CRUD BÁSICO HECHO EN JSP

Autor: Juan Sebastián Pinzón Sánchez

Fecha: 11 de agosto de 2025

Versión: 1.0

Introducción:

- **¿Qué es un CRUD?:** Un CRUD es un conjunto de operaciones básicas para gestionar datos en una aplicación, en total son cuatro funciones las cuales son, Create (crear registros), Read (leer o consultar los registros existentes), Update (actualizar los registros existentes) y Delete (eliminar algún registro existente), estas normalmente se implementan con instrucciones SQL (INSERT, SELECT, UPDATE, DELETE) y se integran a una interfaz que permita al usuario realizar estas acciones de forma segura y eficiente.

- **Lenguaje empleado:** Este CRUD se realizó en JSP porque permite combinar Java y HTML en un mismo archivo, lo que permite una gran facilidad a la hora de crear páginas web dinámicas que interactúan directamente con bases de datos a través de JDBC.

- **¿Para qué sirve?:** Este CRUD sirve para gestionar de forma simple el registro de estudiantes en una base de datos, permitiendo que un usuario autenticado (que inició sesión exitosamente) pueda agregar nuevos estudiantes, consultar la lista de los estudiantes agregados hasta ese momento, editar sus datos y eliminarlos según sea necesario. Esto facilita mantener la información actualizada, organizada y accesible.

- **Objetivo del proyecto:** Desarrollar un sistema web en JSP que permita la gestión eficiente y segura de estudiantes mediante operaciones CRUD.

Requerimientos:

1. **Software utilizado:**

    - **Sistema operativo:** Windows 10 (o superior)
    - **Servidor web:** Apache Tomcat 10 (o compatible con JSP)
    - **Lenguaje:** Java SE 8

- **Base de datos:** MySQL 8
- **Navegador web:** Chrome, Opera o cualquiera de la preferencia del usuario

2. **Herramientas de desarrollo:**

    - **IDE:** Apache NetBeans IDE 23
    - MySQL Workbench para la gestión de la base de datos
    - Conector JDBC para establecer la conexión entre Java y MySQL

3. **Dependencias y librerías:**

    - Conector JDBC de MySQL
    - Configuración del web.xml para el despliegue en Tomcat
    - Drivers para la compatibilidad con MySQL 8

Descripción del CRUD:

- **Login:**

    El Login funciona verificando las credenciales ingresadas por el usuario contra los registros almacenados en la base de datos. Cuando el formulario se envía por medio de un POST, se obtiene el usuario y la contraseña y se ejecuta una consulta SQL preparada para buscar en la tabla usuarios un registro que coincida con ambos valores. Si se encuentra un resultado, se guarda el nombre de usuario en la sesión para mantenerlo autenticado y se redirige a menu.jsp, de lo contrario se muestra un mensaje de error.

```jsp
<%
    String error = null;
    if (request.getMethod().equalsIgnoreCase("POST")) {
        String usuario = request.getParameter("usuario");
        String contrasena = request.getParameter("contrasena");

        Connection con = null;
        PreparedStatement ps = null;
        ResultSet rs = null;

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/crud_estudiantes", "root", "js1216");

            String sql = "SELECT * FROM usuarios WHERE usuario = ? AND contrasena = ?";
            ps = con.prepareStatement(sql);
            ps.setString(1, usuario);
            ps.setString(2, contrasena);
            rs = ps.executeQuery();
```

```jsp
            if (rs.next()) {
                session.setAttribute("usuario", usuario);
                response.sendRedirect("menu.jsp");
            } else {
                error = "Usuario o contraseña incorrectos";
            }
        } catch (Exception e) {
            error = "Error: " + e.getMessage();
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException ignored) {}
            if (ps != null) try { ps.close(); } catch (SQLException ignored) {}
            if (con != null) try { con.close(); } catch (SQLException ignored) {}
        }
    }
%>
```

```jsp
    <div class="login-container">
        <h2>Inicio de Sesión</h2>
        <% if (error != null) { %>
            <p class="error"><%= error %></p>
        <% } %>
        <form action="login.jsp" method="post">
            <input type="text" name="usuario" placeholder="Usuario" required><br>
            <input type="password" name="contrasena" placeholder="Contraseña" required><br>
            <input type="submit" value="Iniciar Sesión">
        </form>
    </div>
</body>
</html>
```

- **Create:**

  Este modulo del CRUD permite registrar un nuevo estudiante en la base de datos mediante un formulario simple que solo pide tres datos, el nombre, la edad y la carrera del estudiante, este formulario está protegido por sesión, es decir que solo los usuarios autenticados pueden realizar este proceso, al enviar los datos con el método POST el sistema obtiene los valores y establece una conexión con MySQL usando el driver JDBC y ejecuta una sentencia INSERT para almacenar los datos en la tabla "estudiantes" de la base de datos, una vez se completa el registro el sistema automáticamente redirige al usuario a la lista de estudiantes para que pueda ver el nuevo dato agregado mas los que ya existen.

```
<%
    if (session.getAttribute("usuario") == null) {
        response.sendRedirect("login.jsp");
    }

    String nombre = request.getParameter("nombre");
    int edad = Integer.parseInt(request.getParameter("edad"));
    String carrera = request.getParameter("carrera");

    Connection con = null;
    PreparedStatement ps = null;

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        // Cambia 'root' y '' a tus credenciales reales de MySQL
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/crud_estudiantes", "root", "js1216");

        String sql = "INSERT INTO estudiantes (nombre, edad, carrera) VALUES (?, ?, ?)";
        ps = con.prepareStatement(sql);
        ps.setString(1, nombre);
        ps.setInt(2, edad);
        ps.setString(3, carrera);
        ps.executeUpdate();
```

```
        response.sendRedirect("listar.jsp");
    } catch (Exception e) {
        out.println("Error: " + e.getMessage());
    } finally {
        if (ps != null) ps.close();
        if (con != null) con.close();
    }
%>
```

```
    </style>
</head>
<body>
    <div class="form-container">
        <h2>Agregar Nuevo Estudiante</h2>
        <form action="insertar.jsp" method="post">
            <input type="text" name="nombre" required placeholder="Nombre"><br>
            <input type="number" name="edad" required placeholder="Edad"><br>
            <input type="text" name="carrera" required placeholder="Carrera"><br>
            <input type="submit" value="Guardar">
        </form>
        <a class="back-link" href="menu.jsp">⬅ Volver al Menú</a>
    </div>
</body>
</html>
```

- **Read:**

El modulo Listar del CRUD muestra todos los registros de estudiantes almacenados en la base de datos, permitiendo al usuario visualizar de forma ordenada la información existente. Al ingresar a listar.jsp, el sistema establece una conexión con MySQL mediante JDBC, ejecuta una consulta preparada y recorre los resultados con un set de resultados, el cual se genera dinámicamente en una tabla HTML con los datos de cada estudiante, además, en cada fila se incluyen botones de editar y eliminar en caso de que el administrador necesite modificar los datos de un estudiante o por el contrario eliminarlo.

```
<%
    if (session.getAttribute("usuario") == null) {
        response.sendRedirect("login.jsp");
    }

    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/crud_estudiantes", "root", "js1216");
        stmt = con.createStatement();
        rs = stmt.executeQuery("SELECT * FROM estudiantes");
%>
```

```
<body>
    <div class="table-container">
        <h2>Lista de Estudiantes</h2>
        <table>
            <tr>
                <th>ID</th>
                <th>Nombre</th>
                <th>Edad</th>
                <th>Carrera</th>
                <th>Acciones</th>
            </tr>
            <%
                while (rs.next()) {
                    int id = rs.getInt("id");
                    String nombre = rs.getString("nombre");
                    int edad = rs.getInt("edad");
                    String carrera = rs.getString("carrera");
            %>
```

```
<tr>
    <td><%= id %></td>
    <td><%= nombre %></td>
    <td><%= edad %></td>
    <td><%= carrera %></td>
    <td>
```

```
<%
    } catch (Exception e) {
        out.println("Error: " + e.getMessage());
    } finally {
        if (rs != null) rs.close();
        if (stmt != null) stmt.close();
        if (con != null) con.close();
    }
%>
```

- **Update:**

  El modulo editar del CRUD permite modificar la información de un estudiante previamente registrado en la base de datos. Primero, en editar.jsp, el sistema recibe el id del estudiante desde la lista, luego realiza una consulta preparada para obtener los datos actuales de ese registro y mostrarlos precargados en un formulario HTML, el usuario puede editar cualquier dato y al enviar el formulario, los datos se envían por el método POST a actualizar.jsp, en este archivo se reciben los valores actualizados y mediante una sentencia UPDATE en MySQL se reemplazan los datos y finalmente el sistema redirige al usuario a la lista para que pueda visualizar los cambios realizados.

```
<%
    if (session.getAttribute("usuario") == null) {
        response.sendRedirect("login.jsp");
    }

    int id = Integer.parseInt(request.getParameter("id"));

    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/crud_estudiantes", "root", "js1216");

        String sql = "SELECT * FROM estudiantes WHERE id = ?";
        ps = con.prepareStatement(sql);
        ps.setInt(1, id);
        rs = ps.executeQuery();

        if (rs.next()) {
            String nombre = rs.getString("nombre");
            int edad = rs.getInt("edad");
            String carrera = rs.getString("carrera");
%>
```

```
<body>
    <div class="form-container">
        <h2>Editar Estudiante</h2>
        <form action="actualizar.jsp" method="post">
            <input type="hidden" name="id" value="<%= id %>">
            <input type="text" name="nombre" value="<%= nombre %>" required placeholder="Nombre"><br>
            <input type="number" name="edad" value="<%= edad %>" required placeholder="Edad"><br>
            <input type="text" name="carrera" value="<%= carrera %>" required placeholder="Carrera"><br>
            <input type="submit" value="Actualizar">
        </form>
        <a class="back-link" href="listar.jsp">⬅ Volver a la Lista</a>
    </div>
</body>
</html>
```

```jsp
<%
    if (session.getAttribute("usuario") == null) {
        response.sendRedirect("login.jsp");
    }

    int id = Integer.parseInt(request.getParameter("id"));
    String nombre = request.getParameter("nombre");
    int edad = Integer.parseInt(request.getParameter("edad"));
    String carrera = request.getParameter("carrera");

    Connection con = null;
    PreparedStatement ps = null;
```

```jsp
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        // Cambia 'root' y '' a tus credenciales reales de MySQL
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/crud_estudiantes", "root", "js1216");

        String sql = "UPDATE estudiantes SET nombre = ?, edad = ?, carrera = ? WHERE id = ?";
        ps = con.prepareStatement(sql);
        ps.setString(1, nombre);
        ps.setInt(2, edad);
        ps.setString(3, carrera);
        ps.setInt(4, id);
        ps.executeUpdate();

        response.sendRedirect("listar.jsp");
    } catch (Exception e) {
        out.println("Error: " + e.getMessage());
    } finally {
        if (ps != null) ps.close();
        if (con != null) con.close();
    }
%>
```

- **Delete:**

  El modulo eliminar del CRUD se encarga de borrar cualquier registro de un estudiante de la base de datos, cuando el usuario hace clic en el botón eliminar desde la lista, se envía el id del estudiante a eliminar.jsp. En este archivo, usando una conexión a MySQL y una consulta preparada se ejecuta la sentencia DELETE para eliminar el registro cuyo id coincida con el recibido. Una vez borrado, el sistema redirige automáticamente a la lista.

```
<%
    if (session.getAttribute("usuario") == null) {
        response.sendRedirect("login.jsp");
    }

    int id = Integer.parseInt(request.getParameter("id"));

    Connection con = null;
    PreparedStatement ps = null;

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        // Cambia 'root' y '' a tus credenciales reales de MySQL
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/crud_estudiantes", "root", "js1216");

        String sql = "DELETE FROM estudiantes WHERE id = ?";
        ps = con.prepareStatement(sql);
        ps.setInt(1, id);
        ps.executeUpdate();

        response.sendRedirect("listar.jsp");
    } catch (Exception e) {
        out.println("Error: " + e.getMessage());
    } finally {
        if (ps != null) ps.close();
        if (con != null) con.close();
    }
%>
```

Prompts utilizados por modulo:

- **Login:**

  Primer prompt: “Genera código JSP que conecte a una base de datos MySQL usando JDBC para validar un login. Recibe usuario y contraseña del formulario, verifica en la tabla usuarios con una consulta preparada que ambos datos existan, y si es correcto crea una sesión y redirige a menu.jsp, si no muestra error.”

  Segundo prompt: “Agrega un formulario de login centrado con fondo claro, caja blanca con bordes redondeados y sombra suave, campos para usuario y contraseña, botón verde con efecto hover, y mensajes de error en rojo.”

- **Create:**

  Primer prompt: “Escribe código JSP que reciba desde un formulario los datos nombre, edad y carrera para un nuevo estudiante, se conecte a MySQL usando JDBC y los inserte en la tabla estudiantes con una sentencia preparada. Luego redirige a la lista de estudiantes.”

  Segundo prompt: “Crea un formulario limpio y centrado para agregar estudiante, con campos para nombre, edad y carrera, fondo blanco, bordes redondeados, botón verde ancho completo con efecto hover, y un enlace para volver al menú principal.”

- **Read:**

  Primer prompt: “Crea código JSP que se conecte a MySQL con JDBC y ejecute una consulta SELECT * FROM estudiantes. Obtén los datos dinámicamente y prepáralos para mostrarlos en una tabla HTML.”

- **Segundo prompt:** “Genera una tabla estilizada con encabezados verdes y texto blanco, filas alternadas con fondo gris claro y efecto hover que resalte la fila. Añade botones pequeños para editar y eliminar con colores verde y rojo respectivamente.”

- **Update:**

  Primer prompt: “Escribe JSP que reciba un parámetro id, conecte a MySQL y obtenga los datos del estudiante con ese ID usando consulta preparada. Precarga estos datos en un formulario para editar.”

  Segundo prompt: “Crea un formulario limpio y centrado, con fondo blanco, bordes redondeados, campos grandes y botón verde con efecto hover para actualizar los datos.”

- **Delete:**

  Primer prompt: “Genera código JSP que reciba un id, se conecte a MySQL y elimine el registro correspondiente de la tabla estudiantes con una consulta preparada, luego redirige a la lista de estudiantes.”

  Segundo prompt: “Crea una página o modal con mensaje centrado, fondo semitransparente y botones para confirmar o cancelar la eliminación, usando colores rojos para confirmar y gris para cancelar.”

Proceso de subida a GitHub:

1. Se abre la terminal (Git bash) y se navega a la carpeta del proyecto con el comando “cd”.

2. Se inicializa un repositorio local de Git.

3. Se crea una nueva rama llamada “feature”

4. Se renombra esa rama y se le pone “main” y se establece como la rama principal.

5. Se crea un archivo gitignore para ignorar los archivos innecesarios.

6. Se agregan todos los archivos necesarios con “git add”

7. Se verifican todos los archivos listos para commit con git status.

8. Se hace el commit con un mensaje descriptivo, esto guarda el estado actual del proyecto en el repositorio.

9. Se crea un repositorio remoto nuevo en GitHub y se vincula con el local.

10. Por ultimo se sube el proyecto al repositorio remoto con “git push”