



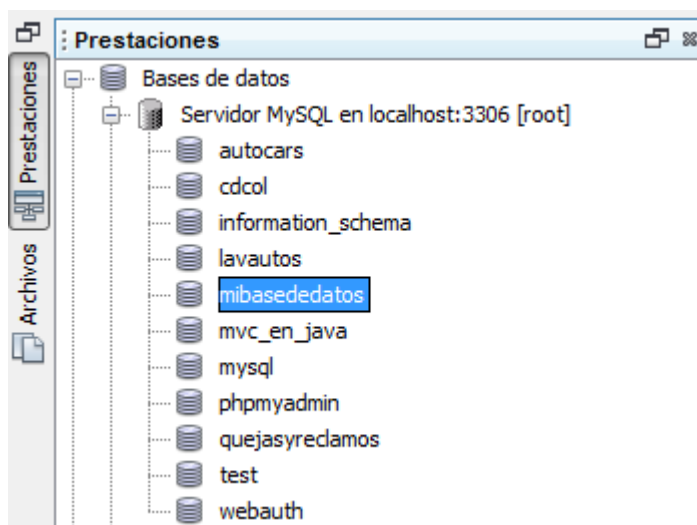
5). clic derecho sobre el servidor y seleccionar la opcion crear base de datos



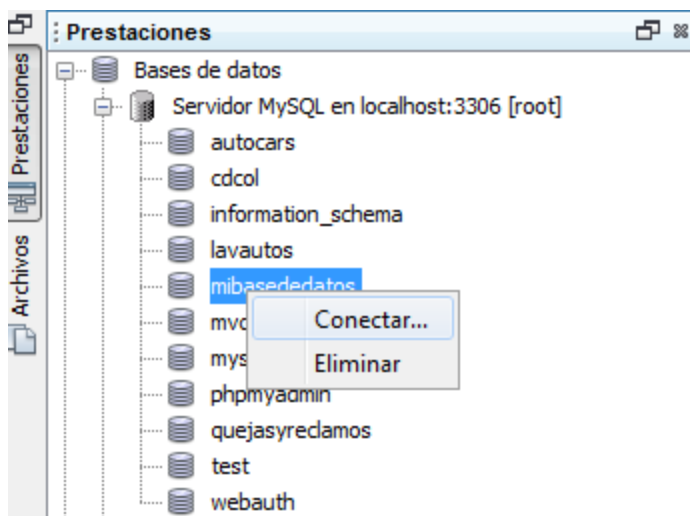
6). darle nombre a la base de datos y escoger el usuario que tendrá permisos sobre la misma



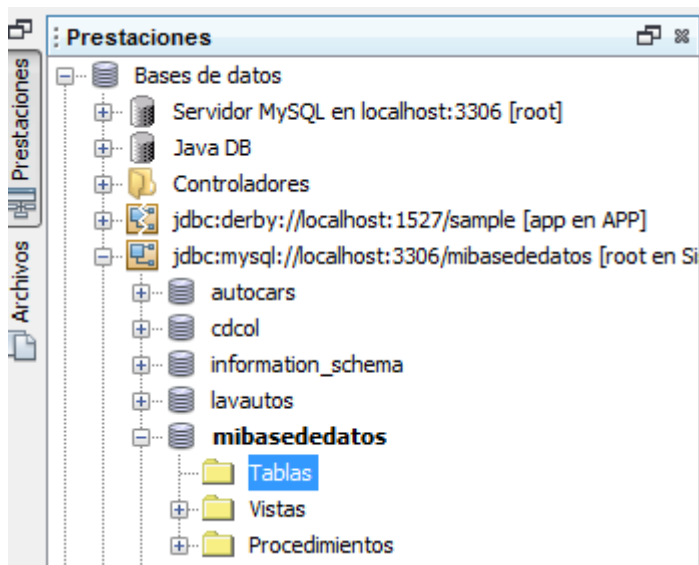
7). observar que la bd es agregada al conjunto de bd.



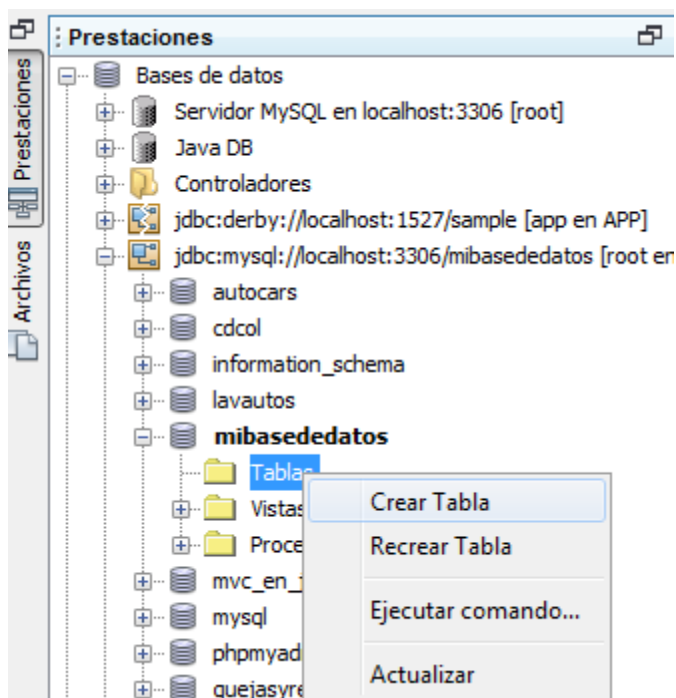
8). clic derecho sobre la nueva base de datos y seleccionar la opción conectar



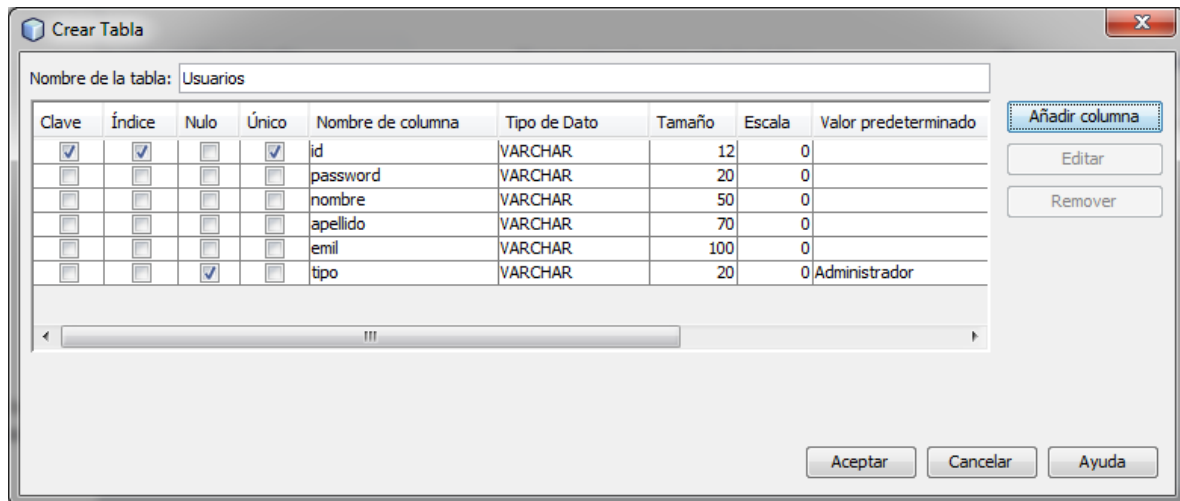
9). Observar que se ha generado una conexión lista para trabajar con la nueva base de datos.



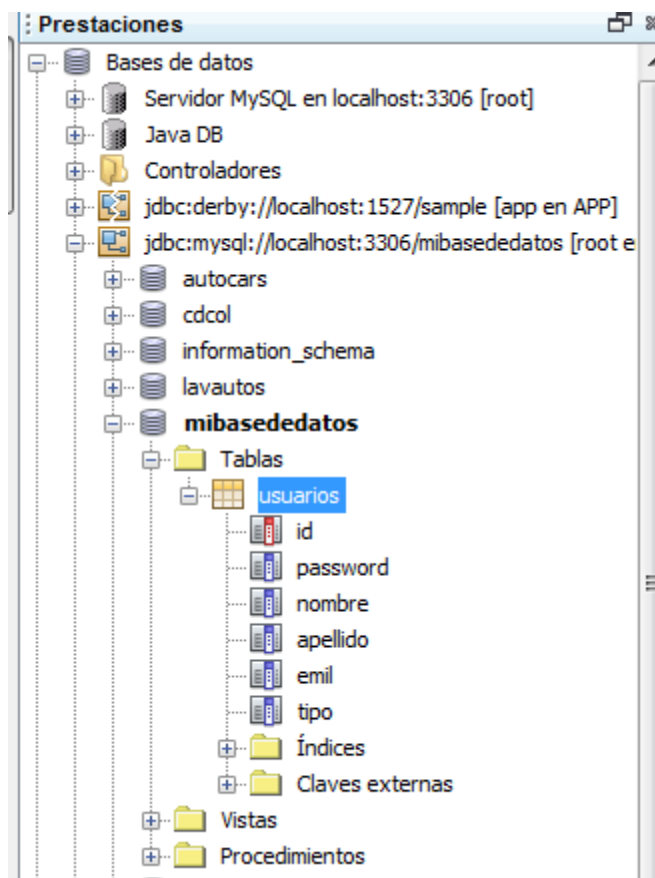
10). Clic derecho sobre la opción tablas de la base de datos creada y seleccionar la opción **Crear Tabla**



11). Agregar en la tabla las columnas necesarias

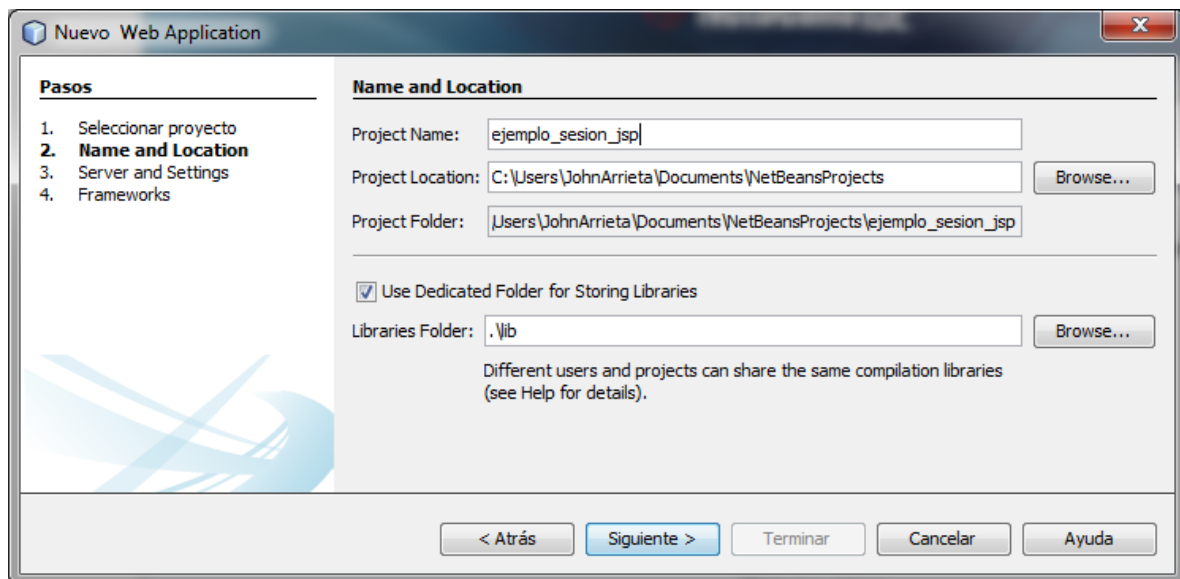
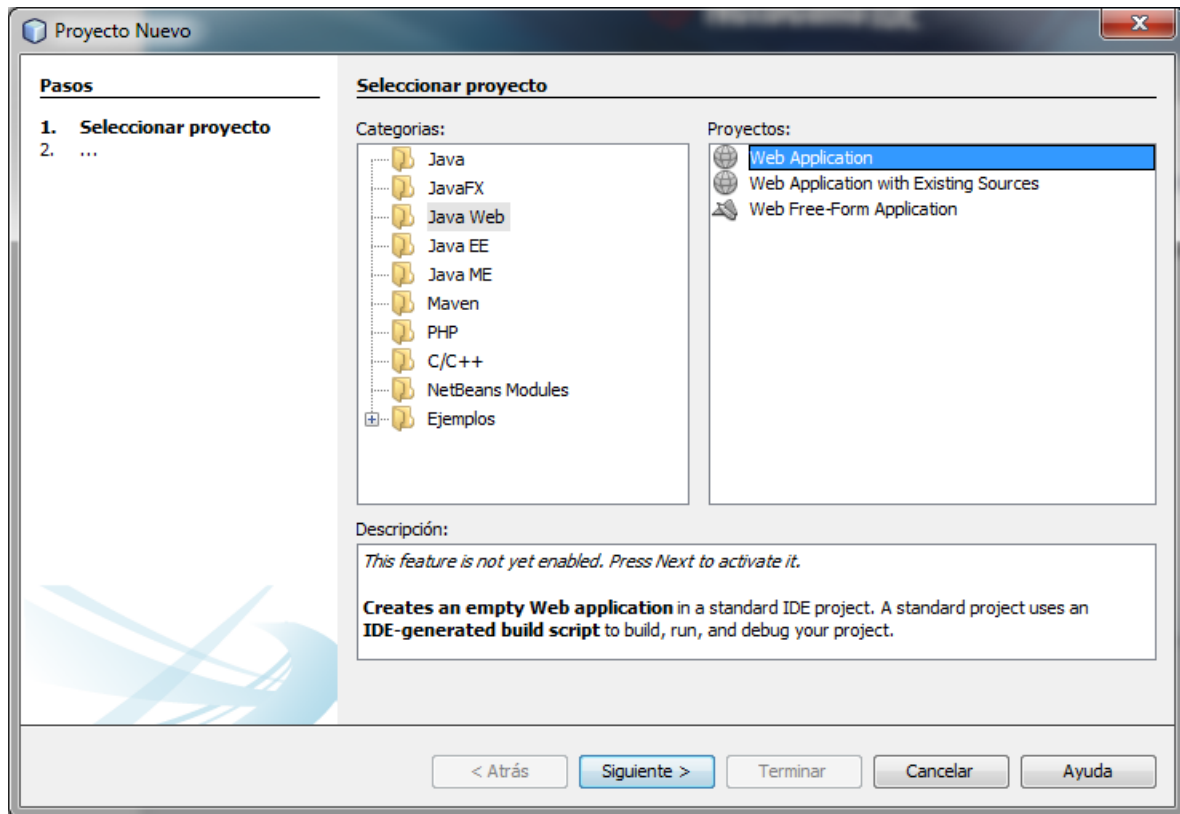


12). observar como la tabla fue creada y agregada a la base de datos

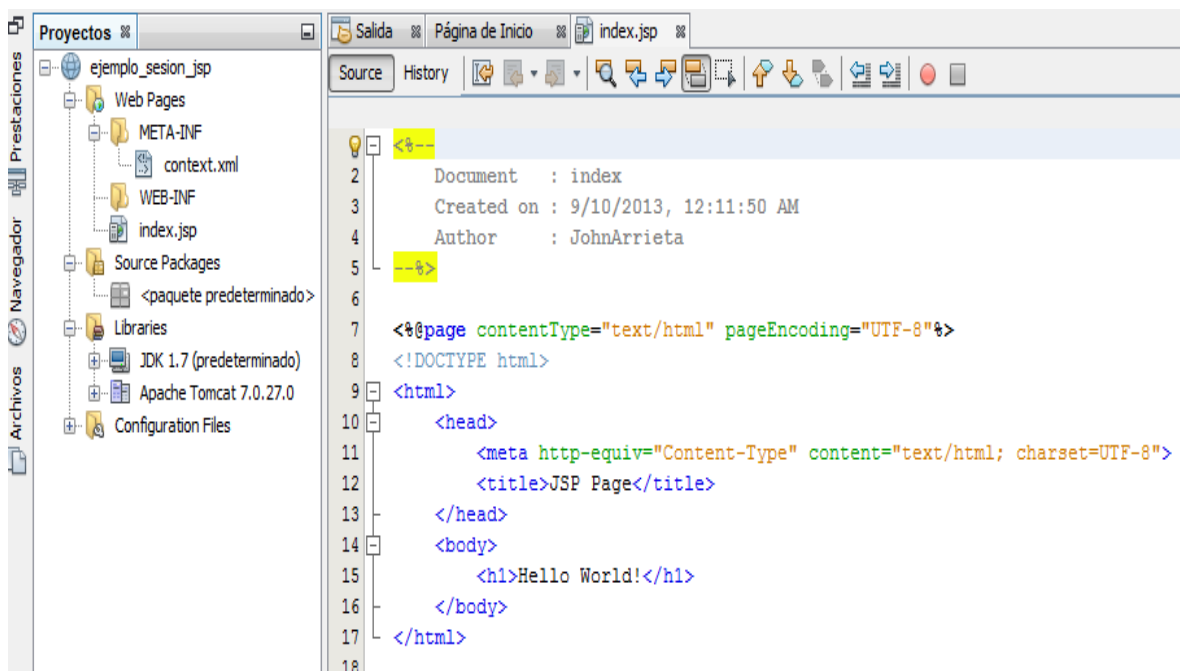
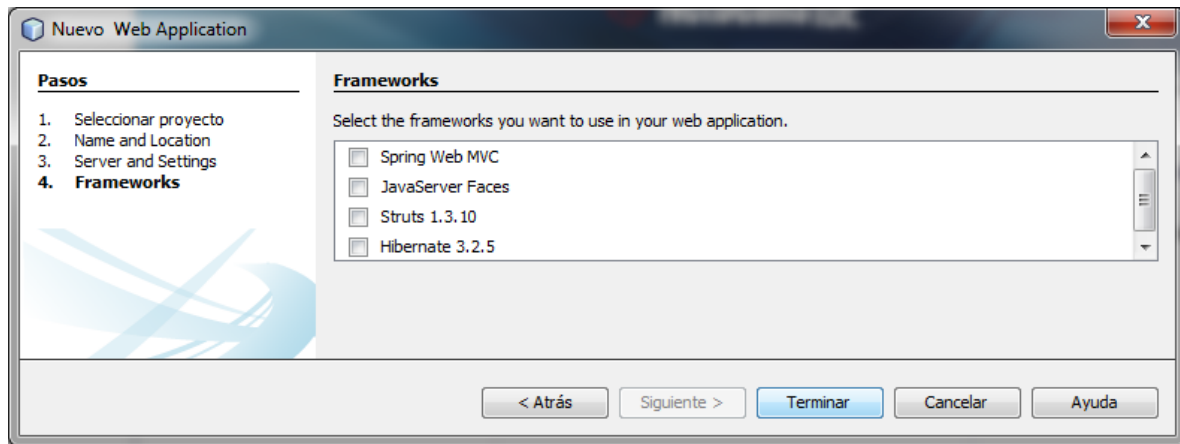


13). Crear un nuevo proyecto Java Web

Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)



Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)



Esta es la estructura del proyecto, la cual es generada automáticamente por el IDE Netbeans.

Web Pages: se colocaran los archivos HTML, JSP, JavaScript, CSS, Imágenes, iconos y Multimedia

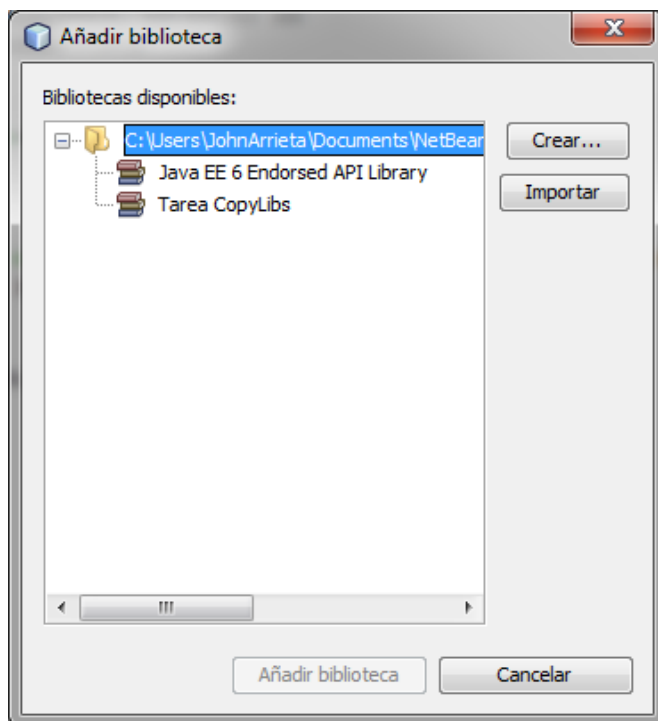
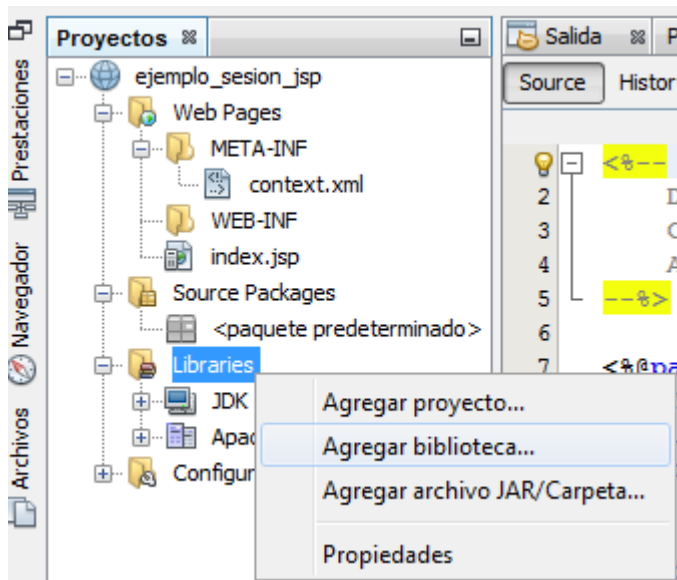
META-INF: contiene información XML sobre la estructura del proyecto (no debemos tocarla)

WEB-INF: contiene un archivo XML en el cual se registran y describen los Servlet y otros detalles importantes del sitio web, como es el caso de la página de inicio, o el HOME o Welcom

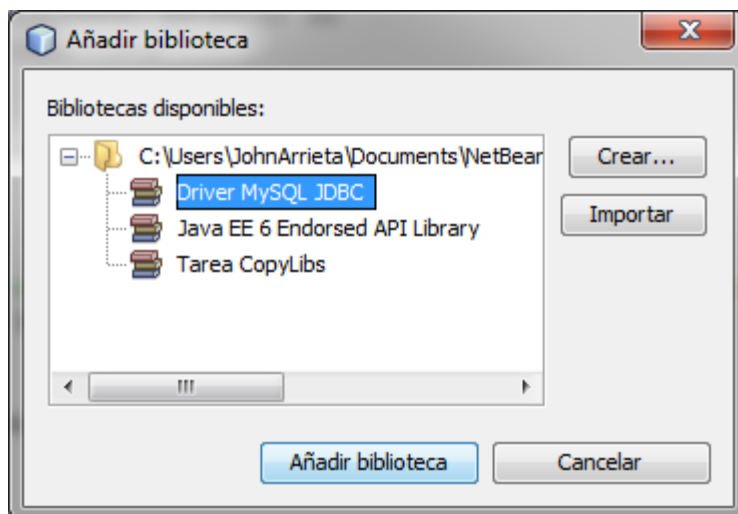
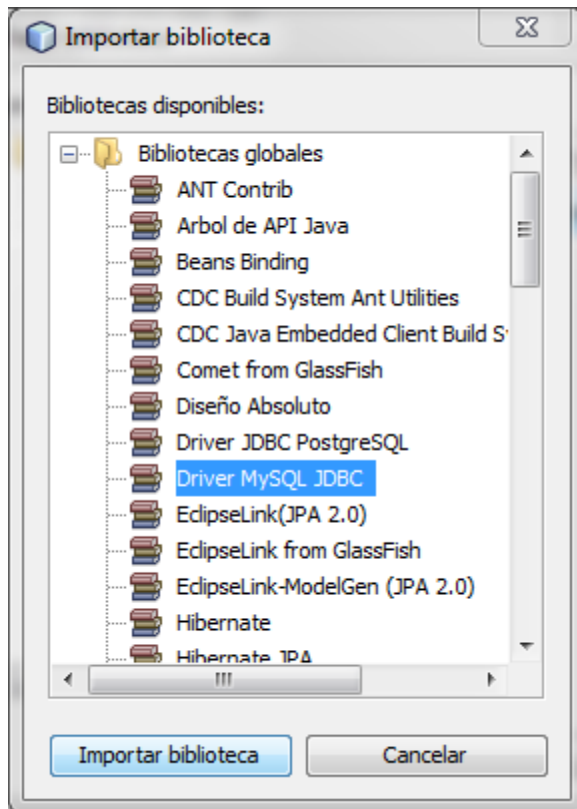
Source Package: Contendrá los archivos de código fuente .java, para el caso de los Servlet y otras clases, como por ejemplo los Java Bean, la clase conexión a bd y las clases utilitarias.

librerías: contiene las librerías o bibliotecas de clases.jar que necesite el proyecto para funcionar, como es el caso de la biblioteca de clases que actúan como Driver o Manejador de conexiones a MySQL.

14). Agregar la biblioteca o librería Diver- MySQL-JDBC.

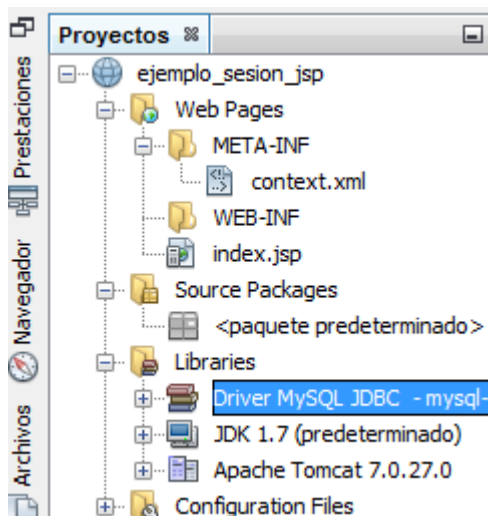


Dar clic en el botón Importar y seleccionar la librería Driver MySQL JDBC.

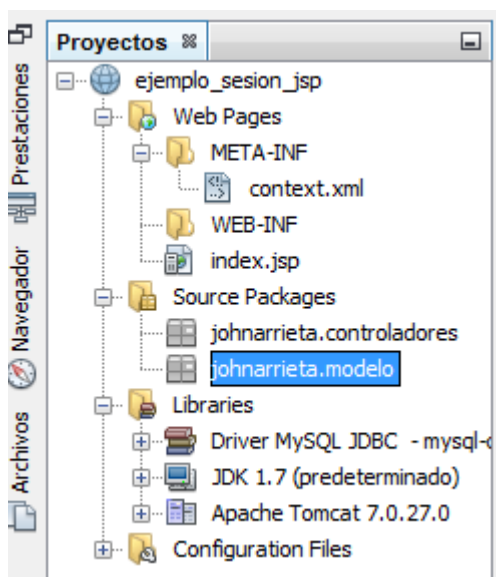
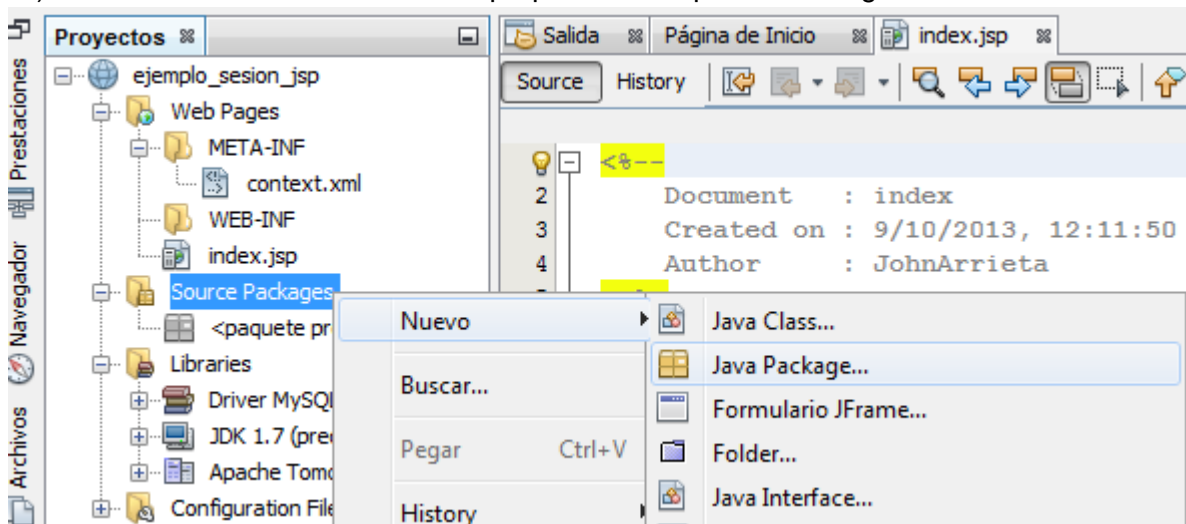


Observemos como fue agregada la librería al conjunto de librerías del proyecto

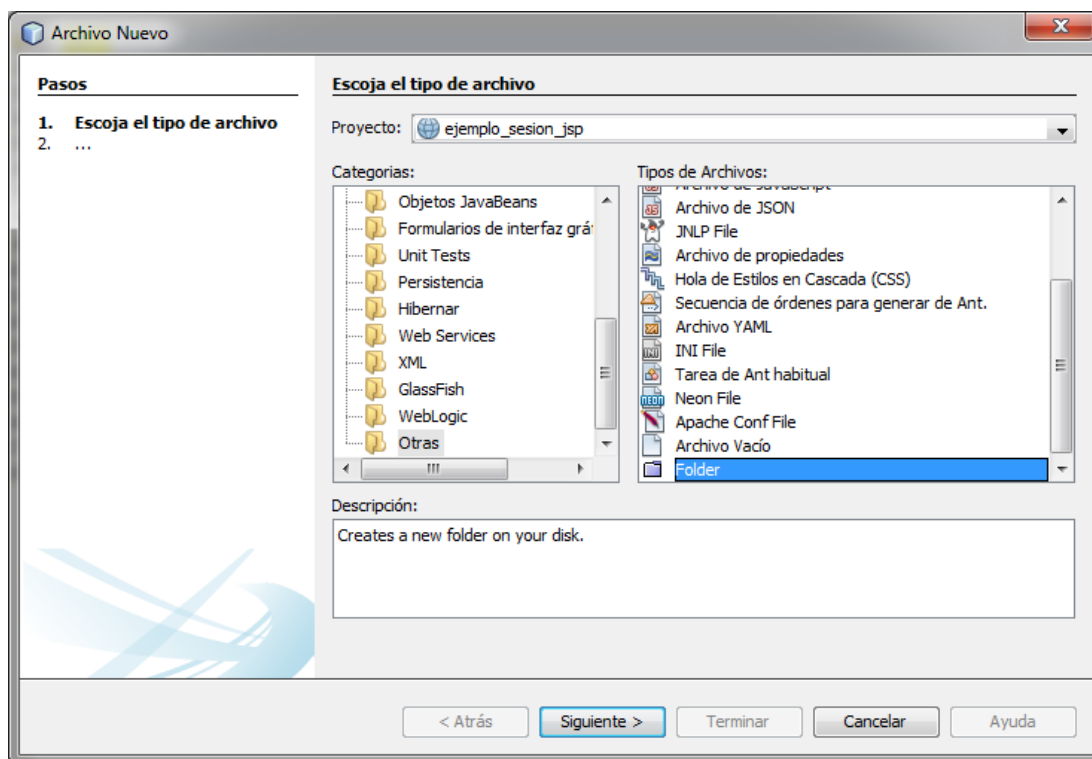
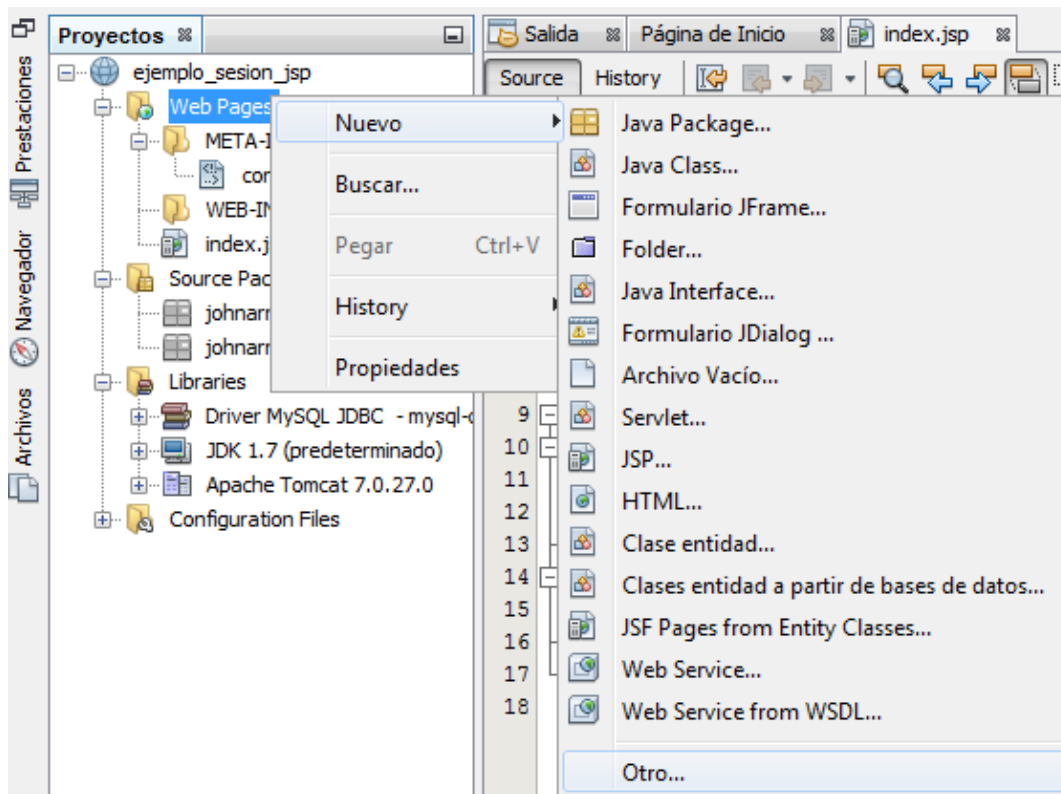
Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)



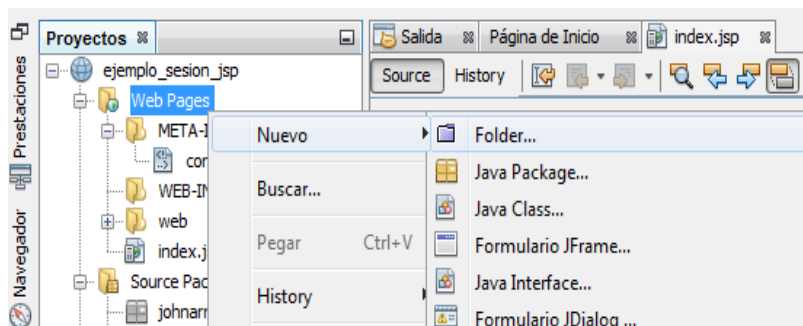
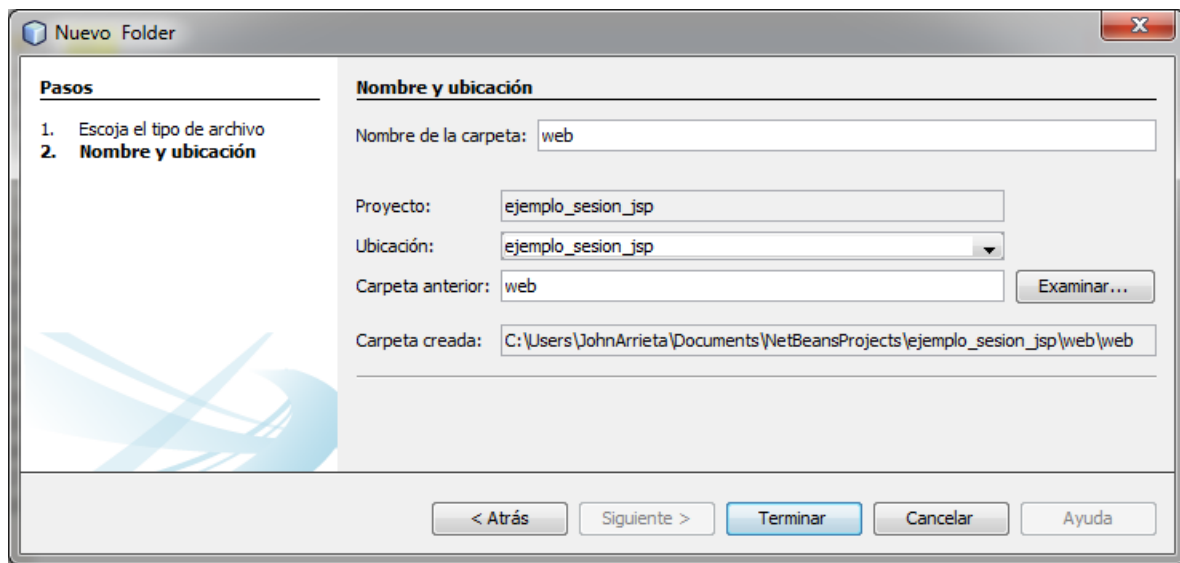
15). crear los paquetes para organizar las clases.



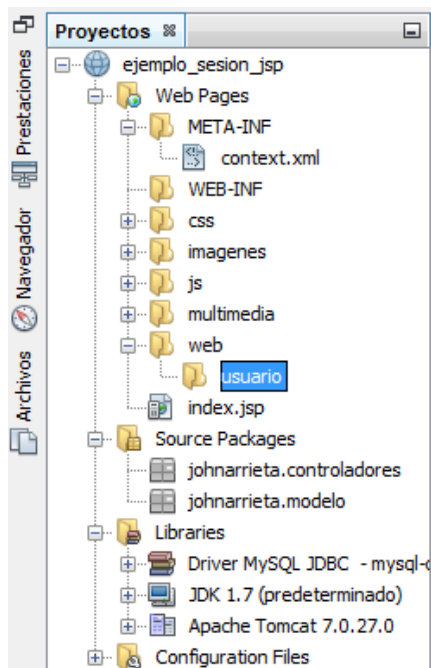
16). En la carpeta Web Pages creemos las carpeta para organizar los archivo html, jsp, css, js y demás,



Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)

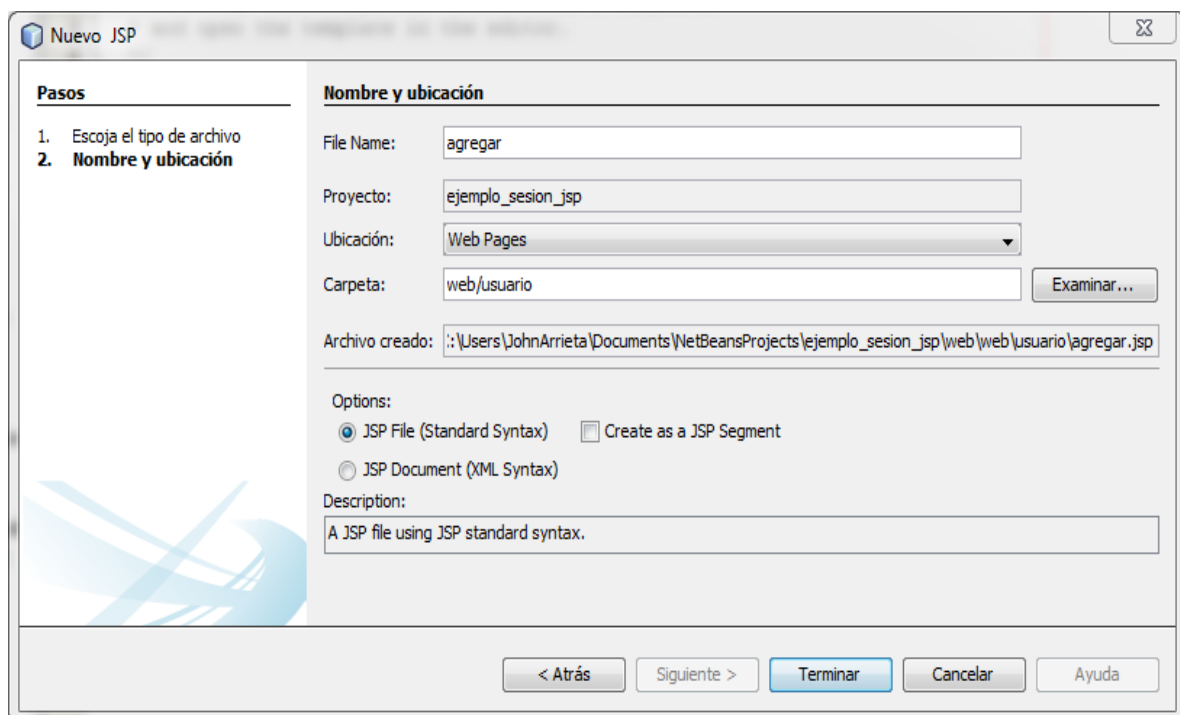
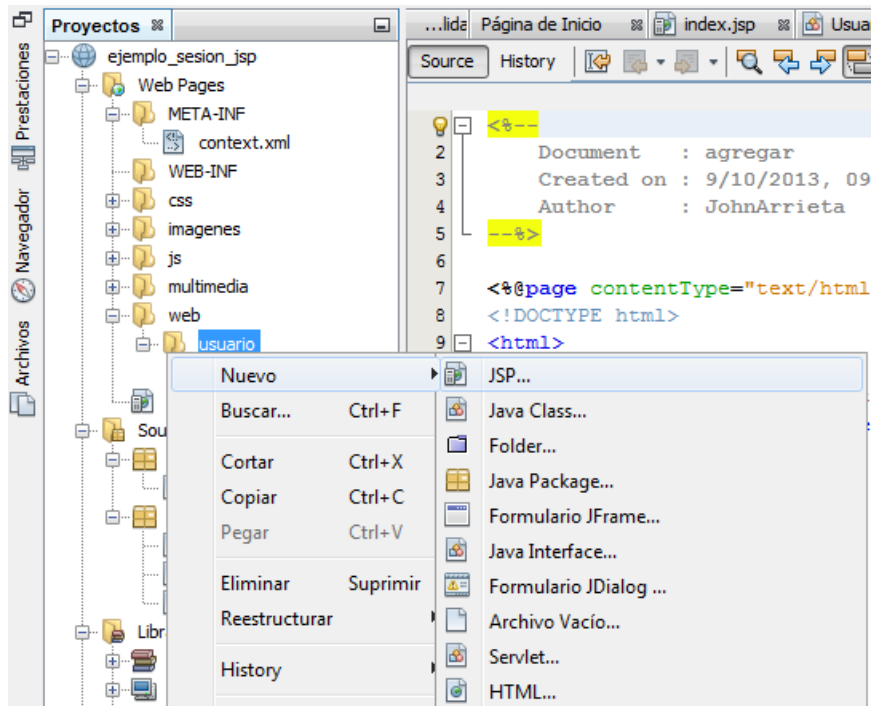


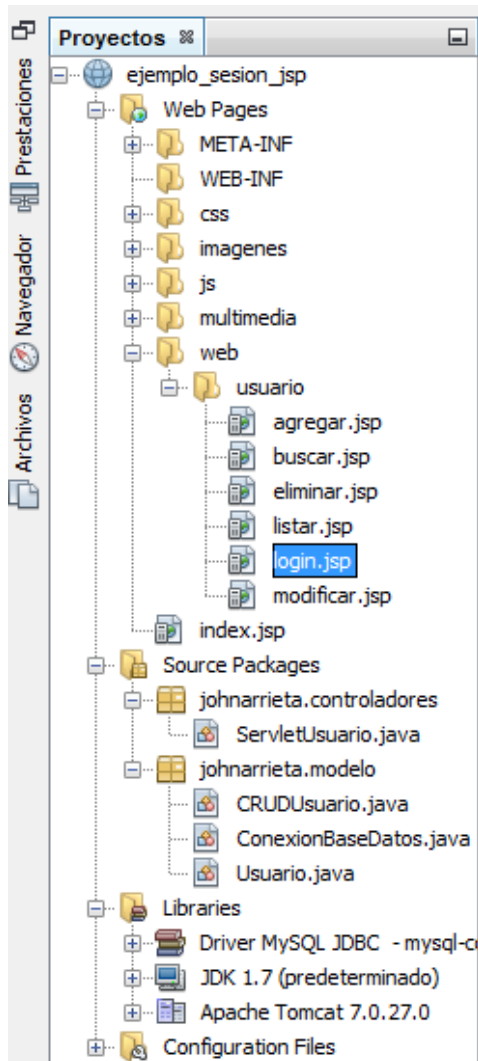
Se recomienda crear una carpeta con el mismo nombre de cada cada Clase de Negocio o Clase Entidad o Tabla de BD del sistema.



Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)

17). En cada carpeta con nombre igual a las tablas de la BD es recomendable crear al menos 5 archivos JSP para cada una de las acciones que comúnmente se realizan sobre dichas tablas, por ejemplo agregar.jsp, eliminar.jsp, modificar.jsp, buscar.jsp, listar.jsp en la carpeta web/usuario.





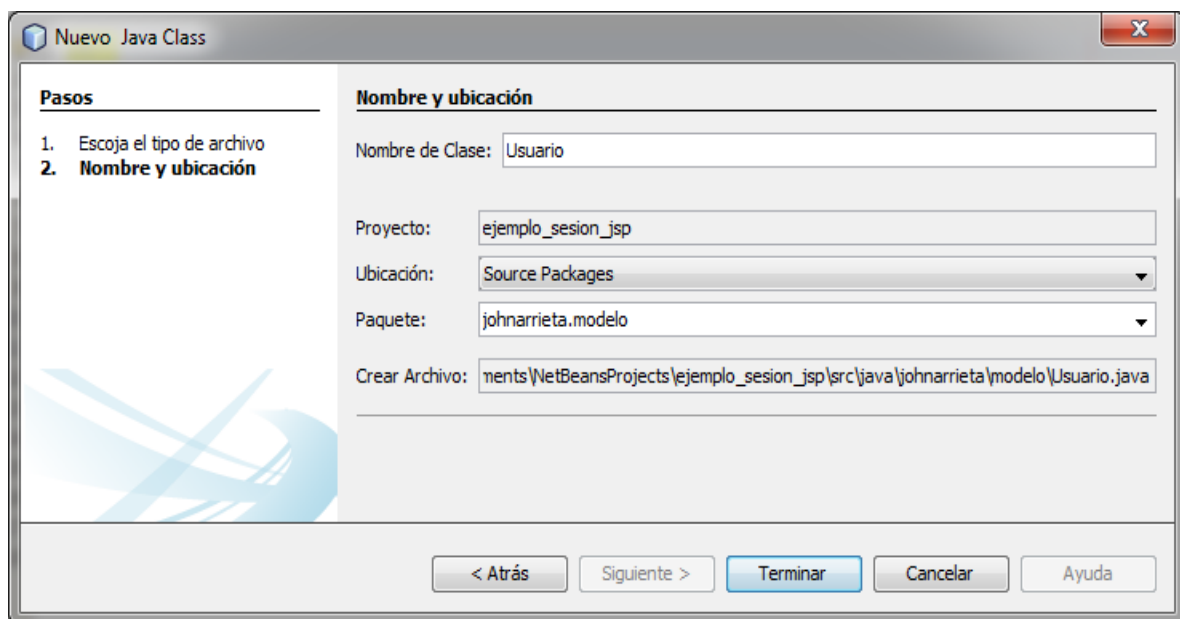
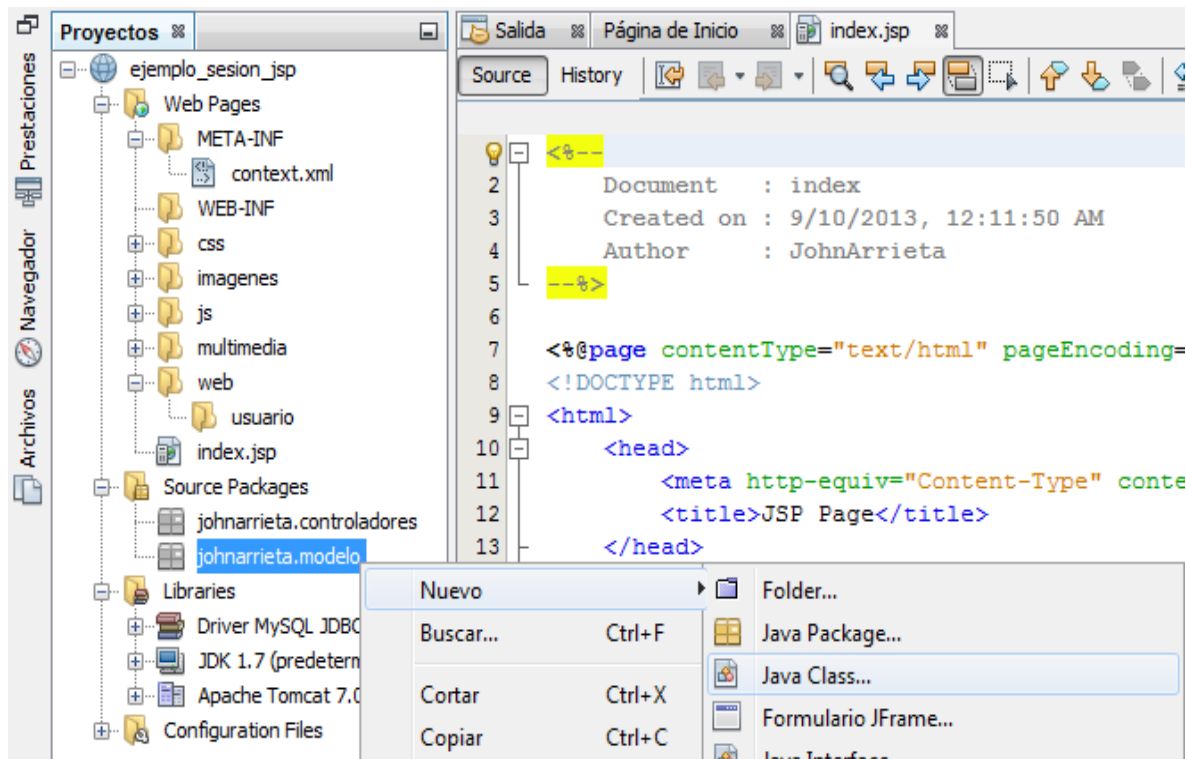
Además, como en este caso estamos tenemos una tabla llamada usuario, es normal que deseemos tener una pagina.jsp que nos permita iniciar sesión en el sistema y otra que nos muestre los mensajes de error, en caso tal de que ocurran, es por eso que también podemos agregar los archivos login.jsp y mensaje.jsp

18). Igual haremos con los demás archivos necesarios, entre ellos se encuentran:

Clases CRUDUsuario.java y ConexionBaseDatos.java en el paquete modelo, CRUD Es la sigla de **C**reate = Crear o Insertar, **R**ead = Leer o Consultar, **U**ppdate = **A**ctualizar o Modificar y **D**elete = Eliminar o Borrar.

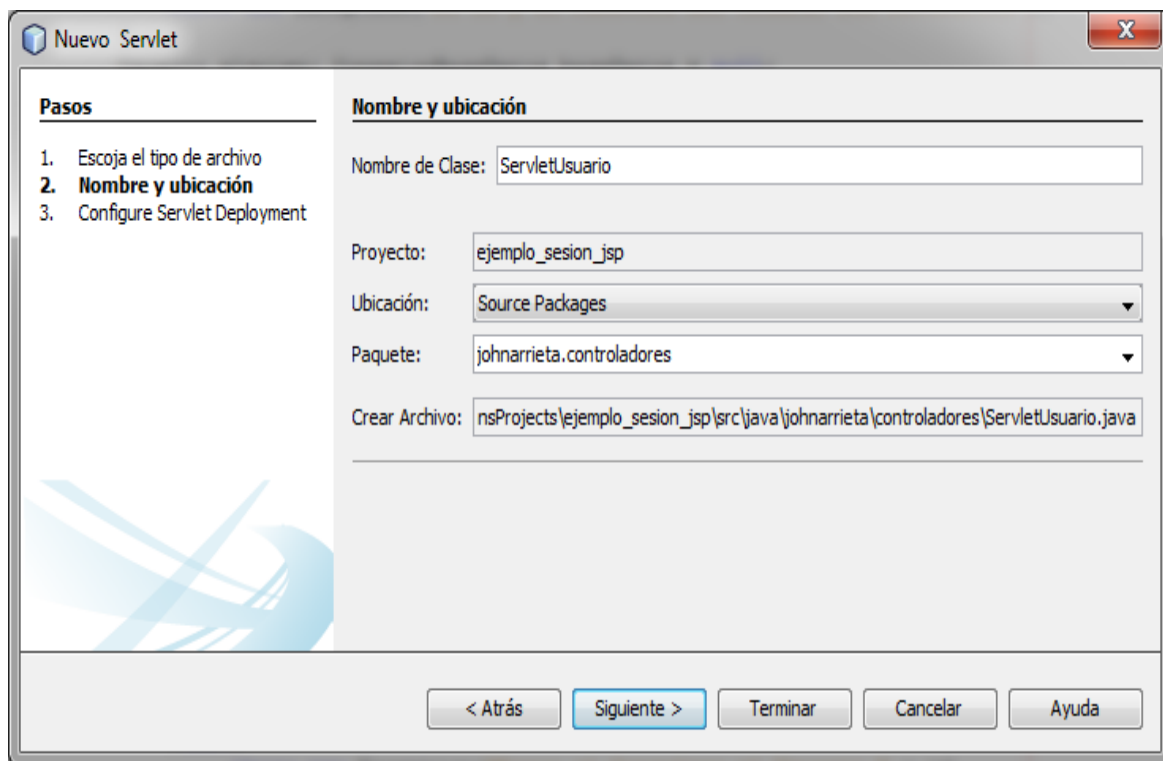
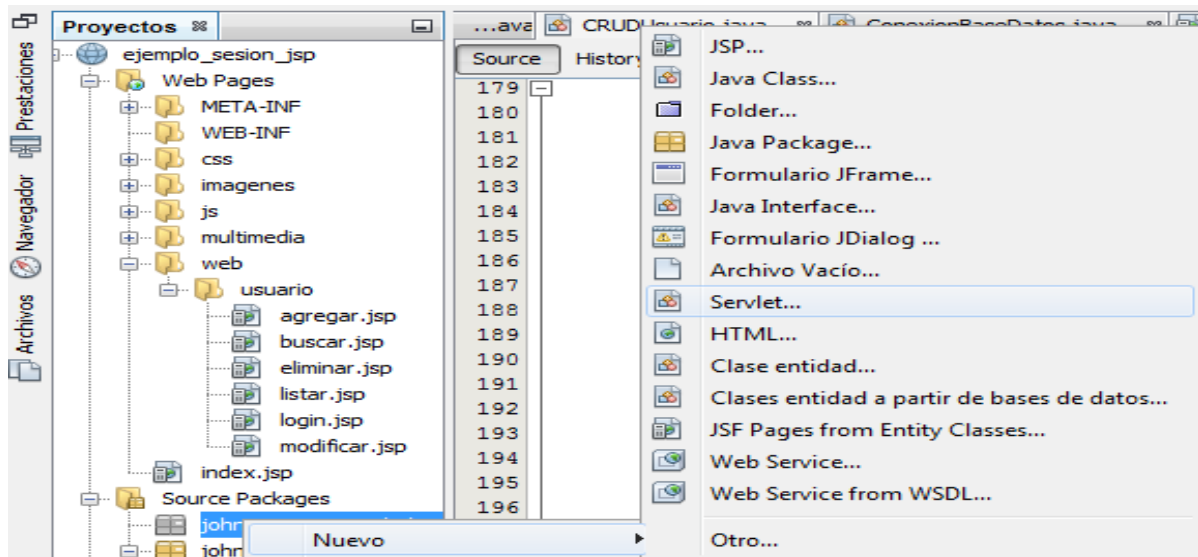
Crear la clase entidad para trabajar con información referente a usuario (generalmente las clases entidades, son una representación en lenguaje de programación a una tabla de la base de datos)

Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)

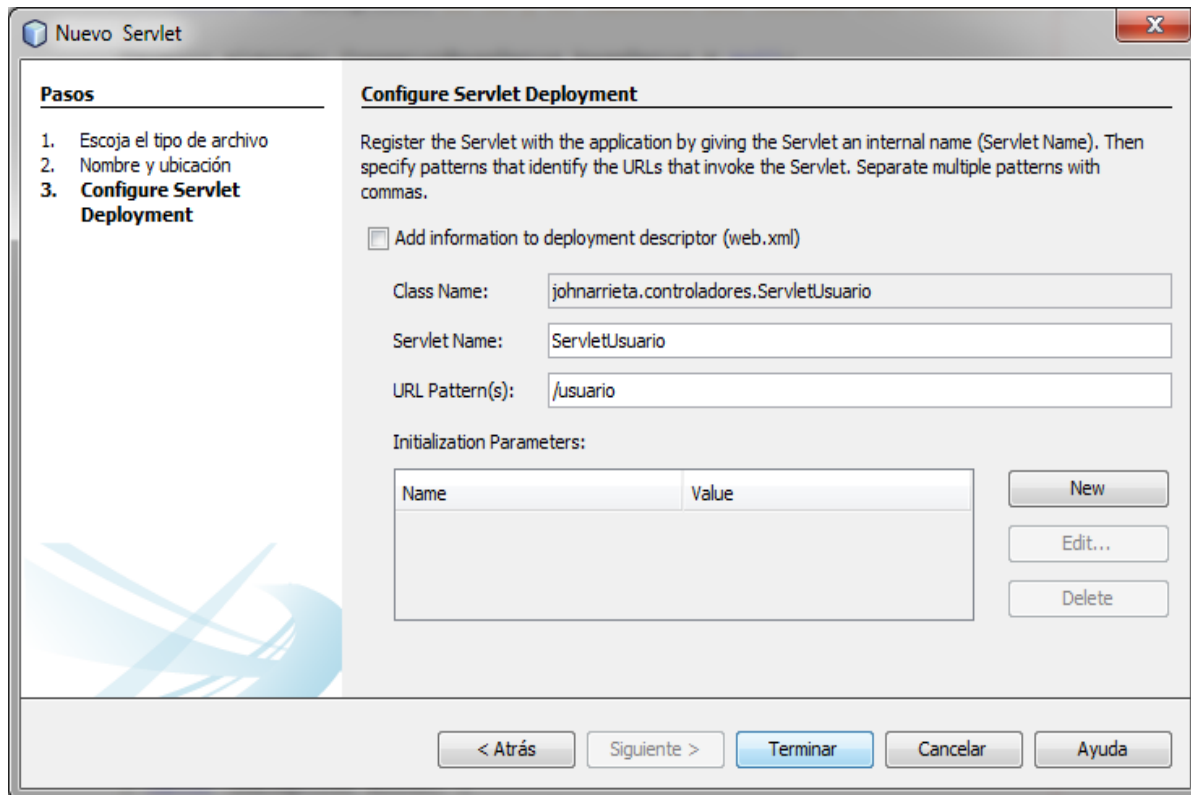


También debemos crear una clase muy importante de tipo Servlet, esta clase se puede llamar por ejemplo SetvletUsuario.java en el paquete **controlador**.

Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)



En la siguiente ventana se debe seleccionar o activar la **Add Information to Deployment Descriptor (web.xml)**, con el fin de generar automáticamente el archivo de despliegue de nuestra aplicación Web, en dicho archivo se agrega automáticamente el alias que deseamos dar al Servlet, en este caso será **/usuario**, también se incluyen datos como cual será la pagina inicial o la Index

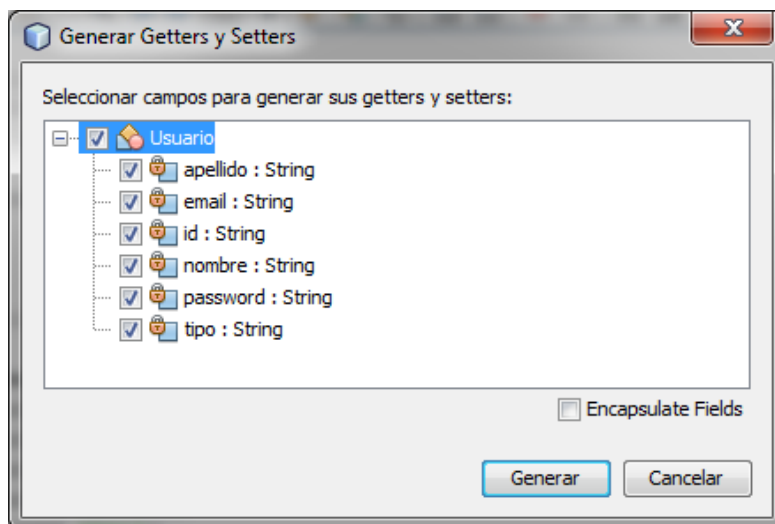
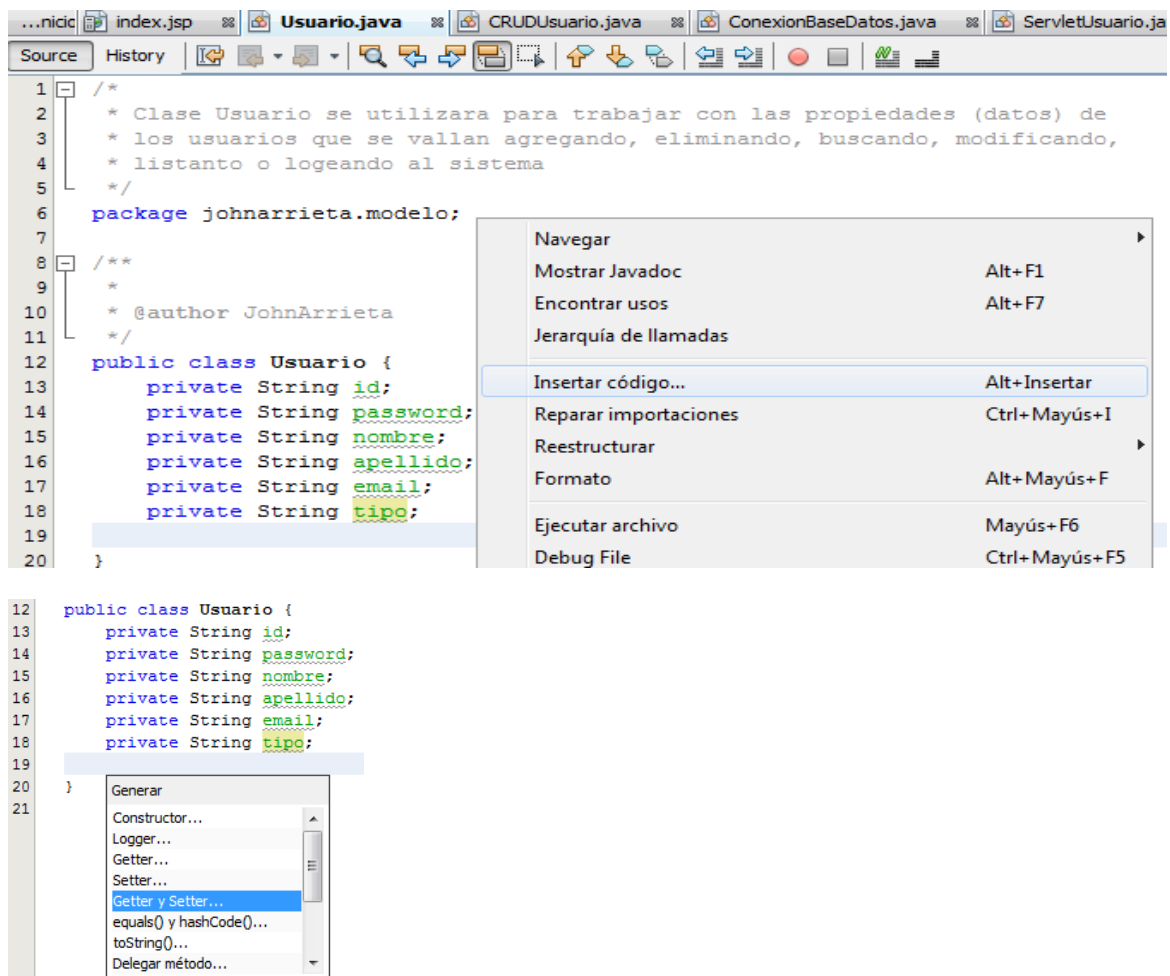


Observemos que se ha agregado nuevos archivo clase .java en los paquetes que hemos deseado.

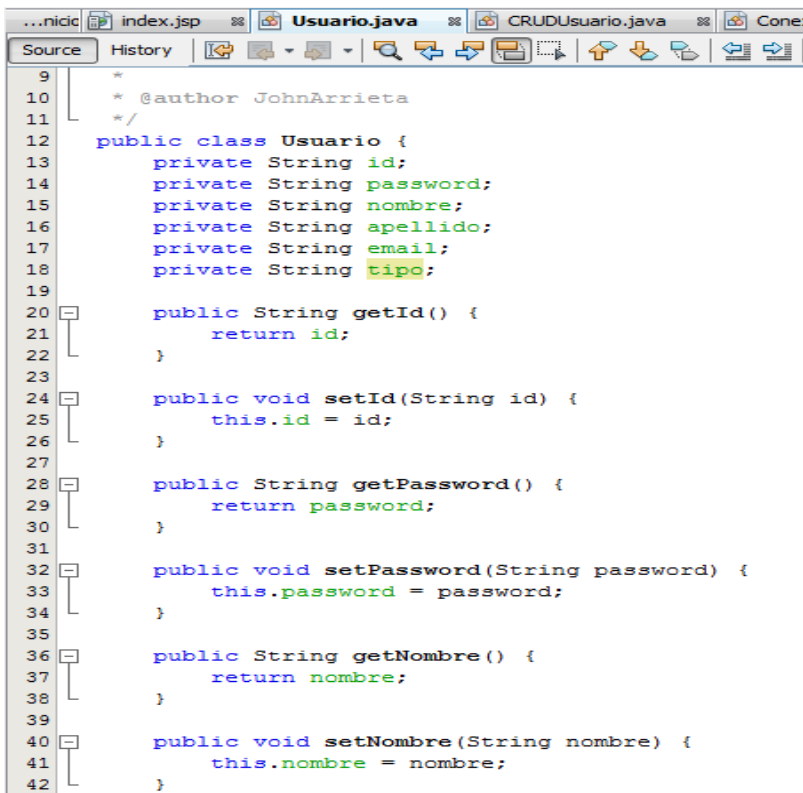
19). Construir los elementos (Propiedades o Atributos y Métodos u Operaciones) de cada clase java, en este caso veremos los elementos de la clase Usuario.java

```
...nicio index.jsp Usuario.java CRUDUsuario.java ConexionBaseDatos.java ServletUsuario.j
Source History
1  /*
2   * Clase Usuario se utilizara para trabajar con las propiedades (datos) de
3   * los usuarios que se vayan agregando, eliminando, buscando, modificando,
4   * listanto o logeando al sistema
5   */
6   package johnarrieta.modelo;
7
8   /**
9    *
10   * @author JohnArrieta
11   */
12   public class Usuario {
13       private String id;
14       private String password;
15       private String nombre;
16       private String apellido;
17       private String email;
18       private String tipo;
19
20   }
21
```


21). Métodos SET y GET para las propiedades de la clase Usuario

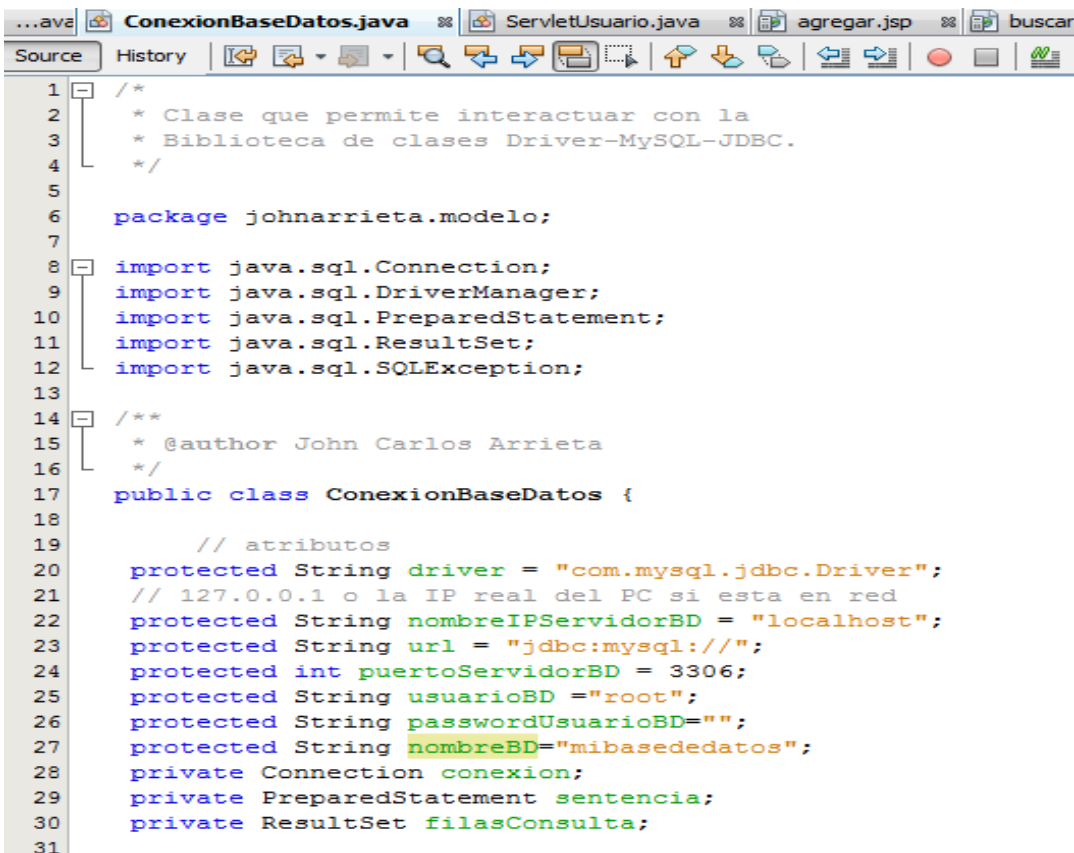


Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)



```
9      *
10     * @author JohnArrieta
11     */
12     public class Usuario {
13         private String id;
14         private String password;
15         private String nombre;
16         private String apellido;
17         private String email;
18         private String tipo;
19
20         public String getId() {
21             return id;
22         }
23
24         public void setId(String id) {
25             this.id = id;
26         }
27
28         public String getPassword() {
29             return password;
30         }
31
32         public void setPassword(String password) {
33             this.password = password;
34         }
35
36         public String getNombre() {
37             return nombre;
38         }
39
40         public void setNombre(String nombre) {
41             this.nombre = nombre;
42         }
43     }
```

22). Propiedades de la clase ConexionBaseDatos.java



```
1      /*
2      * Clase que permite interactuar con la
3      * Biblioteca de clases Driver-MySQL-JDBC.
4      */
5
6     package johnarrieta.modelo;
7
8     import java.sql.Connection;
9     import java.sql.DriverManager;
10    import java.sql.PreparedStatement;
11    import java.sql.ResultSet;
12    import java.sql.SQLException;
13
14    /**
15     * @author John Carlos Arrieta
16     */
17    public class ConexionBaseDatos {
18
19        // atributos
20        protected String driver = "com.mysql.jdbc.Driver";
21        // 127.0.0.1 o la IP real del PC si esta en red
22        protected String nombreIPServidorBD = "localhost";
23        protected String url = "jdbc:mysql://";
24        protected int puertoServidorBD = 3306;
25        protected String usuarioBD = "root";
26        protected String passwordUsuarioBD = "";
27        protected String nombreBD = "mibasededatos";
28        private Connection conexion;
29        private PreparedStatement sentencia;
30        private ResultSet filasConsulta;
31    }
```

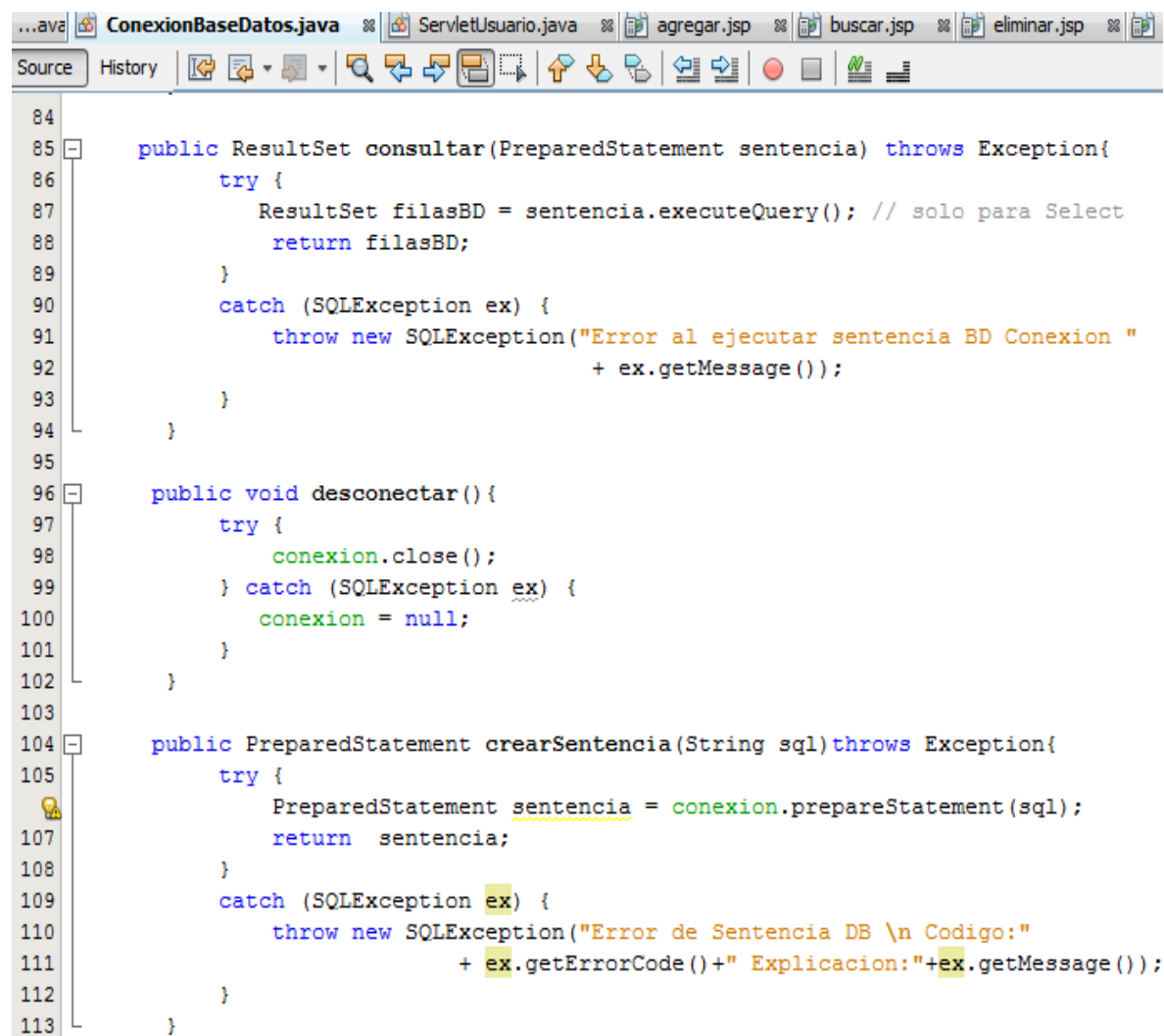
23). métodos de la clase ConexionBaseDatos

```
31
32 // constructores
33 public ConexionBaseDatos( ) throws Exception{
34     url = url+nombreIPServidorBD+": "+puertoServidorBD+"/"+nombreBD;
35     this.conectar();
36 }
37
38 public ConexionBaseDatos(String driver, String servidor, String url,
39     String usuarioBD, String passwordUsuarioBD, String nombreBD)
40     throws Exception
41 {
42     this.driver = driver;
43     this.nombreIPServidorBD = servidor;
44     this.url = url;
45     this.usuarioBD = usuarioBD;
46     this.passwordUsuarioBD = passwordUsuarioBD;
47     this.nombreBD = nombreBD;
48     this.conectar();
49 }

...ava ConexionBaseDatos.java ServletUsuario.java agregar.jsp buscar.jsp eliminar.jsp modificar.jsp
Source History
55 // operaciones sobre BD
56 public void conectar( ) throws Exception {
57     //... colcar aqui el codigo para conectar al SMBD deseado
58     try {
59         Class.forName(driver); // registro el driver de la SMBD
60     }
61     catch (ClassNotFoundException ex) {
62         throw new Exception("Error de Driver "+ex.getMessage());
63     }
64     try {
65         conexion = DriverManager.getConnection(url, usuarioBD, passwordUsuarioBD);
66     }
67     catch (SQLException ex) {
68         throw new Exception("Error de Conexion \nCodigo:"
69             + ex.getErrorCode()+" Explicacion:"+ex.getMessage());
70     }
71 }
72
73
74 public int actualizar(PreparedStatement sentencia) throws Exception {
75     try {
76         int res = sentencia.executeUpdate();
77         return res;
78     }
79     catch (SQLException ex) {
80         throw new SQLException("Error al ejecutar sentencia BD Conexion \nCodigo:"
81             + ex.getErrorCode()+" Explicacion:"+ex.getMessage());
82     }
83 }
```

El método **conectar** de la clase `ConexionBaseDatos`: la ser invocado sus líneas 58-63 contienen un try...catch en el cual se intenta registrar el driver de la base de datos, en este caso las clase `Driver` que se encuentra en el paquete `com.mysql.jdbc` incluidos en la biblioteca de clases `Driver-MySQL-JDBC` agregada al proyecto en el paso No. 14. Líneas 64-70 intentan establecer conexión al servidor MySQL usando la url `jdbc:mysql://localhost:3306/mibasededatos`, usuario y password de la BD.

Método **actualizar** de la clase `ConexionBaseDatos`: al ser invocado recibe como argumentos un objeto de la clase `PreparedStatement`, el cual es utilizado para invocar al método `executeUpdate` de dicha clase, este método envía al Servidor MySQL sentencias SQL de tipo INSERT, UPDATE o DELETE contenidas en el objeto `PreparedStatement`, el servidor ejecuta esta sentencia y devuelve el resultado, todo se hace dentro de un try... catch para poder manejar el error en caso de que este ocurra. Este error es relanzado fuera de este método

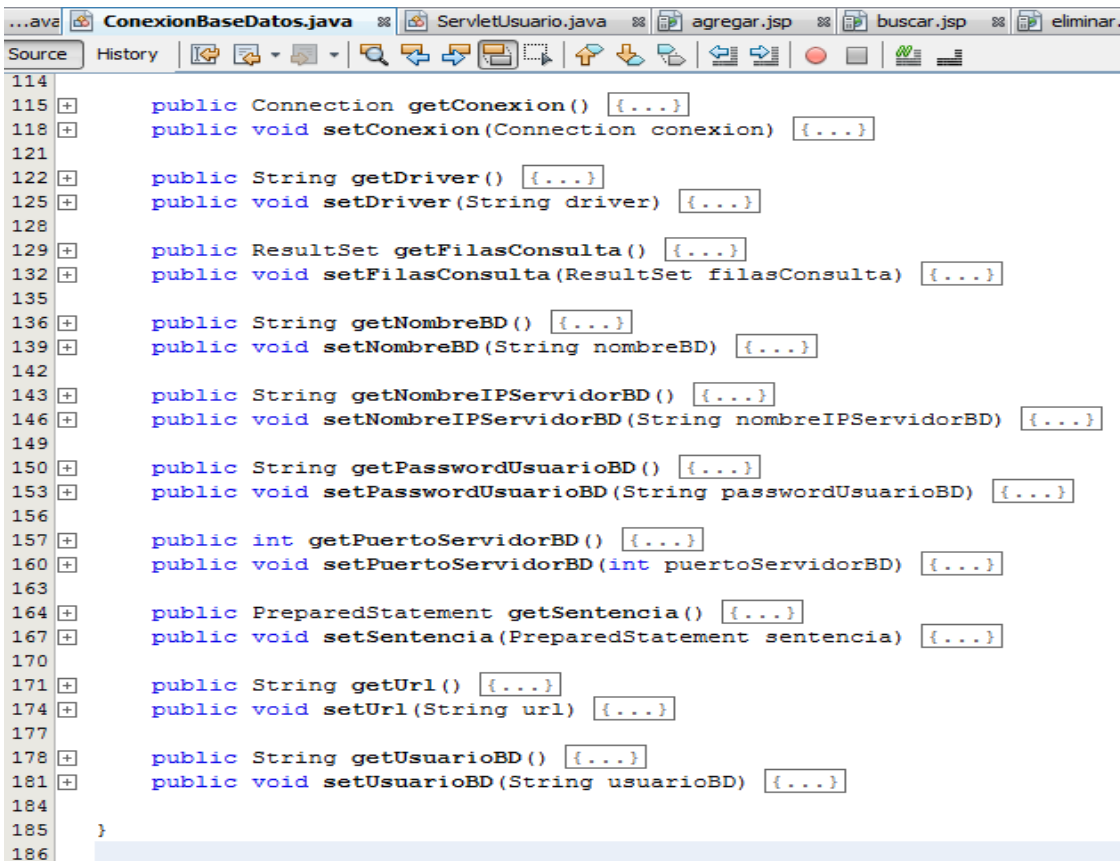


```
84
85 public ResultSet consultar(PreparedStatement sentencia) throws Exception{
86     try {
87         ResultSet filasBD = sentencia.executeQuery(); // solo para Select
88         return filasBD;
89     }
90     catch (SQLException ex) {
91         throw new SQLException("Error al ejecutar sentencia BD Conexion "
92             + ex.getMessage());
93     }
94 }
95
96 public void desconectar(){
97     try {
98         conexion.close();
99     } catch (SQLException ex) {
100         conexion = null;
101     }
102 }
103
104 public PreparedStatement crearSentencia(String sql) throws Exception{
105     try {
106         PreparedStatement sentencia = conexion.prepareStatement(sql);
107         return sentencia;
108     }
109     catch (SQLException ex) {
110         throw new SQLException("Error de Sentencia DB \nCodigo:"
111             + ex.getErrorCode()+" Explicacion:"+ex.getMessage());
112     }
113 }
```

Método **consultar** de la clase `ConexionBaseDatos`: Al ser invocado funciona igual al método `actualizar` pero la diferencia es que el objeto sentencia de tipo `PreparedStatement` contiene sentencias o comando SQL de tipo `SELECT`, los cuales se ejecutan sobre el servidor MySQL al ser invocado el método `executeQuery` de la clase `PreparedStatement`, devolviendo como resultado un objeto de la clase `ResultSet`, el cual contiene información sobre los registros o filas encontradas por el comando `SELECT` en la BD.

El método **desconectar** de la clase `ConexionBaseDatos`: al ser invocado línea 98 invoca al método `close` de la clase `Connection`, método que cierra toda posible conexión con el servidor de BD.

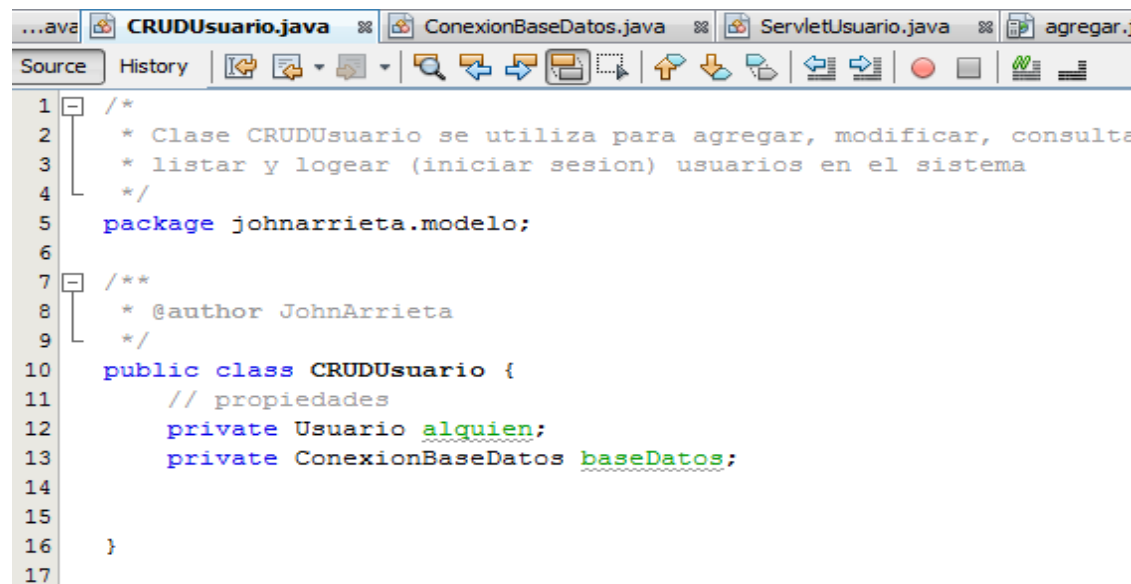
El método **crearSentencia** de la clase `ConexionBaseDatos`: al ser invocado recibe como argumento un objeto de la clase `String`, es decir una cadena de texto, este objeto referenciado usando la variable `sql` contiene la sentencia o comando SQL que será preparada para ser enviada y ejecutada sobre el servidor de BD. La sentencia `sql` se pasa como argumento a la llamada o invocación del método `prepareStatement` de la clase `Connection`, este método trata de preparar dicha sentencia `sql` para que pueda ser enviada sin errores al servidor de BD, si esto es posible, el método `prepareStatement` retorna un objeto de tipo `PreparedStatement`, el cual es utilizado para invocar los métodos `executeUpdate` y `executeQuery` de dicha clase



```
114
115 + public Connection getConexion() {...}
118 + public void setConexion(Connection conexion) {...}
121
122 + public String getDriver() {...}
125 + public void setDriver(String driver) {...}
128
129 + public ResultSet getFilasConsulta() {...}
132 + public void setFilasConsulta(ResultSet filasConsulta) {...}
135
136 + public String getNombreBD() {...}
139 + public void setNombreBD(String nombreBD) {...}
142
143 + public String getNombreIPServidorBD() {...}
146 + public void setNombreIPServidorBD(String nombreIPServidorBD) {...}
149
150 + public String getPasswordUsuarioBD() {...}
153 + public void setPasswordUsuarioBD(String passwordUsuarioBD) {...}
156
157 + public int getPuertoServidorBD() {...}
160 + public void setPuertoServidorBD(int puertoServidorBD) {...}
163
164 + public PreparedStatement getSentencia() {...}
167 + public void setSentencia(PreparedStatement sentencia) {...}
170
171 + public String getUrl() {...}
174 + public void setUrl(String url) {...}
177
178 + public String getUsuarioBD() {...}
181 + public void setUsuarioBD(String usuarioBD) {...}
184
185 }
186
```

Métodos **getXX** y **setXX** de la clase `ConexionBaseDatos`: al invocar los métodos `getXX` estos retornan o devuelven el valor de la propiedad `XX` que lleva su nombre, por ejemplo `getNombreBD()`; devolverá el valor de la propiedad `nombreBD`, mientras que al invocar los métodos `setXX` estos reciben como argumento una variable con el mismo tipo de la propiedad `XX` a la que hace referencia su nombre, con el fin de darle o asignarle un valor a la misma, por ejemplo `setNombreBD("mibasededatos");` asigna el valor "mibasededatos" a la propiedad `setNombreBD` de la clase `ConexionBaseDatos`

24). Propiedades de la clase `CRUDUsuario`



```
1  /*
2   * Clase CRUDUsuario se utiliza para agregar, modificar, consulta
3   * listar y logear (iniciar sesion) usuarios en el sistema
4   */
5   package johnarrieta.modelo;
6
7   /**
8    * @author JohnArrieta
9    */
10  public class CRUDUsuario {
11      // propiedades
12      private Usuario alguien;
13      private ConexionBaseDatos baseDatos;
14
15
16  }
17
```

La clase `CRUDUsuario` es

24). Métodos de la clase `CRUDUsuario` está diseñada con el fin de realizar las operaciones CRUD de usuarios (Create o Insertar, Read o Consultar, Update o Actualizar y Delete o Eliminar por sus siglas en ingles sobre) sobre las base de datos. Para utiliza un objeto de la `ConexionBaseDatos` y otro de la clase `Usuario` con el fin de invocar sus métodos explicados anteriormente.

Esta clase básicamente se encarga de proporcionar métodos para que el Servlet pueda Insertar, Modificar, Eliminar y Consultar datos de los usuarios a partir de la información enviada por formularios jsp.

Los métodos de esta clase mediante los cuales se puede realizar CRUD sobre la base de datos son:

`agregarUsuario`, `modificarUsuario`, `eliminarUsuario`, `consultarUsuario`, `listarTodosLosUsuarios`, e `iniciarSesion`.


```
24 public void agregarUsuario() throws Exception {
25     if (alguien.getId() == null || alguien.getId().isEmpty()) {
26         throw new Exception("El ID del Usuario es Necesario");
27     }
28     // armar el SQL INSERT de forma dinamica
29     String sqlInsert = "INSERT INTO Usuarios "
30         + "(id, password, nombre, apellido,email, tipo) "
31         + "VALUES (?, ?, ?, ?, ?, ?)";
32     try {
33         // Crear una sentencia JDBC mediante la sentencia SQL anterior
34         PreparedStatement sentenciaSQL = baseDatos.crearSentencia(sqlInsert);
35         // Pasarle los datos del usuario a la sentencia SQL
36         sentenciaSQL.setString(1, alguien.getId());
37         sentenciaSQL.setString(2, alguien.getPassword());
38         sentenciaSQL.setString(3, alguien.getNombre());
39         sentenciaSQL.setString(4, alguien.getApellido());
40         sentenciaSQL.setString(5, alguien.getEmail());
41         sentenciaSQL.setString(6, alguien.getTipo());
42         // actualizar la BD usando la sentenciaSQL con los datos del usuario
43
44         baseDatos.actualizar(sentenciaSQL);
45     } catch (Exception error) {
46         throw new Exception("Error al Agregar el Usuario " + alguien.getId()
47             + "<br/>Explicacion: " + error.getMessage());
48     } finally {
49         baseDatos.desconectar();
50     }
51 }
```

EL método **agregarUsuario** de la clase **CRUDUsuario** al ser invocado lanza una excepción si no puede agregar los datos de un usuario en la BD, en las líneas 25 a la 27 se verifica si lanza una excepción si el id del usuario que se desea agregar en la BD este vacío.

Líneas 29 a 31: se crea la sentencia SQL de tipo **INSERT** para insertar los datos del usuario en la tabla usuarios de la BD.

Línea 34: Se invoca al método **crearSentencia** y se le pasa como argumento la sentencia **INSERT**, el objeto **PreparedStatement** devuelto es asignado a una variable referencia del mismo tipo.

Líneas 35 a 41: se utiliza el objeto de tipo **PreparedStatement** para invocar el método **setString(int, String)**, el cual permite reemplazar los símbolos **?** de la sentencia SQL por los valores que serán insertados en la tabla de la base de datos. El primero argumento de tipo **int** del método **setString** es la posición del **?** en la sentencia, mientras que el segundo argumento de tipo **String** corresponde al valor por el cual será reemplazado el **?** en la BD. En este caso se reemplazan el primer **?** por el Id del usuario a agregar en la BD, luego se reemplaza el segundo **?** por el **Password**, el tercer **?** por el **Nombre**, el cuarto **?** por el **Apellido**, el quinto **?** por el **Email** y el sexto **?** por el **Tipo**.

Línea 44: se utiliza la variable `baseDatos` como referencia a un objeto de la clase `ConexionBaseDatos` para poder invocar a su método `actualizar`, el cual recibe como argumento la variable `sentenciaSQL` referencia a un objeto de tipo `PreparedStatement`, este método se ejecuta insertando los datos del usuario como un nuevo registro o fila de la tabla `usuarios` en la BD. Si por algún motivo no se puede agregar el nuevo usuario en la BD, entonces se lanzara una excepción indicando el mensaje de error ocurrido, esta excepción es detectada por el bloque `try` y capturada por el bloque `catch`, el cual relanza nuevamente otra excepción explicando que el error ocurrió al tratar de `agregarUsuario`. Por último, se agregue o no el nuevo usuario a la bd, se ejecuta el bloque `finally` cerrando la conexión a la bd.

```
53 public void modificarUsuario() throws Exception {
54     if (alguien.getId() == null || alguien.getId().isEmpty()) {
55         throw new Exception("El ID del Usuario es Necesario");
56     }
57     // armar el SQL UPDATE de forma dinamica
58     String sqlUpdate = "UPDATE Usuarios "
59         + "SET password=?, nombre=?, apellido=?,email=?, tipo=? "
60         + "WHERE id =?";
61     try {
62         // Crear una sentencia JDBC mediante la sentencia SQL anterior
63         PreparedStatement sentenciaSQL = baseDatos.crearSentencia(sqlUpdate);
64         // Pasarle los datos del usuario a la sentencia SQL
65         //sentenciaSQL.setString(1, alguien.getId());
66         sentenciaSQL.setString(1, alguien.getPassword());
67         sentenciaSQL.setString(2, alguien.getNombre());
68         sentenciaSQL.setString(3, alguien.getApellido());
69         sentenciaSQL.setString(4, alguien.getEmail());
70         sentenciaSQL.setString(5, alguien.getTipo());
71         sentenciaSQL.setString(6, alguien.getId());
72         // actualizar la BD usando la sentenciaSQL con los datos del usuario
73         baseDatos.actualizar(sentenciaSQL);
74     } catch (Exception error) {
75         throw new Exception("Error al Actualizar el Usuario " + alguien.getId()
76             + "<br/>Explicacion: " + error.getMessage());
77     } finally {
78         baseDatos.desconectar();
79     }
80 }
```

El método `modificarUsuario` al ser invocado realizar una operación similar a la realizada por el método `agregarUsuario`, pero la diferencia es que este método envía una sentencia SQL de tipo `UPDATE` en vez de una de tipo `INSERT`.

En la línea 58 se genera la sentencia SQL UPDATE, en la línea 63 se obtiene el objeto para trabajar con la sentencia SQL, mientras que en las líneas 66 a 71 se indican los datos a modificar, y en la línea 73 se realiza la actualización de la BD con dichos datos.

```
82 public void eliminarUsuario() throws Exception {
83     if (alguien.getId() == null || alguien.getId().isEmpty()) {
84         throw new Exception("El ID del Usuario es Necesario");
85     }
86     // armar el SQL DELETE de forma dinamica
87     String sqlDelete = "DELETE FROM Usuarios "
88         + "WHERE id =?";
89     try {
90         // Crear una sentencia JDBC mediante la sentencia SQL anterior
91         PreparedStatement sentenciaSQL = baseDatos.crearSentencia(sqlDelete);
92         // Pasarle los datos del usuario a la sentencia SQL
93         sentenciaSQL.setString(1, alguien.getId());
94         // actualizar la BD usando la sentenciaSQL con los datos del usuario
95         baseDatos.actualizar(sentenciaSQL);
96     } catch (Exception error) {
97         throw new Exception("Error al Eliminar el Usuario " + alguien.getId()
98             + "<br/>Explicacion: " + error.getMessage());
99     } finally {
100         baseDatos.desconectar();
101     }
102 }
```

El método `eliminarUsuario` funciona similar a los métodos `agregarUsuario` y `modificarUsuario`, la diferencia es que este envía a la BD una sentencia SQL DELETE en vez de INSERT y UPDATE.

En la línea 87 se genera la sentencia SQL DELETE, en la línea 91 se obtiene el objeto para trabajar con la sentencia SQL, mientras que en las líneas 93 reemplaza el ? por el Id del usuario a eliminar, y en la línea 95 se realiza la actualización de la BD con dichos datos.

El método `iniciarSesion` de la clase `CRUDUsuario` al ser invocado recibe como argumento el id y el password del usuario que desea iniciar una sesión en el sistema y retorna como resultado un objeto de tipo `Usuario`, el cual contiene los datos del usuario que se ha logeado, en caso de que no pueda loguearse, por ejemplo porque los datos suministrados (id y password) son errados, el método lanza una excepción informado sobre el error ocurrido.

Línea 185: se diseña la sentencia o comando SQL de tipo SELECT, el cual permite consultar información en la base de datos y retorna como resultado un conjunto de filas o registros que coinciden con los criterios de búsqueda especificados en dicha sentencia.

Línea 189: se obtiene el objeto `PreparedStatement` a partir de la sentencia SQL SELECT creada anteriormente. Línea 191: Se reemplaza el primer ? y segundo? por el Id y

el password del usuario a consultar respectivamente. Línea 193: Se invoca el método consultar de la clase ConexionBaseDatos pasándole sentenciaSQL preparada con el fin de obtener un objeto de tipo ResultSet, el cual contiene el resultado de la consulta.

```
179 public static Usuario iniciarSesion( String id, String password) throws Exception {
180     if (id == null || id.isEmpty() || password == null || password.isEmpty()) {
181         throw new Exception("El ID y el Password del Usuario son Necesarios");
182     }
183     Usuario alguien; ConexionBaseDatos baseDatos = null;
184     // armar el SQL SELECT de forma dinamica
185     String sqlSelect = "SELECT * FROM Usuarios WHERE id=? and password=?";
186     try {
187         // Crear una sentencia JDBC mediante la sentencia SQL anterior
188         baseDatos = new ConexionBaseDatos();
189         PreparedStatement sentenciaSQL = baseDatos.crearSentencia(sqlSelect);
190         // Pasarle los datos del usuario a la sentencia SQL
191         sentenciaSQL.setString(1, id); sentenciaSQL.setString(2, password);
192         // Verificar el Resultado de la consulta
193         ResultSet resultado = baseDatos.consultar(sentenciaSQL);
194         if (resultado.next() == true) {
195             alguien = new Usuario();
196             alguien.setId(resultado.getString("id"));
197             alguien.setPassword(resultado.getString("password"));
198             alguien.setNombre(resultado.getString("nombre"));
199             alguien.setApellido(resultado.getString("apellido"));
200             alguien.setEmail(resultado.getString("email"));
201             alguien.setTipo(resultado.getString("tipo"));
202             return alguien;
203         }else{
204             throw new Exception("Error al Consultar el Usuario " + id+"<br/>Explicacion: ");
205         }
206     } catch (Exception error) {
207         throw new Exception(error.getMessage()+"Error en el ID o el Password estan Errados");
208     } finally {
209         if (baseDatos != null) {
210             baseDatos.desconectar();
211         }
212     }
213 }
```

Línea 194 a 205: Se verifica si el resultado tiene disponible un registro como respuesta a la consulta, si esto es verdadero, entonces se crea un nuevo objeto de tipo Usuario y se asigna a una variable llamada alguien, luego se le asignan valores a sus propiedades invocando sus métodos setXX, los cuales reciben como argumento el valor devuelto al invocar el método getString de la clase ResultSet, el cual recibe como argumento el nombre de la columna correspondiente a la tabla de la bd indicada en la sentencia SELECT. Cuando ya se han

cargado todos los datos del usuario logueado, entonces este es retornado por el método culminando así su ejecución, si por el contrario, el resultado no contiene ninguna registro disponible, entonces se lanza una excepción notificando datos sobre el error.

```
104 public static Usuario consultarUsuario( String id) throws Exception {
105     if (id == null || id.isEmpty()) {
106         throw new Exception("El ID del Usuario es Necesario");
107     }
108     Usuario alguien; ConexionBaseDatos baseDatos = null;
109     // armar el SQL SELECT de forma dinamica
110     String sqlSelect = "SELECT * FROM Usuarios WHERE id=?";
111     try {
112         // Crear una sentencia JDBC mediante la sentencia SQL anterior
113         baseDatos = new ConexionBaseDatos();
114         PreparedStatement sentenciaSQL = baseDatos.crearSentencia(sqlSelect);
115         // Pasarle los datos del usuario a la sentencia SQL
116         sentenciaSQL.setString(1, id);
117         // Verificar el Resultado de la consulta
118         ResultSet resultado = baseDatos.consultar(sentenciaSQL);
119         if (resultado.next() == true) {
120             alguien = new Usuario();
121             alguien.setId(resultado.getString("id"));
122             alguien.setPassword(resultado.getString("password"));
123             alguien.setNombre(resultado.getString("nombre"));
124             alguien.setApellido(resultado.getString("apellido"));
125             alguien.setEmail(resultado.getString("email"));
126             alguien.setTipo(resultado.getString("tipo"));
127             return alguien;
128         }else{
129             throw new Exception("Error al Consultar el Usuario " + id+"<br/>Explicacion: ");
130         }
131     } catch (Exception error) {
132         throw new Exception(error.getMessage()+"El usuario No existe en la BD");
133     } finally {
134         if (baseDatos != null) {
135             baseDatos.desconectar();
136         }
137     }
138 }
```

El método `consultarUsuario` de la clase `CRUDUsuario` al ser invocado recibe como argumento el `id` del usuario que se desea consultar en la BD, y devuelve un objeto de tipo `Usuario` con los datos del usuario cuyo `id` en la BD es igual al `id` pasado como argumento. El método retorna una excepción en caso hipotético que no pueda retornar un resultado positivo, es decir que no exista en la BD un usuario con `id` igual al `id` que se indicó como argumento al llamar este método.

Su funcionamiento es básicamente similar al del método `iniciarSesion`, ambos métodos están marcados como `static` lo que significa que no es necesario crear un objeto o instancia de la clase `CRUDUsuario` para poder invocar estos métodos.

```
140 public static Usuario[] listarTodosLosUsuarios() throws Exception {
141     Usuario alguien; ConexionBaseDatos baseDatos = null;
142     // armar el SQL SELECT de forma dinamica
143     String sqlSelect = "SELECT * FROM Usuarios";
144     try {
145         // Crear una sentencia JDBC mediante la sentencia SQL anterior
146         baseDatos = new ConexionBaseDatos();
147         PreparedStatement sentenciaSQL = baseDatos.crearSentencia(sqlSelect);
148         // Verificar el Resultado de la consulta
149         ResultSet resultado = baseDatos.consultar(sentenciaSQL);
150         resultado.last(); // colocarnos en el ultimo registro del resultado
151         Usuario[] listado = new Usuario[resultado.getRow()] ; // la posicion del ultimo
152         resultado.beforeFirst(); // nos colocamos antes del primer registro
153         while (resultado.next() == true) { // nos colocamos en el proximo registro
154             alguien = new Usuario();
155             alguien.setId(resultado.getString("id"));
156             alguien.setPassword(resultado.getString("password"));
157             alguien.setNombre(resultado.getString("nombre"));
158             alguien.setApellido(resultado.getString("apellido"));
159             alguien.setEmail(resultado.getString("email"));
160             alguien.setTipo(resultado.getString("tipo"));
161             listado[resultado.getRow()] = alguien;
162         } if(listado.length <= 0){
163             throw new Exception("Error al Listar los Usuarios "
164                 + "<br/>Explicacion: ");
165         }
166         return listado;
167     } catch (Exception error) {
168         throw new Exception(error.getMessage()+"La BD esta vacia");
169     } finally {
170         if (baseDatos != null) {
171             baseDatos.desconectar();
172         }
173     }
174 }
```

El método `listarTodosLosUsuarios` de la clase `CRUDUsuario` al ser invocado no recibe argumentos y retorna un arreglo de `Usuarios`, este arreglo contiene una lista de todos los objetos de tipo `Usuario` que con información sobre los usuarios encontrados en la BD. El método puede o no retornar una excepción en caso de que no pueda realizar la consulta o la base no contenga ni un solo usuario registrado.

Línea 143: se diseña la sentencia o comando SQL SELECT que será enviado al servidor de base de datos, esta sentencia es una consulta sin criterios o condiciones de búsqueda, es decir

hace un barrido por toda la tabla `Usuarios` de la BD y retorna todos los registros o filas contenidas en ella. Las líneas 146, 147 y 149 ya han sido explicadas anteriormete.

Línea 150: se invoca el método `last` de la clase `ResultSet` con el fin de ubicarnos en la última fila contenida en el resultado devuelto al invocar el método `consultar` de la clase `ConexionBaseDatos`, esto con el fin de obtener el número de la última fila de resultado y hace poder crear un arreglo con tamaño igual al tamaño del resultado, operación que se realiza en la línea 151, al invocar el método `getRow` de la clase `ResultSet` el cual retorna el número de la fila actual (recordemos que en la línea anterior nos ubicamos en la última fila del resultado)

Línea 152: nos volvemos a ubicar en la primera antes de la primera fila del resultado ósea en la fila número 0.

Líneas 153 a 162: realizamos un ciclo de tipo MIENTRAS QUE con el fin de iterar o recorrer el conjunto de filas contenidas en el objeto `resultado` de tipo `ResultSet`. Cada vez que se invoque el mentodo `next` de la clase `ResultSet` y este devuelva verdadero, iniciara una iteración nueva, en la cual se crea un nuevo objeto `Usuario` y se asigna a una variable llamada `alguien`, luego le damos valores a sus propiedades invocando sus métodos `setXX` y pasándole como argumento el valor devuelto al invocar el método `getString` de la clase `ResultSet`, quien recibe como argumento el nombre de la columna de la tabla en la bd especificada en la sentencia `SELECT`

Línea 161: se invoca nuevamente al método `getRow` de la clase `ResultSet` con el fin de obtener el numero o posición de la fila en la que nos encontramos actualmente al recorrer el resultado, este número se pasa como índice al arreglo de `Usuarios` llamado `listado` con el fin de asignar en esa misma posición al objeto usuario recuperado a partir de resultado obtenido por la consulta hecha a la BD.

Línea 162: verificamos si el arreglo tiene una tamaño menor o igual a 0, de ser cierto, esto quiere decir que la consulta no tuvo éxito, ósea que la BD no tiene usuario registrados, entonces se lanza una excepción notificando datos sobre el error.

Si todo anda bien y la ejecución logra llegar a la línea 166, quiere decir que si hubo resultados positivos en la consulta, por ende se debe retornar el arreglo que contiene una lista con los objetos de tipo `Usuario` que contienen en sus propiedades la información requerida.

Clase `ServletUsuario`:

Este clase actúa como controlador de la aplicación, es decir:

1. Posee métodos para capturar los datos enviados por el usuario a través de los formularios HTML ya sea mediante paginas dinámicas JSP o paginas estáticas HTML. Dichos métodos se llaman `doGet` y `doPost`.
2. Procesa los datos o campos enviados por cada formulario determinando si estos cumplen o no con el formato adecuado, si los datos cumplen con el formato, entonces el

Servlet determina cual método de la clase CRUDUsuario debe ser invocados con el fin de agregar, eliminar, actualizar o consultar dichas datos en la BD.

3. Una vez culminada la operación CRUD que se determinó realizar, el Servlet envía al usuario el resultado de dicha operación mediante paginas JSP.

```
8 package johnarrieta.controladores;
9
10 import java.io.IOException;
11 import java.io.PrintWriter;
12 import javax.servlet.ServletException;
13 import javax.servlet.annotation.WebServlet;
14 import javax.servlet.http.HttpServlet;
15 import javax.servlet.http.HttpServletRequest;
16 import javax.servlet.http.HttpServletResponse;
17 import johnarrieta.modelo.CRUDUsuario;
18 import johnarrieta.modelo.Usuario;
19
20 /**
21  * @author JohnArrieta
22  */
23 @WebServlet(name = "ServletUsuario", urlPatterns = {"/usuario"})
24 public class ServletUsuario extends HttpServlet {
25
26     /**...*/
27     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
28         throws ServletException, IOException {
29         response.setContentType("text/html;charset=UTF-8");
30         PrintWriter out = response.getWriter();
31         try {
32             String accion = request.getParameter("accion");// capturar la accion
33             if (accion.equals("agregar")) {
34                 CRUDUsuario crudAlguien = new CRUDUsuario();
35                 crudAlguien.getAlguien().setId(request.getParameter("id"));
36                 crudAlguien.getAlguien().setPassword(request.getParameter("password"));
37                 crudAlguien.getAlguien().setNombre(request.getParameter("nombre"));
38                 crudAlguien.getAlguien().setApellido(request.getParameter("apellido"));
39                 crudAlguien.getAlguien().setEmail(request.getParameter("email"));
40                 crudAlguien.getAlguien().setTipo(request.getParameter("tipo"));
41                 crudAlguien.agregarUsuario();
42             }
43         }
44     }
45 }
```


Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los

```
51 response.sendRedirect( "web/usuario/agregar.jsp?mensaje=Usuario "+request.getParameter("id")+ " Agregado al Sistema" );
52
53 } else if (accion.equals("buscar")) {
54     Usuario alguien = CRUDUsuario.consultarUsuario(request.getParameter("id"));
55     request.getSession().setAttribute("usuario.buscar", alguien);
56     String redirecion = request.getParameter("redir");
57     if (redirecion.equals("borrar")) {
58         response.sendRedirect( "web/usuario/eliminar.jsp" );
59     } else if (redirecion.equals("modificar")) {
60         response.sendRedirect( "web/usuario/modificar.jsp" );
61     } else {
62         response.sendRedirect( "web/usuario/buscar.jsp" );
63     }
64 } else if (accion.equals("modificar")) {
65     CRUDUsuario crudAlguien = new CRUDUsuario();
66     crudAlguien.getAlguien().setId(request.getParameter("id"));
67     crudAlguien.getAlguien().setPassword(request.getParameter("password"));
68     crudAlguien.getAlguien().setNombre(request.getParameter("nombre"));
69     crudAlguien.getAlguien().setApellido(request.getParameter("apellido"));
70     crudAlguien.getAlguien().setEmail(request.getParameter("email"));
71     crudAlguien.getAlguien().setTipo(request.getParameter("tipo"));
72     crudAlguien.modificarUsuario();
73     response.sendRedirect( "web/usuario/modificar.jsp?mensaje=Usuario "+request.getParameter("id")+ " Modificado en el Sistema" );
74 } else if (accion.equals("borrar")) {
75     CRUDUsuario crudAlguien = new CRUDUsuario();
76     crudAlguien.getAlguien().setId(request.getParameter("id"));
77     crudAlguien.eliminarUsuario();
78     response.sendRedirect( "web/usuario/eliminar.jsp?mensaje=Usuario "+request.getParameter("id")+ " Eliminado del Sistema" );
79 } else if (accion.equals("listartodo")) {
80     Usuario[] listado = CRUDUsuario.listarTodosLosUsuarios();
81     request.getSession().setAttribute("usuario.listar", listado);
82     response.sendRedirect( "web/usuario/listar.jsp" );
83 }
84
85     else if (accion.equals("login")) {
86         Usuario alguien = CRUDUsuario.iniciarSesion(request.getParameter("id"), request.getParameter("password"));
87         request.getSession().setAttribute("usuario.login", alguien);
88         response.sendRedirect( "index.jsp?mensaje=Bienvenido al Sistema" );
89     }
90     else if (accion.equals("salir")) {
91
92         request.getSession().setAttribute("usuario.login", null);
93         request.getSession().invalidate();
94         response.sendRedirect( "index.jsp?mensaje=Bienvenido al Sistema" );
95     }
96     else {
97         response.sendRedirect( "web/mensaje.jsp?mensaje=La Accion Solicitada no es Correcta" );
98     }
99 }
100 catch(Exception error){
101     response.sendRedirect( "web/mensaje.jsp?mensaje="+error.getMessage() );
102 }
103 finally {
104     out.close();
105 }
106 }
```

```
107
108 // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign
109 /**...*/
110 @Override
111 protected void doGet(HttpServletRequest request, HttpServletResponse response)
112     throws ServletException, IOException {
113     processRequest(request, response);
114 }
115
116 /**...*/
117 @Override
118 protected void doPost(HttpServletRequest request, HttpServletResponse response)
119     throws ServletException, IOException {
120     processRequest(request, response);
121 }
122
123 /**...*/
124 @Override
125 public String getServletInfo() {
126     return "Short description";
127 }
128 }
129 }
130 }
```

Línea 8: indica que la el archivo de código fuente .java y el archivo compilado .class serán almacenados en la carpeta controladores que a su vez está dentro de la carpeta johnarrieta.

Línea 23: definimos el nombre del servlet y el nombre de url por el cual debemos acceder al mismo.

Línea 24: definimos el nombre de nuestra clase Servlet y al mismo tiempo que lo heredamos de la clase HttpServlet

Líneas 10 a 16: se importan las clases necesarias para poder trabajar con **Servlet**, estas se encuentran en diferentes paquetes, por ejemplo el paquete **java.io** contiene dos clases llamadas **IOException** y **PrintWriter**, la primera nos permite trabajar con Excepciones o posible errores ocurridos en tiempo de ejecución, la segunda nos permite enviar contenido (HTML, CSS, JavaScript, JSON, XML, etc) al cliente web quien generalmente es un navegador web. El paquete **javax.servlet** contiene otros paquetes como por ejemplo **javax.servlet.http**, en el primero se encuentra la clase **ServletException**, la cual utilizamos para manipular los errores en tiempo de ejecución. En el segundo paquete se encuentran las clases **HttpServlet**, **HttpServletRequest** y **HttpServletResponse**. La clase **HttpServlet** es la clase de la cual debemos heredar todos los servlet de nuestras aplicaciones web, esta clase es la que define los métodos **doPost** y **doGet**, necesarios para recuperar la información (campos de formulario GET o POST) enviada en las peticiones o

conexiones realizadas desde el cliente Web o Navegador Web. Por otro lado las clases **HttpServletResponse** pose métodos para responder a dichas peticiones, operación que realiza invocando métodos de un objeto llamado **out** de tipo **PrintWriter**, mientras que la clase **HttpServletRequest** posee métodos para recuperar información enviada por el Cliente Web, esta información generalmente viene en formato POST o GET y se encuentra organizada en forma de campo=valor.

Todo servlet por defecto utiliza una conjunto de variables llamadas **request**, **response**, **out**, **sesión**, entre otras, estas variables son objetos de tipo **HttpServletRequest**, **HttpServletResponse**, **PrintWriter** y **HttpSession** respectivamente.

Línea 36 a 93: inicio o cabecera del método **processRequest**, cuerpo del método y fin de mismo. El método **processRequest** al ser invocado por el método **doPost** o **doGet** recibe como argumento el objeto **request** y el objeto **response** y puede lanzar una **Excepcion** en caso de que ocurra un problema al procesar la petición.

El método **processRequest** es un método propio o de nuestra clase **ServletUsuario**, por consiguiente podemos cambiarle el nombre si así lo deseamos. Dentro del cuerpo de este método debemos escribir el código java que defina la lógica que deseamos ejecutar cuando se realiza una petición hacia el servlet desde un cliente web.

En este caso, la lógica consiste en:

- Recuperar los campos de los formularios HTTP enviados en la petición hecha desde el cliente web.
- Uno de estos campos debe llamarse **acción**, cuyo valor indica justamente la acción que debe realizar el servlet al momento de recibir los datos enviados en la petición, en este ejemplo existen 6 formularios diferentes ubicados en la carpeta **web/usuario** llamados:
 - **agregar.jsp**: envía campos con los valores de un nuevo usuario a ser registrado o agregado al sistema, entre dichos campos también envía un campo especial y oculto (**hidden**) llamado **acción** cuyo valor es **agregar**.
 - **eliminar.jsp**: envía solo dos campos, uno con el **id** del usuario a eliminar y otro campo oculto llamado **acción** cuyo valor es **eliminar**.
 - **modificar.jsp**: envía campos con los valores nuevos valores del usuario a ser modificado o actualizado en el sistema, también envía otro campo oculto llamado **acción** cuyo valor es **modificar**.
 - **buscar.jsp**: envía solo dos campos, uno con el **id** del usuario a buscar y otro campo oculto llamado **acción** cuyo valor es **buscar**.
 - **listar.jsp**: envía solo el campo **acción** cuyo valor es **listartodo**
 - **login.jsp**: envía los campos **id** y **password** del usuario que desea iniciar sesión en el sistema, pero también envía el campo **acción** cuyo valor es **login**
- Una vez el servlet recupera los campos con sus respectivos valores, utiliza el valor del campo **acción** para determinar qué operación o método de la clase **CRUDUsuario** deberá invocar para sea ejecutada sobre la base de datos.

- Si la operación se ejecuta sin excepciones o errores, el servlet redirección el flujo del programa hacia la misma página `.jps` que realizó la petición, notificando con un mensaje explicativo de satisfacción. En caso contrario, es decir si ocurre un error, el servlet redirecciona el flujo del programa hacia una página llamada `mensaje.jsp` enviándole un parámetro con un texto explicativo el error.

Líneas 38 y 39: se le envía como respuesta al cliente web un comando `http` que le indica que todo el contenido siguiente que le será enviado esta en formato de texto `html` codificado con caracteres `UTF-8`. Luego a partir del objeto `response` obtenemos un nuevo objeto de tipo `PrintWriter` llamado `out`, el cual se especializa en enviar caracteres de texto hacia el cliente web

Líneas 41: utilizamos el objeto `request` de tipo `HttpServletRequest` con el fin de invocar al método `getParameter("nombre_parametro")`; con el cual obtenemos el valor del parámetro que se le pasa como argumento, parámetro que se espera haya sido enviado en la petición realizada al servlet, vemos que el parámetro recuperado es el parámetro acción, cuyo valor es asignado a una variable justamente llamada `acción`;

Línea 42: verificamos que si el valor del parámetro `acción` es igual a la palabra `agregar`, si esto es verdadero, entonces, Línea 43 se crea un objeto de tipo `CRUDUsuario` y Línea 44 a 49 lo utilizamos para invocar el método `getAlguien` el cual retorna un objeto de tipo `Usuario`, al que su vez le pasamos valores a sus variables invocando sus métodos `setxx` y pasándole como argumento el valor de los respectivos parámetros `xx` enviado con en la petición.

Línea 50: invocamos el método `agregarUsuario` de la clase `CRUDUsuario`, método que agrega mediante SQL `INSERT` los datos del usuario en la BD.

Línea 51: utilizamos el objeto `response` para invocar al método `sendRedirect` pasándole como argumento la ruta de la pagina `.jsp` que mostrar el mensaje de respuesta al resultado de la operación realizada.

Líneas 53 a 63: verificamos si la `acción` enviada es igual a `buscar`, si esto es verdadero, entonces línea 54 utilizamos el nombre de la clase `CRUDUsuario` para invocar a su método `statico consultarUsuario`, el cual recibe como argumento el `id` del usuario que se desea consultar, `id` que es recuperado como parámetro de los datos enviados en la petición, si la llamada este método arroja un resultado positivo, este retornara o devolverá un objeto de tipo `Usuario` con los datos del usuario que fue encontrado en la bd, de lo contrario lanzara una `excepción` notificando información sobre el error.

Línea 55: utilizamos el objeto `request` para invocar su método `getSession`, el cual retorna un objeto de tipo `HttpSession`, objeto que posee métodos que nos permiten almacenar datos (objetos o valores primitivos) en forma de `atributos` de la petición, de tal forma que puedan ser recuperados por otras `servlets` o paginas `JSP`, es decir, un objeto de tipo `HttpSession` posee métodos que nos permiten compartir información entre diferentes `servlet` o Diferentes paginas `JSP`. Es por eso que en esta misma línea invocamos a su método `setAttribute` el

cual recibe como argumentos el **nombre** con el que será guardado el atributo y el **valor** que le será asignado a dicho atributo, en este caso el nombre será "usuario.buscar" y el valor será el objeto **usuario** que fue encontrado en la bd.

Línea 56: se obtiene el valor del parámetro **redir**, el cual indica al **servlet** cual a cual operación será redireccionado el flujo del programa si se desea por ejemplo modificar o eliminar los datos del usuario consultado.

Líneas 79 a 83: verificamos si la **acción** enviada es igual a **listartodo**, si esto es verdadero, entonces línea 80, utilizamos el nombre de la clase **CRUDUsuario** para invocar a su método **estatico listarTodosLosUsuarios**, el cual no recibe argumentos y retorna un **arreglo** de objetos de tipo **Usuario** como producto de la consulta positiva sobre la BD, de lo contrario, si ocurre un error o la BD está vacía en cuanto a usuarios se refiere, el método lanza una **excepción** notificando información sobre el problema. Posteriormente línea 81, guardamos en un **atributo de Session** el **arreglo** con los **usuarios** consultados en la BD, el nombre con el cual podremos recuperar dicho **arreglo** es **usuario.listar**. Por último línea 82, redireccionamos el flujo del programa hacia la página JSP **listar.jsp**, la cual deberá recuperar el **arreglo** de objetos de tipo **usuario** guardados como **atributo** de la sesión y procederá a mostrarlos uno a uno dentro de una tabla **html**.

Página login.jsp

Iniciar Sesión en el Sistema

ID:	<input type="text"/>
Password:	<input type="password"/>
<input type="button" value="ENTRAR"/>	<input type="button" value="Restablecer"/>

```
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <%
9     if (request.getSession().getAttribute("usuario.login") == null) {
10         getServletContext().getRequestDispatcher("/web/usuario/login.jsp").forward(request,
11     }
12 %>
13 <!DOCTYPE html>
14 <html>
15 <head>
16     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
17     <title>Menu de la Aplicacion</title>
18 </head>
19 <body>
20 <center>
21     <table border="0">
22     <tbody>
23     <tr>
24         <th><h1><a href="web/usuario/agregar.jsp" >Agregar</a></h1></th>
25     </tr>
26     <tr>
27         <td><h1><a href="web/usuario/buscar.jsp" >Buscar</a></h1></td>
28     </tr>
29     <tr>
30         <td><h1><a href="web/usuario/modificar.jsp" >Modificar</a></h1></td>
31     </tr>
32     <tr>
33         <td><h1><a href="web/usuario/eliminar.jsp" >Eliminar</a></h1></td>
34     </tr>
35     <tr>
36         <td><h1><a href="usuario?accion=listartodo" >Listar</a></h1></td>
37     </tr>
38     <tr>
39         <td><h1><a href="/ejemplosesion/usuario?accion=logout" >Salir</a></h1></td>
40     </tr>
41     </tbody>
42 </table>
43 </center>
44 </body>
45 </html>
```

Pagina index.jsp.

Menu de la aplicación

[Agregar](#)
[Buscar](#)
[Modificar](#)
[Eliminar](#)
[Listar](#)
[Salir](#)

```
1 <%--
2 Document : index
3 Created on : 9/10/2013, 12:11:50 AM
4 Author : JohnArrieta
5 --%>
6
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <%
9     if (request.getSession().getAttribute("usuario.login") == null) {
10         getServletContext().getRequestDispatcher("/web/usuario/login.jsp").forward(request, response);
11     }
12 %>
13 <!DOCTYPE html>
14 <html>
15 <head>
16 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
17 <title>Menu de la Aplicacion</title>
18 </head>
19 <body>
20 <center>
21 <h2>Menu de la aplicación</h2>
22 <hr/>
23 <table border="0">
24 <tr>
25 <td><a href="web/usuario/agregar.jsp" >Agregar</a></td>
26 </tr>
27 <tr>
28 <td><a href="web/usuario/buscar.jsp" >Buscar</a></td>
29 </tr>
30 <tr>
31 <td><a href="web/usuario/modificar.jsp" >Modificar</a></td>
32 </tr>
33 <tr>
34 <td><a href="web/usuario/eliminar.jsp" >Eliminar</a></td>
35 </tr>
36 <tr>
37 <td><a href="usuario?accion=listartodo" >Listar</a></td>
38 </tr>
39 <tr>
40 <td><a href="/ejemplo sesion/usuario?accion=logout" >Salir</a></td>
41 </tr>
42 </table>
43 <hr/>
44 </center>
45 </body>
46 </html>
```

Página mensaje.jsp

El ID y el Password del Usuario son Necesarios

[<<< Volver >>>](#)

```
1  <!--
2      Document      : mensaje
3      Created on    : 9/10/2013, 01:00:54 PM
4      Author       : JohnArrieta
5  -->
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>Mensaje del Sistema</title>
13     </head>
14     <body>
15         <center>
16             <h1>
17                 <%=request.getParameter("mensaje") %>
18             </h1>
19             <hr/>
20             <a href="usuario/login.jsp"><<< Volver ::: </a>
21         </center>
22     </body>
23 </html>
```

Página index.jsp

Menu de la aplicación

[Agregar](#)
[Buscar](#)
[Modificar](#)
[Eliminar](#)
[Listar](#)
[Salir](#)

Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)

```
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <%
9     if (request.getSession().getAttribute("usuario.login") == null) {
10         getServletContext().getRequestDispatcher("/web/usuario/login.jsp").forward(request, response);
11     }
12 %>
13 <!DOCTYPE html>
14 <html>
15 <head>
16 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
17 <title>Menu de la Aplicacion</title>
18 </head>
19 <body>
20 <center>
21 <h2>Menu de la aplicación</h2>
22 <hr/>
23 <table border="0">
24 <tr>
25 <td><a href="web/usuario/agregar.jsp" >Agregar</a></td>
26 </tr>
27 <tr>
28 <td><a href="web/usuario/buscar.jsp" >Buscar</a></td>
29 </tr>
30 <tr>
31 <td><a href="web/usuario/modificar.jsp" >Modificar</a></td>
32 </tr>
33 <tr>
34 <td><a href="web/usuario/eliminar.jsp" >Eliminar</a></td>
35 </tr>
36 <tr>
37 <td><a href="usuario?accion=listartodo" >Listar</a></td>
38 </tr>
39 <tr>
40 <td><a href="/ejemplosesion/usuario?accion=logout" >Salir</a></td>
41 </tr>
42 </table>
43 <hr/>
44 </center>
45 </body>
46 </html>
```

Página agregar.jsp

Agregar Usuario

ID:	<input type="text"/>
Password:	<input type="password"/>
Nombre:	<input type="text"/>
Apellido:	<input type="text"/>
Email:	<input type="text"/>
Tipo:	<input type="text" value="Administrador"/>
<input type="button" value="ENTRAR"/> <input type="button" value="Restablecer"/>	

Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)

```
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <%
9 if(request.getSession().getAttribute("usuario.login") == null ){
10     getServletContext().getRequestDispatcher("/web/usuario/login.jsp").forward(request, response);
11 }
12 String mensaje = request.getParameter("mensaje");
13 %>
14 <!DOCTYPE html>
15 <html>
16 <head>
17 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
18 <title>Agregar Usuario al Sistema</title>
19 </head>
20 <body>
21 <center>
22 <h1>Agregar Usuario </h1>
23 <hr/>
24 <form action = "/ejemplosesion/usuario?accion=agregar" method="post">
25 <table>
26 <tr>
27 <th style="text-align: right">ID:</th>
28 <th><input type="text" name="id"/></th>
29 </tr>
30 <tr>
31 <th style="text-align: right">Password:</th>
32 <th><input type="password" name="password"/></th>
33 </tr>
34 <tr>
35 <th style="text-align: right">Nombre:</th>
36 <th><input type="text" name="nombre"/></th>
37 </tr>
38 <tr>
39 <th style="text-align: right">Apellido:</th>
40 <th><input type="text" name="apellido"/></th>
41 </tr>
42 <tr>
43 <th style="text-align: right">Email:</th>
44 <th><input type="text" name="email"/></th>
45 </tr>
46 <tr>
47 <th style="text-align: right">Tipo:</th>
48 <th>
49 <select name="tipo">
50 <option value="Administrador">Administrador</option>
51 <option value="Cliente">Cliente</option>
52 </select>
53 </th>
54 </tr>
55 <tr>
56 <th><input type="submit" value="ENTRAR"></th>
57 <th><input type="reset" name="LIMPIAR"/></th>
58 </tr>
59 </table>
60 </form>
61 <hr/>
62 <p style="color:#FF0000;">
63 <%= (mensaje != null && ! mensaje.isEmpty())?mensaje:"">
64 </p>
65 </center>
66 </body>
67 </html>
```


Página buscar.jsp

Buscar Usuario

ID:

Buscar

Restablecer

Password:

Nombre:

Apellido:

Email:

Tipo:

Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)

```
7 <%@page import="johnarrieta.modelo.Usuario"%>
8 <%@page contentType="text/html" pageEncoding="UTF-8"%>
9 <%
10 if(request.getSession().getAttribute("usuario.login") == null ){
11     getServletContext().getRequestDispatcher("/web/usuario/login.jsp").forward(request, response);
12 }
13 String mensaje = request.getParameter("mensaje");
14 Usuario alguien = (Usuario)request.getSession().getAttribute("usuario.buscar");
15 %>
16 <!DOCTYPE html>
17 <html>
18 <head>
19     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
20     <title>Consultar Usuario</title>
21 </head>
22 <body>
23     <center>
24         <h1>Buscar Usuario </h1>
25         <hr/>
26         <form action ="/ejemplo sesion/usuario?accion=buscar&redir=buscar" method="post">
27             <table>
28                 <tr>
29                     <th style="text-align: right">ID:</th>
30                     <td><input type="text" name="id"/></td>
31                 </tr>
32                 <tr>
33                     <th><input type="submit" value="Buscar"></th>
34                     <td><input type="reset" name="Limpiar"/></td>
35                 </tr>
36                 <tr>
37                     <th style="text-align: right">Password:</th>
38                     <td style="text-align: left"><%= (alguien != null) ? "*****" : "" %></td>
39                 </tr>
40                 <tr>
41                     <th style="text-align: right">Nombre:</th>
42                     <td style="text-align: left"><%= (alguien != null) ? alguien.getNombre() : "" %></td>
43                 </tr>
44                 <tr>
45                     <th style="text-align: right">Apellido:</th>
46                     <td style="text-align: left"><%= (alguien != null) ? alguien.getApellido() : "" %></td>
47                 </tr>
48                 <tr>
49                     <th style="text-align: right">Email:</th>
50                     <td style="text-align: left"><%= (alguien != null) ? alguien.getEmail() : "" %></td>
51                 </tr>
52                 <tr>
53                     <th style="text-align: right">Tipo:</th>
54                     <td style="text-align: left">
55                         <%= (alguien != null) ? alguien.getTipo() : "" %>
56                     </td>
57                 </tr>
58             </table>
59         </form>
60         <hr/>
61         <p style="color:#FF0000;">
62             <%= (mensaje != null && ! mensaje.isEmpty()) ? mensaje : "" %>
63         </p>
64         <%= request.getSession().setAttribute("usuario.buscar", null); %>
65     </center>
66 </body>
67 </html>
```

Página eliminar.jsp

Eliminar Usuario

ID:

Buscar

Restablecer

Password:

Nombre:

Apellido:

Email:

Tipo:

Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)

```
7 <%@page import="johnarrieta.modelo.Usuario"%>
8 <%@page contentType="text/html" pageEncoding="UTF-8"%>
9
10 <%
11     if (request.getSession().getAttribute("usuario.login") == null) {
12         getServletContext().getRequestDispatcher("/web/usuario/login.jsp").forward(request, response);
13     }
14     String mensaje = request.getParameter("mensaje");
15     Usuario alguien = (Usuario) request.getSession().getAttribute("usuario.buscar");
16 %>
17 <!DOCTYPE html>
18 <html>
19 <head>
20 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
21 <title>Eliminar Usuario</title>
22 </head>
23 <body>
24 <center>
25 <h1>Eliminar Usuario </h1>
26 <hr/>
27 <form action ="/ejemplosesion/usuario?accion=buscar&redir=borrar" method="post">
28 <table>
29 <tr>
30 <th style="text-align: right">ID:</th>
31 <td><input type="text" name="id"/></td>
32 </tr>
33 <tr>
34 <th><input type="submit" value="Buscar"></th>
35 <td><input type="reset" name="Limpiar"/></td>
36 </tr>
37 <tr>
38 <th style="text-align: right">Password:</th>
39 <td style="text-align: left"><%= (alguien != null) ? "*****" : ""%></td>
40 </tr>
41 <tr>
42 <th style="text-align: right">Nombre:</th>
43 <td style="text-align: left"><%= (alguien != null) ? alguien.getNombre() : ""%></td>
44 </tr>
45 <tr>
46 <th style="text-align: right">Apellido:</th>
47 <td style="text-align: left"><%= (alguien != null) ? alguien.getApellido() : ""%></td>
48 </tr>
49 <tr>
50 <th style="text-align: right">Email:</th>
51 <td style="text-align: left"><%= (alguien != null) ? alguien.getEmail() : ""%></td>
52 </tr>
53 <tr>
54 <th style="text-align: right">Tipo:</th>
55 <td style="text-align: left">
56 <%= (alguien != null) ? alguien.getTipo() : ""%>
57 </td>
58 </tr>
59 </table>
60 </form>
61 <hr/>
62 <%
63     if (alguien != null) {
64 %>
65 <form action ="/ejemplosesion/usuario?accion=borrar" method="post">
66 <input type="hidden" name="id" value="<%= alguien.getId()%>"
67 <table>
68 <tr><td><input type="submit" value="Eliminar"/></td></tr>
69 </table>
70 </form>
71 <%
72     }
73 %>
74
75
76
77 <p style="color:#FF0000;">
78 <%= (mensaje != null && !mensaje.isEmpty()) ? mensaje : ""%>
79 </p>
80 <%= request.getSession().setAttribute("usuario.buscar", null);%>
81 </center>
82 </body>
83 </html>
```

Página modificar.jsp

```
7 <%@page import="johnarrieta.modelo.Usuario"%>
8 <%@page contentType="text/html" pageEncoding="UTF-8"%>
9
10 <%
11     if (request.getSession().getAttribute("usuario.login") == null) {
12         getServletContext().getRequestDispatcher("/web/usuario/login.jsp").forward(request, response);
13     }
14     String mensaje = request.getParameter("mensaje");
15     Usuario alguien = (Usuario) request.getSession().getAttribute("usuario.buscar");
16 %>
17 <!DOCTYPE html>
18 <html>
19 <head>
20 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
21 <title>Modificar Usuario</title>
22 </head>
23 <body>
24 <center>
25 <h1>Modificar Usuario </h1>
26 <hr/>
27 <form action="/ejemplosesion/usuario?accion=buscar&redir=modificar" method="post">
28 <table>
29 <tr>
30 <th style="text-align: right">ID:</th>
31 <td><input type="text" name="id"/></td>
32 </tr>
33 <tr>
34 <th><input type="submit" value="Buscar"></th>
35 <td><input type="reset" name="Limpiar"/></td>
36 </tr>
37 </table>
38 </form>
39 <hr/>
40
41 <%
42     if (alguien != null) {
43 %>
44 <form action="/ejemplosesion/usuario?accion=modificar" method="post">
45 <table>
46 <tr>
47 <th style="text-align: right">ID:</th>
48 <td><input type="text" name="id" value="<%= (alguien != null) ? alguien.getId() : ""%>" readonly="readonly"/></td>
49 </tr>
50 <tr>
51 <th style="text-align: right">Password:</th>
52 <td><input type="password" name="password" value="<%= (alguien != null) ? alguien.getPassword() : ""%>"/></td>
53 </tr>
54 <tr>
55 <th style="text-align: right">Nombre:</th>
56 <td><input type="text" name="nombre" value="<%= (alguien != null) ? alguien.getNombre() : ""%>"/></td>
57 </tr>
58 <tr>
59 <th style="text-align: right">Apellido:</th>
60 <td><input type="text" name="apellido" value="<%= (alguien != null) ? alguien.getApellido() : ""%>"/></td>
61 </tr>
62 <tr>
63 <th style="text-align: right">Email:</th>
64 <td><input type="text" name="email" value="<%= (alguien != null) ? alguien.getEmail() : ""%>"/></td>
65 </tr>
66 <tr>
67 <th style="text-align: right">Tipo:</th>
68 <td>
69 <select name="tipo">
70 <option value="Administrador"
71 <%= (alguien != null && alguien.getTipo().equals("Administrador")) ? "selected" : ""%> >
72 Administrador</option>
73 <option value="Cliente"
74 <%= (alguien != null && alguien.getTipo().equals("Cliente")) ? "selected" : ""%> >
75 Cliente</option>
76 </select>
77 </td>
78 </tr>
79 <tr>
80 <td><input type="submit" value="Modificar"></td>
81 <td><input type="reset" name="Limpiar"/></td>
82 </tr>
83 </table>
84 </form>
85 <hr/>
86 <%
87     }
88 %>
89 <p style="color:#FF0000;">
90 <%= (mensaje != null && !mensaje.isEmpty()) ? mensaje : ""%>
91 </p>
92 <% request.getSession().setAttribute("usuario.buscar", null);%>
93 </center>
94 </body>
95 </html>
```

Página a listar.jsp

Todos los Usuarios Agregados al Sistema

Item	ID	Nombre	Apellido	Email	Tipo
1	admin	John Carlos	Arrieta Arrieta	arrietajohn@hotmail.com	Administrador
2	sebastian	Sebastian Camilo	Arrieta Villarreal	sebasravi@hotmail.com	Cliente

[<<: VOLVER AL MENU](#)

Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)

```
1  <%--
2      Document      : listar
3      Created on    : 9/10/2013, 09:26:43 AM
4      Author       : JohnArrieta
5  --%>
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <%@page session="true" %>
9  <%@page import="johnarrieta.modelo.Usuario" %>
10
11  <%
12      Usuario listado[] = (Usuario[]) session.getAttribute("usuario.listar");
13      String mensaje = null;
14      if (listado == null || listado.length <= 0) {
15          mensaje = "Resultado: 0 Usuarios encontrados en el Sistema";
16      }
17  %>
18
19  <!DOCTYPE html>
20  <html>
21  <head>
22      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
23      <title>JSP Page</title>
24  </head>
25  <body>
26      <center>
27          <h1>Todos los Usuarios Agregados al Sistema </h1>
28          <%
29              if (mensaje != null) {
30                  out.print(mensaje);
31              } else {
32
33                  <table border="1">
34                      <thead>
35                          <tr>
36                              <th>Item</th>
37                              <th>ID</th>
38                              <th>Nombre</th>
39                              <th>Apellido</th>
40                              <th>Email</th>
41                              <th>Tipo</th>
42                          </tr>
43                      </thead>
44                      <tbody>
45                          <%
46                              int contador = 0;
47                              for (Usuario alguien : listado) {
48                                  contador = contador + 1;
49
50                                  <tr>
51                                      <td><%= contador%></td>
52                                      <td><%= alguien.getId()%></td>
53                                      <td><%= alguien.getNombre()%></td>
54                                      <td><%= alguien.getApellido()%></td>
55                                      <td><%= alguien.getEmail()%></td>
56                                      <td><%= alguien.getTipo()%></td>
57                                  </tr>
58                              <%
59                                  }
60                              %>
61                          </tbody>
62                      </table>
63
64                      <%
65                          }
66                      %>
67                  <hr>
68                  <a href="../../index.jsp"><:: VOLVER AL MENU</a>
69              </center>
70  </body>
71  </html>
```


Guía didáctica: aprender a desarrollar una aplicación básica para internet mediante el uso de los lenguajes HTML, SQL, Java y tecnología Servlet JSP (AUTOR: ING. JOHN CARLOS ARRIETA ARRIETA)