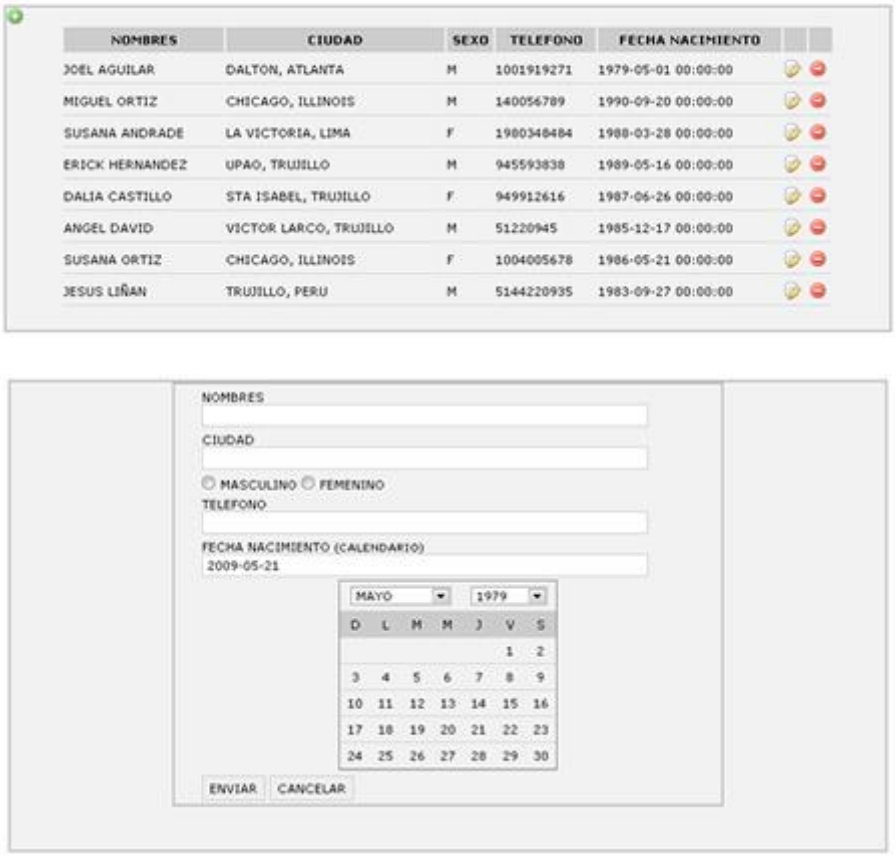
Practica 9 . jQuery, Ajax y CORS

En esta práctica se dispondrá de un código de una pequeña appweb que deberéis mejorar. La práctica se divide en varias secciones o partes con sucesivos puntos que no son excluyentes.

El siguiente ejemplo muestra como realizar una gestión completa de una tabla Mysql con una página web mediante la librería **jQuery**. El resultado se muestra a continuación.



Vamos a realizar un pequeño **mantenimiento de datos de unos clientes**, esto nos permitirá visualizar, agregar, modificar ó eliminar sus datos. Haremos uso de MySQL y PHP pues es la forma de probarlo en un servidor web.

Empezaremos con la tabla cliente, que tiene la siguiente estructura:

```
--
-- Estructura de tabla para la tabla `cliente`
--

CREATE TABLE IF NOT EXISTS `cliente` (
  `id` tinyint(7) NOT NULL auto_increment,
  `nombres` varchar(50) NOT NULL,
  `ciudad` varchar(50) NOT NULL,
  `sexo` char(1) NOT NULL,
  `telefono` varchar(10) NOT NULL,
  `fecha_nacimiento` datetime NOT NULL,
```

```
  KEY `id` (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

Deberemos crear con phpMyadmin una base de datos empresa y ejecutar la consulta anterior para generar la tabla de clientes.

Vamos a crear dos clases en PHP: 1) para la conexión con el servidor y 2) una clase cliente, con cinco métodos básicos.

conexion.class.php

```
<?php
class DBManager{
        var $conect;

        var $BaseDatos;
        var $Servidor;
        var $Usuario;
        var $Clave;
        function DBManager(){
                $this->BaseDatos = "base-datos";
                $this->Servidor = "ip";
                $this->Usuario = "user";
                $this->Clave = "pwd";
        }

         function conectar() {
                if(!($con=@mysql_connect($this->Servidor,$this-
>Usuario,$this->Clave))){
                        echo"<h1> [:(] Error al conectar a la base de
datos</h1>";
                        exit();
                }
                if (!@mysql_select_db($this->BaseDatos,$con)){
                        echo "<h1> [:(] Error al seleccionar la base
de datos</h1>";
                        exit();
                }
                $this->conect=$con;
                return true;
        }
}
?>
```

cliente.class.php

```
<?php
include_once("conexion.class.php");

class Cliente{
 //constructor
        var $con;
        function Cliente(){
                $this->con=new DBManager;
        }

        function insertar($campos){
                if($this->con->conectar()==true){
                        return mysql_query("INSERT INTO cliente
```

```
(nombres, ciudad, sexo, telefono, fecha_nacimiento) VALUES
('".$campos[0]."',
'".$campos[1]."','".$campos[2]."','".$campos[3]."','".$campos[4]."')")
;
                }
        }

        function actualizar($campos,$id){
                if($this->con->conectar()==true){
                        return mysql_query("UPDATE cliente SET nombres
= '".$campos[0]."', ciudad = '".$campos[1]."', sexo =
'".$campos[2]."', telefono = '".$campos[3]."', fecha_nacimiento =
'".$campos[4]."' WHERE id = ".$id);
                }
        }

        function mostrar_cliente($id){
                if($this->con->conectar()==true){
                        return mysql_query("SELECT * FROM cliente
WHERE id=".$id);
                }
        }

        function mostrar_clientes(){
                if($this->con->conectar()==true){
                        return mysql_query("SELECT * FROM cliente
ORDER BY id DESC");
                }
        }

        function eliminar($id){
                if($this->con->conectar()==true){
                        return mysql_query("DELETE FROM cliente WHERE
id=".$id);
                }
        }
}
?>
```

Los procesos básicos de un mantenimiento de datos están separados en varios archivos: *consulta.php*, *nuevo.php*, *actualizar.php*, *eliminar.php*.

consulta.php

Este archivo se encarga de *mostrar todos los clientes* de la tabla, contienen un enlace a *nuevo.php* para agregar un nuevo cliente, además contiene dos enlaces para *modificar* y *eliminar* que enlazarán a los archivos *actualizar.php* y *eliminar.php* respectivamente, con sus variables *GET* (Ej. *actualizar.php?id=40*). También colocamos algo de JavaScript para estos dos enlaces.

```
<?php
require('clases/cliente.class.php');
$objCliente=new Cliente;
$consulta=$objCliente->mostrar_clientes();
?>
<script type="text/javascript">
$(document).ready(function(){
        // mostrar formulario de actualizar datos
        $("table tr .modi a").click(function(){
```

```
                $('#tabla').hide();
                $("#formulario").show();
                $.ajax({
                        url: this.href,
                        type: "GET",
                        success: function(datos){
                                $("#formulario").html(datos);
                        }
                });
                return false;
        });

        // llamar a formulario nuevo
        $("#nuevo a").click(function(){
                $("#formulario").show();
                $("#tabla").hide();
                $.ajax({
                        type: "GET",
                        url: 'nuevo.php',
                        success: function(datos){
                                $("#formulario").html(datos);
                        }
                });
                return false;
        });
});

</script>
<span id="nuevo"><a href="nuevo.php"><img src="img/add.png"
alt="Agregar dato" /></a></span>
        <table>
                <tr>
                        <th>Nombres</th>
                <th>Ciudad</th>
                <th>Sexo</th>
                <th>Telefono</th>
            <th>Fecha Nacimiento</th>
            <th></th>
            <th></th>
        </tr>
<?php
if($consulta) {
        while( $cliente = mysql_fetch_array($consulta) ){
        ?>

                  <tr id="fila-<?php echo $cliente['id'] ?>">
                          <td><?php echo $cliente['nombres'] ?></td>
                          <td><?php echo $cliente['ciudad'] ?></td>
                          <td><?php echo $cliente['sexo'] ?></td>
                          <td><?php echo $cliente['telefono'] ?></td>
                          <td><?php echo $cliente['fecha_nacimiento']
?></td>
                          <td><span class="modi"><a
href="actualizar.php?id=<?php echo $cliente['id'] ?>"><img
src="img/database_edit.png" title="Editar" alt="Editar"
/></a></span></td>
                          <td><span class="dele"><a
onClick="EliminarDato(<?php echo $cliente['id'] ?>); return false"
href="eliminar.php?id=<?php echo $cliente['id'] ?>"><img
src="img/delete.png" title="Eliminar" alt="Eliminar"
/></a></span></td>
```

```
                    </tr>
        <?php
        }
}
?>
    </table>
```

nuevo.php

Este archivo muestra un formulario donde escribiremos los datos del nuevo cliente, pero también actúa cómo un proceso para agregar el cliente a la base de datos. Es por eso que primero evalúa si existen definidas alguna variable POST. Si es así, llama a la *clase cliente* y hace uso del método *insertar*. En la parte del formulario, en el evento *onsubmit*, llamamos a la función en JavaScript: *GrabarDatos()*.

```
<?php
require('functions.php');
if(isset($_POST['submit'])){
        require('clases/cliente.class.php');

        $nombres = htmlspecialchars(trim($_POST['nombres']));
        $ciudad = htmlspecialchars(trim($_POST['ciudad']));
        $sexo = htmlspecialchars(trim($_POST['alternativas']));
        $telefono = htmlspecialchars(trim($_POST['telefono']));
        $fecha_nacimiento =
htmlspecialchars(trim($_POST['fecha_nacimiento']));

        $objCliente=new Cliente;
        if ( $objCliente-
>insertar(array($nombres,$ciudad,$sexo,$telefono,$fecha_nacimiento))
== true){
                echo 'Datos guardados';
        }else{
                echo 'Se produjo un error. Intente nuevamente';
        }
}else{
?>
<form id="frmClienteNuevo" name="frmClienteNuevo" method="post"
action="nuevo.php" onsubmit="GrabarDatos(); return false">
  <p><label>Nombres<br />
  <input class="text" type="text" name="nombres" id="nombres" />
  </label>
  </p>
  <p>
    <label>Ciudad<br />
    <input class="text" type="text" name="ciudad" id="ciudad" />
    </label>
  </p>
  <p>
    <label>
    <input type="radio" name="alternativas" id="masculino" value="M"
/>
    Masculino</label>
    <label>
    <input type="radio" name="alternativas" id="femenino" value="F" />
    Femenino</label>
  </p>
  <p>
    <label>Telefono<br />
```

```
    <input class="text" type="text" name="telefono" id="telefono" />
    </label>
  </p>
  <p>
    <label>Fecha Nacimiento <a onclick="show_calendar()"
style="cursor: pointer;">
<small>(calendario)</small>
</a><br />
    <input readonly="readonly" class="text" type="text"
name="fecha_nacimiento" id="fecha_nacimiento" value="<?php echo
date("Y-m-j")?>" />
    <div id="calendario" style="display:none;"><?php calendar_html()
?></div>
    </label>
  </p>
  <p>
    <input type="submit" name="submit" id="button" value="Enviar" />
    <label></label>
    <input type="button" class="cancelar" name="cancelar"
id="cancelar" value="Cancelar" onclick="Cancelar()" />
  </p>
</form>
<?php
}
?>
```

actualizar.php

Este archivo, similar a *nuevo.php*, muestra un formulario con los datos del cliente seleccionado, pero también actúa como un proceso para modificar los datos del cliente. Es por eso que primero evalúa si existen definidas alguna variable POST. Si es así llama a la *clase cliente* y hace uso del método *actualizar*. En la parte del formulario, en el evento *onsubmit*, llamamos a la función en JavaScript: *ActualizarDatos()*.

```
<?php
require('functions.php');
if(isset($_POST['submit'])){
        require('clases/cliente.class.php');
        $objCliente=new Cliente;

        $cliente_id = htmlspecialchars(trim($_POST['cliente_id']));
        $nombres = htmlspecialchars(trim($_POST['nombres']));
        $ciudad = htmlspecialchars(trim($_POST['ciudad']));
        $sexo = htmlspecialchars(trim($_POST['alternativas']));
        $telefono = htmlspecialchars(trim($_POST['telefono']));
        $fecha_nacimiento =
htmlspecialchars(trim($_POST['fecha_nacimiento']));

        if ( $objCliente-
>actualizar(array($nombres,$ciudad,$sexo,$telefono,$fecha_nacimiento),
$cliente_id) == true){
                echo 'Datos guardados';
        }else{
                echo 'Se produjo un error. Intente nuevamente';
        }
}else{
        if(isset($_GET['id'])){

                require('clases/cliente.class.php');
```

```
                $objCliente=new Cliente;
                $consulta = $objCliente->mostrar_cliente($_GET['id']);
                $cliente = mysql_fetch_array($consulta);
        ?>
        <form id="frmClienteActualizar" name="frmClienteActualizar"
method="post" action="actualizar.php" onsubmit="ActualizarDatos();
return false">
        <input type="hidden" name="cliente_id" id="cliente_id"
value="<?php echo $cliente['id']?>" />
        <p>
          <label>Nombres<br />
          <input class="text" type="text" name="nombres" id="nombres"
value="<?php echo $cliente['nombres']?>" />
          </label>
      </p>
          <p>
                <label>Ciudad<br />
                <input class="text" type="text" name="ciudad"
id="ciudad" value="<?php echo $cliente['ciudad']?>" />
                </label>
          </p>
          <p>
                <label>
                <input type="radio" name="alternativas" id="masculino"
value="M" <?php if($cliente['sexo']=="M") echo "checked=\"checked\""?>
/>
                Masculino</label>
                <label>
                <input type="radio" name="alternativas" id="femenino"
value="F" <?php if($cliente['sexo']=="F") echo "checked=\"checked\""?>
/>
                Femenino</label>
          </p>
          <p>
                <label>Telefono<br />
                <input class="text" type="text" name="telefono"
id="telefono" value="<?php echo $cliente['telefono']?>" />
                </label>
          </p>
          <p>
        <label>Fecha Nacimiento <a onclick="show_calendar()"
style="cursor: pointer;"><small>(calendario)</small></a><br />
        <input readonly="readonly" class="text" type="text"
name="fecha_nacimiento" id="fecha_nacimiento" value="<?php echo
$cliente['fecha_nacimiento'] ?>" />
        <div id="calendario" style="display:none;"><?php
calendar_html() ?></div>
        </label>
          </p>
          <p>
                <input type="submit" name="submit" id="button"
value="Enviar" />
                <label></label>
                <input type="button" name="cancelar" id="cancelar"
value="Cancelar" onclick="Cancelar()" />
          </p>
        </form>
        <?php
        }
}
?>
```

eliminar.php

Este archivo, como su nombre lo indica, se encarga de borrar los datos del cliente especificado.

```
<?php
require('clases/cliente.class.php');

$cliente_id=$_GET['id'];
$objCliente=new Cliente;
if( $objCliente->eliminar($cliente_id) == true){
        echo "Registro eliminado correctamente";
}else{
        echo "Ocurrió un error";
}
?>
```

Ahora veamos las funciones en JavaScript relacionadas con los procesos PHP que acabamos de explicar de manera general.

jQuery.functions.js

Las cuatro funciones *ActualizarDatos(), ConsultaDatos(), EliminaDato()* y *GrabarDatos(),* hacen llamadas AJAX, mediante la función ***$.ajax({...})*** que nos proporciona jQuery. Cada función hace una llamada a los procesos en PHP respectivamente.

```
        function ActualizarDatos(){
                var cliente_id = $('#cliente_id').attr('value');
                var nombres = $('#nombres').attr('value');
                var ciudad = $('#ciudad').attr('value');
                var alternativas =
$("input[@name='alternativas']:checked").attr("value");
                var telefono = $("#telefono").attr("value");
                var fecha_nacimiento =
$("#fecha_nacimiento").attr("value");

                $.ajax({
                        url: 'actualizar.php',
                        type: "POST",
                        data:
"submit=&nombres="+nombres+"&ciudad="+ciudad+"&alternativas="+alternat
ivas+"&telefono="+telefono+"&fecha_nacimiento="+fecha_nacimiento+"&cli
ente_id="+cliente_id,
                        success: function(datos){
                                alert(datos);
                                ConsultaDatos();
                                $("#formulario").hide();
                                $("#tabla").show();
                        }
                });
                return false;
        }

        function ConsultaDatos(){
                $.ajax({
                        url: 'consulta.php',
                        cache: false,
```

```
                        type: "GET",
                        success: function(datos){
                                $("#tabla").html(datos);
                        }
                });
        }

        function EliminarDato(cliente_id){
                var msg = confirm("Desea eliminar este dato?")
                if ( msg ) {
                        $.ajax({
                                url: 'eliminar.php',
                                type: "GET",
                                data: "id="+cliente_id,
                                success: function(datos){
                                        alert(datos);
                                      $("#fila-"+cliente_id).remove();
                                }
                        });
                }
                return false;
        }

        function GrabarDatos(){
                var nombres = $('#nombres').attr('value');
                var ciudad = $('#ciudad').attr('value');
                var alternativas =
$("input[@name='alternativas']:checked").attr("value");
                var telefono = $("#telefono").attr("value");
                var fecha_nacimiento =
$("#fecha_nacimiento").attr("value");

                $.ajax({
                        url: 'nuevo.php',
                        type: "POST",
                        data:
"submit=&nombres="+nombres+"&ciudad="+ciudad+"&alternativas="+alternat
ivas+"&telefono="+telefono+"&fecha_nacimiento="+fecha_nacimiento,
                        success: function(datos){
                                ConsultaDatos();
                                alert(datos);
                                $("#formulario").hide();
                                $("#tabla").show();
                        }
                });
                return false;
        }
        //esta funcion es para el boton cancelar del form
        function Cancelar(){
                $("#formulario").hide();
                $("#tabla").show();
                return false;
        }
```

index.php

Finalmente todo esto aparecerá en nuestro archivo principal *index.php*, que tendrá la siguiente estructura:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Mantenimiento de Clientes</title>
<script src="js/jquery-1.3.1.min.js" type="text/javascript"></script>
<script src="js/jquery.functions.js" type="text/javascript"></script>
<link type="text/css" rel="stylesheet" href="css/estilo.css" />
</head>

<body>
<div id="contenedor">
        <div id="formulario" style="display:none;">
        </div>
            <div id="tabla">
            <?php include('consulta.php') ?>
            </div>
</div>
<span style="text-
align:center;width:300px;margin:auto;display:block;margin-
top:20px;">DAW 2013</span>
</body>
</html>
```

Desarrollo:

La práctica pretende poner en práctica la mayoría de conceptos aprendidos durante el curso y ofrece a los alumnos avanzados la posibilidad de realizar ampliaciones a modo de opcionales.

La práctica debe crear una aplicación web para la gestión de una tabla de clientes mediante la aproximación a un "Rich Client". Para ello, al alumno se le dota de los ficheros iniciales de la API que realizarán todas las tareas sobre la BD. Esta carpeta es la única que tendrá la obligación de estar en el servidor Web ya que lo que la aplicación hará es una conexión con esta API que gestiona la BD.

La API está hecha con phpDAO. Si el alumno debe hacer algún cambio en la BD debe modificar la API utilizando phpDAO.

Módulo DIW

El proyecto debe de cumplir:

Todas las interfaces deberán ser realizadas con el framework css en Boostrap 4.

Se deben utilizar iconografías en formato glyph con Fontawesome.

Las interfaces deben ser responsive y por tanto adecuarse a los dispositivos móviles.

Se deben utilizar plantillas de cliente con handlebars para mostrar la tabla de los clientes (ListaClientes) y los datos de un cliente (Cliente) nuevo/modificado. Las fechas deben mostrarse correctamente por lo que deberéis crear un helper que transformar la fecha en formato 2015-12-02 00:00:00 a 02/12/2015.

Para mostrar los datos de un cliente, al modificarlos o crear uno nuevo, se debe de usar un modal de bootstrap o algo similar.

Se debe incluir un calendario en la interfaz cliente. Existen varios calendarios, en jqueryUi hay uno, pero se puede incluir el que se considere oportuno. Si se utiliza, controles HTML5 para el formulario, se debe incluir un polifyll para que, en caso de no estar soportado por el navegador, nos muestre el calendario alternativo.

Módulo DWC

El proyecto debe de cumplir:

La aplicación debe ser un single Page App con entry point index.htm.

Utilizar la librería jquery (última versión estable de la web oficial).

Utilizar el patrón modular para crear los objetos de lista de clientes y cliente (en este caso es recomendable usar un además un Factory pattern).

Utilizar el patrón MVC para la lista de usuarios, utilizando el objeto event visto en clase y crear los ficheros View asociados.

La lista cliente será la encargada de hacer las peticiones Ajax para leer/escribir en la base de datos para Crear(C)/leer(R)/modificar(U)/eliminar(D) (–CRUD-) los clientes.

Las peticiones Ajax se deben realizar con el método $.post de Jquery. Los parámetros deben ser enviados como data en formato json.

El objeto cliente debe sincronizarse con la vista del cliente y cada vez que cambie un cliente, debe ejecutar un método que lo cambie y "avisar" a la lista de clientes que ejecutará un método que debe actualizar la lista de clientes y por tanto la base de datos. De esta forma siempre está sincronizado el modelo con la base de datos.

OPCIONAL (**Una vez terminada la práctica. No se valorará ningún opcional si la práctica está incompleta**):

Incluir la dirección y la provincia y fecha de alta en cada cliente. Modificar las interfaces para que se gestionen también los nuevos campos. (0.5 puntos en cada módulo en la 2ª evaluación)

Incluir el api de google de geocoding/places para guardar la ubicación del domicilio del cliente. Añadir también un mapa en la ficha del cliente donde se vea una chincheta en la ubicación de su domicilio. (1 punto en cada módulo en la 2ª evaluación)

Modificar la App para que los clientes puedan tener varios vehículos (infinitos) que pueden ser de varios tipos: coche, moto, camión, furgoneta. El vehículo debe tener matrícula, fecha de fabricación, marca y modelo. Hacer todas las modificaciones necesarias para que se pueda gestionar y mostrar estos vehículos con la App. (1.5 puntos en cada módulo en la 2ª evaluación)

Incluir un sistema de subidas de ficheros para cada cliente y vehículo. Con este sistema se gestiona la posible subida de documentos pdf de cada usuario relacionado al vehículo. Pueden ser contratos, facturas etc… La interfaz debe de mostrar la lista de los ficheros ya subidos y por cada uno de ellos la opción de eliminar y descargar. (1,5 en el módulo de DWC y 1 punto en DIW).

Crear un acceso para los clientes con formulario de registro que le permita al usuario poder ver su ficha, añadir una foto de perfil y subir/descargar documentos. El usuario no puede eliminar ningún documento. (1 punto en cada módulo en la 2ª evaluación)

Incluir un menú de navegación Bootstrap para la interfaz del cliente y del administrador. (0.5 puntos en DIW en la 2ª evaluación)

Empaquetar todo el proyecto con webpack, creando una versión de producción y otra de desarrollo (solo de la parte obligatoria) (0.5 punto en DWC en la 2ª evaluación)

Utilizar la API de google maps para añadir una opción al usuario que le permita reportar un siniestro con un vehículo de los que tiene. El usuario debe de poder localizar la dirección y posicionar su vehículo en el mapa, así como el vehículo contrario (con polígonos), incluir la

dirección de movimiento de cada vehículo y situación primer impacto en el mapa. Solo hay que hacer la interfaz y enviar la petición al guardar pero no se debe guardar nada en la BD. (1 punto en DWC en la 2ª evaluación)

NOTA: Todos los opcionales deben de seguir los mismos criterios que se han establecido para la parte obligatoria. Por ejemplo, para realizar la vista de los vehículos se deberá de crear con el patrón modular una clase vehículos, una lista de vehículos y sus correspondientes vistas. Si algún opcional no cumple con dichos criterios no será considerado como correcto.