# 

[EL FRONTEND](#_2c7se0fe64to) 3

[Página principal:](#_otxf0mcxlpls) 3

[Página de servicios ofrecidos:](#_5vtu0jnpc22q) 4

[Página de contacto:](#_1ow8d6icp58d) 4

[Login al backend:](#_eob182gufi4o) 5

[Alta y baja de vehículos:](#_3ai4dtpox6nj) 5

[Alta y baja de usuarios:](#_ha9jdz28gf5i) 6

[Mostrar coches:](#_1wryv67adkt2) 6

[Mostrar clientes:](#_ghvsl7s7tnps) 6

[EL BACKEND](#_i6jypxbi80p3) 7

[Login y conexión a la base de datos](#_igju0rtv5dza) 7

[Interfaz de conexión a la base de datos para las funciones de la aplicación BD.php](#_g7z6z81kggnv) 8

[Alta cliente php](#_a50upw2z0yj0) 9

[Baja cliente php](#_9se4yqzanwfu) 9

[Mostrar clientes](#_crbth664jc3b) 10

[Alta y baja vehiculos](#_3i6chuli6n0y) 10

[**LA BASE DE DATOS**](#_8yjacxkeuiq2) **11**

[Creación del esquema de base de datos:](#_sb9eauxqkwta) 11

[Creación de la tabla adm](#_seayb0cxqogo) 12

[Creación de tabla cliente](#_x40beeovg5xy) 12

[Creación tabla coches](#_u9bl6dlh8t82) 12

[Comprobaciones adicionales](#_97jynfn1mau7) 13

[Ningún campo de un formulario puede quedar vacío excepto las observaciones](#_o87f3hjrny6f) 13

[Comprobación de campos de texto email y teléfono](#_3950s70t0r9) 13

[**BIBLIOGRAFÍA:**](#_59mnq0rfrwek) **13**

[**Créditos:**](#_br23d2lk1w91) **14**

## EL FRONTEND

Para la realización del frontend se ha decidido basarse en boostrap por completo

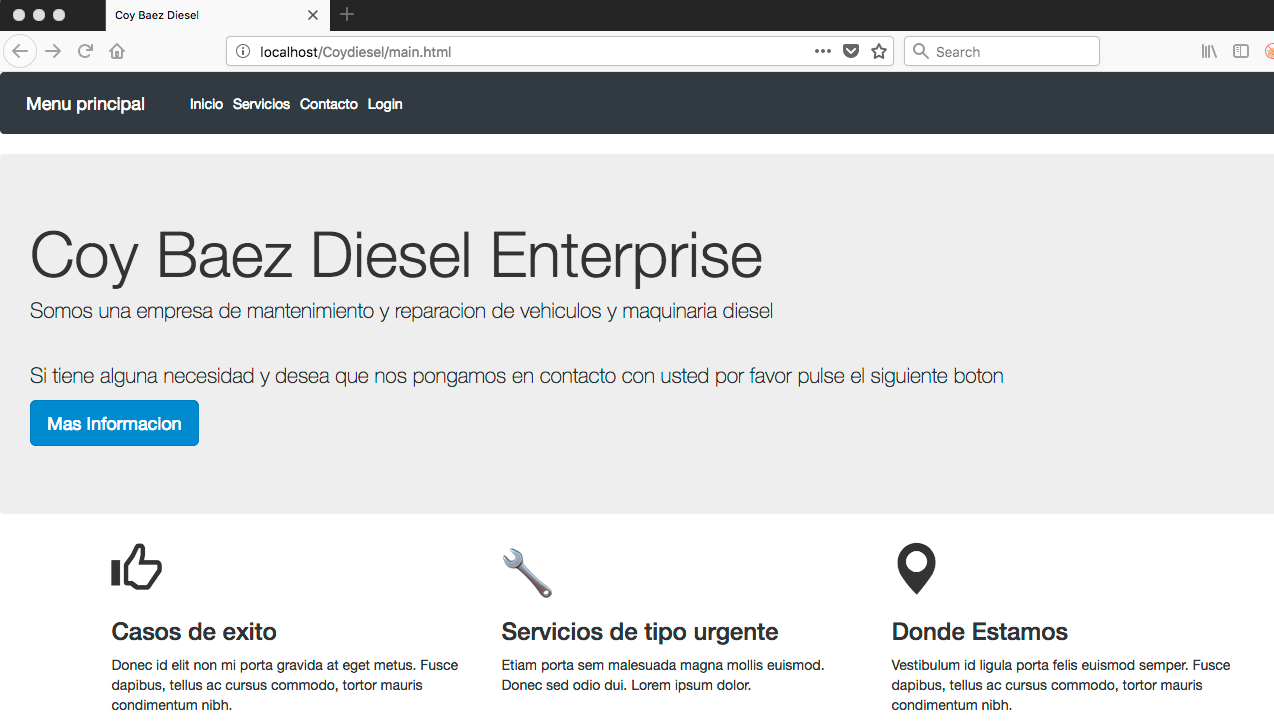
para el diseño visual de cara al cliente. También se ha determinado abstraer completamente la parte visual y de presentación de la aplicación, es decir, todas las funciones de cara al cliente están implementadas en los archivos HTML como se observa a continuación. La decisión de implementar boostrap está basada en la facilidad que tiene este para ser implementado además de hacer y garantizar la compatibilidad con la mayoría de navegadores web, dispositivos multimedia y un diseño “responsive”.

Los diseños se han tomado de la misma documentación en boostrap y los ejemplos presentados en:

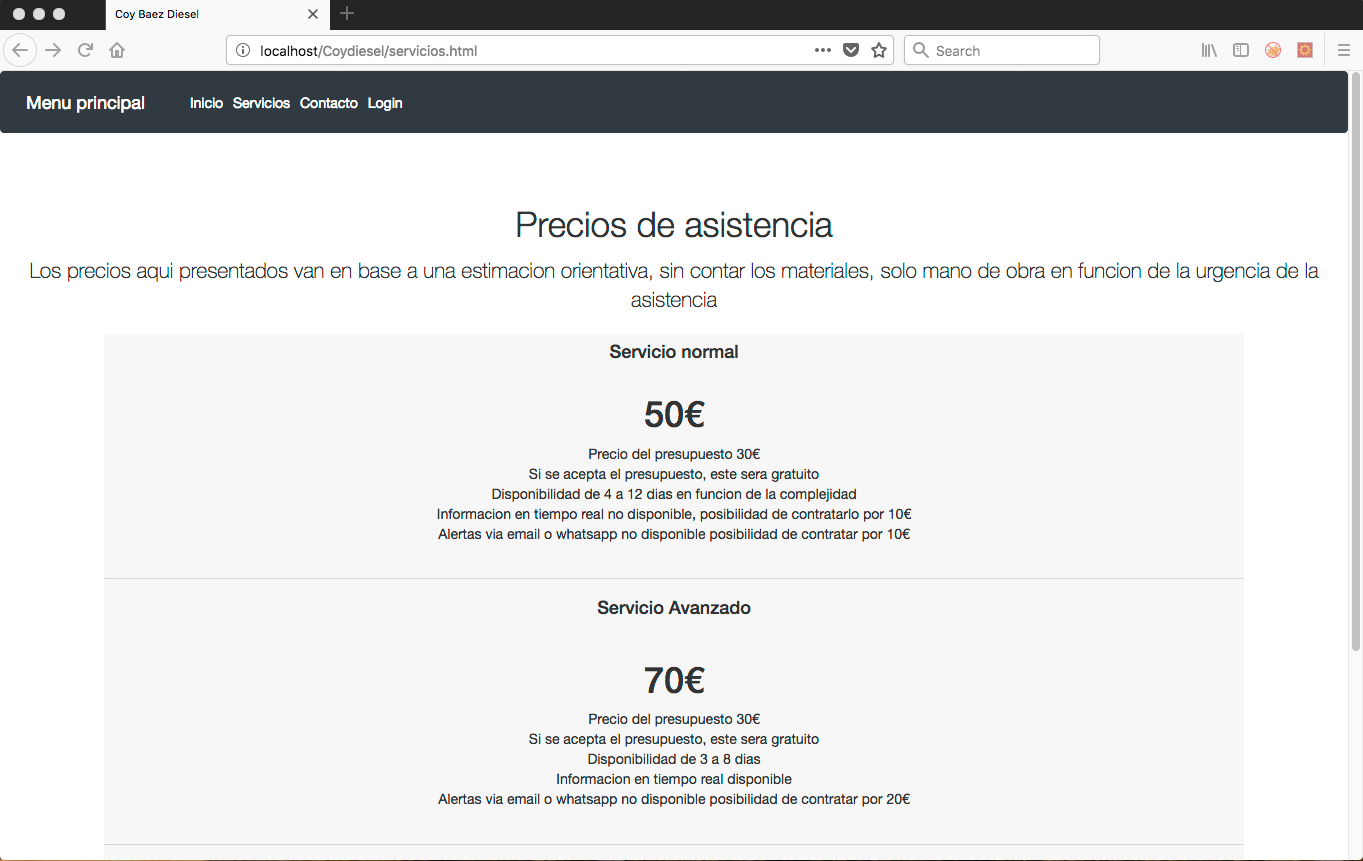
<https://getbootstrap.com/docs/4.0/examples/>

Capturas de la adaptación y la aplicación desarrollada:

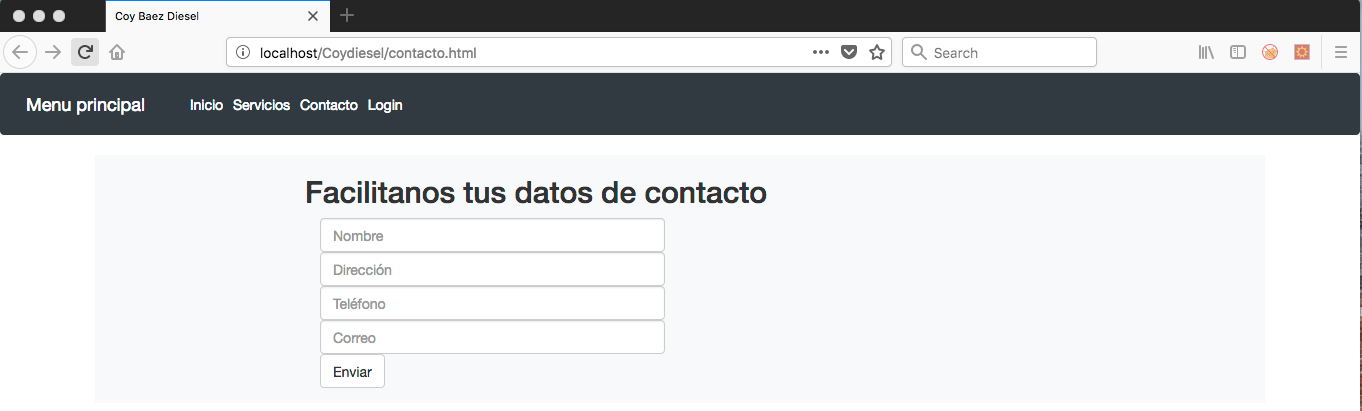
### Página principal:



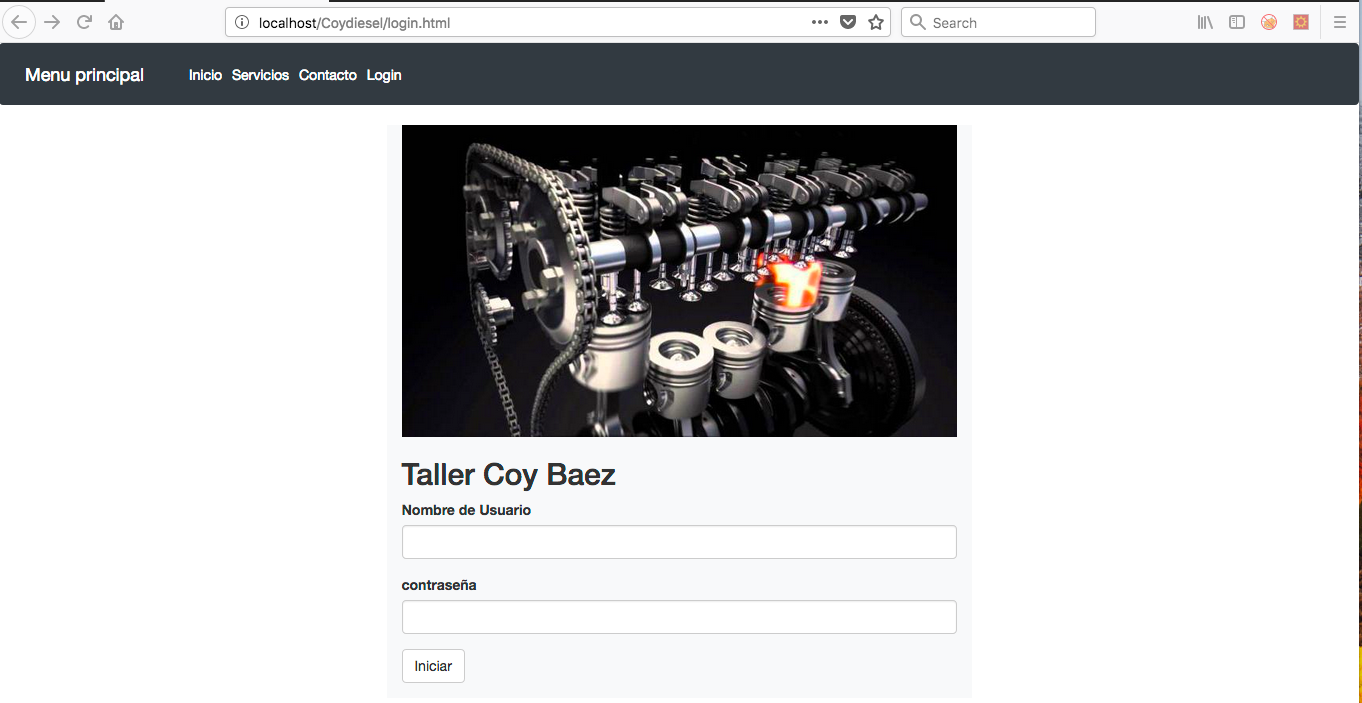
### Página de servicios ofrecidos:



### Página de contacto:

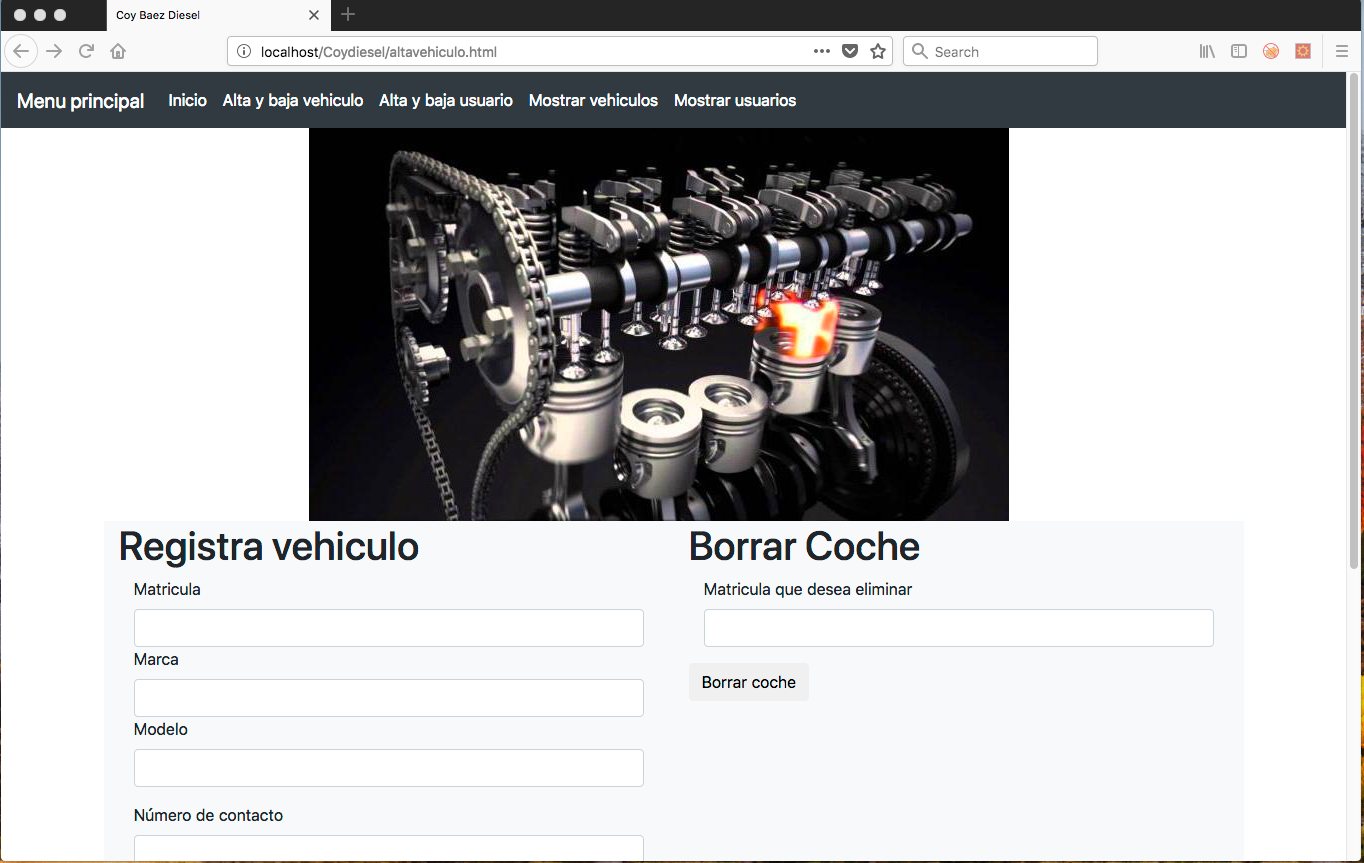


### Login al backend:

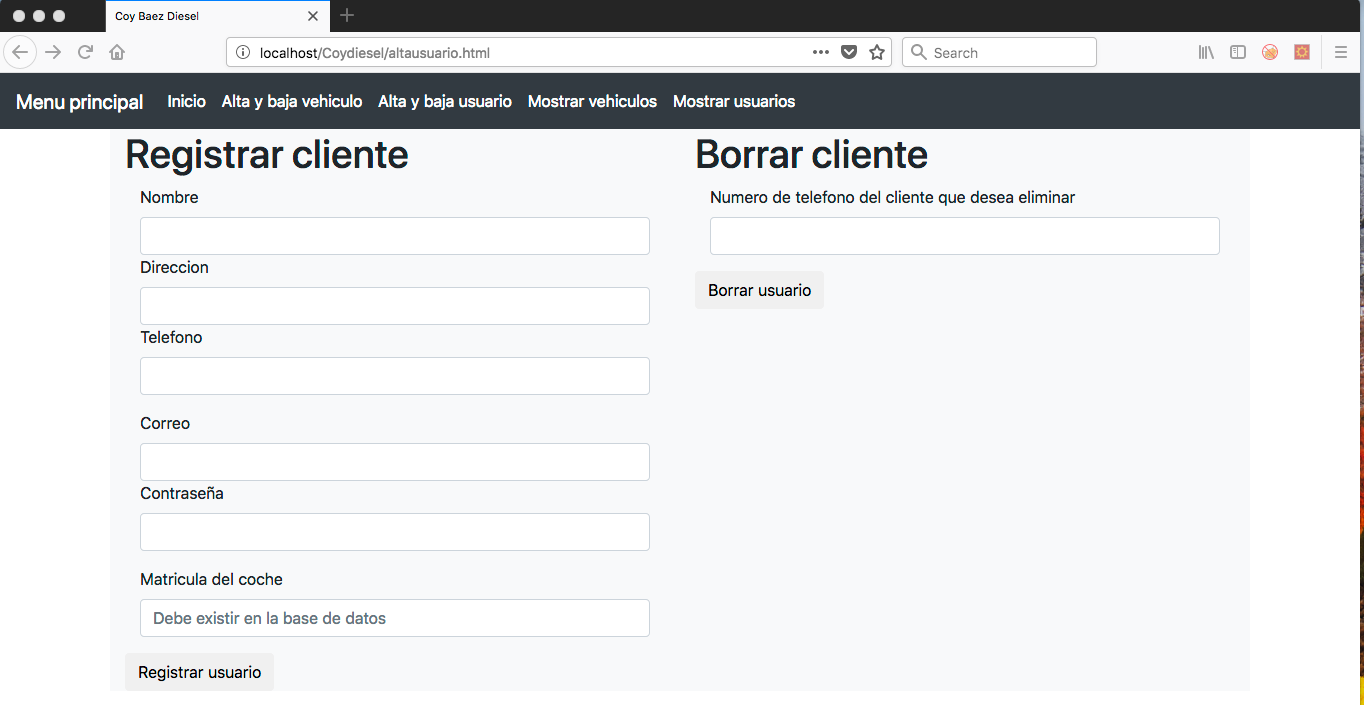


Como se ha comentado anteriormente, el objetivo a conseguir con la aplicación es abstraer totalmente la parte visual, el front del cliente del back como tal, por tanto la parte visual del backend también reposa sobre html

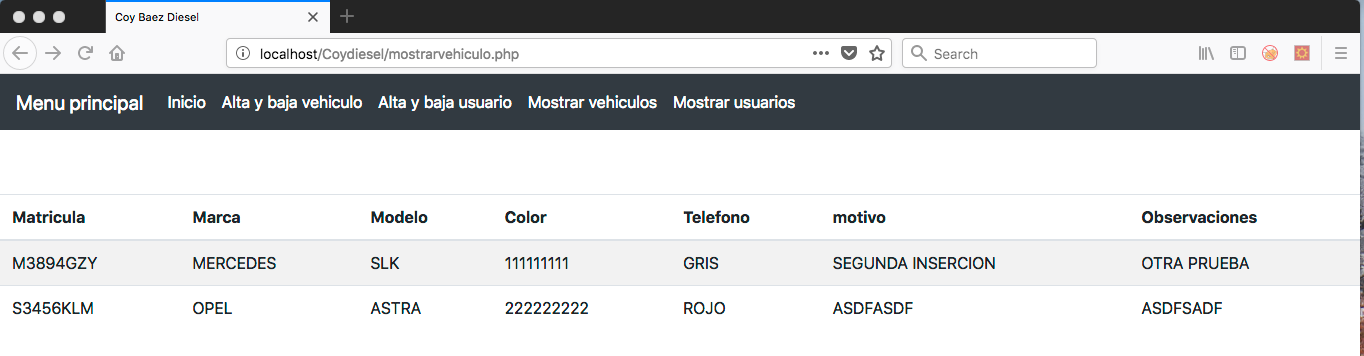
### Alta y baja de vehículos:



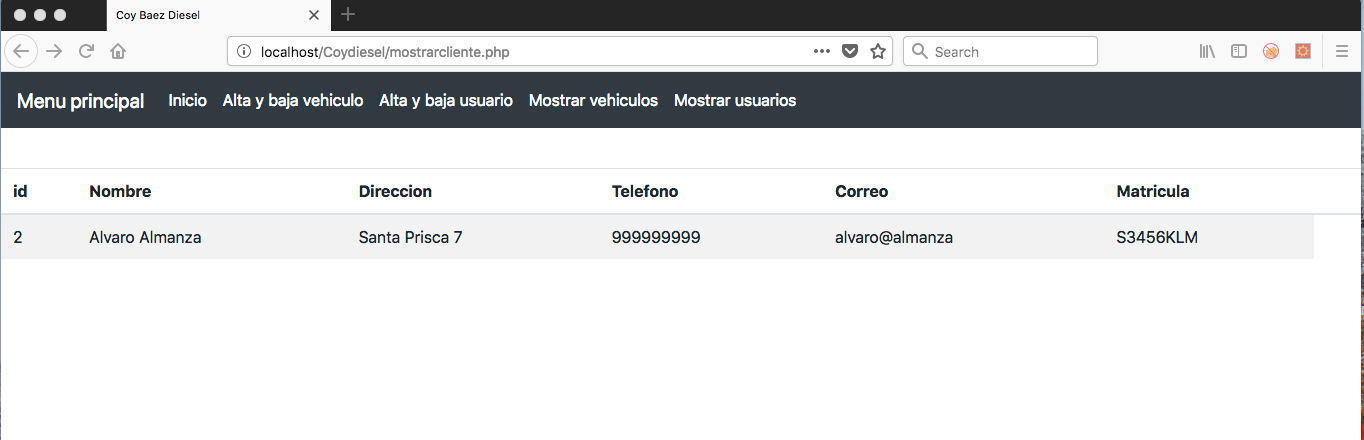
### Alta y baja de usuarios:



### Mostrar coches:



### Mostrar clientes:



Actualmente se omite todo el código html el cual estará presente en el archivo de entrega y se puede comprobar el mismo directamente.

## EL BACKEND

Este está implementado en código php, el cual interactúa con una base de datos mysql donde extrae y guarda datos. La base de datos se da por creada desde un principio (mirese apartado de base de datos a continuación).

El primer apartado es la funcionalidad del login, el cual lo que hace es recoger los datos del formulario html para después consultarlos en la base de datos.

### Login y conexión a la base de datos

<?php

//$login=false;

$con=false;

$datos=$\_POST["usuario"];

$link;

$host="localhost";

$puerto = "3306";

$usuario = "ui1";

$pass = "almanza";

$baseDeDatos ="administracion";

//$tabla = "personas";

ComprobacionInicial();

//FUNCION PARA COMPROBAR LA CONEXION A LA BASE DE DATOS Y LA CONEXION DE LOS USUARIOS

function ComprobacionInicial()

{

global $datos,$usuario,$pass, $link;

$seguir=Conectarse($usuario,$pass);

if($seguir){

ComprobarUsuario($datos["nombre"],$datos["contrasenia"]);

}

}

//FUNCION PARA COMPROBAR LA CONEXION A LA BASE DE DATOS

function Conectarse($aux1,$aux2)

{

global $host, $puerto, $con, $link, $baseDeDatos;

$link= mysqli\_connect ($host.":".$puerto, $aux1, $aux2, $baseDeDatos);

if ($link){

/\*if (!mysqli\_select\_db($baseDeDatos)) {

include 'errorconexion.html';

} else {

$\_SESSION["conexion"]=$link;

$con=true;

}\*/

$con=true;

//print\_r($link);

}else{

include 'errorconexion.html';

$con=false;

}

return $con;

}

//FUNCION PARA COMPORBAR LA EXISTENCIA DEL USUARIO EN LA TABLA ADM

function ComprobarUsuario($aux1,$aux2)

{

global $link;

$\_SESSION["con"]=$link;

$query= "SELECT usuario,pass FROM administracion.adm WHERE usuario='".$aux1."'";

$result = mysqli\_query($link,$query);

$row = mysqli\_fetch\_array(mysqli\_query($link,$query));

if (!$result) {die('Consulta no válida: ' . mysql\_error());}

if($aux1==$row["usuario"] && $aux2==$row["pass"])

{

include 'dashboard.html';

mysqli\_close($link);

}else

{

include 'errorusuario.html';

mysqli\_close($link);

}

}

?>

A grandes rasgos, el código anterior lo que hace es realizar dos comprobaciones por medio de dos funciones, una realiza un “test” sobre el socket de la base de datos para verificar que el programa puede conectarse a la misma, en caso de ser cierto, pasa a la siguiente función de comprobación de datos de autenticación a la aplicación para finalmente mostrar la página inicial de backend.

En caso de encontrar algún tipo de error de conexión o autenticación, la aplicación muestra un diálogo modal indicando el error.

### Interfaz de conexión a la base de datos para las funciones de la aplicación BD.php

<?php

$con=false;

$link;

$host="localhost";

$puerto = "3306";

$usuario = "ui1";

$pass = "almanza";

$baseDeDatos ="administracion";

function Conectarse()

{

global $host, $puerto, $con, $link, $baseDeDatos, $usuario, $pass;

$link= mysqli\_connect ($host.":".$puerto, $usuario, $pass, $baseDeDatos);

if (!$link){

include 'errorconexion.html';

} else{

return $link;

}

}

?>

Esta interfaz permite abstraer la conexión de cada uno de los elementos a la BD cuando se pretende realizar alguna acción (Insercion, eliminacion, etc..) lo unico que se debe hacer en las demás tareas es realizar una llamada a la función implementada anteriormente y está devolverá un objeto el cual permitirá realizar una “query”.

### Alta cliente php

<?php

include 'bd.php';

$datos=$\_POST['usuario'];

$con= Conectarse();

$query= "INSERT INTO cliente(nombre,direccion,telefono,correo,pass,matricula) VALUES ('".$datos["nombre"]."','".$datos['direccion']."','".$datos['telefono']."','".$datos['correo']."','".$datos['contrasenia']."','".$datos['matricula']."');";

//print\_r($query);

$result = mysqli\_query($con,$query);

if($result) {

echo "<strong>Se ingresaron los registros con éxito</strong>. <br/>";

} else {

echo "No se ingresaron los registros. <br/>";

}

?>

Esta función incluye como parámetro la interfaz de conexión “bd.php” para obtener la conexión a la base de datos, recoge los datos del formulario php para fabricar la “query” a ejecutar de la base de datos y almacenar los mismos.

### Baja cliente php

<?php

include 'bd.php';

$datos=$\_POST['baja'];

$con= Conectarse();

$query= "DELETE FROM cliente WHERE telefono='".$datos["telefono"]."'";

//print\_r($query);

$result = mysqli\_query($con,$query);

if($result) {

echo "<strong>Se ha eliminado el usuario con éxito</strong>. <br/>";

} else {

echo "Error en la eliminacion. <br/>";

}

?>

Este código repite la constante de comportamiento anterior, hace una llamada a la función de conexión de base de datos recoge los datos del formulario html y fabrica la query para ser ejecutada en la instancia de base de datos.

### Mostrar clientes

<?php

include 'bd.php';

$con= Conectarse();

$query= "SELECT \* FROM cliente;";

//print\_r($query);

$result = mysqli\_query($con,$query);

while($row = mysqli\_fetch\_array($result)) { printf("<tr><td>%s </td><td>%s </td><td>%s </td><td>%s </td><td>%s </td></td><td>%s </td></tr><br/>",

$row["id"],$row["nombre"],$row["direccion"],$row["telefono"],$row["correo"],$row["matricula"]); }

?>

La funcion aqui presentada lo unico que hace es obtener un objeto de conexión a la base de datos, ejecutar una consulta a toda la base de datos de clientes y presentarlos en una tabla.

### Alta y baja vehiculos

Estas funciones siguen la misma filosofía anterior, obtener objeto de conexión a la base de datos creación de la consulta a ejecutar y lanzar la misma contra la base de datos con la información deseada.

No se incluye el código pues se puede observar el el fichero adjunto.

# LA BASE DE DATOS

Para la implementación de la base de datos se ha tomado las siguientes decisiones:

1. Creación de una tabla de datos común para gestionar los usuarios que pueden acceder al backend
2. Creación de una tabla para la gestión de clientes
3. Creación de una tabla para la gestion de vehiculos.

El único requisito que se ha tenido en cuenta para la creación de esta base de datos es crear una llave foránea desde el campo de “matrícula” de la tabla de cliente a la clave primaria de la tabla “coches”, por tanto, se fuerza que:

1. Debe existir un coche o “matrícula” dada de alta en la base de datos para poder realizar el alta del cliente
2. Por cada usuario dado de alta, existiría si o si un coche asociado
3. Se puede implementar el “borrado en cascada”, por tanto, en caso de eliminar el usuario se puede eliminar el coche, sin embargo, estimo que no es una práctica recomendada pues aunque se elimine el cliente, es bastante acertado mantener la información relativa a los coches.

### Creación del esquema de base de datos:

CREATE DATABASE `administracion` /\*!40100 DEFAULT CHARACTER SET utf8 COLLATE utf8\_spanish\_ci \*/;

Para crear esta base de datos se ha tenido en cuenta la modalidad de caracteres para que la misma sea capaz de gestionar caracteres propios del castellano.

### Creación de la tabla adm

CREATE TABLE `adm` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`usuario` varchar(45) COLLATE utf8\_spanish\_ci NOT NULL,

`pass` varchar(45) COLLATE utf8\_spanish\_ci NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8 COLLATE=utf8\_spanish\_ci;

### Creación de tabla cliente

CREATE TABLE `cliente` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`nombre` varchar(100) COLLATE utf8\_spanish\_ci NOT NULL,

`direccion` varchar(45) COLLATE utf8\_spanish\_ci NOT NULL,

`telefono` varchar(45) COLLATE utf8\_spanish\_ci NOT NULL,

`correo` varchar(100) COLLATE utf8\_spanish\_ci NOT NULL,

`pass` varchar(45) COLLATE utf8\_spanish\_ci NOT NULL,

`matricula` varchar(10) COLLATE utf8\_spanish\_ci DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `ref\_coche\_idx` (`matricula`),

CONSTRAINT `ref\_coche` FOREIGN KEY (`matricula`) REFERENCES `coches` (`matricula`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8 COLLATE=utf8\_spanish\_ci;

### Creación tabla coches

CREATE TABLE `coches` (

`matricula` varchar(10) COLLATE utf8\_spanish\_ci NOT NULL,

`marca` varchar(45) COLLATE utf8\_spanish\_ci NOT NULL,

`modelo` varchar(45) COLLATE utf8\_spanish\_ci NOT NULL,

`color` varchar(45) COLLATE utf8\_spanish\_ci NOT NULL,

`telefono` varchar(45) COLLATE utf8\_spanish\_ci NOT NULL,

`motivo` varchar(500) COLLATE utf8\_spanish\_ci DEFAULT NULL,

`observaciones` varchar(1000) COLLATE utf8\_spanish\_ci DEFAULT NULL,

PRIMARY KEY (`matricula`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_spanish\_ci;

## Comprobaciones adicionales

### Ningún campo de un formulario puede quedar vacío excepto las observaciones

Para realizar esta tarea, se ha decidido utilizar la función de las etiquetas en html pues esta ya tiene implementada la tarea tan solo indicando en la etiqueta input él “required”.

### Comprobación de campos de texto email y teléfono

Llegados a este punto se pudo observar que, el haber implementado boostrap suponía un pequeño inconveniente para realizar estas comprobaciones por medio de JavaScript, pues se intentó modificar o sobrescribir el script del cdn, sin embargo, es una limitación pues no funcion en un primer instante.

Dado el anterior problema, se procedió a buscar una solución sobre boostrap y se encontró una solución adicional basada en bootstrap la cual añade esta funcionalidad también basada en JavaScrtip:

<https://bootstrap-validate.js.org/>

En el código del alta del usuario (altausuario.html) se puede observar:

<script>  
 bootstrapValidate('#telefono', 'min:9:debe introducir al menos 9 digitos');  
 bootstrapValidate('#telefono', 'max:9:debe introducir un maximo de 9 digitos');  
 bootstrapValidate('#email', 'email:Introduzca una direccion de correo valida');  
 </script>

# BIBLIOGRAFÍA:

1. Documentación PHP
   1. <http://php.net/manual/en/>
2. Documentación Boostrap
   1. <https://getbootstrap.com/docs/4.0/getting-started/introduction/>
3. Documentacion mysql
   1. <https://dev.mysql.com/doc/refman/5.6/en/sql-syntax-data-manipulation.html>
4. Documentación boostrap validate Rules
   1. <https://bootstrap-validate.js.org/rules.html>

# Créditos:

Este trabajo ha podido convertirse en realidad gracias a:

1. Asignaturas:
   1. Desarrollo de aplicaciones Web, Universidad Internacional Isabel Primera de Castilla. profesor: Francisco Javier Almeida.
   2. Administración de bases de datos, Universidad Internacional Isabel Primera de Castilla. profesor: José Antonio Marcos