ESP_DOMOTIC

Por: Navarro Guerrero, Jose Antonio

Curso: Administración de sistemas informáticos en red

Fecha presentación: 01/06/2022

Tutor: Fernández Ortega, Jaime

Centro: IES Zaidín Vergeles (Granada)

Índice

0. Introducción

La principal finalidad del proyecto es poder domotizar una estancia o incluso llegar a domotizar una casa, mediante el uso de una aplicación web alojada en un VPS, o en un servidor local, de forma que con un simple vistazo podemos ver y gestionar diferentes dispositivos, ya sean lámparas, persianas, etc.

Para la gestión de los dispositivos se conectarán directamente a un Arduino, con una conexión a internet para poder acceder a la base de datos con los estados de los dispositivos. Posteriormente, la aplicación web está hecha en su mayoría con PHP junto a un framework de JavaScript (Jquery). Dicha aplicación tendrá un login de usuarios para evitar que todo el que se conecte a la aplicación pueda gestionar los dispositivos, para ello dichos usuarios serán almacenados en una base de datos relacional y todos los campos de los formularios de login y registro tendrán que estar protegidos para evitar SQLi. Para la actualización de los estados en la base de datos de los diferentes dispositivos, a través de la aplicación, se implementará el uso de funciones Ajax para hacer las operaciones de forma asíncrona, sin que la aplicación sea refrescada cada vez que se modifica el estado de un dispositivo. Para la parte de frontend se utilizará CSS y junto con Boostrap (un framework de CSS).

Para la administración de los dispositivos en Arduino utilizaremos el lenguaje de programación C++.

1. Metodología de desarrollo

Una metodología de trabajo se refiere a un framework o marco de trabajo que es usado para estructurar, planificar y controlar el desarrollo del software. Lo primero que se tiene que analizar son los objetivos finales de dicha aplicación, una vez determinados dichos objetivos, realizaremos el diseño de dicha aplicación y las interacciones del usuario con la misma, mediante el uso de diagramas y esquemas. Y una vez diseñada la aplicación se comenzará con el desarrollo de código.

1.1. Objetivos

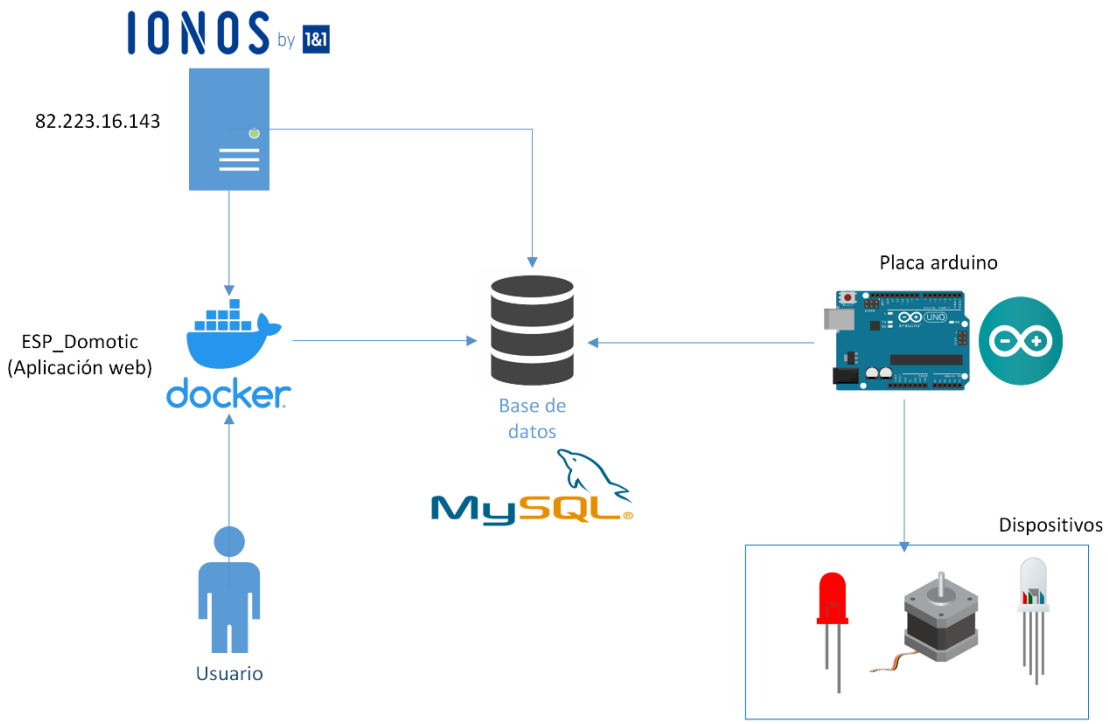
La aplicación debe disponer de un sistema de autenticación de usuarios para que puedan acceder a la administración de sus dispositivos, y evitar que cualquiera pueda controlarlos. También la aplicación debe disponer de una conexión con la base de datos para poder actualizar los estados de los dispositivos en tiempo real, y que estos cambios en la base de datos sean leídos por el Arduino y se vean reflejados en cambios visibles por los diferentes dispositivos conectados y configurados.

1.2. Diseño

Se ha decidido que la aplicación sea una aplicación web, la cual para el login y registro de usuarios se hará mediante acceso a un base de datos, junto con registro y la administración de los diferentes dispositivos asignados a los usuarios, por lo que antes de desarrollar el código necesario para la aplicación, es necesario planificar los diferentes diagramas de interacción del usuario con la aplicación, como el diagrama de aplicación, diagrama de flujo o el diagrama de base de datos que muestra la estructura de la misma.

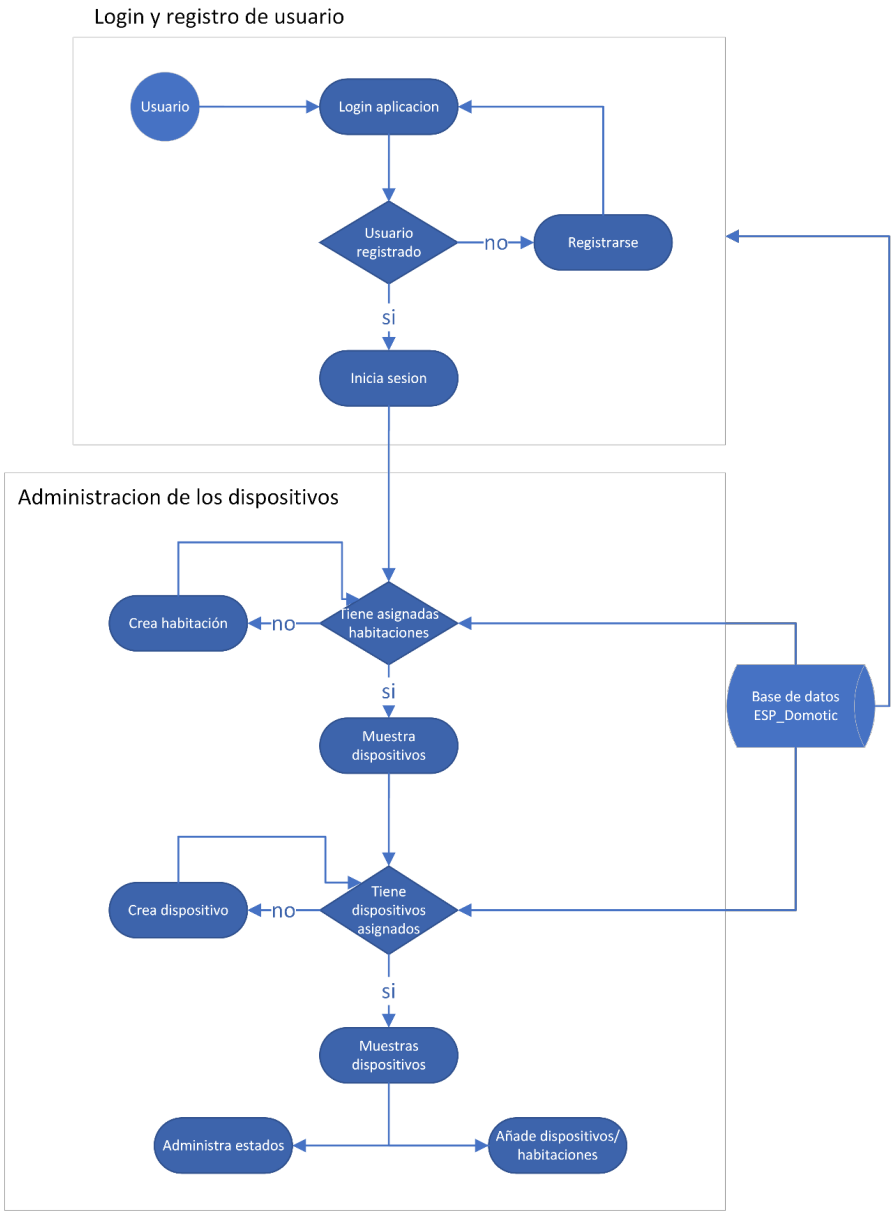
1.2.1. Diagrama de aplicación

El diagrama de aplicación es una herramienta que se utiliza para mostrar de forma gráfica el contenido de la aplicación, de sus componentes y su forma de despliegue. En este caso consta de una aplicación alojada en un VPS de la plataforma IONOS, aunque la aplicación está implementada mediante el uso de Docker y la base de datos está implementada con MySQL mediante el uso de una base de datos relacional. Otra parte fundamental del proyecto es la parte del Arduino que es la encargada de gestionar los dispositivos conectados al mismo. Para ello es necesario que el Arduino se comunique con la base de datos.


IONOS by 1&1
82.223.16.143
ESP_Domotic
(Aplicación web)
docker
Base de
datos
MySQL
Placa arduino
Dispositivos
Usuario

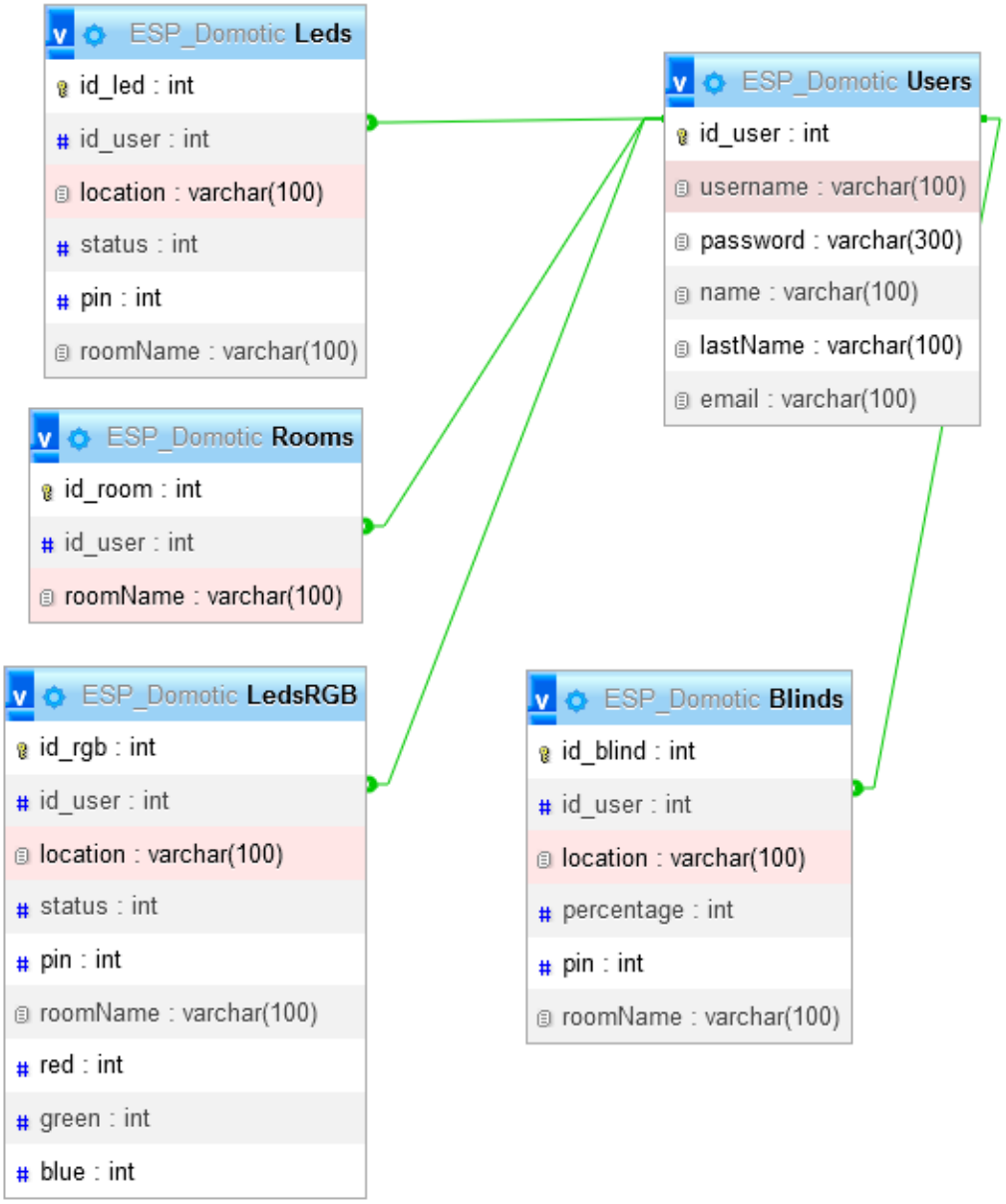
1.2.2. Diagrama de flujo

El diagrama de flujo es utilizado para representar la secuencia de actividades en un proceso, en este caso en la utilización de una aplicación, para ello muestra el comienzo de proceso, los puntos de decisión y el final del mismo.



1.2.3. Diagrama de base de datos

El diagrama de base de datos se utiliza para visualizar y construir una base de datos, por lo tanto, este es el siguiente diagrama de base de datos que describe las diferentes tablas de la base de datos que se ha diseñado para la aplicación web de ESP_Domotic. Aquí se muestra las relaciones de la tabla **Usuarios(Users)** con las demás tablas de los dispositivos.

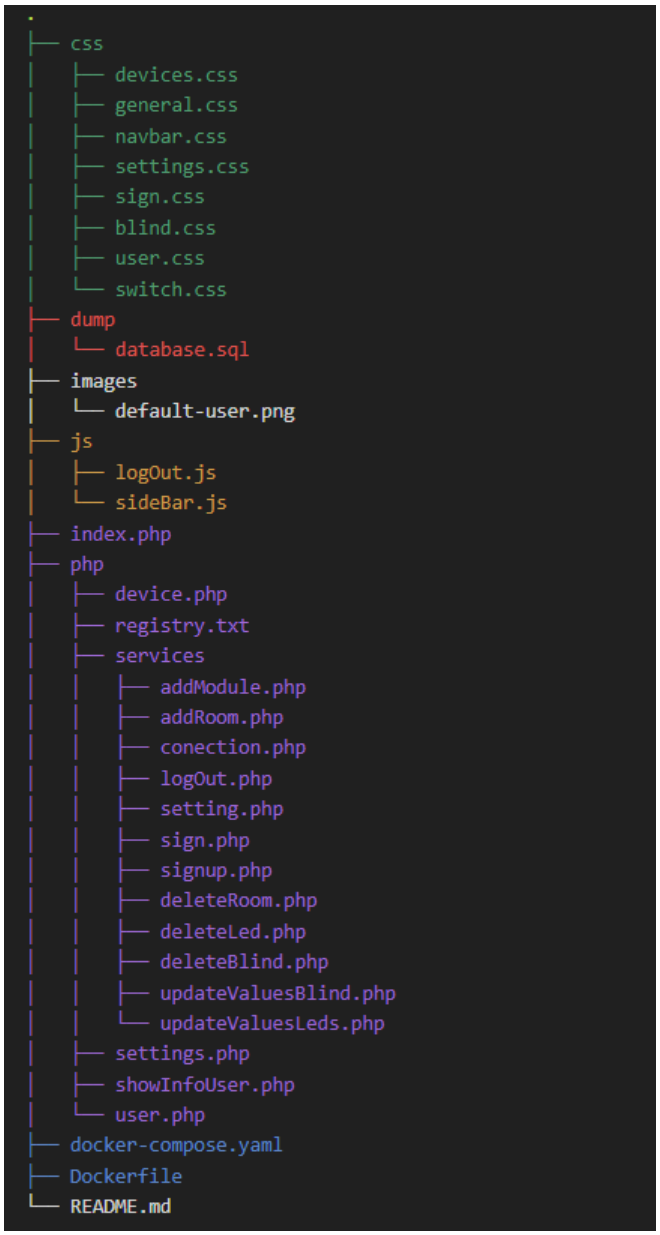


2. Desarrollo

2.1. Estructura

Como se ha especificado anteriormente la aplicación consta de dos partes, la parte de la aplicación web está programada en PHP y Jquery (JavaScript) que será la alojada en el servidor VPS con Docker y la otra parte que se ejecutará en el Arduino programada en C++.

2.1.1. Aplicación web (Servidor)



2.1.2. Domotización (Arduino)

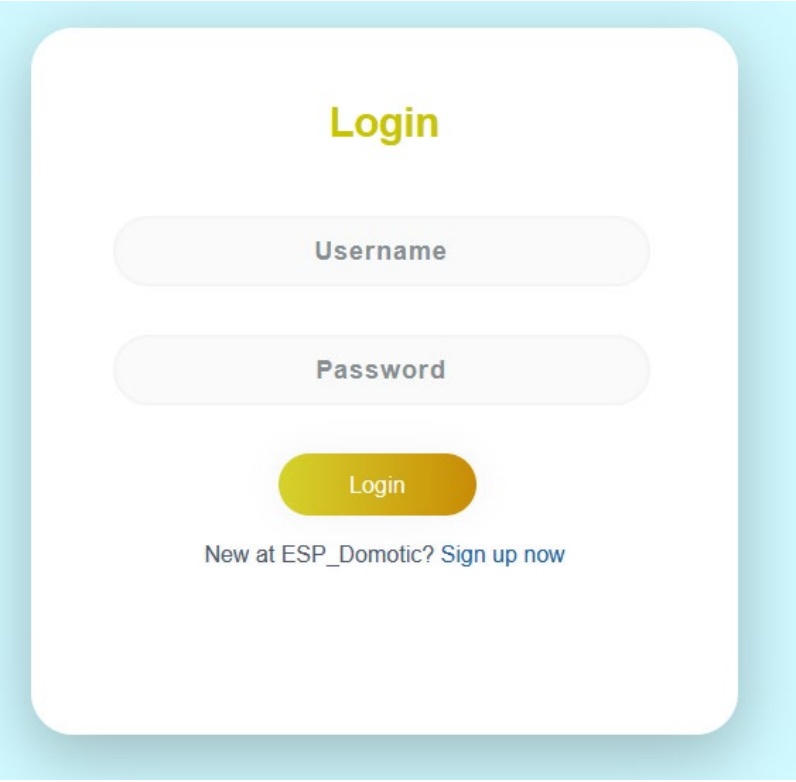
```
.
├── connectDatabase.h
├── setup.h
└── main.ino
```

2.2. Desarrollo del código

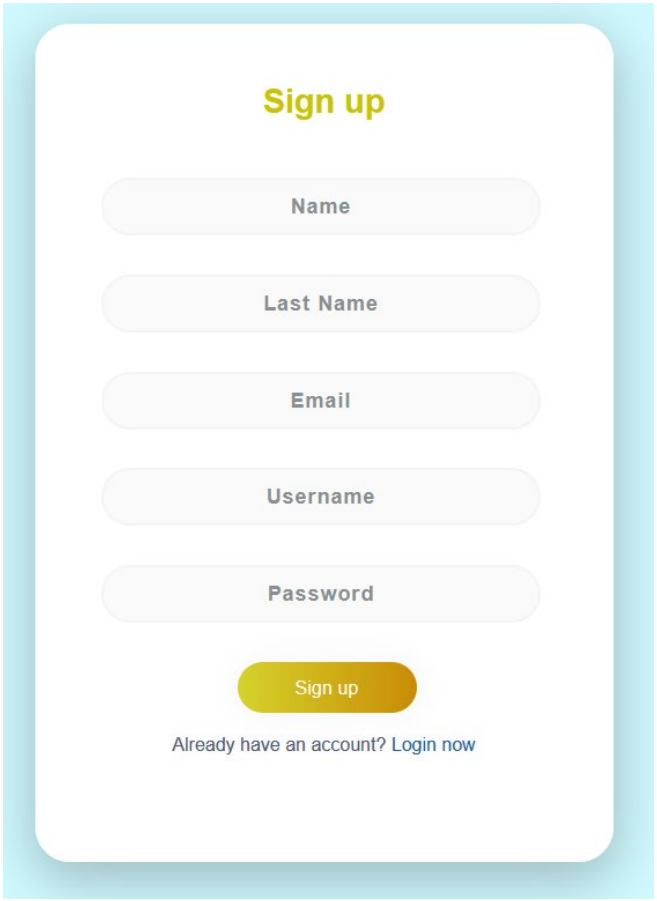
2.2.1. Login y registro de usuarios

Nota aclarativa: Todo el código se ha desarrollado en ingles por motivos de naturalidad y constubre.

La primera parte del código de la aplicación web es el formulario de **login y registro** de usuarios que cuando se rellena el formulario completo ya sea el de login o registro se llama a sus respectivas funciones que son las encargadas de recoger los datos de los formularios respectivos.



```php
<div class="login">
    <p class="sign">Login</p>
    <form class="form1" action="<?php $_SERVER["PHP_SELF"];?>" method="POST">
        <input type="text" class="un" name="username_log" align="center" placeholder="Username" require/>
        <input type="password" class="pass" name="password_log" align="center" placeholder="Password" require/>
        <input type="submit" class="submit" name="login" value="Login"/>
    </form>
```




```php
<div class="signup">
    <p class="sign" align="center">Sign up</p>
    <form class="form1" action="<?php $_SERVER["PHP_SELF"];?>" method="POST">
        <input type="text" class="un" name="name" placeholder="Name" required />
        <input type="text" class="un" name="lastName" placeholder="Last Name" required />
        <input type="email" class="un" name="email" placeholder="Email" required />
        <input type="text" class="un" name="username" placeholder="Username" required />
        <input type="password" class="pass" name="password" placeholder="Password" required />
        <input type="submit" class="submit" name="signup" value="Sign up"/>
    </form>

    <p class="text">Already have an account? <a href="../../index.php">Login now</a></p>

</div>
```

La primera parte del login y registro de usuarios seria donde se reciben los datos de los respectivos formularios. A la hora de guardar los datos en variables, estos datos recibidos pasan a través de un filtro que escapa los caracteres especiales y entrecomilla en caso de que sea necesario.

```php
$name=$con->quote($_POST['name']);
$lastName=$con->quote($_POST['lastName']);
$email=$con->quote($_POST['email']);
$user=$con->quote($_POST['username']);
$password=$con->quote($_POST['password']);
//Encryp the password with the sha1;
$pass_encrypted= sha1($password);
```

Una vez se guardan los datos de los formularios en variables, el siguiente paso en el caso del **registro de usuarios** a la hora de guardar la contraseña en la base de datos es necesario que no se guarde en texto plano sin codificar, para ello se utiliza la función **sha** para cifrar la contraseña y añadirle una capa más de seguridad a la aplicación.

```php
//Encryp the password with the sha1;
$pass_encrypted= sha1($password);
```

Para evitar que se pueda hacer una **SQLi** en vez de hacer una consulta con el método **query** se hacen con los métodos **prepare** y **execute,** que evitan las SQLi, ya que sustituyen el valor del marcador por el valor seguro que tiene almacenadas las variables. La primera consulta que aparece es la que se encarga de comprobar si existe un usuario con ese nombre de usuario.

```php
$sql_user="SELECT id_user from Users where username=:username";
//The query return the id_user
$result= $con->prepare($sql_user);
$result->execute(array(':username' => $user));
//Count the rows of the returned query
$rows = $result->rowCount();
```

Una vez que se ha comprobado que no existe en la base de datos un usuario con dicho **nombre de usuario** (**username**) se hacer la consulta de inserción en la base de datos junto con la comprobación de que la consulta de inserción se ha realizado con éxito y la posterior redirección al formulario de **login.**

```php
//Count the rows of the returned query
$rows = $result->rowCount();

//If the num rows isnt 0 means that the username exists in the database
if($rows>0){
    //Print a message and redirect to index.php
    echo "<script>
    alert('User entered already exists');
    </script>";

//If the num rows is 0 means that the username not exists in the database
}else{
    $sqlinsert= $con -> prepare("INSERT INTO Users (name,lastName,email,username,password) values (:name,:lastName,:email,:user,:password)");
    $sqlinsert->execute(array(':name' => $name,':lastName' => $lastName,':email' => $email,':user' => $user,':password' => $pass_encrypted));

    //If the query was good do the scrpit in the echo
    header("Location: ../../index,php");

}
```

En el caso de la función de **login** una vez que se reciben los datos y se guardan de la misma forma que en el registro de usuarios se realiza una la consulta con los métodos **prepare y execute.**

```php
//The query return the id_user
$sql = "SELECT id_user from Users where username=:username and password=:password";
$sql_result=$con->prepare($sql);
$sql_result->execute(array(':username' => $username, ':password' => $pass_encrypted));
```

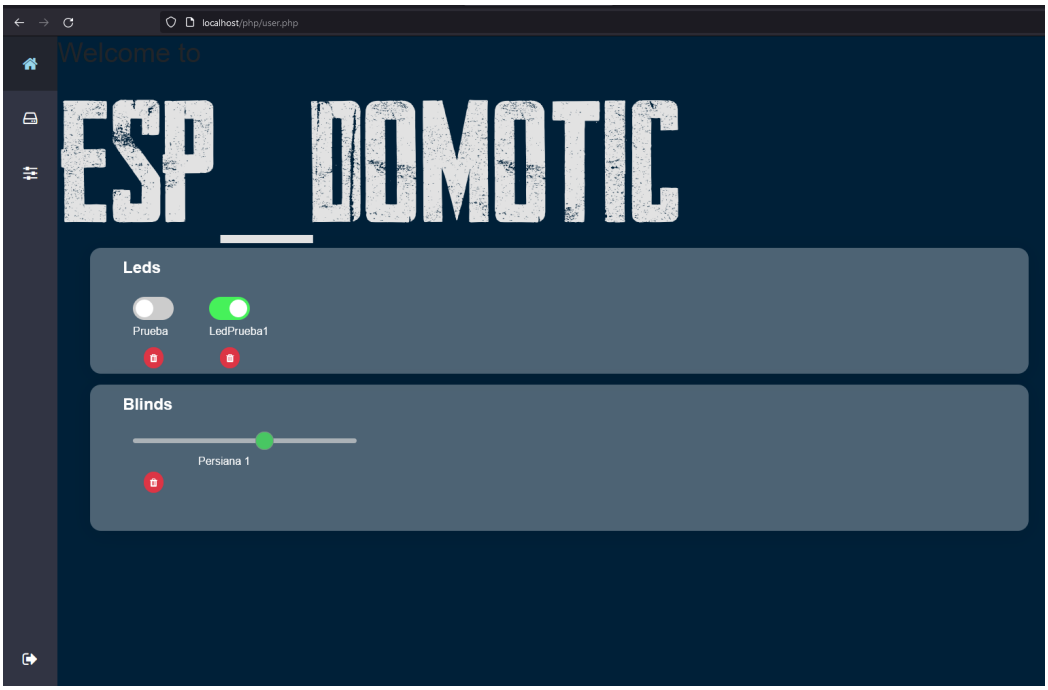
Si existe el usuario con los datos introducidos en el formulario y en caso de coincida una tupla, inicia sesión y redirige al usuario a una página de usuario.

```php
//Count the rows of the query
$rows=$sql_result->rowCount();

//If the num of rows create a SESION variable
if($rows>0){
    //Use the function for creating a new session
    session_start();
    //Asign the string that corresponds the row returned
    $row = $sql_result->fetch(5);
    $_SESSION['id_user']=$row->id_user;
    //Redirect to the user's page
    header("Location:./php/user.php");
```

En caso contrario muestra un mensaje de aviso de que alguno de los datos introducidos nos es correcto.

```
}else{
     //Print message and redirect to the index.php
     echo "<script>
             alert('User or password incorrect');

         </script>";
 }
```

Una vez que el usuario se loguea en la aplicación la primera página que verá será **user.php** donde le aparecerán los dispositivos que tiene asignados.



2.2.2. Barra de navegación

Para el caso de la **barra de navegación** lateral será un elemento común en **todas las páginas accesibles** por el usuario, como se puede observar en la barra de navegación, aparecen cuatro botones, tres botones para redirigir en al usuario en las diferentes páginas y uno posicionado en la parte inferior de la barra de navegación que es el encargado de **desloguear** al usuario y redirigirlo a la página de **login.**

```
    <nav class="sidebar-navigation">
    <ul>
        <li class="active" id="home">
            <i class="fa fa-home"></i>
            <span class="tooltip">Home</span>
        </li>
        <li id="device">
            <i class="fa fa-hdd-o"></i>
            <span class="tooltip">Devices</span>
        </li>
        <li id="setting">
            <i class="fa fa-sliders"></i>
            <span class="tooltip">Settings</span>
        </li>
        <li class="logOut">
            <i class="fa fa-sign-out" ></i>
            <span class="tooltip">Log Out</span>
        </li>
    </ul>
</nav>

<script src="../js/logOut.js"></script>
<script src="../js/sideBar.js"></script>
```

Para poder redirigir al usuario cuando haga click en alguno de los iconos es necesario que se incluya código de **JavaScript** con **Jquery**. En el caso de los tres primeros iconos es necesario el archivo **sideBar.js.** Cuando se hace click sobre algún icono que tiene asignado un Id se redirige a la página asignada.

```
$('#home').on('click', function() {
    location.replace("./user.php");
});

$('#device').on('click', function() {
    location.replace("./device.php");
});

$('#setting').on('click', function() {
    location.replace("./settings.php");
});
```

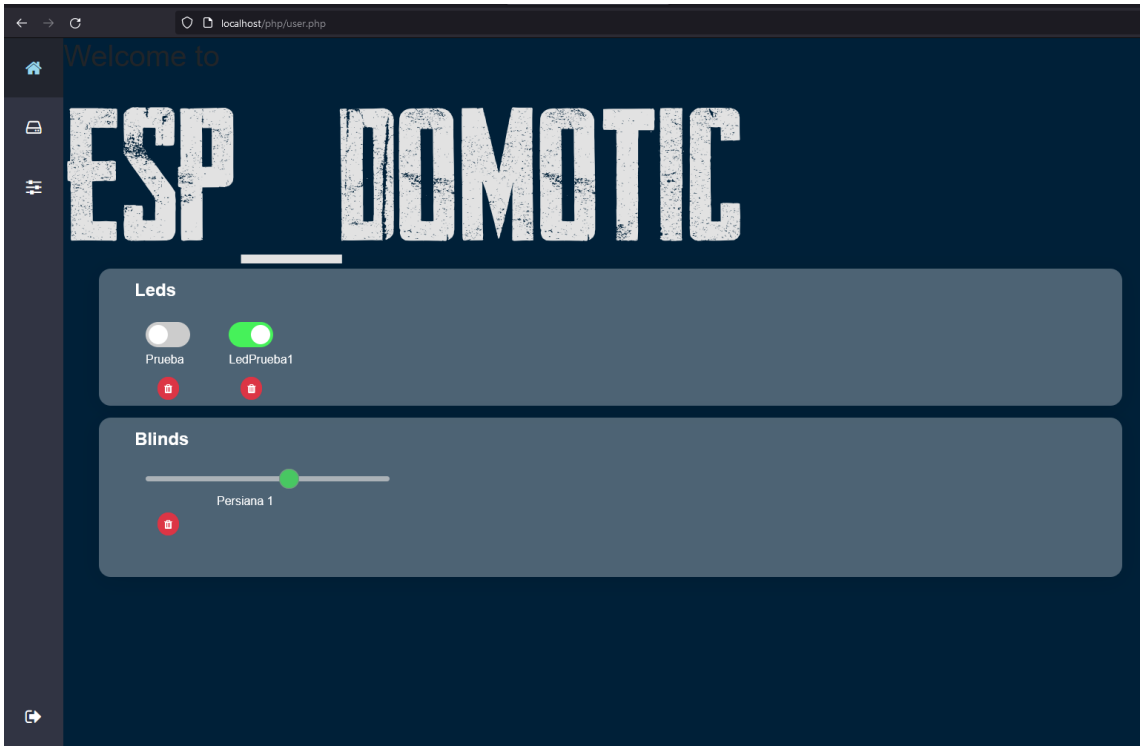
Para el icono de deslogueo es necesario el archivo **logOut.js,** Para hacer el deslogueo del usuario es necesario hacer una llamada Ajax a una función de logOut definida en el fichero **logOut.php**

```
$('li.logOut').on('click', function() {
    $.ajax({
        method: "POST",
        url: "./services/logOut.php",

    });
    location.replace("../index.php");


});
```

```
session_start();

session_destroy();

$_SESSION[] = NULL;
```

2.2.3. Página de inicio (Home)

La página de inicio para el usuario es **user.php,** como se ha dicho anteriormente, es en esta página donde se muestran los dispositivos asignados al usuario correspondiente que se haya logueado.



Para mostrar los dispositivos lo que hace es una consulta a la base de datos donde el criterio de búsqueda es el **id_user.**

```php
$username = $_SESSION['id_user'];
$sql_muestra_led = "SELECT * from Leds where id_user = '$username'";

$con = conectDatabase();

$result=$con->query($sql_muestra_led);
```

```php
<?php
$username = $_SESSION['id_user'];
$sql = "SELECT * from Blinds where id_user = '$username'";

$result=$con->query($sql);
if ($result->rowCount() != 0 ) {
  while ($row = $result->fetch(5)){
```

Una vez que se hace la consulta lo que se necesita es hacer un bucle **while,** puesto que no se sabe cuántos dispositivos tiene asignados.

```
while ($row = $result->fetch(5)){

    $id_led=$row->id_led;
    $location=$row->location;
    $status=$row->status;

        echo"<div class=\"col-sm-1 d-inline\">
                <label class='switch'>";
        if($status==0){
              echo "<input type='checkbox' id='$id_led' value='$status'>";
        }else{
              echo "<input type='checkbox' id='$id_led' value='$status' checked>";
        }
        echo "<span class='slider round'></span>
                </label>
                <p class ='$id_led text_id_led'>$id_led</p>
                <p>$location</p>";
          echo "<li class='list-item'>";
            echo "<button id='led_$id_led' class=\"btn btn-danger btn-sm rounded-circle ml-3 mb-2\"
                        type=\"button\" data-toggle=\"tooltip\" data-placement=\"top\"
                        title=\"Delete\">
                        <i class=\"fa fa-trash\"></i>
                    </button>";
          echo "</li>";
          echo "<script>
                  $( \"#led_$id_led\" ).click(function(){
                      $.ajax({
                          method: \"POST\",
                          url: \"./services/deleteLed.php\",
                          data: { text: $(\"p.$id_led\").text() }
                      }).done(function(){
                        location.reload(true);
                      });

                  });
                </script>";
```

Dentro del bucle es donde se muestra los dispositivos en función de sus valores que tienen en la base de datos y además necesitamos que mediante el uso de una función Ajax tanto para que se actualice el valor de estado en la base de datos como para eliminar los dispositivos de la base de datos.

```
echo "<script>
      $(\"#$id_led\").change( function(){
          $.ajax({
              method: \"POST\",
              url: \"./services/updateValuesLed.php\",
              data: { text: $(\"p.$id_led\").text() }
          });
      });
  </script>";
```

La función **Ajax** hace una llamada asíncrona al fichero **updateValues{dispositivo}.php** donde se **hace la actualización de la base de datos**, para actualizar los valores se hace una consulta a la base de datos donde se coge el estado del dispositivo y se actualiza con el valor contrario.

```php
include 'conection.php';

$id_led = $_POST['text'];
$con = conectDatabase();

$query = $con->query("SELECT * from Leds where id_led = '$id_led'");
$led = $query->fetch(5);

if($led->status == "1"){
  $con->query("UPDATE Leds SET status = 0 WHERE id_led = '$id_led'");
}elseif($led->status == "0"){
  $con->query("UPDATE Leds SET status = 1 WHERE id_led = '$id_led'");
}
```

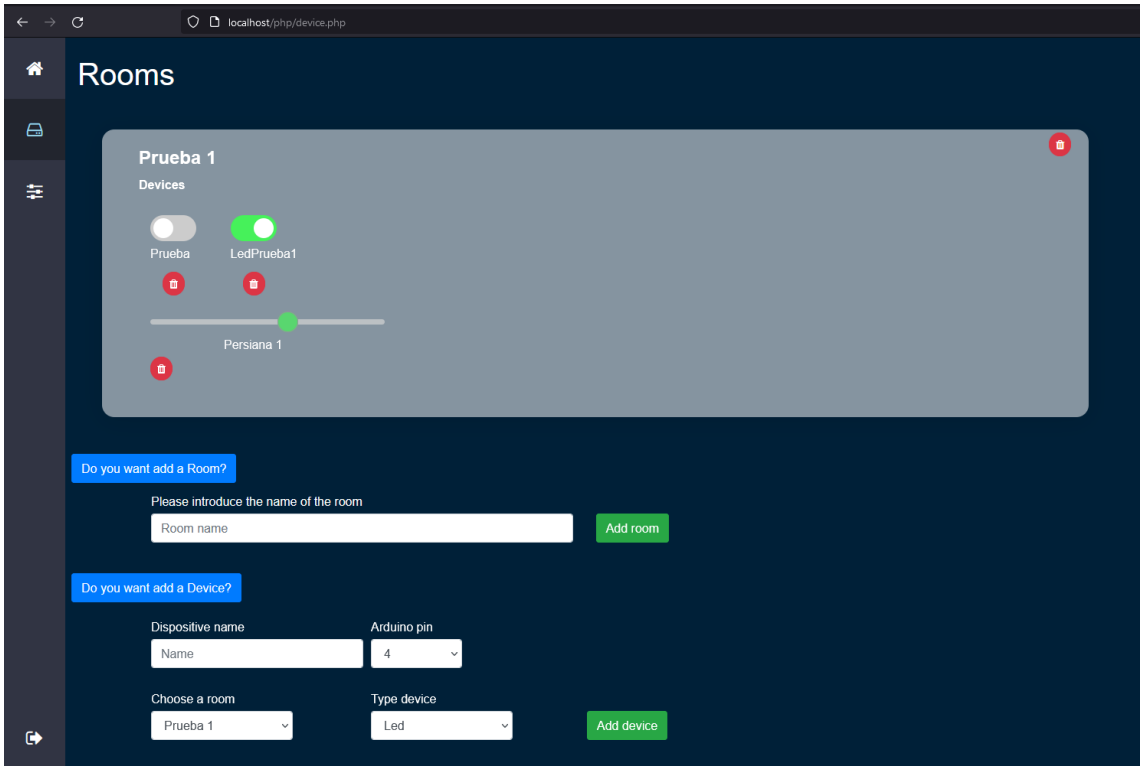
UpdateValuesLed.php

```php
$id_blind = $_POST['text'];
$percentage = $_POST['rangeValue'];
#var_dump($_POST);
var_dump($_POST['text']);

$con = conectDatabase();

$con->query("UPDATE Blinds SET percentage = $percentage WHERE id_blind = '$id_blind'");
```

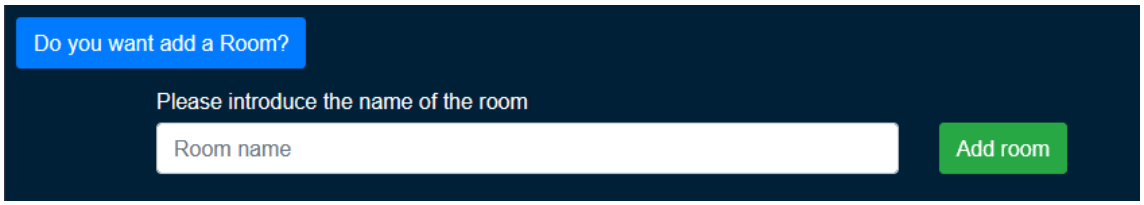
UpdateValuesBlind.php

2.2.4. Dispositivos (Device)

Otra página importante es la de **Device.php** que es la que muestra todos los dispositivos, pero a diferencia de la página de inicio, esta página los muestra separados por **habitaciones.**



También es en esta página donde se puede añadir **habitaciones.** Para ello se muestra un formulario donde recoge el nombre de la habitación que es lo primero que podemos ver en el comienzo de la pagina




```
<label>Do you want add a Room?</label>
<form action="<?php $_SERVER["PHP_SELF"];?>" method="post">
    <label>Please introduce the name of the room</label>
    <input type="text" name="roomName" placeholder="Room name"><br>
    <input type="submit" name="addRoom" value="Add room">
</form>
```

Lo siguiente que aparece en la página de Device son las diferentes habitaciones asociadas al usuario logueado, con los diferentes dispositivos de asociados a la habitación si lo tuviese.

Para mostrar las habitaciones lo primero que se hace es una consulta a la base de datos.

```php
$con = conectDatabase();

$id_user = $_SESSION['id_user'];

$rooms = $con -> query("SELECT * from Rooms where id_user='$id_user'");
```

Una vez que se tiene todas las habitaciones asignadas al usuario logueado se crea un **div** para cada habitación.

```php
while ($room = $rooms->fetch(5)){

   echo "<div class=\"room m-5\">";
     echo "<div class=\"row\">";
       echo "<p class=\"row col-10 mt-4 ml-5 roomName\">$room->roomName</p>";
       echo "<li class='list-item mt-1 ml-5 pl-4'>";
         echo "<button id='room_$room->id_room' class=\"btn btn-danger btn-sm rounded-circle ml-5\"
          type=\"button\" data-toggle=\"tooltip\" data-placement=\"top\" title=\"Delete\">
                  <i class=\"fa fa-trash\"></i>
              </button>";
       echo "</li>";
       echo "<spam class ='room_$room->id_room text_id_led'>$room->id_room</spam>";
       echo "<script>
               $( \"#room_$room->id_room\" ).click(function(){
                   $.ajax({
                       method: \"POST\",
                       url: \"./services/deleteRoom.php\",
                       data: { text: $(\"spam.room_$room->id_room\").text() }
                   }).done(function(){
                     location.reload(true);
                   });

               });
             </script>";
```

Una vez dentro del **div** se vuelve a hacer una consulta de los dispositivos asignados al usuario y al nombre de la habitación.

```
$result=$con->query("SELECT * from Leds where id_user = '$id_user' and roomName = '$room->roomName' UNION
                     SELECT * from Blinds where id_user = '$id_user' and roomName = '$room->roomName'");

if ($result->rowCount() != 0 ) {
```

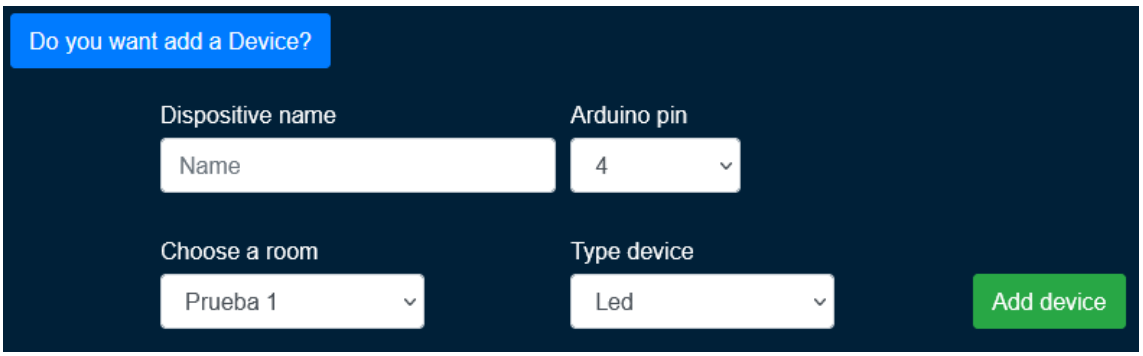
Si el número de dispositivos es distinto de cero la muestra de la misma forma que se hacía en la página de **user.php.**

```
// Show all Leds
echo "<div class=\"row mt-3 ml-5\">";
$result=$con->query("SELECT * from Leds where id_user = '$id_user' and roomName = '$room->roomName'");

while ($row = $result->fetch(5)){

    $id_led=$row->id_led;
    $location=$row->location;
    $status=$row->status;

        echo"<div class=\"col-sm-1 d-inline\">
                <label class='switch'>";
        if($status==0){
              echo "<input type='checkbox' id='$id_led' value='$status'>";
        }else{
              echo "<input type='checkbox' id='$id_led' value='$status' checked>";
        }
        echo "<span class='slider round'></span>
                </label>
                <p class ='$id_led text_id_led'>$id_led</p>
                <p>$location</p>";
          echo "<li class='list-item'>";
            echo "<button id='led_$id_led' class=\"btn btn-danger btn-sm rounded-circle ml-3 mb-2\"
                        type=\"button\" data-toggle=\"tooltip\" data-placement=\"top\"
                        title=\"Delete\">
                        <i class=\"fa fa-trash\"></i>
                  </button>";
          echo "</li>";
```

En caso de que el número de dispositivos sea cero, muestra un mensaje donde indica que esa habitación no tiene ningún dispositivo asignado.

```
}else{
    echo "<label id=\"empty\">This Room dont have any device yet<label>";
}
```

Por último, en la página aparece un formulario donde se puede añadir los dispositivos.



Para ello el formulario necesita que se le pase un nombre para el dispositivo.

```
<div class="form-group col-md-2">
  <label>Dispositive name</label>
  <input type="text" name="location" class="form-control" placeholder="Name" required/><br/>
</div>
```

Los siguiente en el formulario es necesario que se le asigne un **pin**, pero para que no se dé el caso de que varios dispositivos tengan asignados el mismo pin, para ello es necesario hacer una consulta con para sacar los pines utilizados. Una vez que se tiene los pines utilizados, se almacenan en un array.

```
<div class="form-group col">
  <label for="inputPinArduino" class="d-block">Arduino pin</label>
  <select name="pinArduino" class="form-control  col-sm-1" id="inputPinArduino">
  <?php
    $id_user = $_SESSION['id_user'];

    $pins = $con->query("(SELECT pin FROM Leds) UNION (SELECT pin FROM Blinds)");

    $used_pin[]=NULL;

    if ($pins->rowCount()!=0) {
        while($pin = $pins->fetch(5)){
            $used_pin[]=$pin->pin;
        }
    }
```

Y por último como el Arduino utilizado tiene 54 pines digitales, se hace un bucle **for** hasta 54 donde se compara si cada número está dentro del array de pines utilizados y en caso de que no esté, muestra la etiqueta **opción** con los valores que estén disponibles.

```php
for ($i=1; $i <= 54; $i++) {
    if (!in_array($i, $used_pin)) {
        echo "<option value=\"$i\">$i</option>";
    }
}
```

En el formulario se muestra un **select** donde se muestran las posibles habitaciones para asignar los dispositivos. Para ello se hace una consulta a la base de datos donde recoge la información de las habitaciones asignadas al usuario.

```php
$rooms = $con -> query("SELECT * from Rooms where id_user='$id_user'");
```

Una vez hecha la consulta se hace un bucle donde se muestra un **option** con el nombre de la habitación.

```php
 while ($room = $rooms->fetch(5)){
      echo "<option value=\"$room->roomName\">$room->roomName</option>";

 }
```

Lo último que aparece en el formulario seria un select donde se puede elegir el tipo de dispositivo que se quiere añadir.

```html
<div class="form-group col-md-2">
    <label for="inputRoomName">Type device</label>
    <select class="form-control col-md-8" name="type_device" id="inputTypeDevice">
      <option value="led">Led</option>
      <option value="blind">Blind</option>
      <option value="led_rgb" disabled>Led RGB</option>
  </select>
</div>
```

A la hora de hacer la inserción en la base de datos de los diferentes dispositivos se hace mediante una función definida en el fichero **addModule.php.** En la función se recogen los datos del formulario.

```php
<?php
if(isset($_POST['addRoom'])){
    addRoom();
}

if (isset($_POST['addDevice'])) {
  if($_POST['type_device']=="led"){
      addLED();
  }elseif ($_POST['type_device']=="blind") {
      addBlind();
  }
}
```

Una vez que se tienen los datos, se hace la consulta de inserción en la base de datos.

```php
$sql_result=$con->prepare("INSERT INTO Leds (id_user,location,pin,roomName) values (:id_user,:location,:pin,:roomName)");
$sql_result->execute(array(':id_user' => $id_user, ':location' => $location, ':pin' => $pinArduino,':roomName' => $roomName));
```

```php
$sql_result=$con->prepare("INSERT INTO Blinds (id_user,location,pin,roomName,percentage) values (:id_user,:location,:pin,:roomName,1)");
$sql_result->execute(array(':id_user' => $id_user, ':location' => $location, ':pin' => $pinArduino,':roomName' => $roomName));
```

Cuando se hace la consulta, se almacena si el resultado de la consulta ha salida bien o mal en un fichero de log.

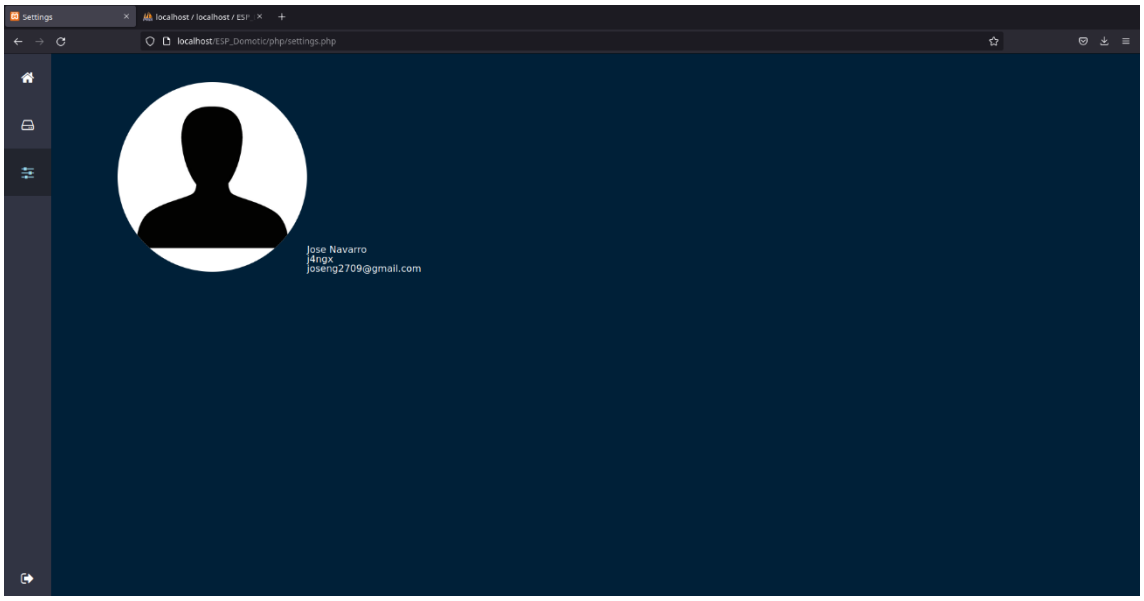
```php
//If the result is true print a Succesfull message and date in a registry.txt
if($sql_result != False){
    system("echo 'Led succesfully added | User: $id_user' >> registry.txt | date >> registry.txt");
//Else print a Error message and date in the registry.txt
}else{
    system("echo 'Error adding led to database | User: $id_user' >> registry.txt | date >> registry.txt");
}
```

Por último, se hace una actualización de la página para que aparezcan de forma inmediata

```php
echo "<script>window.location = window.location.href.split(\"#\")[0];</script>";
```

2.2.5. Información del usuario

Otra de las vistas que tiene la aplicación es una página de opciones donde el usuario logueado puede ver su información.



Para ver dicha información se hace una consulta a la base de datos con la condición de la clave de usuario almacenada en una variable de sesión.

```php
$con = conectDatabase();

$sql_result = $con->query("SELECT * from Users where id_user='$id_user'");
```

```php
$user = $sql_result->fetch(5);

$content_user = "<div id=\"content_user\"> ";
$content_user .= "<img src=\"../images/default-user.png\">";
$content_user .= "<div id=\"user_information\">";
$content_user .= "<p>$user->name $user->lastName</p>";
$content_user .= "<p>$user->username</p>";
$content_user .= "<p>$user->email</p>";
$content_user .= "</div>";
$content_user .= "</div>";
```

Más adelante se quiere implementar que el usuario pueda cambiar la imagen de su perfil.

2.2.6. Arduino

Para el desarrollo del código del Arduino se hace en el lenguaje de programación C++. La primera parte es la que se encarga de establecer la conexión con el WIFI. Para ello establecemos los valores iniciales prepara la conexión con el router que se encontraran en el fichero **setup.h.**

```
// WIFI data connection
byte mac[6];
WiFiServer server(80);
IPAddress ip(192, 168, 178, 33);
IPAddress gateway(192, 168, 178, 1);
IPAddress subnet(255, 255, 255, 0);
```

Lo siguiente que se tiene que hacer para hacer las conexiones con la base de datos seria establecer los valores iniciales que se encuentra en el fichero **connectDatabase.h.**

```
// MySQL connection data
IPAddress server_addr(192,168,178,67);  // IP of the MySQL server
char user[] = "myuser";                 // MySQL user login username
char passwordDb[] = "mypassword";         // MySQL user login password
```

Cuando se han declarado todos los valores necesarios para el funcionamiento del Arduino, lo siguiente que les seguiría sería el fichero **main.ino** que es el fichero principal que se carga en el Arduino. Lo primero que aparece es la declaración de los objetos de conexión al router y la base de datos.

```
WiFiClient dbClient;

MySQL_Connection conn((Client *)&dbClient);
```

Cuando se hace un programa para que se ejecute en Arduino se tienen que diferenciar dos funciones principales y proporcionadas por **ArduinoIDE**. La primera sería **setup.** Es esta función la que primero se ejecuta, en esta función donde se configuran todos los pines de salida o entrada, o se inicializan los objetos.

En este caso lo primero que se hace en esta función sería la conexión con el Wifi del router y mostrar un mensaje de confirmación en caso de que se establezca conexión.

```
void setup() {
  Serial.begin(9600);

  // Connect to the WIFI
  WiFi.mode(WIFI_STA);
  WiFi.config(ip,gateway,subnet);
  WiFi.begin(ssid, password);

  // Show the status when is connecting
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

  // Show the local IP of the ESP
  Serial.println(WiFi.localIP());

  WiFi.macAddress(mac);
  server.begin();
```

Lo siguiente que se hace en esta función sería establecer la conexión con la base de datos en función de los parámetros que anteriormente se definieron en el fichero **connectDatabase.h**.

```
// Conect to database
while (conn.connect(server_addr, 3306, user, passwordDb) != true) {
    delay(200);
    Serial.print ( "." );
}
```

Por ultimo se tienen que inicializar los pines como “pines de salida” y no de lectura. Para ello se hace un bucle **for** desde 1 hasta 54 que es el numero total de pines que tiene el Arduino donde se establecen dichos pines

```
//Initialize all the pins
for (int i = 1; i < 54; i++) {
  pinMode(i, OUTPUT);
}
```

La otra función principal es **loop**, se trata de un bucle infinito que se ejecuta después de ejecutar la función **setup**. Es aquí donde se hacen las consultas a la base de datos y modifica los valores de los pines según sus correspondientes valores en las tablas de los dispositivos guardados en la base de datos.

Lo primero que se hace en esta función es crear un cursor que será el encargado de hacer las consultas a la base de datos

```
void loop(){

  delay(1000);
  Serial.println("Check database");

  // Initialize the cursor of MySQL
  MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);
```

Lo siguiente que se tiene que hacer sería hacer la consulta a la base de datos y la modificación de los valores de los pines. Para ello se hace una consulta a la base de datos para extraer los valores que nos interesan, en este caso para los **Leds** los únicos valores que interesaría extraer de la base de datos son el estado que tiene en la base de datos (0 para apagado y 1 para encendido) y el pin que tiene asignado.

```
String queryLed = "SELECT status,pin FROM ESP_Domotic.Leds";
if(cur_mem->execute(queryLed.c_str())) {
  row_values *row = NULL;
  column_names *columns = cur_mem->get_columns();
  row = cur_mem->get_next_row();
  if (row != NULL) {

    if (row->values[0] == 0){
      digitalWrite(row->values[1], LOW);
    }else{
      digitalWrite(row->values[1], HIGH);
    }
    show_ok();
    while((row = cur_mem->get_next_row()) != NULL);

  }else {
    show_error();
  }
}
delete cur_mem;
```

Al igual que se ha hecho con los Leds también se hace con los diferentes dispositivos de la base de datos como las Persianas (Blinds).

```
String queryLed = "SELECT percentage,pin FROM ESP_Domotic.Blinds";
if(cur_mem->execute(queryLed.c_str())) {
  row_values *row = NULL;
  column_names *columns = cur_mem->get_columns();
  row = cur_mem->get_next_row();
  if (row != NULL) {

    servoMotor.write(row->values[0]);
    show_ok();
    while((row = cur_mem->get_next_row()) != NULL);

  }else {
    show_error();
  }
}
delete cur_mem;
```

3. Instalación

3.1. Instalación Docker

La aplicación se ha optado por alojarla en un servidor VPS con el sistema operativo Debian 11 y para alojar la aplicación se ha optado por utilizar Docker. **Docker** es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

3.1.1. Prerrequisitos

1. Lo primero que se ha de hacer es actualizar los repositorios de Debian.

```
sudo apt update && sudo apt upgrade -y
```

2. Una vez actualizado los repositorios, el siguiente paso es instalar los siguientes paquetes que son necesarios para comenzar a instalar Docker

```
sudo apt install ca-certificates
                 curl
                 gnupg
                 lsb-release
```

3. Una vez instalados los paquetes los siguiente que se ha de hacer es añadir la clave GPG oficial de Docker

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings
/docker-archive-keyring.gpg
```

4. El siguiente paso en la instalación de Docker es añadir el repositorio necesario para instalar Docker

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
      https://download.docker.com/linux/debian \ $(lsb_release -cs) stable" |
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

3.1.2. Instalar Docker Engine

Una vez se ha configurado los repositorios se actualizará los repositorios y se ha de instalar los siguientes repositorios

```
sudo apt update &&

sudo apt install docker-ce docker-ce-cli container.io docker-compose-plugin
```

3.2. Montar la aplicación

Una vez instalado Docker ya se puede crear los contenedores que contendrá la aplicación y la base de datos. para ello se utilizará Docker-compose para montar los contenedores con un solo comando.

Para utilizar Docker Compose es necesario dentro de la carpeta raíz del proyecto crear un fichero **docker-compose.yaml.** Este fichero contine las instrucciones necesarias para desplegar los diferentes contenedores.

```yaml
version: "2"
services:
    www:
        build: .
        ports:
            - "80:80"
        volumes:
            - .:/var/www/html/
        links:
            - db
        networks:
            - default
    db:
        image: mysql
        ports:
            - "3306:3306"
        environment:
            MYSQL_DATABASE: ESP_Domotic
            MYSQL_USER: user
            MYSQL_PASSWORD: test
            MYSQL_ROOT_PASSWORD: test
        volumes:
            - ./dump:/docker-entrypoint-initdb.d
            - persistent:/var/lib/mysql
        networks:
            - default
    phpmyadmin:
        image: phpmyadmin/phpmyadmin
        links:
            - db:db
        ports:
            - 8000:80
        environment:
            MYSQL_USER: user
            MYSQL_PASSWORD: test
            MYSQL_ROOT_PASSWORD: test
volumes:
    persistent:
```

3.2.1. www

El primer servicio que se desplegaría sería **www** (la aplicación web).

```
www:
    build: .
    ports:
        - "80:80"
    volumes:
        - .:/var/www/html/
    links:
        - db
    networks:
        - default
```

La primera directriz que aparece en ese servicio es un **build** del contendor, ya que en esa misma ruta esta un fichero **Dockerfile**, el cual es el encargado de montar el contendor con la imagen correcta.

```
FROM php:8.1-apache

# PHP extensions
RUN docker-php-ext-install pdo pdo_mysql

WORKDIR /var/www/html

COPY . .
```

En este caso como la aplicación se ejecuta con **PHP**, la imagen base que se necesita para que pueda correr es una que contenga **PHP** y **Apache**. Además, se necesitan instalar las extensiones para utilizar **PDO** (Orientación a objetos). Por ultimo se especifica que el directorio de trabajo será **/var/www/html** y que todos los ficheros y directorios que estén donde se ejecuta el Dockerfile se copien al directorio de trabajo del contenedor.

Lo siguiente que aparece en el docker-compose.yaml son los puertos que utilizará el contenedor, en este caso el puerto 80. A continuación aparece los volúmenes que utilizara para sincronizar los ficheros en caso de apagado o cambio en dichos ficheros. Lo siguiente indica que deberá estar sincronizado con el contenedor **db,** que más adelante se creará. Y por último se especifica la red interna que se le asignará, en este caso por defecto.

3.2.2. db

El siguiente servicio que se despliega es **db** (Base de datos), para ello se utiliza una imagen oficial **mysql** extraída de los repositorios de **Docker Hub.**

```
db:
    image: mysql
    ports:
        - "3306:3306"
    environment:
        MYSQL_DATABASE: ESP_Domotic
        MYSQL_USER: user
        MYSQL_PASSWORD: test
        MYSQL_ROOT_PASSWORD: test
    volumes:
        - ./dump:/docker-entrypoint-initdb.d
        - persistent:/var/lib/mysql
    networks:
        - default
```

De nuevo los siguiente que se asigna son los puertos que tendrá el contenedor, para el caso de la base de datos será 3306 que es lo más común. Lo siguiente que aparece en la configuración del contenedor de la base de datos son variables de configuración del contenedor requeridas por la imagen de **mysql,** como la base de datos que se creará el usuario con el que se accede y contraseña de administrador.

Los siguiente que aparece son los volúmenes con los cuales se utilizaran para dumpear o rellenar la base de datos y los lo más importante de esta sección es indicar que los datos que se agreguen sean persistentes y no se borren. Y al igual que con el contenedor **www** se le asigna la red por defecto.

3.2.3. phpmyadmin

El ultimo contendor que se utiliza es para administrar la base de datos de forma gráfica. La configuración es muy similar que la dos anteriores.

```
phpmyadmin:
    image: phpmyadmin/phpmyadmin
    links:
        - db:db
    ports:
        - 8000:80
    environment:
        MYSQL_USER: user
        MYSQL_PASSWORD: test
        MYSQL_ROOT_PASSWORD: test
```

3.2.4. Docker-compose

Una ver que se tiene el fichero docker-compose.yaml completo con toda la configuración de los contenedores, lo ultimo que se ha de hacer es ejecutar el siguiente comando en la ruta donde se encuentre el docker-compose.yaml.

```
docker-compose up -d
```

Una vez que se termine de ejecutar el comando, ya estarán desplegados los tres contenedores necesarios para que funcione la aplicación web. Para ver que los contenedores están funcionando se utiliza el comando **docker ps.**

```
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS                               NAMES
c1bc0d0f9f49   esp_domotic_www        "docker-php-entrypoi…"   56 seconds ago   Up 54 seconds   0.0.0.0:80->80/tcp                  esp_domotic-www-1
3dcf1a02a5ca   phpmyadmin/phpmyadmin  "/docker-entrypoint.…"   56 seconds ago   Up 54 seconds   0.0.0.0:8000->80/tcp                esp_domotic-phpmyadmin-1
07ce84cab52a   mysql                  "docker-entrypoint.s…"   56 seconds ago   Up 19 seconds   0.0.0.0:3306->3306/tcp, 33060/tcp   esp_domotic-db-1
```

4. Webgrafía y bibliografía

- Sean Kane, (2018) Docker: Up & Running: Shipping Reliable Containers in Production
- https://stackoverflow.com
- https://github.com
- https://docker.com
- https://hub.docker.com
- https://php.net
- https://ionos.es
- https://arduino.com
- https://boostrap.com
- https://jquery.com