Índice

[0. Introducción 2](#_Toc104893547)

[1. Metodología de desarrollo 3](#_Toc104893548)

[1.1. Objetivos 3](#_Toc104893549)

[1.2. Diseño 3](#_Toc104893550)

[1.2.1. Diagrama de aplicación 3](#_Toc104893551)

[1.2.2. Diagrama de flujo 5](#_Toc104893552)

[1.2.3. Diagrama de base de datos 6](#_Toc104893553)

[2. Desarrollo 7](#_Toc104893554)

[2.1. Estructura 7](#_Toc104893555)

[2.1.1. Aplicación web (Servidor) 7](#_Toc104893556)

[2.1.2. Domotización (Arduino) 8](#_Toc104893557)

[2.2. Desarrollo del código 8](#_Toc104893558)

[2.2.1. Login y registro de usuarios 8](#_Toc104893559)

[2.2.2. Barra de navegación 13](#_Toc104893560)

[2.2.3. Página de inicio (Home) 15](#_Toc104893561)

[2.2.4. Dispositivos (Device) 18](#_Toc104893562)

[2.2.5. Información del usuario 24](#_Toc104893563)

[2.2.6. Arduino 25](#_Toc104893564)

[3. Instalación 29](#_Toc104893565)

[3.1. Instalación Docker 29](#_Toc104893566)

[3.1.1. Prerrequisitos 29](#_Toc104893567)

[3.1.2. Instalar Docker Engine 30](#_Toc104893568)

[3.2. Montar la aplicación 31](#_Toc104893569)

[3.2.1. www 32](#_Toc104893570)

[3.2.2. db 33](#_Toc104893571)

[3.2.3. phpmyadmin 34](#_Toc104893572)

[3.2.4. Docker-compose 34](#_Toc104893573)

[4. Webgrafía y bibliografía 35](#_Toc104893574)

# 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++.

# 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.

## Objetivos

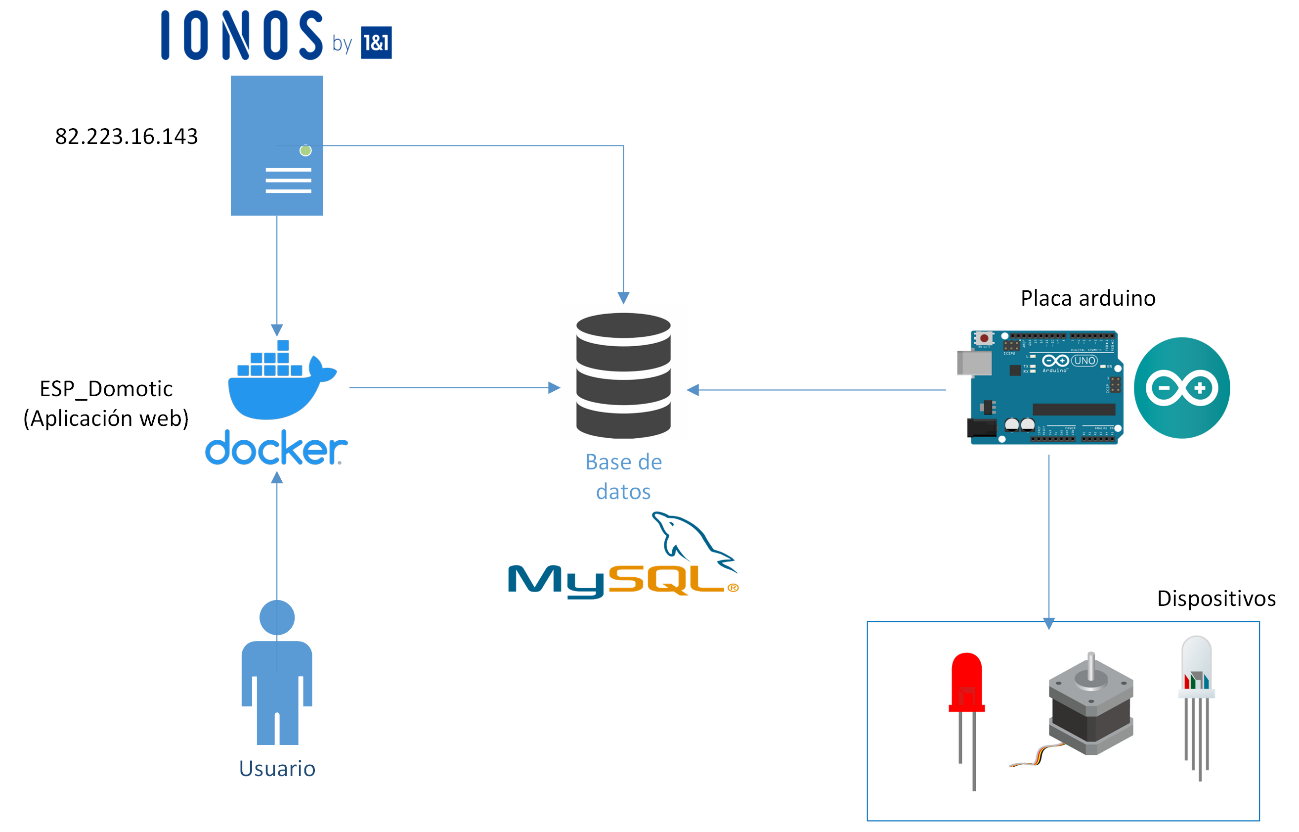
La aplicación debe disponer de un sistema de autenticación de usuarios para que puedan acceder a la administración de sus dispositivos, y evitar que cualquiera pueda controlarlos. También la aplicación debe disponer de una conexión con la base de datos para poder actualizar los estados de los dispositivos 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.

## 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.

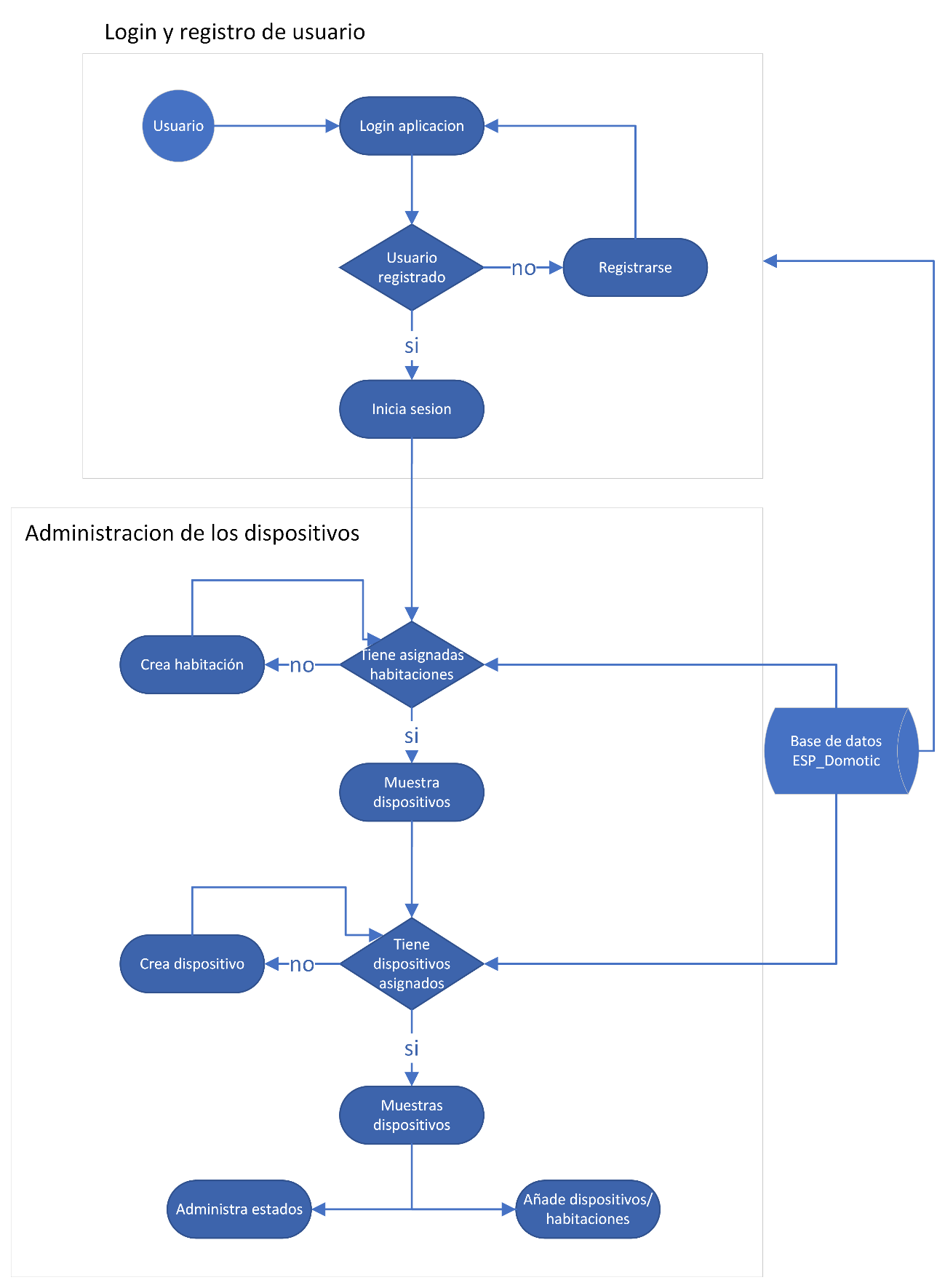
### Diagrama de aplicación

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.



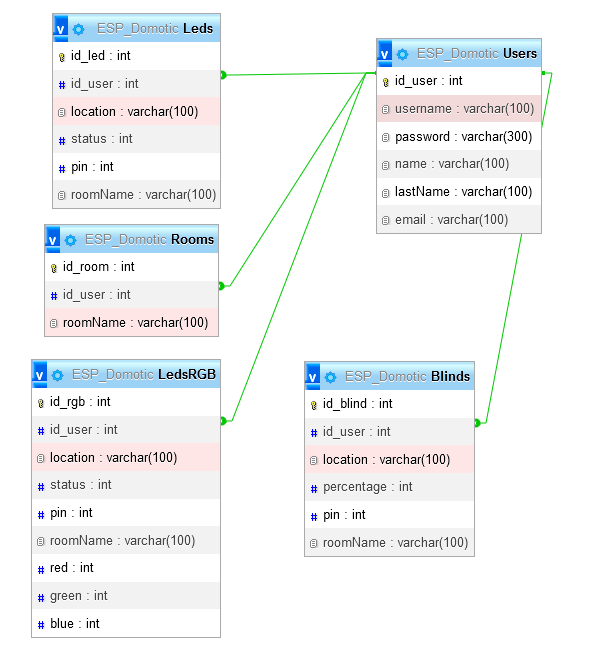
### Diagrama de flujo

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.



### 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.

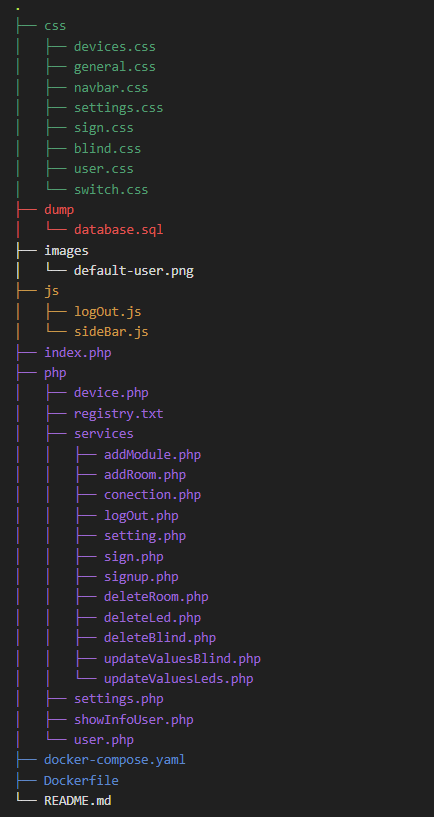


# Desarrollo

## 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++.

### Aplicación web (Servidor)



### Domotización (Arduino)

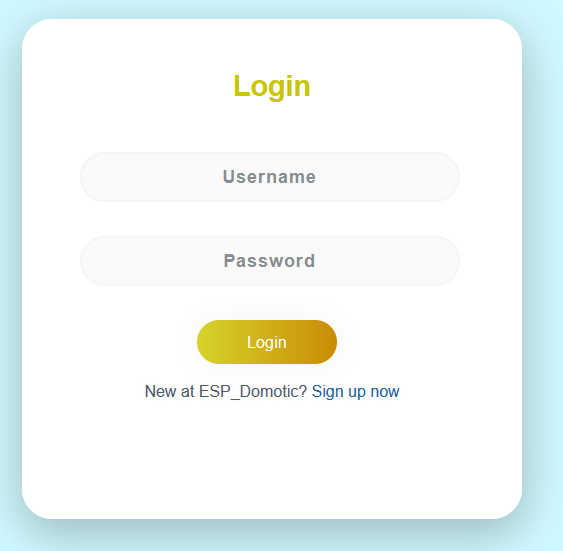


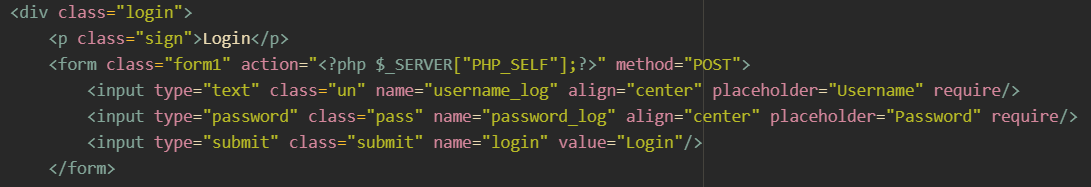
## Desarrollo del código

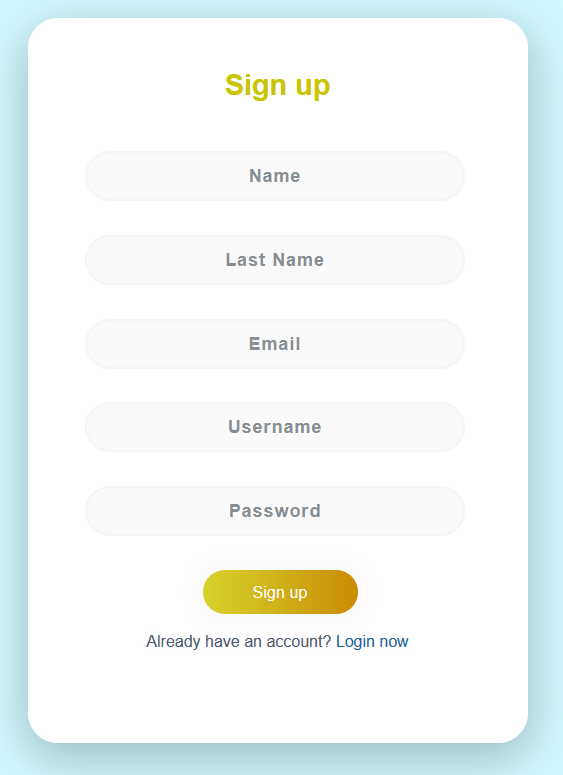
### Login y registro de usuarios

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.

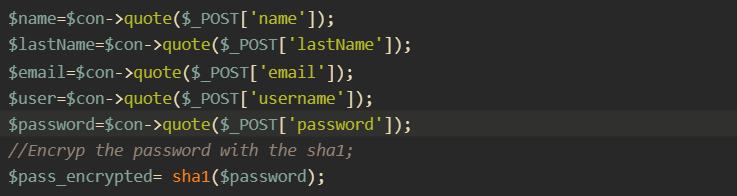






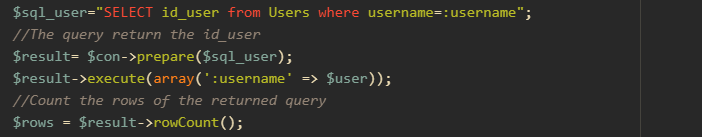


La primera parte del login y registro de usuarios seria donde se reciben los datos de los respectivos formularios. A la hora de guardar los datos en variables, estos datos recibidos pasan a través de un filtro que escapa los caracteres especiales y entrecomilla en caso de que sea necesario.

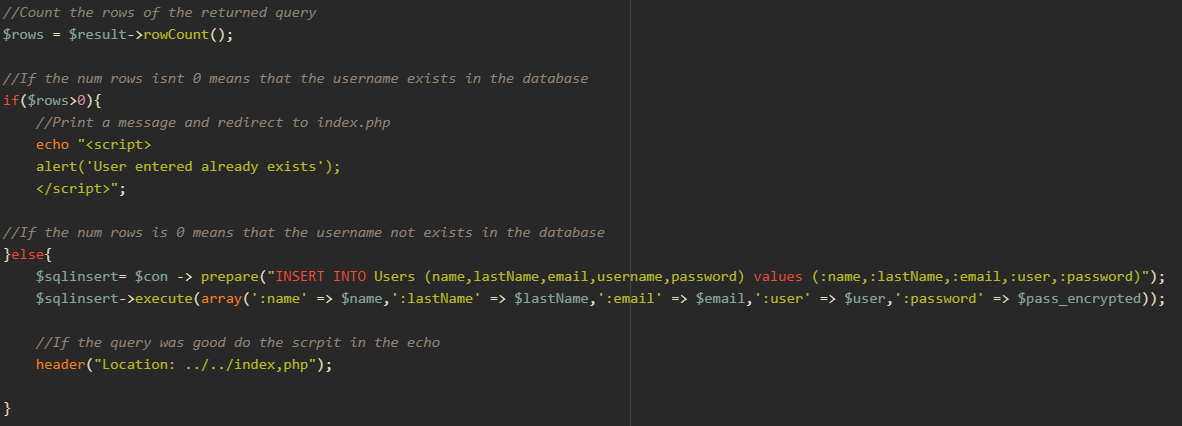


Una vez se guardan los datos de los formularios en variables, el siguiente paso en el caso del **registro de usuarios** a la hora de guardar la contraseña en la base de datos es necesario que no se guarde en texto plano sin codificar, para ello se utiliza la función **sha** para cifrar la contraseña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

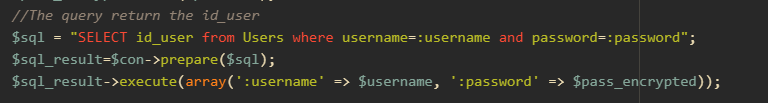
Para evitar que se pueda hacer una **SQLi** en vez de hacer una consulta con el método **query** se hacen con los métodos **prepare** y **execute,** que evitan las SQLi, ya que sustituyen el valor del marcador por el valor seguro que tiene almacenadas las variables. La primera consulta que aparece es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



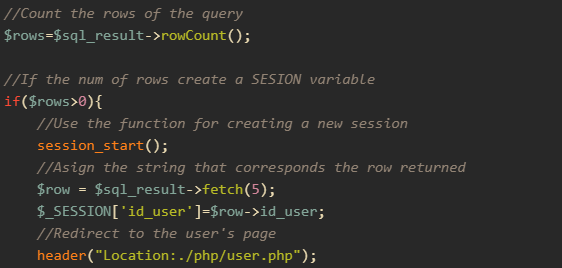
Una vez que se ha comprobado que no existe en la base de datos un usuario con dicho **nombre de usuario** (**username**) se hacer la consulta de inserción en la base de datos junto con la comprobación de que la consulta de inserción se ha realizado con éxito y la posterior redirección al formulario de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



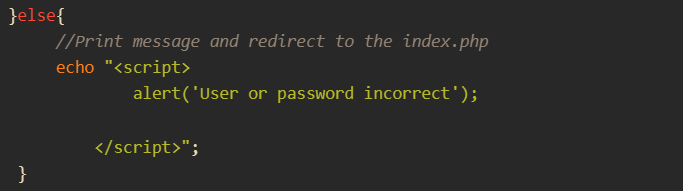
En el caso de la función de **login** una vez que se reciben los datos y se guardan de la misma forma que en el registro de usuarios se realiza una la consulta con los métodos **prepare y execute.**



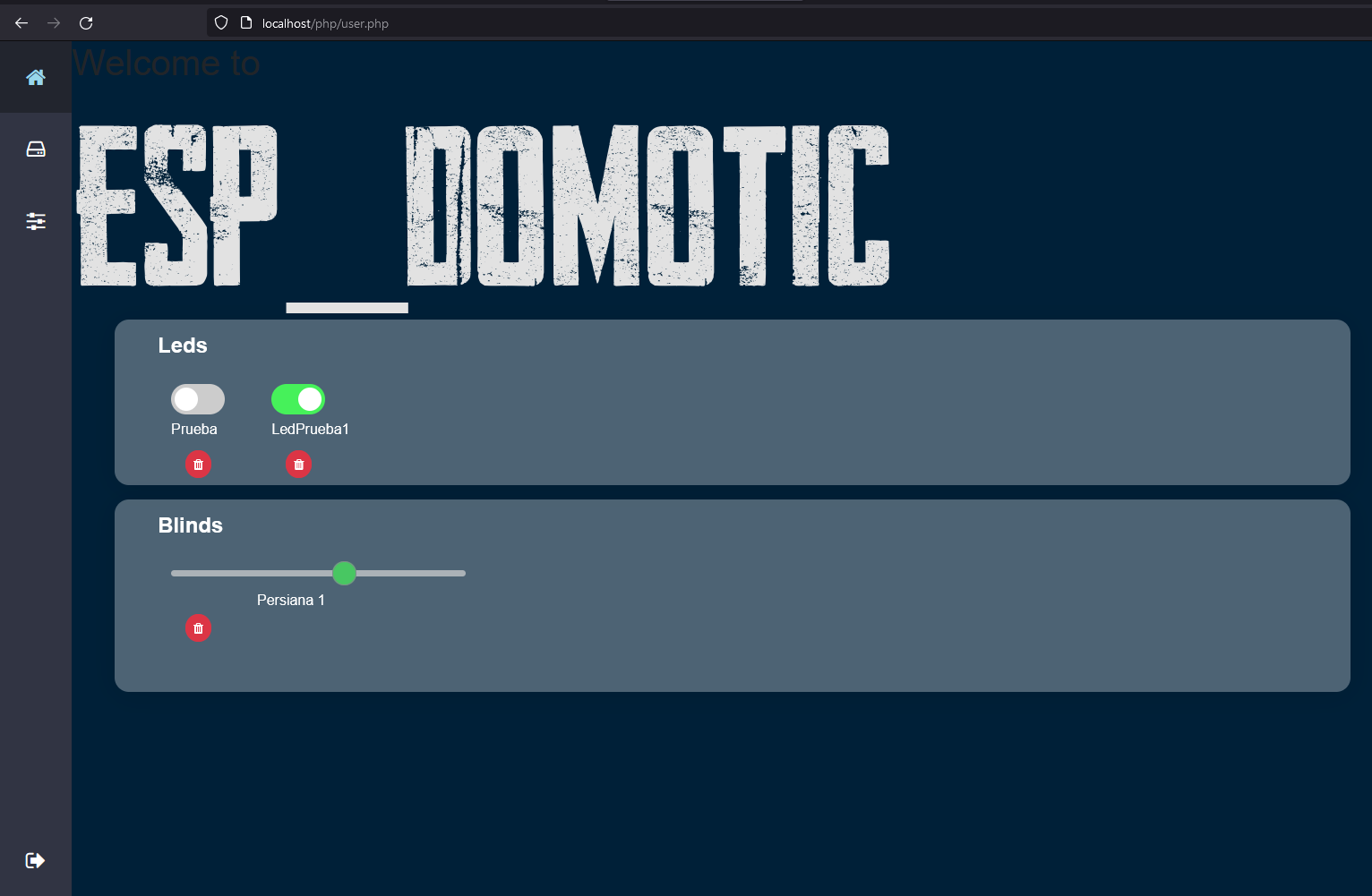
Si existe el usuario con los datos introducidos en el formulario y en caso de coincida una tupla, inicia sesión y redirige al usuario a una página de usuario.



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



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.



### 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.**





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.



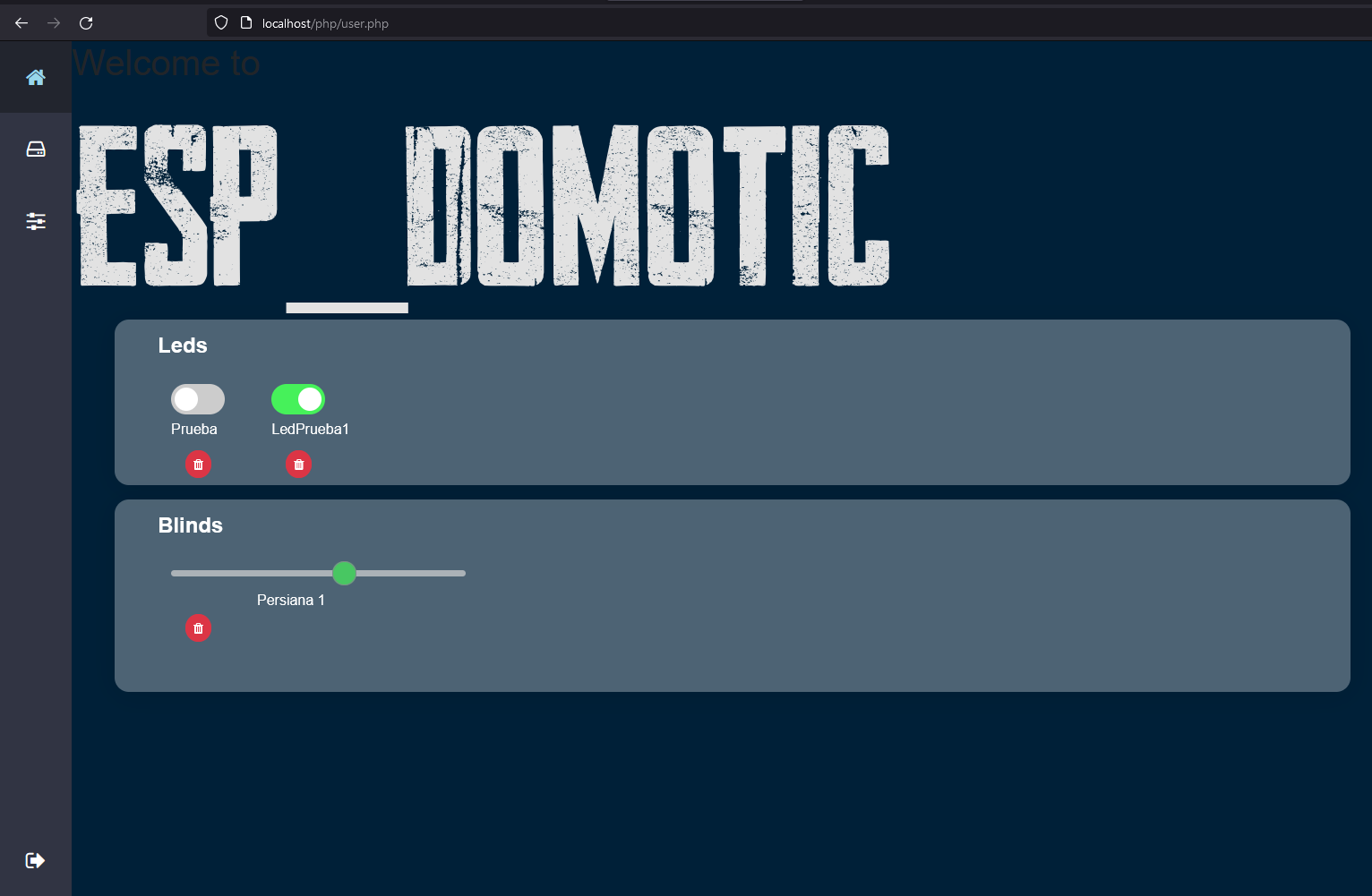
Para el icono de deslogueo es necesario el archivo **logOut.js,** Para hacer el deslogueo del usuario es necesario hacer una llamada Ajax a una función de logOut definida en el fichero **logOut.php**



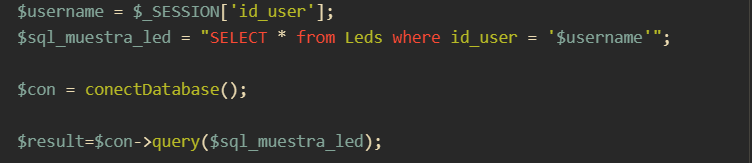
****

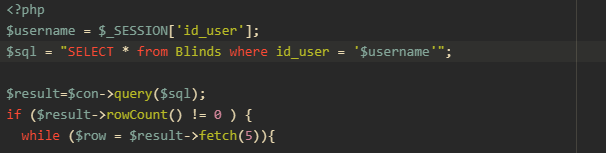
### Página de inicio (Home)

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.

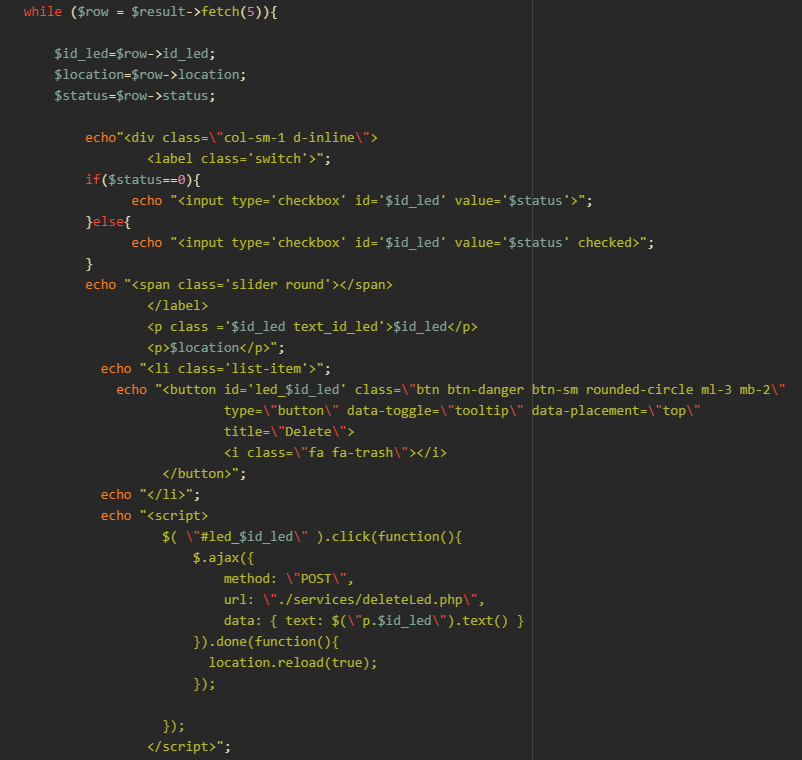


Para mostrar los dispositivos lo que hace es una consulta a la base de datos donde el criterio de búsqueda es el **id\_user.**

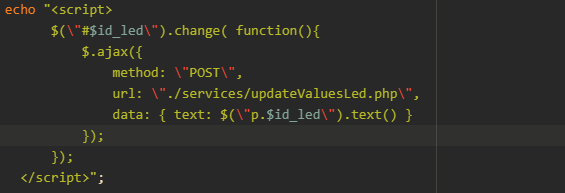




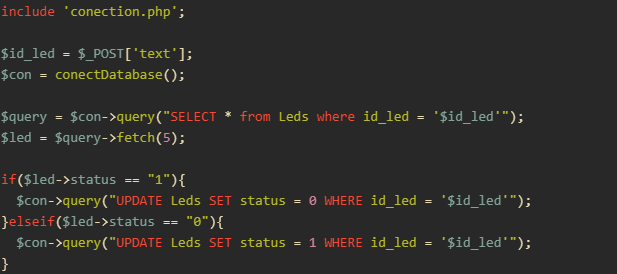
Una vez que se hace la consulta lo que se necesita es hacer un bucle **while,** puesto que no se sabe cuántos dispositivos tiene asignados.



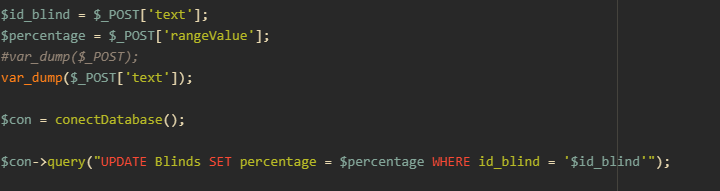
Dentro del bucle es donde se muestra los dispositivos en función de sus valores que tienen en la base de datos y además necesitamos que mediante el uso de una función Ajax tanto para que se actualice el valor de estado en la base de datos como para eliminar los dispositivos de la base de datos.



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.



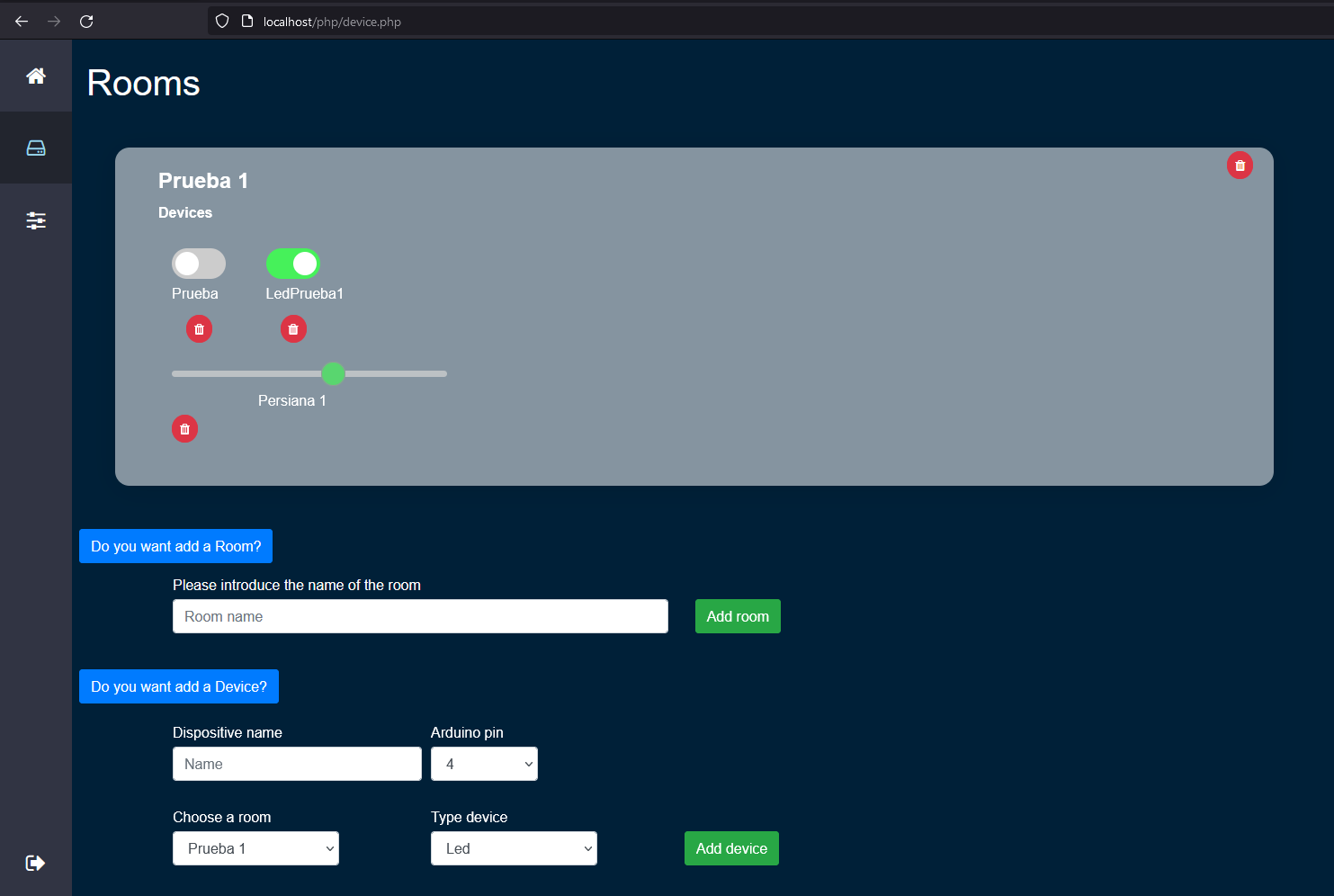
UpdateValuesLed.php



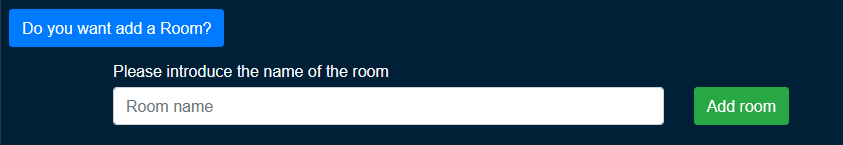
UpdateValuesBlind.php

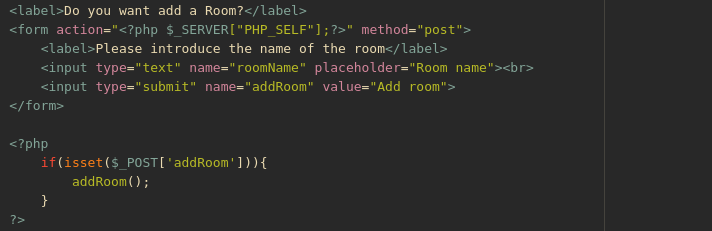
### Dispositivos (Device)

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

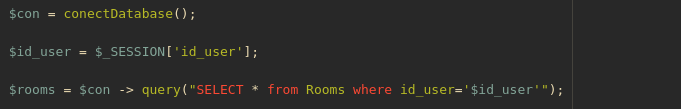




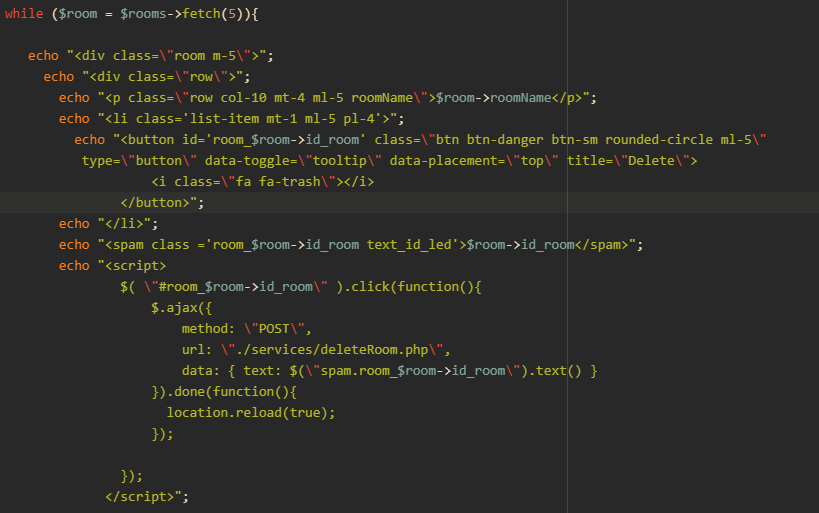
Lo siguiente que aparece en la página de Device son las diferentes habitaciones asociadas al usuario logueado, con los diferentes dispositivos de asociados a la habitación si lo tuviese.



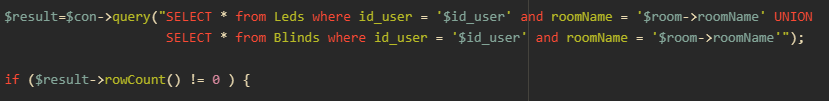
Para mostrar las habitaciones lo primero que se hace es una consulta a la base de datos.



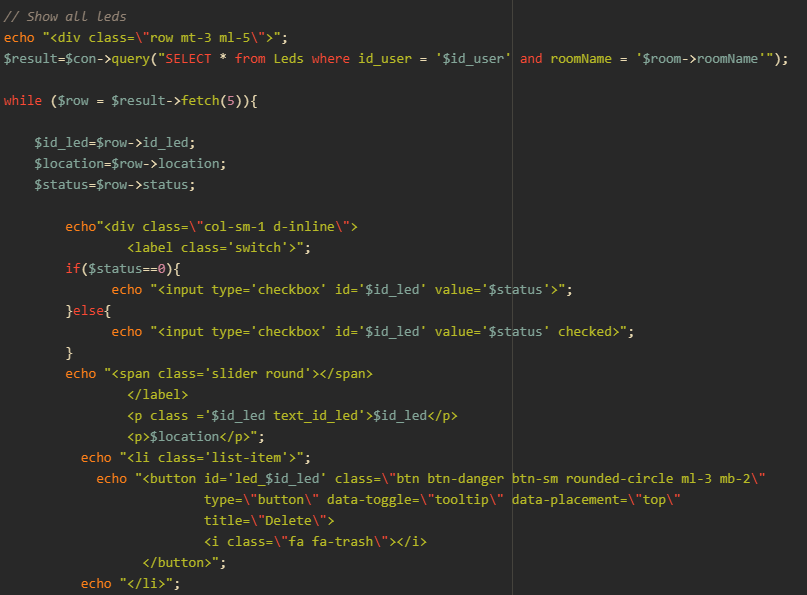
Una vez que se tiene todas las habitaciones asignadas al usuario logueado se crea un **div** para cada habitación.



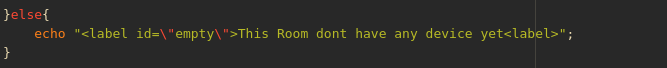
Una vez dentro del **div** se vuelve a hacer una consulta de los dispositivos asignados al usuario y al nombre de la habitación.



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

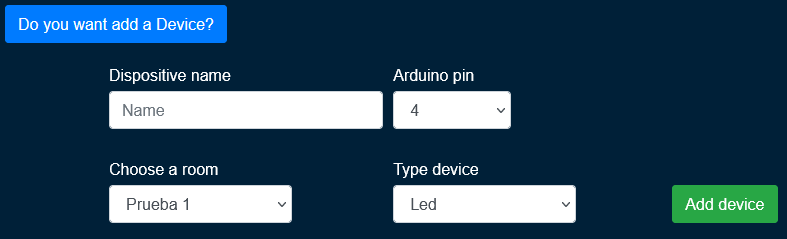


En caso de que el número de dispositivos sea cero, muestra un mensaje donde indica que esa habitación no tiene ningún dispositivo asignado.

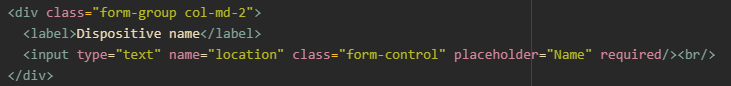




Por último, en la página aparece un formulario donde se puede añadir los dispositivos.



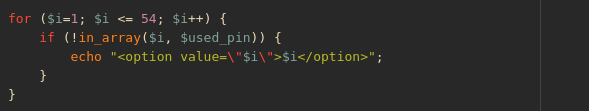
Para ello el formulario necesita que se le pase un nombre para el dispositivo.



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.



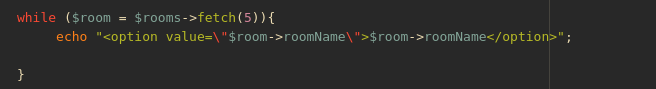
Y por último como el Arduino utilizado tiene 54 pines digitales, se hace un bucle **for** hasta 54 donde se compara si cada número está dentro del array de pines utilizados y en caso de que no esté, muestra la etiqueta **opción** con los valores que estén disponibles.



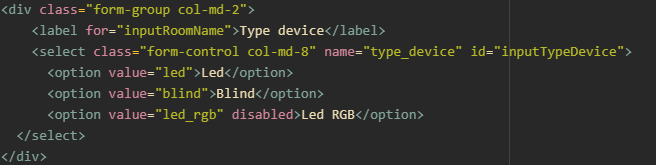
En el formulario se muestra un **select** donde se muestran las posibles habitaciones para asignar los dispositivos. Para ello se hace una consulta a la base de datos donde recoge la información de las habitaciones asignadas al usuario.



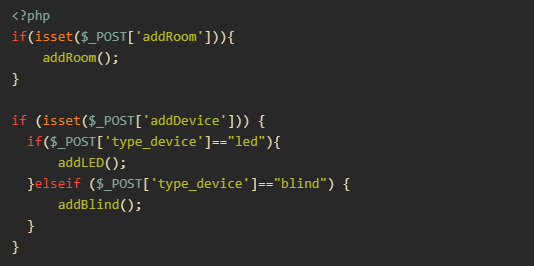
Una vez hecha la consulta se hace un bucle donde se muestra un **option** con el nombre de la habitación.



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



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.

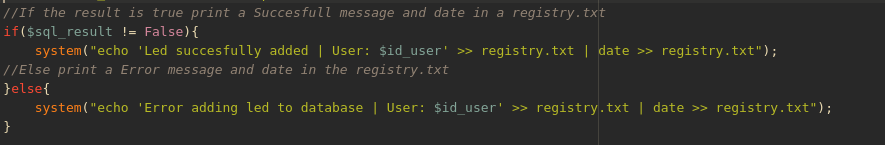


Una vez que se tienen los datos, se hace la consulta de inserción en la base de datos.





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

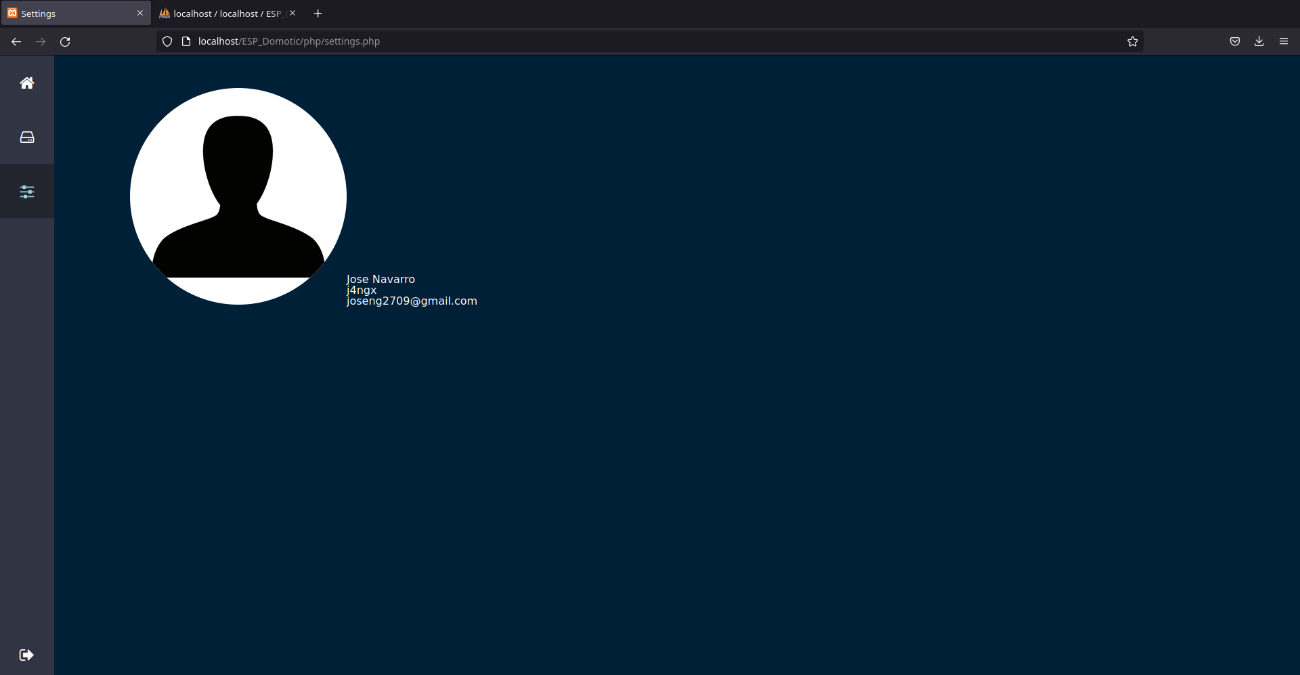


Por último, se hace una actualización de la página para que aparezcan de forma inmediata

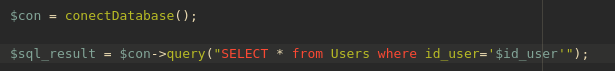


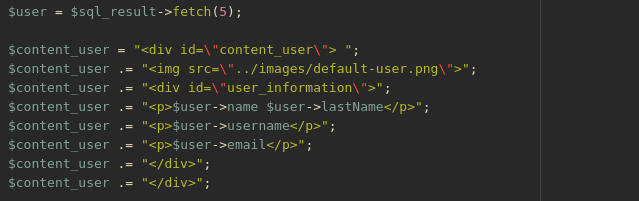
### Información del usuario

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.

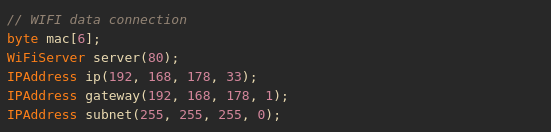




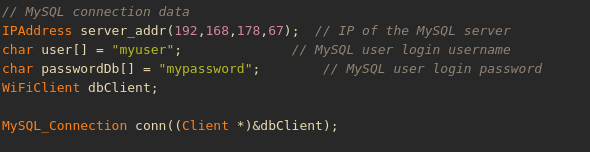
Más adelante se quiere implementar que el usuario pueda cambiar la imagen de su perfil.

### Arduino

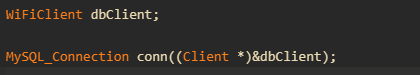
Para el desarrollo del código del Arduino se hace en el lenguaje de programación C++. La primera parte es la que se encarga de establecer la conexión con el WIFI. Para ello establecemos los valores iniciales prepara la conexión con el router que se encontraran en el fichero **setup.h.**



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.**

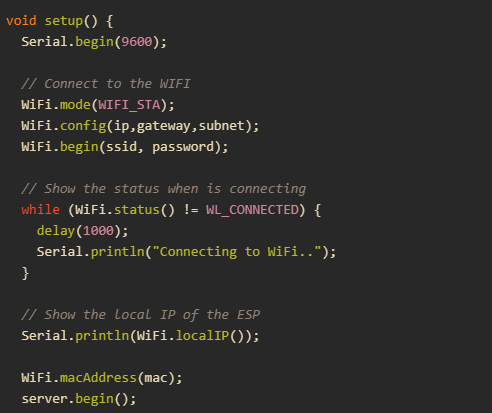


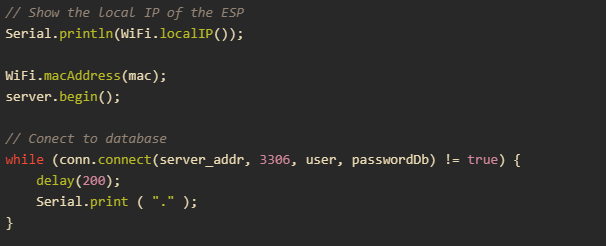
Cuando se han declarado todos los valores necesarios para el funcionamiento del Arduino, lo siguiente que les seguiría sería el fichero **main.ino** que es el fichero principal que se carga en el Arduino. Lo primero que aparece es la declaración de los objetos de conexión al router y la base de datos.



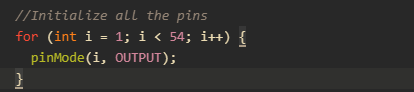
Cuando se hace un programa para que se ejecute en Arduino se tienen que diferenciar dos funciones principales y proporcionadas por **ArduinoIDE**. La primera sería **setup.** Es esta función la que primero se ejecuta, en esta función donde se configuran todos los pines de salida o entrada, o se inicializan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



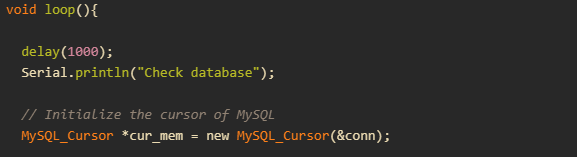
Lo siguiente que se hace en esta función sería establecer la conexión con la base de datos en función de los parámetros que anteriormente se definieron en el fichero **connectDatabase.h**.

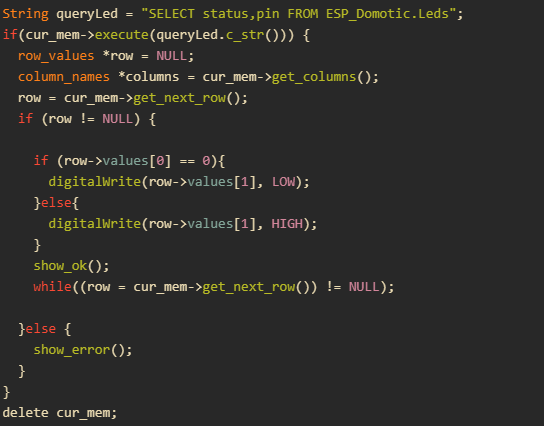
Por ultimo se tienen que inicializar los pines como “pines de salida” y no de lectura. Para ello se hace un bucle **for** desde 1 hasta 54 que es el numero total de pines que tiene el Arduino donde se establecen dichos pines



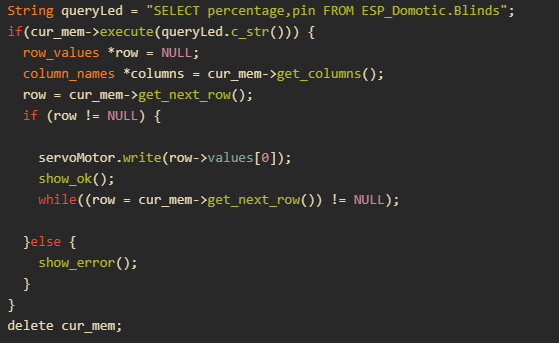
La otra función principal es **loop,** se trata de un bucle infinito que se ejecuta después de ejecutar la función **setup.** Es aquí donde se hacen las consultas a la base de datos y modifica los valores de los pines según sus correspondientes valores en las tablas de los dispositivos guardados en la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



 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.

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).



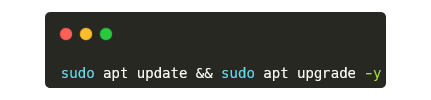
# Instalación

## Instalación Docker

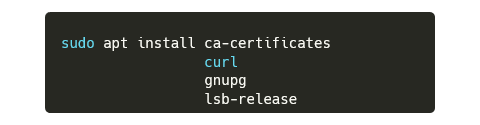
La aplicación se ha optado por alojarla en un servidor VPS con el sistema operativo Debian 11 y para alojar la aplicación se ha optado por utilizar Docker. **Docker** es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa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

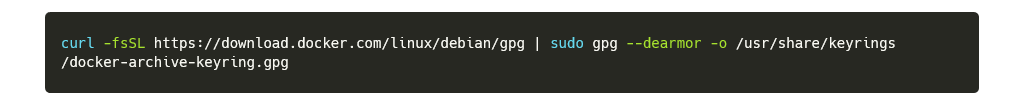
### Prerrequisitos

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

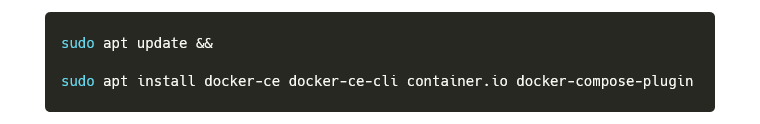


1. Una vez actualizado los repositorios, el siguiente paso es instalar los siguientes paquetes que son necesarios para comenzar a instalar Docker



1. Una vez instalados los paquetes los siguiente que se ha de hacer es añadir la clave GPG oficial de Docker
2. El siguiente paso en la instalación de Docker es añadir el repositorio necesario para instalar Docker

### Instalar Docker Engine

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

## 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.

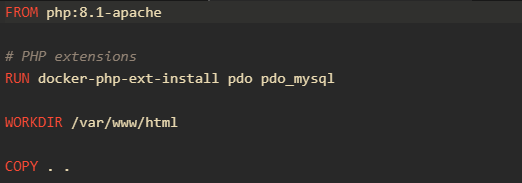


### www

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



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.

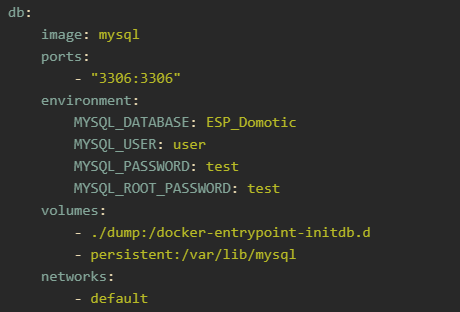


En este caso como la aplicación se ejecuta con **PHP**, la imagen base que se necesita para que pueda correr es una que contenga **PHP** y **Apache**. Además, se necesitan instalar las extensiones para utilizar **PDO** (Orientación a objetos). Por ultimo se especifica que el directorio de trabajo será **/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.

### 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.**

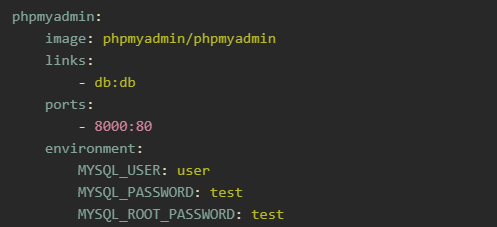


De nuevo los siguiente que se asigna son los puertos que tendrá el contenedor, para el caso de la base de datos será 3306 que es lo más común. Lo siguiente que aparece en la configuración del contenedor de la base de datos son variables de configuración del contenedor requeridas por la imagen de **mysql,**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

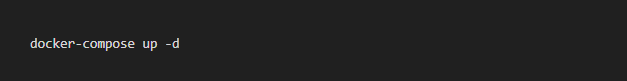
### phpmyadmin

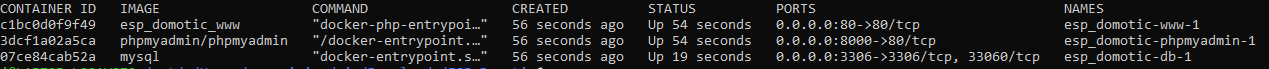
El ultimo contendor que se utiliza es para administrar la base de datos de forma gráfica. La configuración es muy similar que la dos anteriores.



### 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.



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.**

# 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>