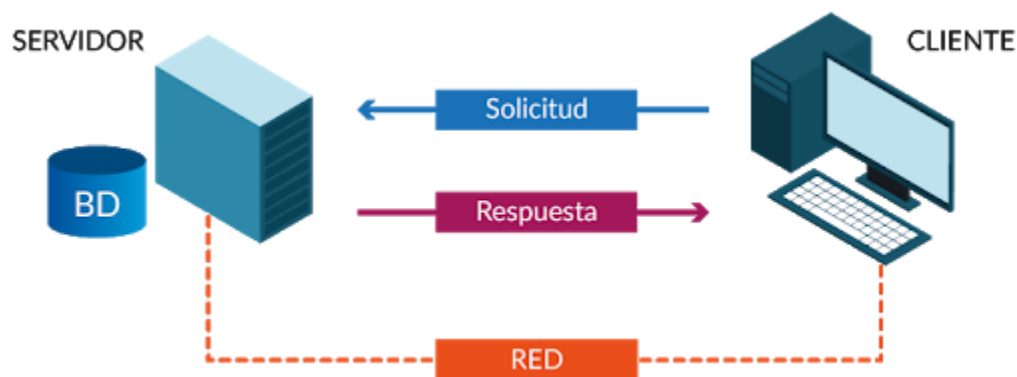


## Arquitectura Cliente – Servidor

Gracias a la evolución de internet, esta trajo consigo un cambio revolucionario en el ámbito de las tecnologías, interconectando a todo el mundo. Siguiendo una arquitectura y estructura específica para la comunicación digital. Esta es la popular arquitectura Cliente-Servidor, que resulta ser un modelo informático, donde se da a conocer la manera en que se comunican los dispositivos por medio de internet, explicando concisamente cuando un cliente emite una petición a un servidor, y este a su vez le responde a lo pedido. Esta arquitectura también se denomina estructura de computación de red, porque cada solicitud o petición y sus servicios asociados se distribuyen a través de una red.



En la ilustración anterior podemos visualizar el total funcionamiento de la arquitectura Cliente-Servidor gráficamente. Para entender mas detalladamente explicáramos cada una de estas partes.

**Cliente:** Este es el encargado de realizar peticiones o solicitudes por medio de internet o una red en general.

**Solicitud:** Es la petición realizada por el cliente, la cual puede ser un recurso digital cualquiera. Esta generalmente se realiza mediante el protocolo de red HTTP.

**Servidor:** Este se asocia generalmente con un equipo de compute llamado servidor que consta con unas características elevadas con respecto a los ordenadores normales, capaz de proveer información o dar respuesta a múltiples “Clientes”, según

determinada petición. También, este puede tener una base de datos incrustada en él o por aparte.

**Respuesta:** Son los datos que le entrega el Servidor a una determinada solicitud que realizó un determinado cliente por medio de la red.

Un ejemplo muy sencillo podría ser, cuando nosotros intentamos solicitar un sitio web ([www.unipamplona.edu.co](http://www.unipamplona.edu.co), esta sería la petición), por medio de un navegador de internet (Google Chrome, Mozilla Firefox, Safari, etc. Este sería el cliente). Esta petición la recibe internet, y ella es la encargada de direccionarla al Servidor asociado (si existe, claro está), donde posteriormente el Servidor la recibe y procesa dicha solicitud mediante su lógica, para determinar cual debe ser la respuesta que este debe entregar. Para este caso, su respuesta debería ser un archivo HTML, el cual el Cliente (navegador de internet) finalizará el proceso, renderizando en la vista esa respuesta (el archivo HTML).

### Implementando Arquitectura en un Sitio Web

Para poder implementar esta arquitectura, primero aclararemos algunos conceptos básicos sobre los sitios web.

La construcción de un sitio web requiere algunos conocimientos previos, como lenguajes de programación, lenguajes de marcado o etiquetas y gestores de bases de datos. Estos conocimientos requeridos se especifican para ciertas partes del sitio web que se vaya a construir.

Para la parte de la vista específicamente se utilizan los lenguajes de marcado y programación.

Lenguajes de Marcado	Lenguajes de Programación
HTML CSS	JavaScript

Cuando hablamos de la parte lógica o la lógica de negocio, es aquella que debe estar en el servidor, para esta únicamente se utilizan lenguajes de programación como:

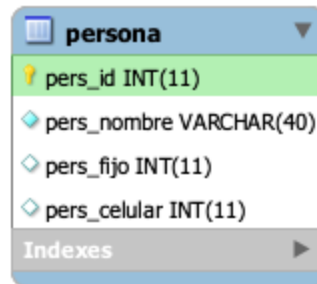
- ✓ PHP
- ✓ Java
- ✓ Python
- ✓ Ruby
- ✓ Go
- ✓ Node

Y por último cuando necesitamos almacenar todos nuestros datos que se generen u obtengamos mediante nuestro sitio web se deben utilizar gestores de bases de datos.

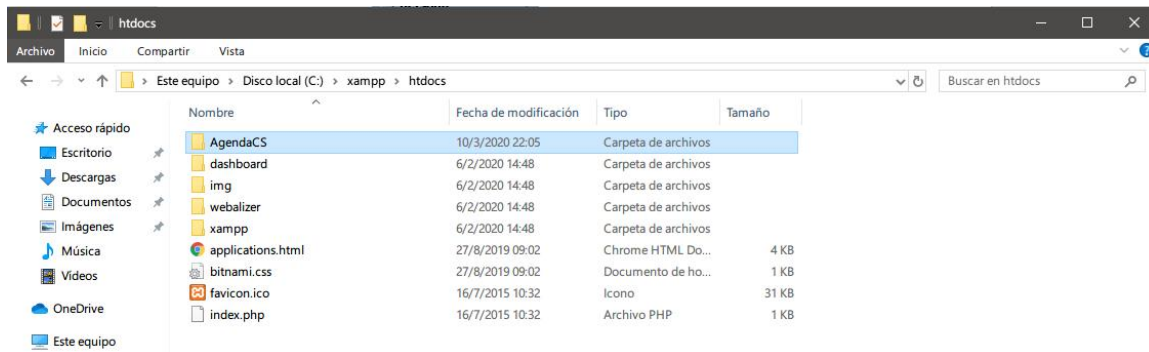
SQL	NoSQL
<ul style="list-style-type: none"><li>✓ Oracle</li><li>✓ SQL Server</li><li>✓ PostgreSQL</li><li>✓ MySQL</li><li>✓ MariaDB</li></ul>	<ul style="list-style-type: none"><li>✓ MongoDB</li><li>✓ Redis</li><li>✓ Cassandra</li><li>✓ CouchDB</li></ul>

Para esta práctica o implementación de la arquitectura Cliente-Servidor, utilizaremos los lenguajes para el cliente mencionados anteriormente, para la parte del servidor utilizaremos el lenguaje de programación PHP y para almacenar los datos, usaremos el gestor de base de datos MySQL. Se puede minimizar la complejidad de instalación de cada una de estas herramientas, instalando únicamente el software XAMPP que ya trae las herramientas que utilizaremos.

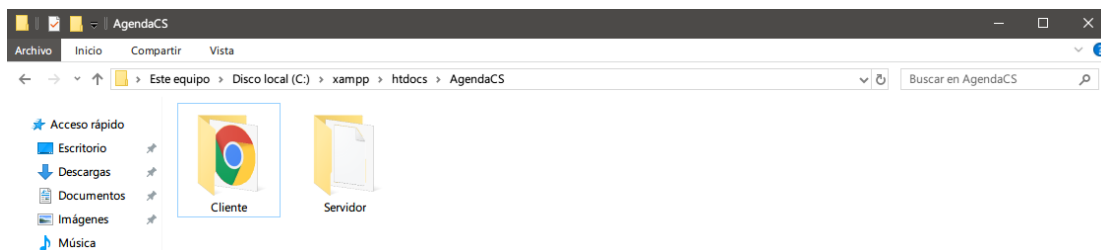
Realizaremos un mini proyecto, que consistirá en una agenda personal. Es decir crearemos un sitio web para registrar números de contacto personales. Para comenzar, empezaremos realizando el modelo relacional el cual nos dará a entender la lógica del software, para este caso realizamos dicho modelo utilizando la herramienta de ORACLE llamada MySQL Workbench en su versión 8.0.



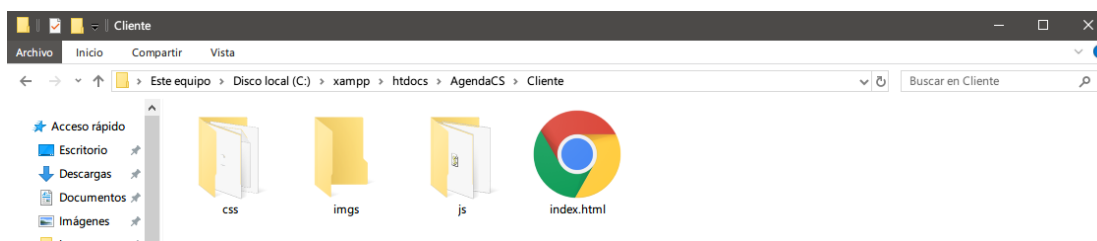
Luego, continuaremos construyendo la arquitectura la cual nos compete que viene siendo Cliente-Servidor, para ello creamos la carpeta de nuestro proyecto dentro de la carpeta “htdocs”, que se encuentra a su vez dentro de la carpeta principal de “xampp”.



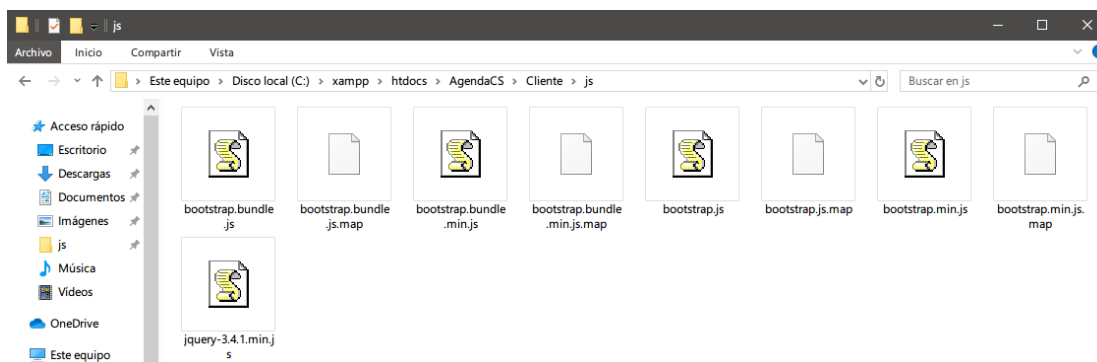
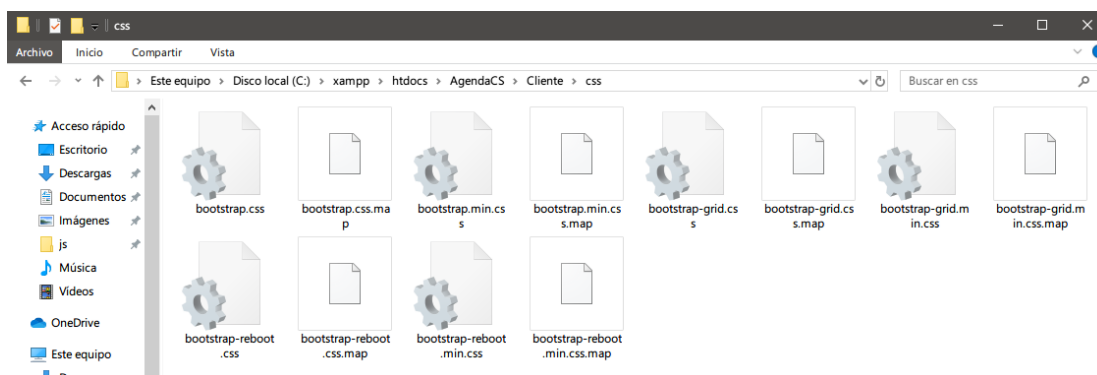
Dentro de ella crearemos la estructura necesaria, para construir nuestra aplicación web de tal manera, que logremos implementar la arquitectura Cliente-Servidor.



En la carpeta cliente, trabajaremos todo lo que tenga que ver con los lenguajes de la vista, que es lo que el usuario final o el cliente terminará visualizando por medio de un navegador para nuestro caso que es una aplicación web.

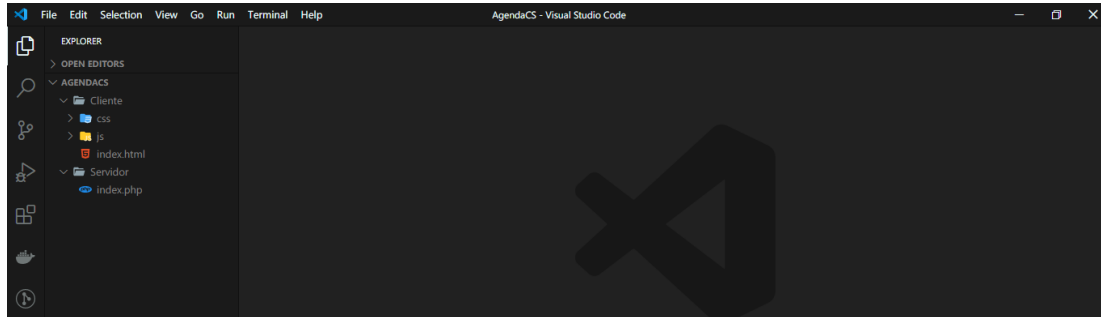


Para la maquetación o construcción de la interfaz gráfica de nuestro sitio web, utilizaremos una librería llamada Bootstrap la cual nos ayudará a agilizar este proceso mucho más. Esta librería podemos descargarla desde <https://getbootstrap.com/docs/4.4/getting-started/download>, el cual nos compartirá archivos de CSS como también de JavaScript.





Procederemos a abrir nuestro proyecto con el editor de texto de su preferencia, para este caso utilizaremos “Visual Studio Code”, que ha sido últimamente el más utilizado por la comunidad de desarrollo de software.



Luego, empezaremos construyendo la interfaz de usuario UI con el lenguaje de maquetado HTML, utilizando la librería Bootstrap como comentamos anteriormente.



Universidad de Pamplona  
Pamplona - Norte de Santander - Colombia  
Tels: (7) 5685303 - 5685304 - 5685305 - Fax: 5682750 - [www.unipamplona.edu.co](http://www.unipamplona.edu.co)

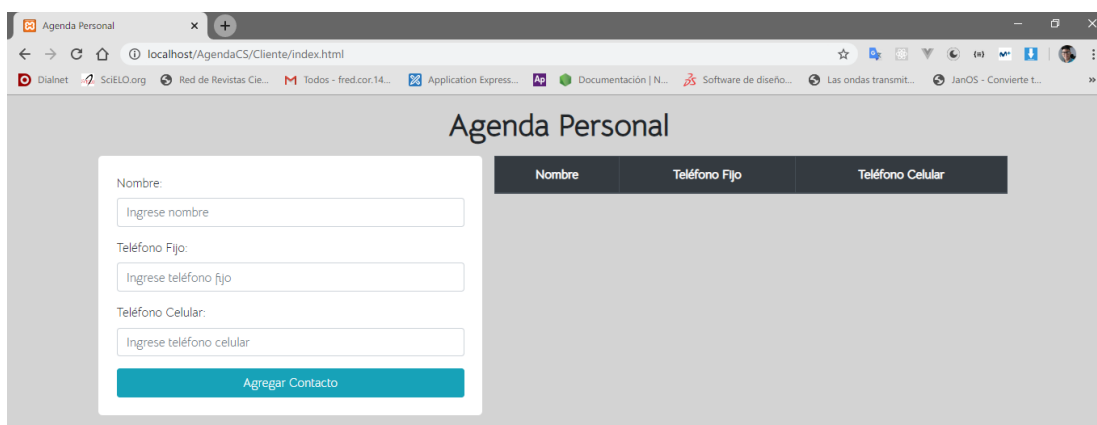
```
<!DOCTYPE html>
<html Lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Agenda Personal</title>
  <link href="https://fonts.googleapis.com/css?family=Bellota+Text&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link rel="stylesheet" href="css/index.css">
</head>
<body>
  <main class="container mt-3">
    <h1 class="text-center mb-3"><b>Agenda Personal</b></h1>
    <section class="row">
      <div class="col-sm-5 cajaContacto p-4">
        <form name="frmContacto" method="POST" action="../Servidor/index.php">
          <div class="form-group">
            <label for="txtNombre">Nombre: </label>
            <input type="text" name="txtNombre" id="txtNombre" class="form-control" placeholder="Ingrese nombre">
          </div>
          <div class="form-group">
            <label for="txtNombre">Teléfono Fijo: </label>
            <input type="text" name="txtTelFijo" id="txtTelFijo" class="form-control" placeholder="Ingrese teléfono fijo">
          </div>
          <div class="form-group">
            <label for="txtNombre">Teléfono Celular: </label>
            <input type="text" name="txtTelCel" id="txtTelCel" class="form-control" placeholder="Ingrese teléfono celular">
          </div>
          <input type="submit" class="btn btn-info btn-block" value="Agregar Contacto">
        </form>
      </div>
      <div class="col-sm-7">
        <div class="table-responsive">
          <table class="table table-bordered table-hover">
            <thead class="thead-dark text-center">
              <tr>
                <th>Nombre</th>
                <th>Teléfono Fijo</th>
                <th>Teléfono Celular</th>
              </tr>
            </thead>
            <tbody>
              <tr>
                <td></td>
                <td></td>
                <td></td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </section>
  </main>
  <script src="js/jquery-3.4.1.min.js"></script>
  <script src="js/bootstrap.min.js"></script>
</body>
</html>
```

Una interfaz sencilla, que nos permitirá realizar la implementación de nuestra arquitectura de interés. También se agregaron unos estilos propios a través del lenguaje CSS.



```
body {  
    font-family: 'Bellota Text', cursive;  
    background-color: lightgray;  
}  
.cajaContacto {  
    background-color: white;  
    border-radius: 5px;  
}
```

Con el código escrito anteriormente, logramos construir la siguiente UI.



Es momento de darle funcionalidad con el lenguaje de programación JavaScript para lograr validar los campos requeridos por el formulario y su respectivo envío al servidor para ser tratados. Podemos ayudarnos usando una librería llamada “Toastr” que nos permitirá lanzar mensajes de alerta cuando no sea válido algún dato que intentemos enviar al servidor. Podemos obtener el código de esta librería mediante el siguiente enlace, <https://github.com/CodeSeven/toastr>.

```
<link href="https://fonts.googleapis.com/css?family=Bellota+Text&display=swap" rel="stylesheet"  
>  
<link rel="stylesheet" href="css/bootstrap.min.css">  
<link rel="stylesheet" href="css/toastr.min.css">  
<link rel="stylesheet" href="css/index.css">
```



```
<script src="js/jquery-3.4.1.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/toastr.min.js"></script>
<script src="js/validaciones.js"></script>
<script src="js/index.js"></script>
```

Este sería el código resultante después de realizar las importaciones tanto de los estilos css que conlleva la librería “Toast” como también los archivos js, tomando en cuenta los archivos de validaciones.js y nuestro index.js que serán los encargados de validar y comunicarse con el servidor respectivamente.

A continuación se muestra, como tratamos de conectar el formulario de contactos al servidor mediante JavaScript, enfocandonos primeramente en la validacion de dicho formulario.

```
toastr.options.preventDuplicates = true

function registrarContacto() {
  let inpNombre = document.getElementById("txtNombre")
  let inpTel = document.getElementById("txtTelFijo")
  let inpTel2 = document.getElementById("txtTelCel")
  if(validarNombre(inpNombre, 1, 40) && validarTelefonos(inpTel, inpTel2, 6, 15)) {
    alert("enviando al server")
  }
}
```

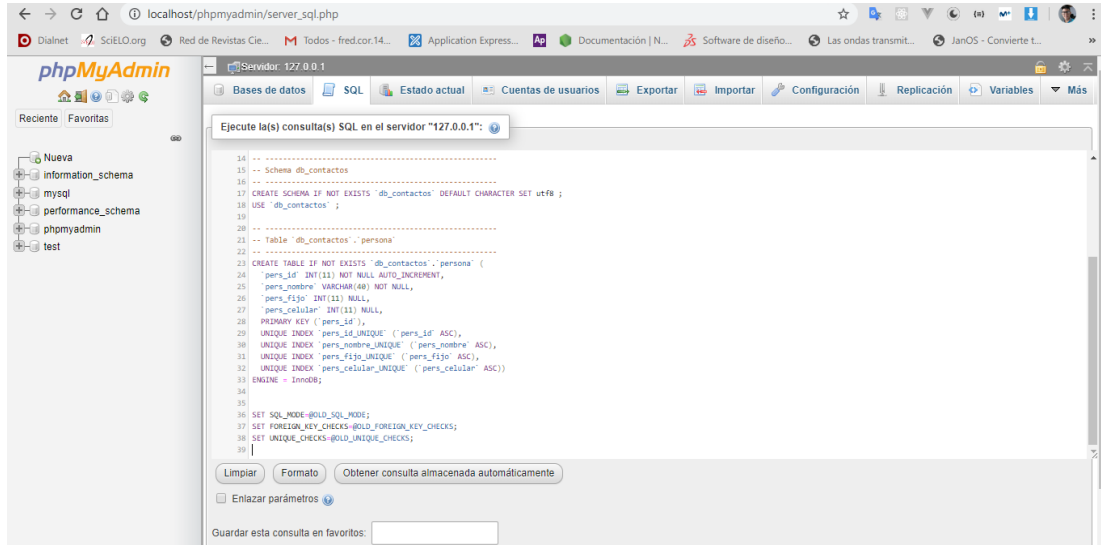
Ahora debemos preocuparnos por crear y construir la logica de las funciones de validaciones que utilizamos en nuestra funcion de registrarContacto.

```
function validarNombre(input, minLength, maxLength) {
  let nombre = input.value, valid = true
  if(nombre == "") {
    input.focus()
    toastr.warning("Debe digitar un nombre de contacto", "Proceso detenido", {timeout: 2000})
    valid = false
  } else if(nombre.length < minLength) {
    input.focus()
    toastr.warning("La cantidad de caracteres mínimos permitidos para el nombre son ${minLength}", "Proceso detenido", {timeout: 2000})
    valid = false
  } else if(nombre.length > maxLength) {
    input.focus()
    toastr.warning("La cantidad de caracteres máximos permitidos para el nombre son ${maxLength}", "Proceso detenido", {timeout: 2000})
    valid = false
  }
  return valid
}

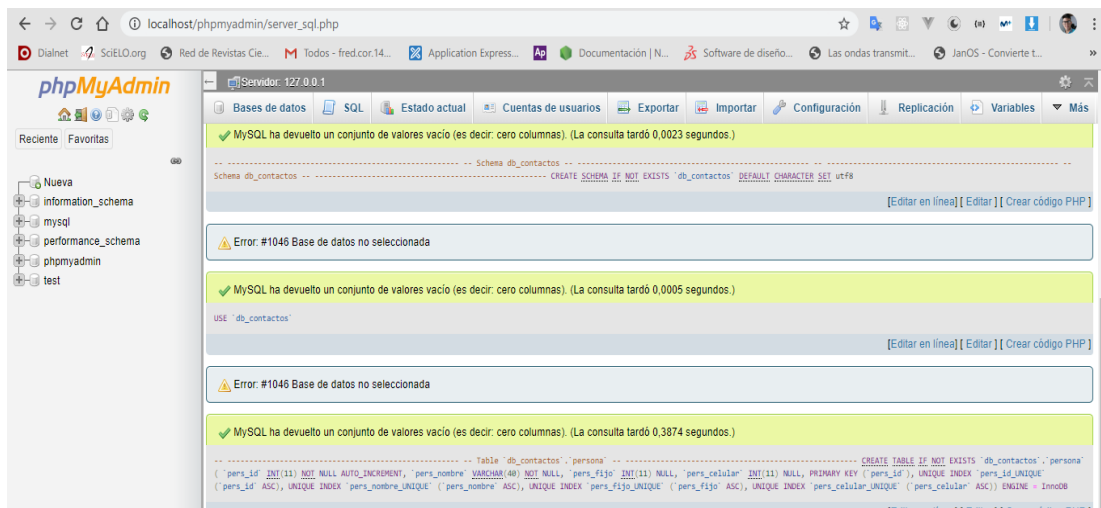
function validarTelefonos(input, input2, minLength, maxLength) {
  let telFijo = input.value, telCel = input2.value, valid = true
  if(telFijo == "" && telCel == "") {
    input.focus()
    toastr.warning("Debe digitar por lo menos un teléfono de contacto", "Proceso detenido", {timeout: 2000})
    valid = false
  }
  if(telFijo != "") {
    if(isNaN(telFijo)) {
      input.focus()
      toastr.warning("El teléfono fijo solo puede contener números", "Proceso detenido", {timeout: 2000})
      valid = false
    } else if(telFijo.length < minLength) {
      input.focus()
      toastr.warning("La cantidad de caracteres mínimos permitidos para el teléfono fijo son ${minLength}", "Proceso detenido", {timeout: 2000})
      valid = false
    } else if(telFijo.length > maxLength) {
      input.focus()
      toastr.warning("La cantidad de caracteres máximos permitidos para el teléfono fijo son ${maxLength}", "Proceso detenido", {timeout: 2000})
      valid = false
    }
  }
  if(telCel != "") {
    if(isNaN(telCel)) {
      input2.focus()
      toastr.warning("El teléfono celular solo puede contener números", "Proceso detenido", {timeout: 2000})
      valid = false
    } else if(telCel.length < minLength) {
      input2.focus()
      toastr.warning("La cantidad de caracteres mínimos permitidos para el teléfono celular son ${minLength}", "Proceso detenido", {timeout: 2000})
      valid = false
    } else if(telCel.length > maxLength) {
      input2.focus()
      toastr.warning("La cantidad de caracteres máximos permitidos para el teléfono celular son ${maxLength}", "Proceso detenido", {timeout: 2000})
      valid = false
    }
  }
  return valid
}
```

Continuaremos construyendo la parte del servidor para este caso, utilizaremos el modelo presentado anteriormente, el cual fue modelado mediante el software MySQL

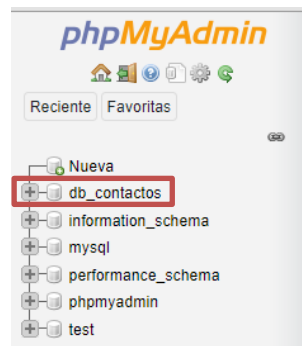
Workbench y nos ofrece la posibilidad de generar el código SQL para ejecutarlo dentro de nuestra interfaz “phpmyadmin” quien se encargará de crear la base de datos. La llamaremos “db\_contactos”.



Cuando ejecutemos el script de nuestra base de datos, nos debería arrojar una respuesta como esta.



Luego de que su ejecución haya sido exitosa, procedemos a recargar el sitio, para verificar que nuestra base de datos “db\_contactos” ha sido creada.



Luego de dicho procedimiento, seguiremos con la lógica que debemos manejar para la parte del servidor. Para este caso utilizaremos el lenguaje de programación PHP como se había mencionado anteriormente y con el trataremos de enviarle datos al cliente. Empezaremos construyendo una función que nos permita conectarnos con la base de datos que creamos anteriormente. Para ello utilizaremos una clase nativa de PHP llamada PDO que nos proporciona funciones para poder interactuar con bases de datos. También existe otra alternativa como mysqli.

```
<?php
function getConnection() {
    $credentials = array(
        'local' => array(
            'hostname' => "localhost",
            'database' => "db_contactos",
            'username' => "root",
            'password' => ""
        )
    );
    try {
        $conn = new PDO(
            "mysql:host={$credentials['local']['hostname']};dbname={$credentials['local']['database']}",
            $credentials['local']['username'],
            $credentials['local']['password'],
            [
                PDO::ATTR_EMULATE_PREPARES => FALSE,
                PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
            ]
        );
        $stm = $conn->prepare("SET NAMES 'utf8'");
        $stm->execute();
        return $conn;
    } catch (PDOException $ex) {
        return $ex->getMessage();
    }
}
```

También debemos crear un archivo que llamaremos “registrar.php” que como su nombre lo indica ejecutará la acción de registro en la base de datos.

```
<?php
require "../index.php";

$conn = getConnection();
if(is_string($conn)) {
    echo $conn;
} else {
    $pers_nombre = $_POST["pers_nombre"];
    $pers_fijo = $_POST["tele_fijo"];
    $pers_celular = $_POST["tele_celular"];
    if($pers_fijo != "" && $pers_celular != "") {
        $query = "INSERT INTO persona (pers_nombre, pers_fijo, pers_celular)
        VALUES ('{$pers_nombre}', {$pers_fijo}, {$pers_celular})";
    } else if($pers_fijo != "") {
        $query = "INSERT INTO persona (pers_nombre, pers_fijo)
        VALUES ('{$pers_nombre}', {$pers_fijo})";
    } else {
        $query = "INSERT INTO persona (pers_nombre, pers_celular)
        VALUES ('{$pers_nombre}', {$pers_celular})";
    }
    try {
        $stm = $conn->prepare($query);
        $stm->execute();
        echo 1;
    } catch (PDOException $e) {
        echo $e->getMessage();
    }
}
```

Si la ejecución o el registro se realiza correctamente, el servidor nos retornará un 1 con el cual nos indicará que todo ha salido bien, por el contrario, nos devolverá un mensaje con el error que llegue a suceder.

A continuación, seguiremos con la construcción de nuestra función de registrarContacto implementada en JavaScript, la cual nos permitirá realizarle la petición al servidor para que este pueda realizar dicha tarea. Realizaremos las peticiones al servidor mediante la librería de JQuery, la cual posee un método Ajax que nos facilita las peticiones http.

```
function registrarContacto() {  
    let inpNombre = document.getElementById("txtNombre")  
    let inpTel = document.getElementById("txtTelFijo")  
    let inpTel2 = document.getElementById("txtTelCel")  
    if(validarNombre(inpNombre, 1, 40) && validarTelefonos(inpTel, inpTel2, 6, 15)) {  
        let data = new FormData()  
        data.append("pers_nombre", inpNombre.value)  
        data.append("tele_fijo", inpTel.value)  
        data.append("tele_celular", inpTel2.value)  
        $.ajax({  
            url: "http://localhost/AgendaCS/Servidor/registrar.php",  
            data,  
            cache: false,  
            contentType: false,  
            processData: false,  
            type: 'POST',  
            success: res => {  
                //console.log(res)  
                if(res == 1) {  
                    toastr.success('Registrado Correctamente', 'Proceso Exitoso')  
                    inpNombre.value = ""  
                    inpTel.value = ""  
                    inpTel2.value = ""  
                    llenarTablaContactos()  
                }  
                else toastr.error(res, "Algo ha salido mal")  
            }  
        })  
    }  
}
```

Se debe tener en cuenta que el nombre que le asignemos a cada atributo del FormData es con el cual vamos a poder hacer referencia en el servidor por medio de las variables POST para este caso de PHP. Allí nosotros intentamos hacer uso de una función llamada llenarTablaContactos. Por lo tanto, es necesario, empezar a construir dicha función que nos permitirá ir mostrando todos los contactos registrados en la tabla.

```
function llenarTablaContactos() {  
    $.ajax({  
        url: "http://localhost/AgendaCS/Servidor/consultar.php",  
        data: {},  
        cache: false,  
        contentType: false,  
        processData: false,  
        type: 'POST',  
        success: res => {  
            //console.log(res)  
            try {  
                let data = JSON.parse(res)  
                let tr = ""  
                data.results.forEach(element => {  
                    tr += `  
                        <tr>  
                            <td>${element.pers_nombre}</td>  
                            <td>${element.pers_fijo ? element.pers_fijo : "No tiene"}</td>  
                            <td>${element.pers_celular ? element.pers_celular : "No tiene"}</td>  
                        </tr>  
                    `;  
                })  
                $("#tablaContactos > tr").remove()  
                $("#tablaContactos").append(tr)  
            } catch (error) {  
                toastr.error(error, "Algo ha salido mal")  
            }  
        }  
    });  
}
```

De igual manera para obtener los datos de los contactos registrados, es necesario realizar la petición al servidor. Entonces construiremos un archivo llamado “consultar.php” que nos permite realizar dicha acción.

```
<?php  
    require "../index.php";  
  
    $conn = getConnection();  
    if(is_string($conn)) {  
        echo $conn;  
    } else {  
        $stm = $conn->prepare("SELECT * FROM persona");  
        $stm->execute();  
        $data = $stm->fetchAll(PDO::FETCH_ASSOC);  
        $data = array("results" => $data);  
        echo json_encode($data);  
    }  
}
```



De igual forma para finalizar, debemos agregar en nuestra estructura HTML, el código JavaScript que vamos a utilizar. Por ejemplo, el llenado de la tabla con los contactos que se encuentran registrados, se puede realizar en el evento de carga del body utilizando el atributo “onload” y asignándole el nombre de nuestra función. También para el registro del contacto fue necesario utilizar el atributo “onclick” de la etiqueta a. Por lo tanto, el código de maquetación resultante sería el siguiente.

```
<!DOCTYPE html>
<html Lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Agenda Personal</title>
  <link href="https://fonts.googleapis.com/css?family=Bellota+Text&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link rel="stylesheet" href="css/toastr.min.css">
  <link rel="stylesheet" href="css/index.css">
</head>
<body onload="llenarTablaContactos()">
  <main class="container mt-3">
    <h1 class="text-center mb-3"><b>Agenda Personal</b></h1>
    <section class="row">
      <div class="col-sm-5 cajaContacto p-4">
        <form name="frmContacto" method="POST" action="../Servidor/index.php">
          <div class="form-group">
            <label for="txtNombre">Nombre: </label>
            <input type="text" name="txtNombre" id="txtNombre" class="form-control" placeholder="Ingrese nombre">
          </div>
          <div class="form-group">
            <label for="txtNombre">Teléfono Fijo: </label>
            <input type="text" name="txtTelFijo" id="txtTelFijo" class="form-control" placeholder="Ingrese teléfono fijo">
          </div>
          <div class="form-group">
            <label for="txtNombre">Teléfono Celular: </label>
            <input type="text" name="txtTelCel" id="txtTelCel" class="form-control" placeholder="Ingrese teléfono celular">
          </div>
          <a onclick="registrarContacto()" class="btn btn-info btn-block">Agregar Contacto</a>
        </form>
      </div>
      <div class="col-sm-7">
        <div class="table-responsive">
          <table class="table table-bordered table-hover">
            <thead class="thead-dark text-center">
              <tr>
                <th>Nombre</th>
                <th>Teléfono Fijo</th>
                <th>Teléfono Celular</th>
              </tr>
            </thead>
            <tbody id="tablaContactos" style="background-color: white;">
              <tr>
                <td></td>
                <td></td>
                <td></td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </section>
  </main>
  <script src="js/jquery-3.4.1.min.js"></script>
  <script src="js/bootstrap.min.js"></script>
  <script src="js/toastr.min.js"></script>
  <script src="js/validaciones.js"></script>
  <script src="js/index.js"></script>
</body>
</html>
```



Universidad de Pamplona  
Pamplona - Norte de Santander - Colombia  
Tels: (7) 5685303 - 5685304 - 5685305 - Fax: 5682750 - www.unipamplona.edu.co

De esta forma podemos visualizar como funciona correctamente la funcionalidad de registro como la de consulta de todos los contactos.

localhost/AgendaCS/Cliente/index.html

Agenda Personal

Proceso Exitoso Registrado Correctamente

Nombre:

Ingrese nombre

Teléfono Fijo:

Ingrese teléfono fijo

Teléfono Celular:

Ingrese teléfono celular

Agregar Contacto

Nombre	Teléfono Fijo	Teléfono Celular
Camila Hernández	5772345	2147483647

DQS is member of:



*Formando líderes para la construcción de un nuevo país en paz*