

CRUD con AJAX

Ejemplo funcional

- Ver ejemplo final

Agregar ventana modal

Ver <https://getbootstrap.com/docs/5.0/components/modal/>

Cambiar el botón Alta Cliente para que este muestre el modal.

Agregar el div que contiene el modal al final del main.jsp

```
<button type="button"
  class="btn btn-success"
  data-bs-toggle="modal"
  data-bs-target="#editarClienteModal">Alta Cliente</button>
```

```
<div class="modal fade" id="editarClienteModal" tabindex="-1"
  aria-labelledby="modalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header bg-info text-white">
        <h5 class="modal-title" id="modalLabel">Datos Cliente</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal"
          aria-label="Close"></button>
      </div>
      <div class="modal-body">...</div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary"
          data-bs-dismiss="modal">Cancelar</button>
        <button type="button" class="btn btn-primary">Aceptar</button>
      </div>
    </div>
  </div>
</div>
```

Incliur la pantalla de captura

- Modificar el archivo main.jsp para incluir datosCliente.jsp en el cuerpo de nuestra ventana modal.

```
<%@include file="./datosCliente.jsp"%>
```

- Adicionalmente, eliminar los botones del modal en main.jsp para que sean los botones de datosCliente.jsp los que realicen la acción.
- Finalmente, quitar el include de menu.jsp y quitar el botón de cancelar en el archivo de datosCliente.jsp, puesto que ya no los vaoms a necesitar.

Modificar EditarClienteAction

- Quitar el método init, ya que no lo vamos a utilizar.
- Quitar el método validate, puesto que las validaciones las dejaremos sólo en el front end?

Modificar struts.xml

- Agregar un nuevo paquete e incluir allí los action AJAX
- Incluir el stack de interceptor de JSON en el nuevo paquete.

```
<package name="ajax" namespace="/" extends="json-default">
  <interceptors>
    <interceptor-stack name="defaultStack">
      <interceptor-ref name="json">
        <param name="enableSMD">true</param>
      </interceptor-ref>
    </interceptor-stack>
  </interceptors>
```

- Mover editarCliente a este package y cambiar el result type de success para que regrese JSON.

```
<action name="editarCliente"
  class="com.action.clientes.EditarClienteAction"
  method="execute">
  <result name="success" type="json"/>
    <result name="correcto" type="redirect">consultar</result>
    <result name="input">/WEB-INF/clientes/datosCliente.jsp</result>
  </action>
```

Función enviar datosCliente.jsp

- Agregar la funcionalidad de JQuery para mandar a llamar el action al hacer click en el botón de Enviar.
- Se requiere hacer uso de la función JSON.stringify para enviar correctamente el objeto cliente.

```
$("#btnEnviar").click(function() {  
    if (validator.form()){  
        var id = $('#idCliente');  
        var nombre = $('#nombre');  
        var apellidos = $('#apellidos');  
        var email = $('#email');  
        var telefono = $('#telefono');
```

```
var cliente = { "cliente" : {  
    "idCliente" : id.val() ? id.val() : null,  
    "nombre" : nombre.val(),  
    "apellidos" : apellidos.val(),  
    "email" : email.val(),  
    "telefono" : telefono.val()  
}};
```

```
var data1 = JSON.stringify(cliente);
```

- Agregar llamada por medio de ajax

```
$.ajax({  
    url : "editarCliente",  
    type : 'POST',  
    datatype : 'json',  
    contentType: 'application/json',  
    data : data1,  
    async: true,  
    success : function(response) {
```

Función enviar datosCliente.jsp

- Si la llamada de ajax es exitosa, entonces debemos agregar una nueva línea a la tabla de clientes.
- En caso de error hacemos sólo mostrarnos los mensajes en consola.

```
let linea = '<tr><td>'+clienteR.idCliente+'</td>';
linea = linea + '<td>'+clienteR.nombre+'</td>';
linea = linea + '<td>'+clienteR.apellidos+'</td>';
linea = linea + '<td>'+clienteR.email+'</td>';
linea = linea + '<td>'+clienteR.telefono+'</td>';
linea = linea + '<td><i class="bi bi-pencil-square"></i></td>';
linea = linea + '<td><i class="fas fa-trash"></i></td>';
linea = linea + '<td><i class="fas fa-calendar-plus"></i></td>';
linea = linea + '</tr>';

$("#bodyClientes").append(linea);
```

```
error : function(jqxhr, textstatus, errorthrown) {
    console.log("jqxhr:%o", jqxhr);
    console.log("textstatus:%o", textstatus);
    console.log("errorthrown:%o", errorthrown);
    $("#closeModal").click();
}
```

- Por lo pronto sólo incluimos los íconos sin ninguna acción.
- Independientemente de si es correcta o con error, hacemos click en el botón de cerrar del modal y regresamos false para que no se haga en envío del formulario.

```
$("#closeModal").click();
return false;
```

Probar el alta

- Ejecutar la aplicación y probar
- Se debe abrir la ventana modal y permitir la captura.
- Al hacer click en enviar debe de hacer la llamada al action, crear el registro en la base y actualizar la tabla.
- Qué pasa si envío el formulario vacío?

Unir las funciones de javaScript

- Dentro de la carpeta de scripts, crear un nuevo archivo js llamado adminCliente.js
- Agregar el código que tenemos en el jsp de datosCliente.js a este archivo.
- Incluirllo en el jsp de datos cliente y probar que todo sigue funcionando igual.
- Agregar el código de datosCliente.js al archivo adminCliente.js y asignar la función a una variable:

```
let validator = $('#formulario').validate({
```

- Agregar un if para comprobar que forma ha sido validada:

```
if (validator.form()){
```

- Probar nuevamente intentando enviar el formulario vacío.

Unir las funciones de javaScript

- Podemos observar que los valores utilizados en el alta anterior se quedan escritos en nuestro modal. Para evitar esto vamos a limpiar la forma seteando un valor vacío a nuestro elementos Ej. `nombre.val('')`
- Además, debemos “limpiar” la forma para que nuestro validate sepa que debe realizar las validaciones nuevamente:

```
validator.resetForm();  
validator.reset();
```

- Finalmente, vamos a utilizar Toaster, para mostrar un mensaje de confirmación cuando el alta fue exitosa

Incluir Toaster

- Para incluirlo, debemos agregar un div a nuestro main.jsp que sea el contenedor de nuestro mensaje.

```
<div aria-live="polite" aria-atomic="true" class="position-relative">
  <div class="toast-container position-absolute top-0 end-0 p-3"
    id="toastContainer">
    <div class="toast text-white" role="alert" aria-live="assertive"
      aria-atomic="true" id="toastMsg">
      <div class="d-flex">
        <div class="toast-body" id="toastBody"></div>
        <button type="button"
          class="btn-close btn-close-white me-2 m-auto"
          data-bs-dismiss="toast" aria-label="Close"></button>
      </div>
    </div>
  </div>
</div>
```

- Toaster viene incluido en bootstrap si utilizamos el bootstrap.bundle.min.js.
- Aquí, en el div con id="toastBody" es donde vamos a mostrar nuestro mensaje. Para ello antes de limpiar la forma incluimos lo siguiente:

```
$("#toastMsg").removeClass("bg-danger").addClass("bg-success");
$("#toastBody").html('Alta Exitosa');

new bootstrap.Toast(toastMsg).show();
```

Modificar la funcionalidad de Editar Cliente

- Vamos a modificar la funcionalidad para que al seleccionar algún cliente de la lista, se edite utilizando AJAX.
- Para esto, vamos a modificar nuestro tag `<s:a>` en `main.jsp` para que en lugar de enviar los parámetros como parte del link, enviemos el id de cliente en el href:

```
<s:a href="%{idCliente}" id="editCte"><i class="bi bi-pencil-square"></i></s:a>
```

- Crear el archivo `main.js` dentro de la carpeta `scripts` en incluirlo en `main.jsp`
- Agregar la siguiente función a `main.js` para que al hacer click en el botón de `editCliente`, mande la información al modal y lo abra en pantalla.

```
$(document).ready(function() {  
    $(document).on("click", "#editCte", function(){  
        console.log('editCte');  
  
        var idCliente = $(this).attr('href');  
  
        $('#idCliente').val(idCliente);  
        $('#altaClienteModal').click();  
  
        return false;  
    });  
});
```

Modificar la funcionalidad de Editar Cliente

- Como podemos observar sólo se está enviando el id, por lo que debemos leer el resto de la información. Para facilitar el funcionamiento, vamos a crear un nuevo action que consulte el cliente por medio de id.
- Para esto, hay que crear un nuevo action en el paquete com.action.clientes, llamado ConsultaClienteAction.java que tenga una variable de cliente y en el método execute busque al cliente recibido.
- Agregar el action creado a nuestro struts.xml en el paquete de ajax para utilizarlo posteriormente.
- Dentro de main.js manda a llamar la función que acabamos de crear para para obtener los datos y enviarlos al modal.
- Adicionalmente, en el tr de cada cliente le vamos a poner un id para saber cuál línea debemos de modificar. `<tr id="${idCliente}" align="center">`

Modificar la funcionalidad de Editar Cliente

- En nuestro archivo adminCliente.js agregamos la funcionalidad para que si no es un alta, busque la línea de la tabla a editar y actualice los datos

```
let trow = $("#bodyClientes tr#" + id.val());
let linea = '<td>' + clienteR.idCliente + '</td>';
linea = linea + '<td>' + clienteR.nombre + '</td>';
linea = linea + '<td>' + clienteR.apellidos + '</td>';
linea = linea + '<td>' + clienteR.email + '</td>';
linea = linea + '<td>' + clienteR.telefono + '</td>';
linea = linea + '<td><a id="editCte" href="' + clienteR.idCliente + '"><i class="bi bi-pencil-square"></i></a></td>';
linea = linea + '<td><i class="fas fa-trash"></i></td>';
linea = linea + '<td><i class="fas fa-calendar-plus"></i></td>';

trow.html(linea);
```

- Notar que el link para editar, estamos agregando los datos de referencia requeridos:

```
'<td><a id="editCte" href="' + clienteR.idCliente + '"><i class="bi bi-pencil-square"></i></a></td>';
```

```
linea = linea +
```

- Finalmente, vamos a modificar los mensajes de toaster para que muestre un mensaje más adecuado a la acción.
- Y en nuestro archivo main.js vamos a crear una función para limpiar el modal cada que este se cierre:

```
$("#editarClienteModal").on('hidden.bs.modal', function() {
    $('#idCliente').val('').removeClass('success').addClass('normal');
    $('#nombre').val('').removeClass('success').addClass('normal');
    $('#apellidos').val('').removeClass('success').addClass('normal');
    $('#email').val('').removeClass('success').addClass('normal');
    $('#telefono').val('').removeClass('success').addClass('normal');
});
```

Modificar la funcionalidad de Baja Cliente

- El último paso es cambiar la funcionalidad de baja para utilizar AJAX.
- De igual forma vamos a cambiar el link para incluir el id del cliente en el href. Y agregamos un id="bajaCte" en nuestro tag <a>
- Cambiamos el action bajaCliente al paquete de ajax y el result type a json.
- Agregamos la función de baja a nuestro main.js para mandar a llamar al action y realizar la baja en la base de datos.
- Y en nuestro archivo main.js vamos a crear una función para limpiar el modal cada que este se cierre.
- Además, antes de hacer la llamada con ajax, vamos a pedirle al usuario que confirme que quiere realizar la baja:

```
beforeSend: function() {  
    return confirm('¿Está seguro de que quiere realizar la baja?');  
},
```

Modificar la funcionalidad de Baja Cliente

- Si todo es correcto quitamos la fila de la tabla y mostramos un mensaje de baja exitosa.

```
let trow = $("#bodyClientes tr#" + idCliente);  
trow.remove();  
  
$("#toastMsg").removeClass("bg-danger")  
                .addClass("bg-success");  
$("#toastBody").html('Baja Exitosa');  
new bootstrap.Toast(toastMsg).show();
```

-
- Si queremos, también podríamos dejar datosCliente.jsp como un simple html.

Práctica

- Realizar los ajustes necesarios para que alta de Inventario se realice utilizando AJAX.