Funciones de actualización de datos:

· **Gestion de capas de formularios** → se carga el contenido de una capa dentro de gestionClientes.html dinámicamente desde a la capa formularios del index.html. Lo convertimos a un diálogo y se le realizan varios cambios a los campos.

· **Cargar selects** → Se deshabilitan y se realiza una petición al servidor y este le responde con los datos más importantes de cada objeto que se necesite. Tras esto, se vacía, se genera y se habilita.

· **Rellenar Formulario según el Select** → Si el select principal de cada capa es seleccionado, se realiza una petición al servidor enviando la clave (id o dni) serializada y recibiendo todos los datos que pertenecen a esa clave. Se analiza la respuesta y se añaden al formulario.

· **Validacion de formularios** → Antes de que el formulario sea enviado al servidor, se realiza una validación del mismo, en el que se comprueban todos los datos. En caso de que uno o más campos sean erróneos, será notificado. Si todo está correcto, devolverá un objetoRespuesta con todo lo necesario (válido, oObjeto, sMensaje).

· **Gestion altas** → Todos campos están habilitados. Si todo es correcto, se obtiene objetoRespuesta muestra el sMensaje y enviará oObjeto al servidor para que inserte una nueva fila con los datos.

· **Gestion modificaciones** → Campos deshabilitados excepto el select principal, se obtiene objetoRespuesta, muestra el sMensaje y enviará oObjeto al servidor en formato json para que actualice la fila con los nuevos datos.

· **Gestion bajas/anulaciones** → Campos deshabilitados excepto el select principal. En ambos casos se envía al servidor la clave (id o dni) en formato json. Si se da de baja, se elimina la fila de la base de datos, pero si se anula o finaliza se actualizará un valor en su fila. Bajas: cliente, maquinaria, empleado. Anulaciones: incidencia. Finalizar: alquiler.

· **Gestión listados** → Para listar las maquinarias, se enviará un string para filtrar (listar todo, tipo eléctrica o tipo combustible). Para los demás listados, se realiza una llamada (no envía nada) al servidor. El servidor le devolverá un archivo XML con todos los datos que pertenecen al objeto. Se procesará el XML, se rellenará la tabla y será mostrada.

· **LocalStorage** → La primera vez que entre al formulario de Clientes o de Empleados, se realizará una petición al servidor y obtendrá todas las provincias y lo almacenará en el almacenamiento local del cliente (key => value). Ya no necesitará realizar más esta petición. Tras esto, se rellenará el select de provincias.

Uso de las llamadas:

- **.load** → Carga Formularios Externos
  - Estas llamadas solo están en el index.html. Si se carga correctamente el formulario, se obtiene un script para gestionar el diálogo se mantiene a la espera del usuario.
- **$.getScript** → Obtener Scripts Externos
  - La utilizo justo después de la llamada load.
- **$.get** → Petición Ajax Get Consulta al Servidor
- **$.post** → Petición Ajax Post.
- **$.ajax** → Petición Ajax Genérica
  - **type**: 'GET', datatype: 'json'
  - **type**: 'POST', datatype: 'json' →Es utilizado para el alta y la actualización de datos, el tipo de formato que envía es un objeto json, Cuando termine esta llamada se mostrara un mensaje informativo.
- **Ajax sin jQuery** → XMLHTTPRequest.
  - **oXHR** 'GET'→ realiza una petición que no envía y recibe un texto con la última id insertada en la tabla. Al ser una cadena ('00005'), necesitamos sumarle 1, lo transformamos a integer (5), le sumamos (5+1) y lo transformamos a cadena de nuevo ('00006'). Esto se cargará automáticamente en un input. También la utilizo para cargar los selects, y para traer los datos de una fila al formulario.
- Los datos que se envían siempre son serializados

Funciones:

· Funciones de inserción/borrado/modificación de datos:

- **altaCliente.js línea 25:** $.ajax({type: "POST", url: "php/altaCliente.php"…}); JSON
- **bajaMaquinaria.js línea 28:** → $.ajax({type: "POST", url: "php/bajaMaquinaria.php"…}); JSON
- **modificaEmpleado.js línea 29:** → $.ajax({type: "POST", url: "php/modificaEmpleado.php"…});

· Funciones para generación de listados:

- **listadoClientes.js línea 4:** → peticion.open("GET", "php/listadoClientes.php", true), XML
- **listadoEmpleados.js línea 4:** → peticion.open("GET", "php/listadoEmpleados.php", true), XML
- **listadoIncidencias.js línea 4:** → peticion.open("GET", "php/listadoIncidencias.php", true), XML
- **listadoAlquileres.js línea 4:** → peticion.open("GET", "php/listadoAlquileres.php", true), XML

· Funciones para generación de listados mediante formulario con filtrado de datos :

- **listadoMaquinarias.js línea 4:** → peticion.open("GET", "php/listadoMaquinarias.php", true)XML

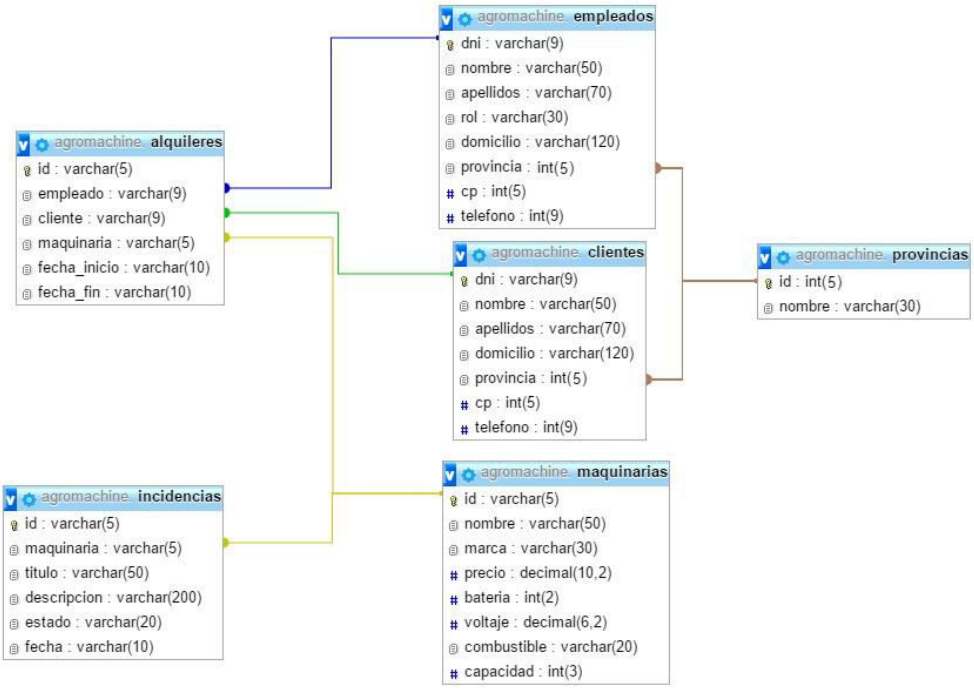
· Funciones para generación de selects, en funcionesSelect.js:

- **cargarSelectProvincia(select) línea 4:** → peticion.open("GET", "php/cargarProvincias.php", true);
- **cargarSelectCliente(select,cadena) línea 51:** → peticion.open("GET","php/selectClientes.php", true);
- **cargarSelectMaquinaria(select,cadena).js línea 120:** → peticion.open("GET", "php/selectMaquinarias.php", true);
- **cargarSelectMaquinariaDisponible (select,cadena).js línea 146:** → peticion.open("GET", "php/selectMaquinariasDisponibles.php", true);

· Funciones para cargar datos al formulario, en funcionesSelect.js:

- **cargarDatosCliente() línea 80:** → peticion.open("GET", "php/infoCliente.php?dni='seleccionado'", true);
- **cargarDatosAlquiler() línea 231:** → peticion.open("GET", "php/infoAlquiler.php?id='seleccionado'", true);

Diseño SQL:



Modelo Entidad Relación: