MANUAL TÉCNICO

DESARROLLO DE UN SIMULADOR WEB PARA EL CONTROL DE RIEGO EN CULTIVOS AGRICOLAS MEDIANTE UN MICROCONTROLADOR ESP32.

AUTORES:

González Zambrano Josué Emmanuel

Obando Garces Max Alejandro

Instituto Superior Tecnológico Tena

Tecnólogo Superior de en Desarrollo de Software.

TUTOR: Ing. Oswaldo Patricio Bonifaz Vallejo Mgs.

Fecha: Julio 2025

Tena - Ecuador

2025-IS

Tabla de contenido

INDICE DE TABLAS

INDICE DE FIGURAS

Introducción

Este manual técnico está dirigido a usuarios, desarrolladores y personal técnico que requieran conocer el funcionamiento, instalación y mantenimiento del sistema de riego semiautomático desarrollado mediante una aplicación web y un microcontrolador ESP32. El sistema permite monitorear el nivel de humedad del suelo y controlar el riego desde una interfaz web accesible. Su objetivo principal es mejorar el uso del agua y reducir el esfuerzo humano en procesos agrícolas.

Requisitos del Sistema

Requisitos de Hardware

- ESP32 (con Wi-Fi)
- Sensor de humedad YL-69
- Módulo relé
- Bomba de agua (3v-5v)
- Módulo de alimentación mb 102 (3v-5v)
- Protoboard y cables
- Clable de alimentación tipo b y tipo ()

Requisitos de Software

- Navegador web moderno (Chrome, Firefox)
- Servidor local XAMPP (PHP 8, MySQL)
- Arduino IDE (para programar el ESP32)
- Visual Code para HTML , CSS ,JS y PHP
- Sublime text(opcional)
- Fritizing
- Dia
- Excalidraw

Requisitos del Usuario

- Conocimientos en creación de aplicaciones o sitios web
- Conocimiento básico en lenguaje PHP
- Conocimiento básico en Arduino
- Conocimiento en base de datos y lenguaje sql
- Conexión a red Wi-Fi

Arquitectura del Sistema

El sistema se organiza en tres capas principales:

- Front-End: Interfaces web desarrolladas con HTML, CSS y JavaScript.
- Back-End: Lógica del sistema en PHP, conectada a una base de datos MySQL.
- Hardware: ESP32 programado para controlar el riego automáticamente desde datos censados.

Se sigue un modelo cliente-servidor donde el navegador envía solicitudes al servidor web, que procesa la información y responde, activando dispositivos a través del ESP32.

El simulador está organizado en carpetas dentro de riegos (carpeta del proyecto) tenemos al mismo nivel carpetas llamadas:

- PHP
- CSS
- Imágenes
- JS
- Todos los archivos del frontend

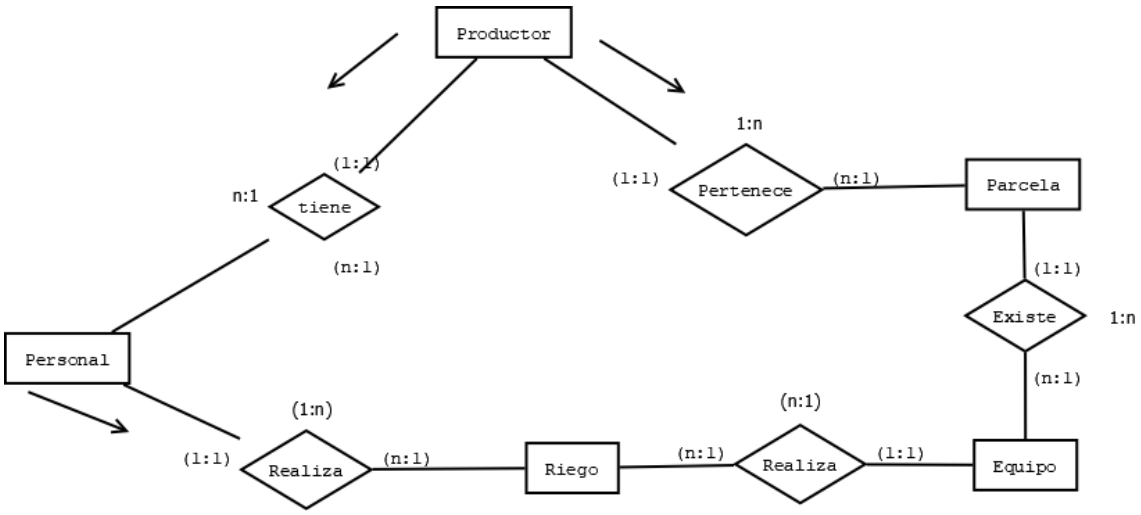
Dentro de las carpetas php y js manejamos por carpetas el nombre de las entidades por ejemplo dentro de php y js tenemos una carpeta llamada parcela y dentro de esta tenemos archivos como guardar.php y guardar.js respectivamente.

Diseño del Sistema

Diseño de la Base de Datos

Figura 1

Diagrama entidad-relación



Diccionario de datos.

Tabla 1

Diccionario

Tabla	Descripción
Productor	Contiene la informacion del productor este puede ser de tipo empresa o unitario.
Personal	Contiene la información personal del usuario, la asignación de roles su usuario y contraseña.
Parcela	Contiene la información de las parcelas(terrenos,invernaderos,fincas) el tamaño el tipo de cultivo.
Equipo	Contiene la información del equipo.
Riego	Se trata de la tabla de reportes aquí se guardada los riegos el equipo y usuario que lo hizo.

Atributos

Tabla 2

Productor

Atributo	Tipo de dato	Descripción
idPro	Int(PK)-Auto incremental	Identificador unico para el personal
tipPro	Varchar(50)	Tipo de productor (empresa o personal)
nombre	Varchar(50)	Nombre de la empresa o parcela
ciudad	Varchar(30)	Ciudad de la parcela

Tabla 3

Personal

Atributo	Tipo de dato	Descripción
idPer	Int(PK)-Auto incremental	Identificador unico para el personal
nomPer	Varchar(50)	Nombres del personal
apePer	Varchar(50)	Apellido del personal
rol	Varchar(30)	Rol asigado
corrPer	Varchar(100)	Correo del personal
usuario	Varchar(100)	Usuario
contra	Varchar(50)	Contraseña

Tabla 4

Parcela

Atributo	Tipo de dato	Descripción
idPar	Int(PK)-Auto incremental	Identificador unico para la parcela
idpro	Int(FK)	
nomPar	Varchar(75)	Nombres del terreno o parcela

tamaño	double	Tamaño del terreno en metros cuadrados
cultivo	Varchar(50)	Tipo de cultivo o terreno
foto	Varchar(255)	Ruta de la imagen

Tabla 5

Equipos

Atributo	Tipo de dato	Descripción
idPar	Int(PK)-Auto incremental	Identificador unico para la parcela
idpro	Int(FK)	
nomPar	Varchar(75)	Nombres del terreno o parcela
tamaño	double	Tamaño del terreno en metros cuadrados
cultivo	Varchar(50)	Tipo de cultivo o terreno
foto	Varchar(255)	Ruta de la imagen

Tabla 6

Riego

Atributo	Tipo de dato	Descripción
idRie	Int(PK)-Auto incremental	Identificador unico del riego
tipo	Varchar(30)	Tipo de riego (manual, auto)
fecha	date	Fecha del riego
hora	time	Hora del riego
duracion	double	Duracion del Riego
humedad	varchar(30)	Humedad registrada
idequi	Int(FK)	Llave foranea con la tabla equipo
idPer	Int(FK)	Llave foranea con la tabla personal

Descripción de llaves primarias y foráneas

Llaves primarias (PK)

Las llaves primarias permiten identificar de forma **única** cada fila de una tabla. En este sistema, cada tabla posee su propia llave primaria:

idPro: Llave primaria de la tabla *Productor*.

idPer: Llave primaria de la tabla *Personal*.

idPar: Llave primaria de la tabla *Parcela*.

idEqui: Llave primaria de la tabla *Equipo*.

idRie: Llave primaria de la tabla *Riego*.

Llaves foráneas (FK)

Las llaves foráneas permiten establecer relaciones entre las distintas tablas. Estas claves garantizan la integridad referencial en la base de datos:

idPro en *Parcela* → referencia a *Productor*.

idPar en *Equipo* → referencia a *Parcela*.

idEqui en *Riego* → referencia a *Equipo*.

idPer en *Riego* → referencia a *Personal*.

Diseño de la Interfaz Web

En cuanto al diseño de la interfaz web, se utilizaron herramientas como Excalidraw para crear prototipos de las pantallas principales, incluyendo el inicio de sesión, el panel de control y el historial de riego. Cada módulo fue esquematizado visualmente antes de su implementación, permitiendo prever el flujo de navegación y la disposición de botones, formularios e indicadores.

Por otro lado, se realizó el diseño físico del sistema, representado por un esquema de conexión electrónica en el cual se integran componentes como el ESP32, una bomba

de agua,rele y sensor de humedad. Este diseño sirvió como base para el armado de la maqueta de simulación, en la que se recreó el funcionamiento del sistema automatizado.

Como señala Luján (2018), un diseño bien estructurado permite anticipar errores de funcionalidad y facilita la comprensión del sistema por parte de los usuarios y desarrolladores, al representar de forma clara las partes que lo componen y su interacción.

Diseño de interfaces del sistema web en excalidraw.

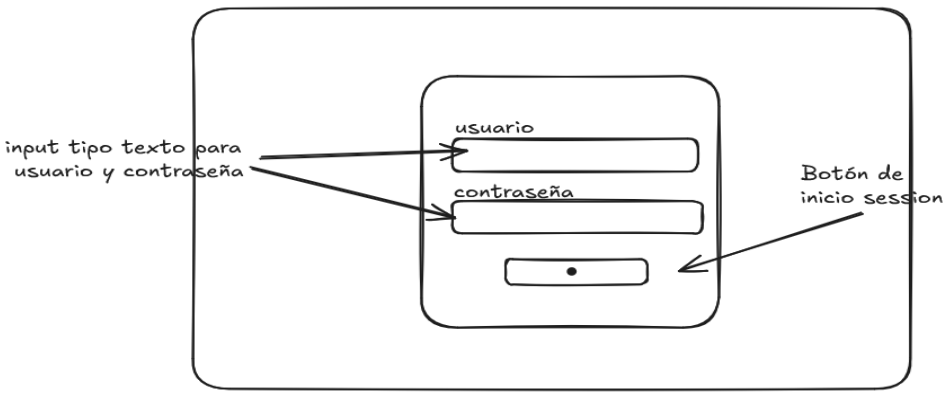
El diseño de la interfaz gráfica del sistema se centró en ofrecer una experiencia de usuario clara y accesible. Se elaboraron pantallas funcionales como el login, panel principal, gestión de ubicaciones agrícolas y control del riego. Las interfaces fueron construidas en base a un diseño limpio, utilizando colores suaves y elementos visuales que faciliten la interacción del usuario con el sistema.

Login

- **Objetivo:** Permitir el acceso de los usuarios autorizados.
- **Componentes:** Campo usuario, contraseña, botón ingresar.
- **Diseño:**

Figura 2

Login



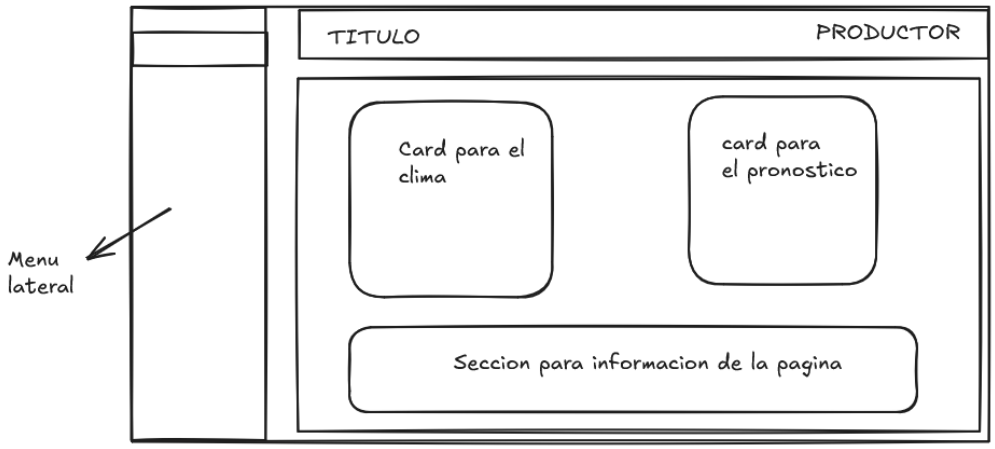
- **Tecnologías:** HTML, CSS, JS.

Panel principal

- **Objetivo:** Entrada para el usuario aquí se muestra información de la página, clima local y pronostico además del productor (empresa o agrícola independiente).
- **Componentes:** Menú, toolbar, página central con Cards para mostrar el clima y pronostico.
- **Diseño:**

Figura 3

Panel Principal



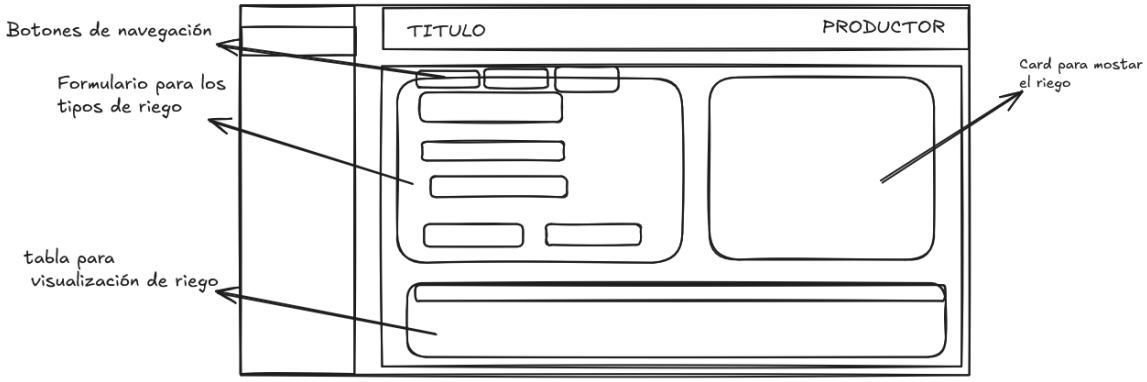
- **Tecnologías:** HTML, CSS, JS

Panel Riego Administrador

- **Objetivo:** Controlar el riego manualmente, el riego programado y el límite de humedad para el riego también se podrá visualizar el riego de manera interactiva y se podrá observar el listado mediante tabla para editar o eliminar.
- **Componentes:** Menú, toolbar, página central con Cards para navegar entre los dos tipos de riego y configuración además del control de humedad cada uno con su respectivo formulario con sus propios inputs y botones además una tabla para ver el listado de riegos
- **Diseño:**

Figura 4

Panel Riego Administrador



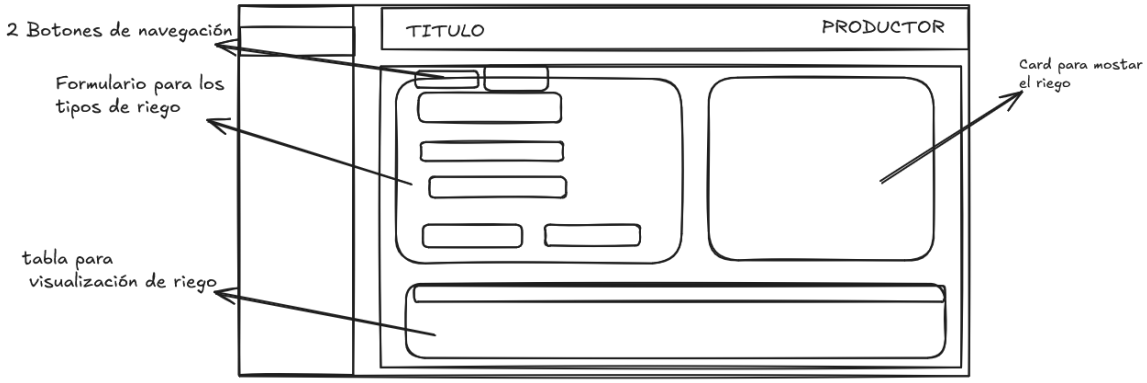
- **Tecnologías:** HTML, CSS, JS

Panel Riego usuario

- **Objetivo:** En esta sección se controla el riego de forma manual y el riego programado también se podrá visualizar el riego de manera interactiva y se podrá observar el listado mediante tabla para editar o eliminar.
- **Componentes:** Menú, toolbar, página central con Cards para navegar entre los dos tipos de riego y configuración además del control de humedad cada uno con su respectivo formulario con sus propios inputs y botones además una tabla para ver el listado de riegos
- **Diseño:**

Figura 5

Panel Riego Usuario



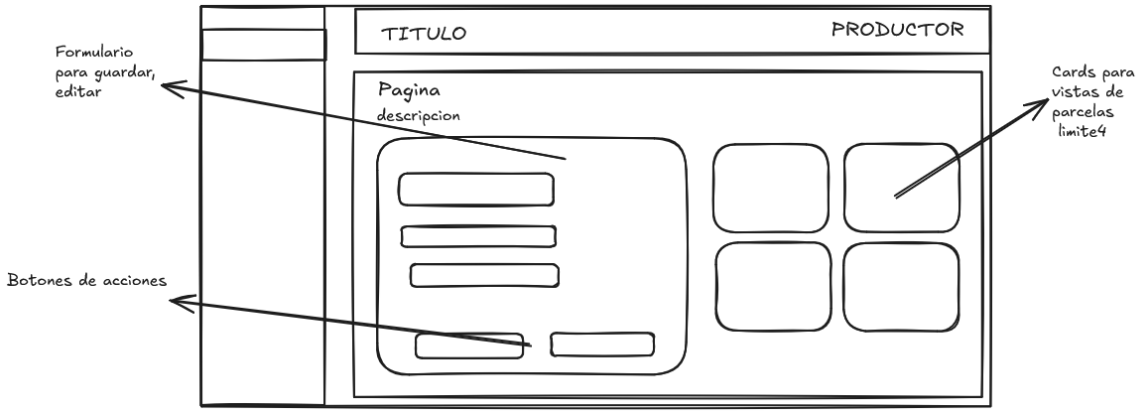
- **Tecnologías:** HTML, CSS, JS

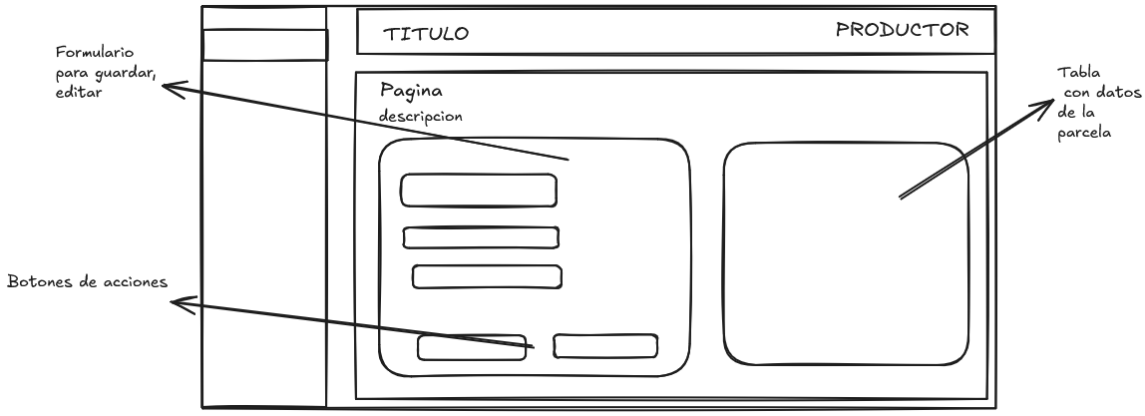
Panel Parcelas

- **Objetivo:** Agregar parcelas (terrenos, fincas) y poder editarlas o eliminar según convenga además de una tabla y cards para la visualización.
- **Componentes:** Menú, toolbar, página central dividida en dos de manera horizontal con un título, descripción y un botón para alternar. Un formulario y la otra con las cards de presentación, así como una tabla se alternarán.
- **Diseño:**

Figura 6

Panel Parcelas





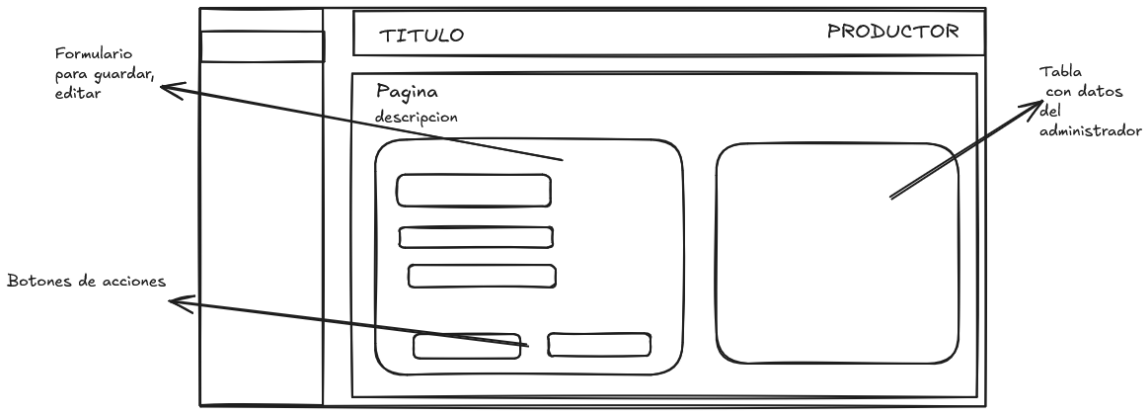
- **Tecnologías:** HTML, CSS, JS

Panel Administradores

- **Objetivo:** Agregar administradores, superusuarios poder editarlos o eliminar según convenga además de una tabla para la visualización.
- **Componentes:** Menú, toolbar, página central dividida en dos de manera horizontal con un título y una descripción. Un formulario y una tabla para la visualización.
- **Diseño:**

Figura 7

Panel Administrador



- **Tecnologías:** HTML, CSS, JS

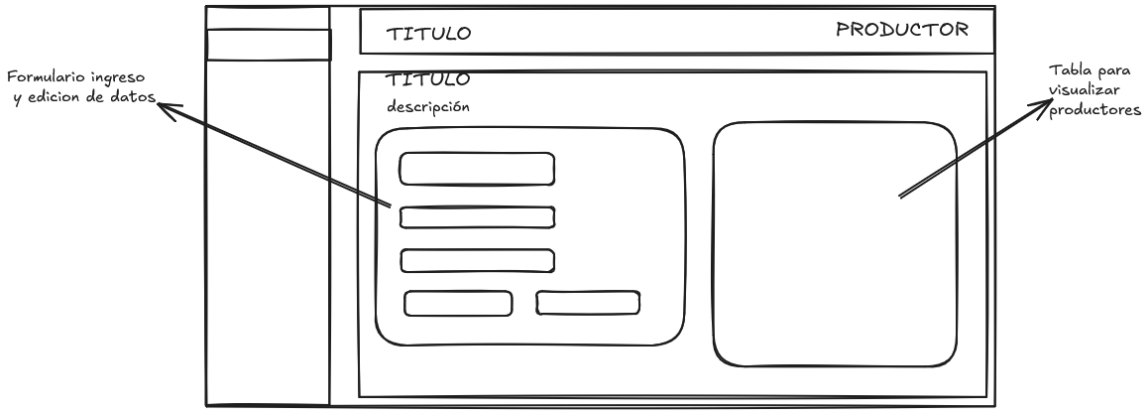
Productor

- **Objetivo:** Aquí podremos manejar el agregado del productor (empresa o agrícola independiente) editarlos o eliminar según convenga además de una tabla para la visualización.

- **Componentes:** Menú, toolbar, página central dividida en dos de manera horizontal con un título y una descripción. Un formulario y una tabla para la visualización.
- **Diseño:**

Figura 8

Productor

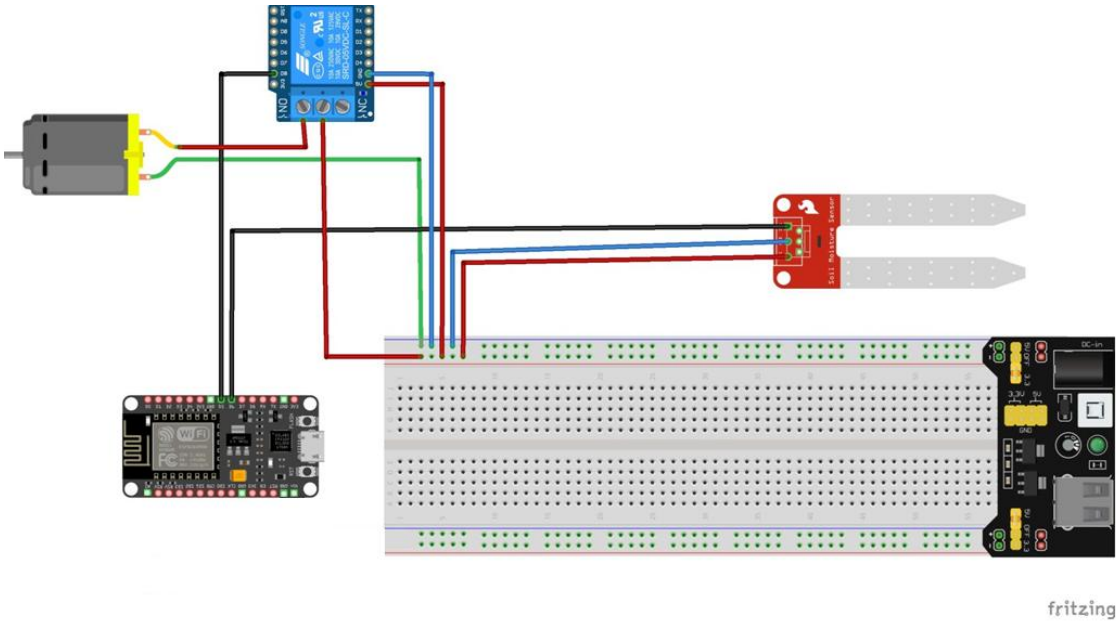


- **Tecnologías:** HTML, CSS, JS

Diseño del Sistema Electrónico

Figura 9

Esquema eléctrico o diagrama de conexión



Explicación de los componentes y conexiones físicas

ESP32 (con conectividad Wi-Fi)

El **ESP32** es el microcontrolador principal del sistema. Permite la lectura de sensores analógicos, el control de dispositivos electrónicos (como relés) y la conexión a redes inalámbricas Wi-Fi para el envío y recepción de datos.

- **Alimentación**: mediante USB o por el pin **VIN (3V)**
- **Pines utilizados**:
    - **GPIO 34**: Entrada analógica del sensor de humedad
    - **GPIO 23**: Salida digital para el control del relé

Sensor de humedad del suelo (YL-69 con módulo)

Este sensor detecta la humedad del sustrato. La lectura se realiza mediante una salida analógica, lo que permite determinar el nivel de humedad y tomar decisiones de riego automáticas.

- **Conexiones**:
  - **VCC** → 3.3V del ESP32
  - **GND** → GND del ESP32
  - **A0 (analógico)** → GPIO 34 del ESP32

Módulo relé (1 canal, 5V)

El relé funciona como un interruptor controlado electrónicamente. Se activa cuando el ESP32 envía una señal digital a través del pin asignado. Este módulo permite energizar o desenergizar la bomba de agua sin que el microcontrolador reciba directamente la carga de corriente.

- **Conexiones**:
  - **VCC** → 5V (proporcionado por el módulo MB102)
  - **GND** → Tierra común (protoboard)
  - **IN** → GPIO 23 del ESP32 (señal de control)
  - **COM y NO** → conexiones con la bomba de agua

Nota: El relé se considera "activo en bajo", por lo que se enciende al recibir una señal LOW.

Bomba de agua (3V – 5V)

La bomba se activa por medio del relé, permitiendo regar cuando se detecta un nivel de humedad por debajo del umbral definido. Opera a bajo voltaje, compatible con el sistema de alimentación auxiliar.

- Conexiones:
    - Cable positivo → Relé (NO)
    - Cable negativo → GND (fuente de alimentación)

Módulo de alimentación MB102

Este módulo permite alimentar la protoboard con 3.3V y 5V, según se requiera. En este caso, se usa para alimentar el módulo relé y la bomba.

- Entradas: alimentación por jack o puerto USB tipo B
- Salidas:
    - 5V → Línea positiva de la protoboard (para relé y bomba)
    - GND → Línea negativa de la protoboard (conectada a tierra común)

Protoboard y cables jumper

Se utilizan para montar el circuito sin necesidad de soldadura. Permite conectar todos los componentes mediante cables dupont/jumper y facilita la distribución de voltajes y señales.

- Líneas de alimentación:
    - Roja (+) → 5V del MB102
    - Azul (–) → GND común (MB102, ESP32, relé y sensor)

Cables de alimentación tipo B y tipo jack

Se utilizan para suministrar energía al módulo MB102. Pueden conectarse mediante:

- Cable USB tipo B, o
- Fuente de 7V a 12V con conector tipo jack (2.1mm)
- **Fotografías del prototipo:**

Figura 10

Prototipo

Implementación

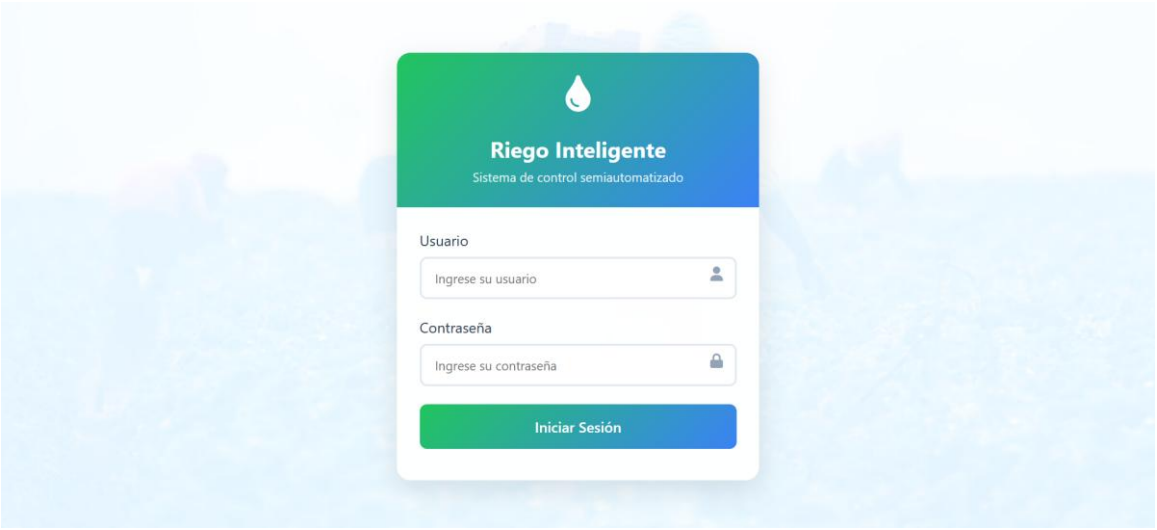
Front-End (HTML, CSS, JS).

Login

- **Diseño:**

Figura 11

Login



- **Tecnologías:** HTML, CSS, JS.

Panel principal

- **Diseño:**

Figura 12

Panel Principal

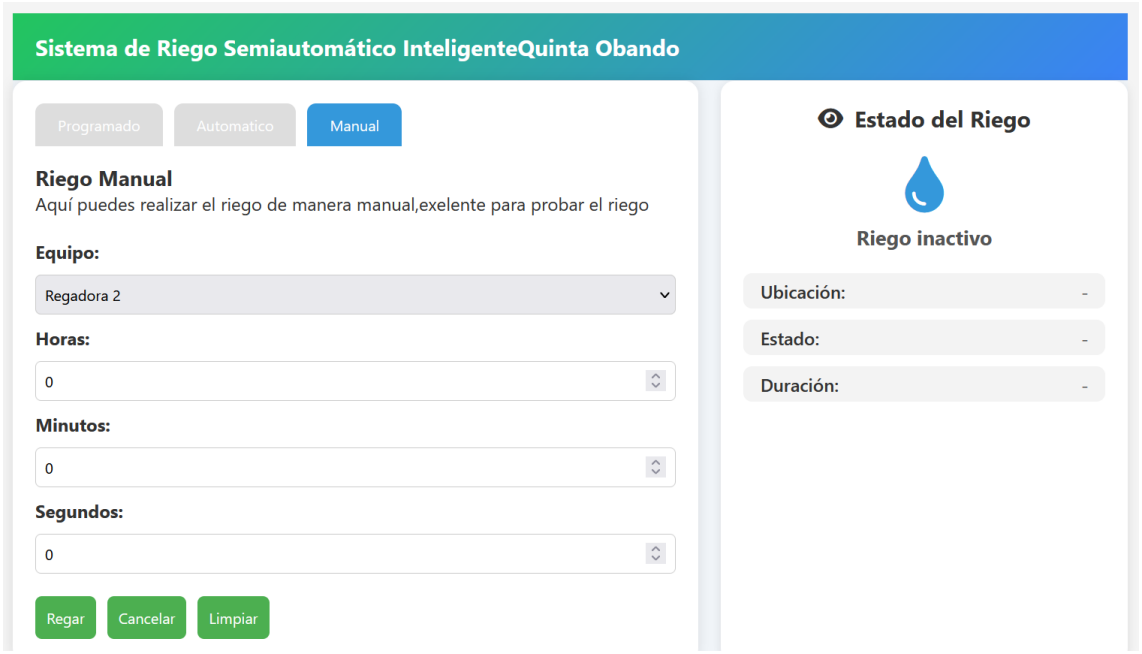
- **Tecnologías:** HTML, CSS, JS

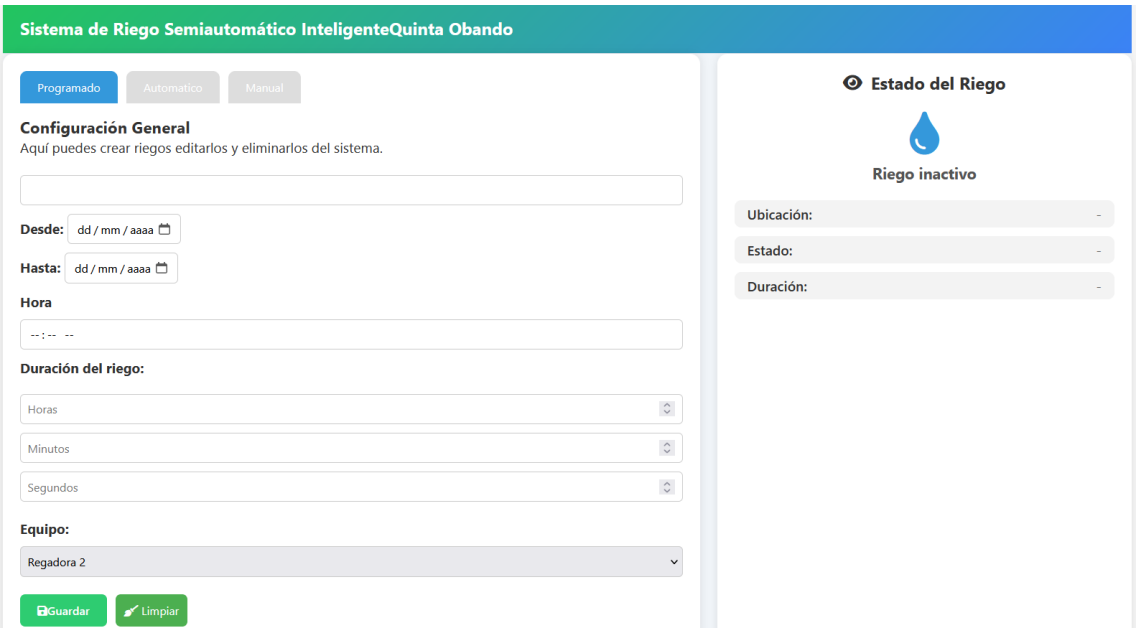
Panel Riego- Administrador

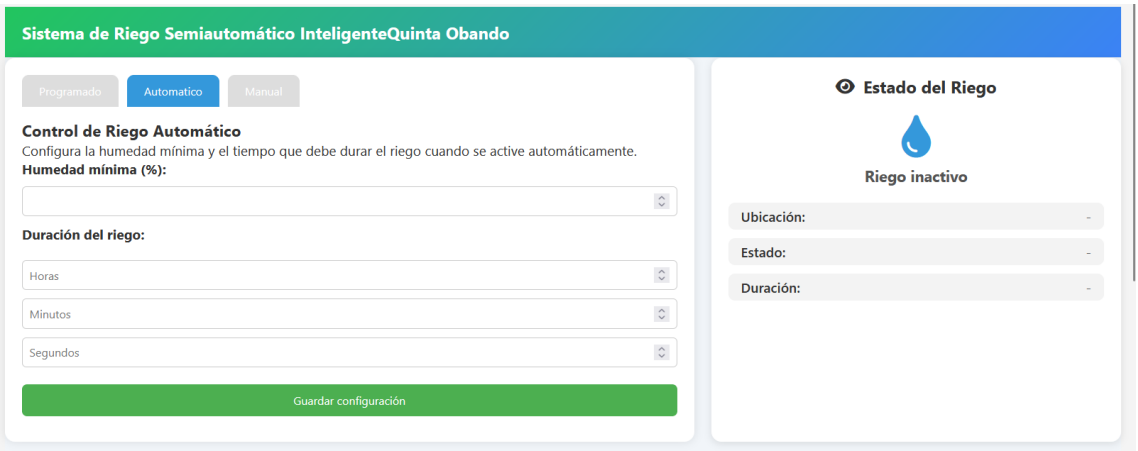
- **Diseño:**

Figura 13

Panel Riego- Administrador







ID	Tipo	Fecha	Hora	Duración	Ubicación	Usuario	Acciones
1	manual	2025-07-08	00:00:20	60:00:00	7	1	Editar Eliminar
2	manual	2025-07-08	00:00:20	60:00:00	8	1	Editar Eliminar
3	manual	2025-07-08	00:00:20	02:00:00	7	1	Editar Eliminar
4	manual	2025-07-08	00:00:21	123:00:00	9	21	Editar Eliminar
5	manual	2025-07-08	00:00:22	00:00:00	9	21	Editar Eliminar
6	programado	2025-07-09	10:00:00	02:00:00	8	1	Editar Eliminar
7	programado	2025-07-10	10:00:00	02:00:00	8	1	Editar Eliminar
8	programado	2025-07-11	10:00:00	02:00:00	8	1	Editar Eliminar

- **Tecnologías:** HTML, CSS, JS

Panel Riego-Usuario

- **Diseño:**

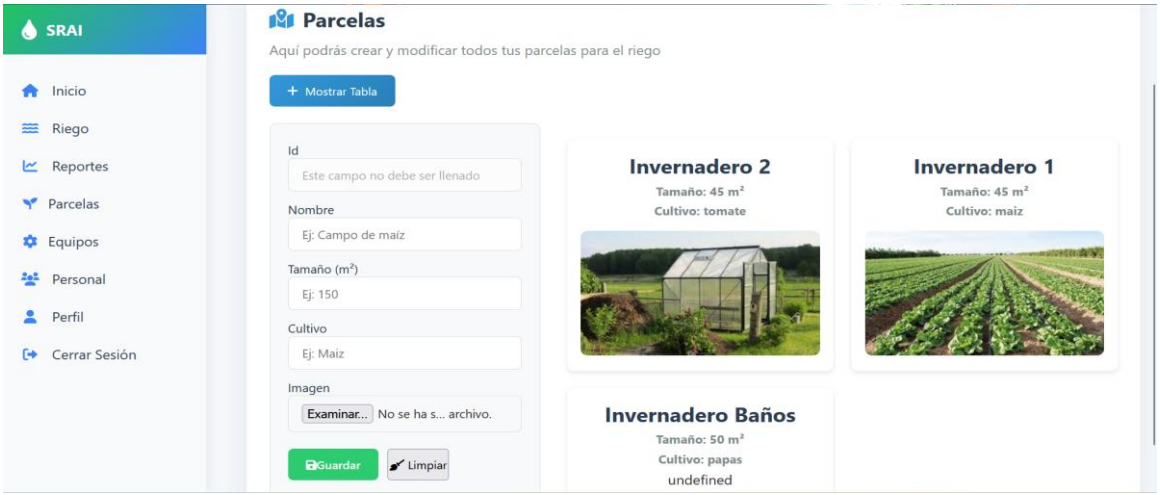
- **Tecnologías:** HTML, CSS, JS

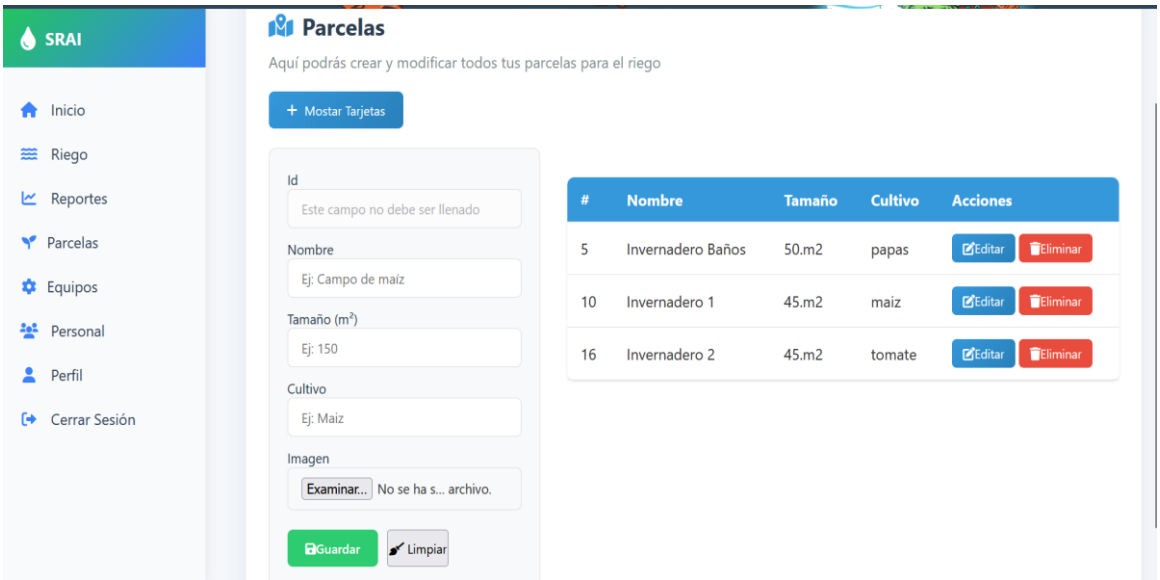
Panel Parcelas- Usuarios/Administradores

- **Diseño:**

Figura 14

Panel Parcelas- Usuarios/Administradores





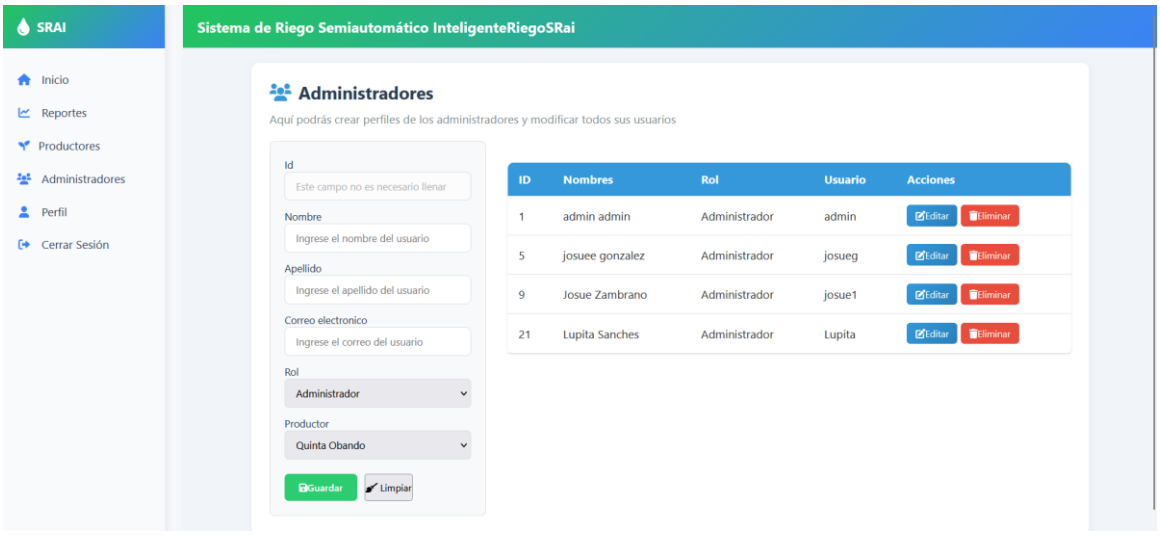
- **Tecnologías:** HTML, CSS, JS

Panel Administradores-Super Administrador

- **Diseño:**

Figura 15

Panel Administradores-Super Administrador



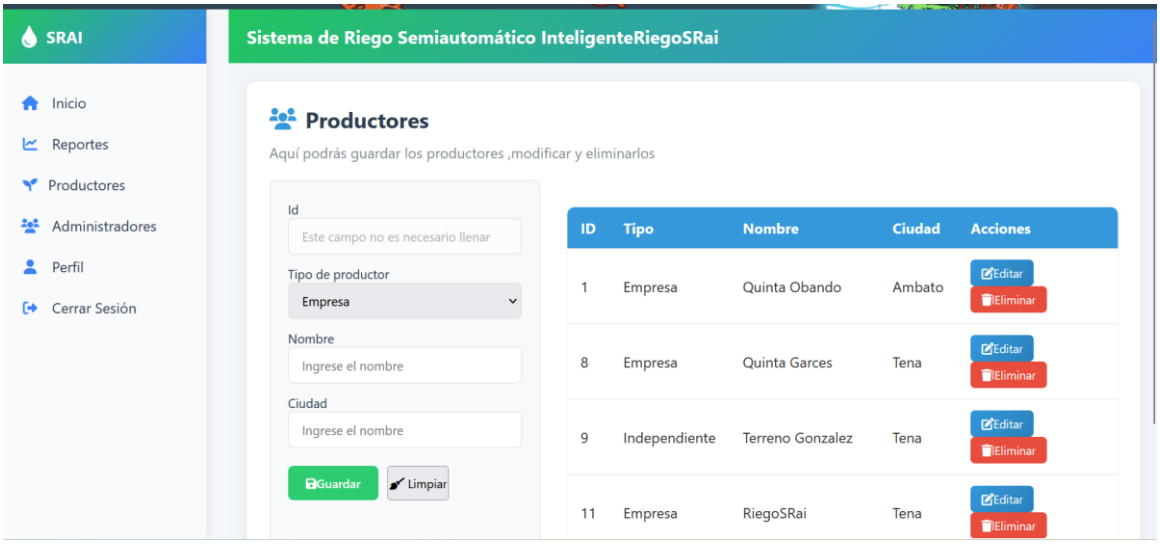
- **Tecnologías:** HTML, CSS, JS

Productor-Super administrador

- **Diseño:**

Figura 16

Productor-Super administrador



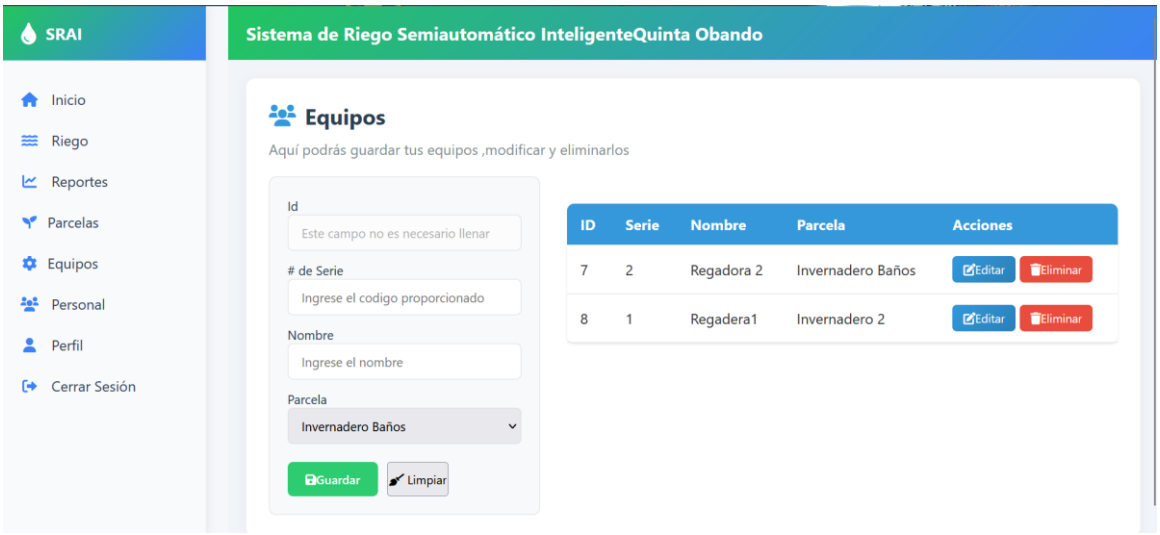
- **Tecnologías:** HTML, CSS, JS

Equipos-Usuarios/Administradores

- **Diseño:**

Figura 17

Equipos-Usuarios/Administradores



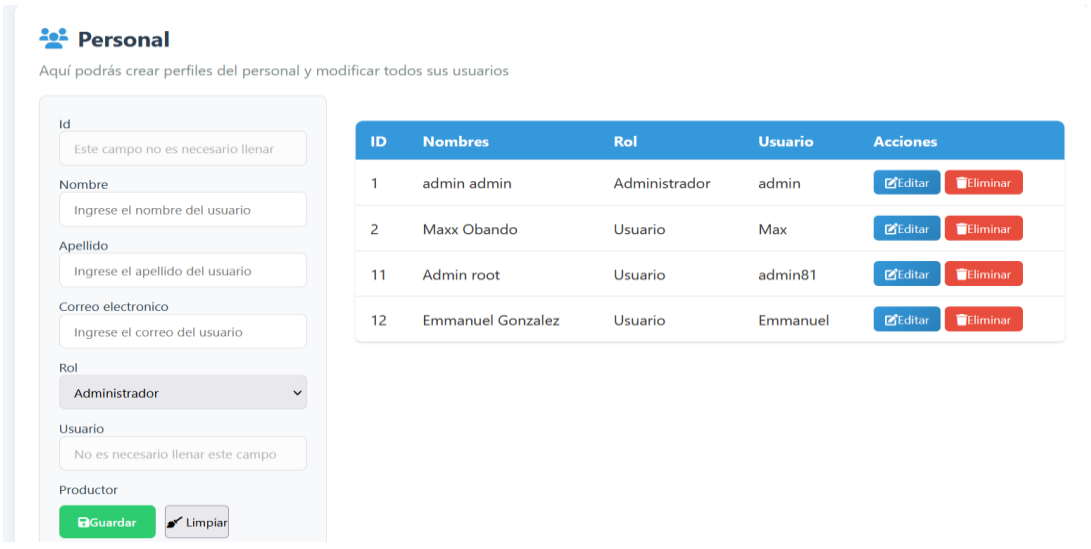
- **Tecnologías:** HTML, CSS, JS

Personal-Usuarios/Administradores

- **Diseño:**

Figura 18

Personal-Usuarios/Administradores



- **Tecnologías:** HTML, CSS, JS

Código Back-End (PHP)

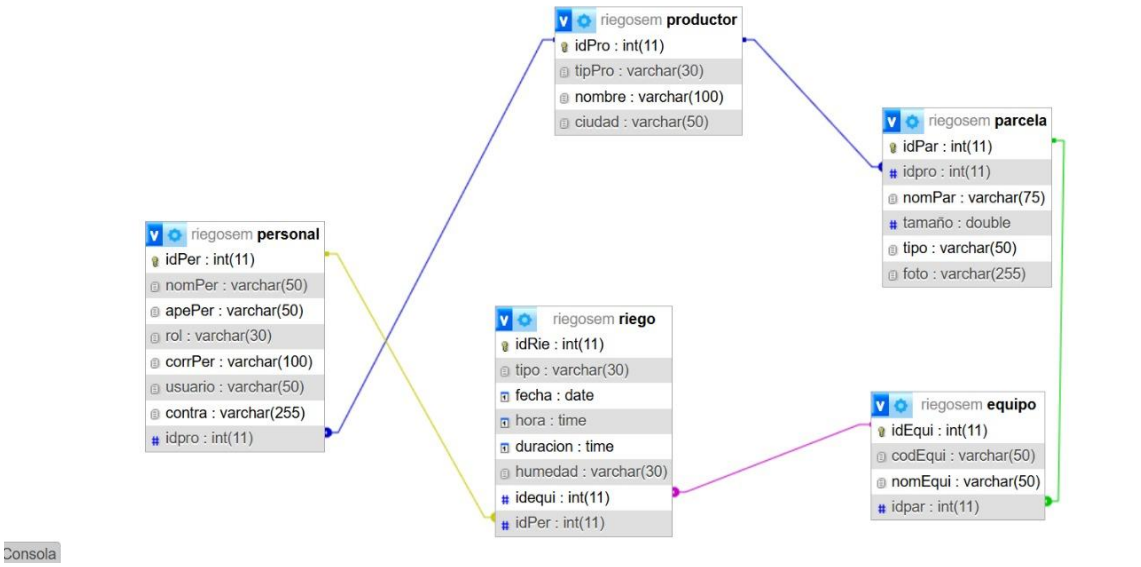
Implementación de la base de datos en phpmyadmin

En el siguiente fragmento se muestra el script de la creación de la base de datos el script completo se encuentra en el anexo 1 del manual técnico también el diagrama entidad relación extraído del propio xampp.

```
CREATE DATABASE riegosem
CREATE TABLE productor (
idPro int AUTO_INCREMENT PRIMARY KEY,
   tipPro varchar(30),
   nombre varchar (100),
   cuidad varchar(50)
);
```

Figura 19

Entidad-Relación



Conexión a la base de datos

En el siguiente código se aprecia la conexión a la base de datos el archivo ira colocado en la carpeta PHP con el nombre conexion.php y la variable será llamada $cnx,

al ser un simulador el servidor será localhost , el usuario root este ira sin contraseña además del nombre de la base de datos es el que colocamos este se ve en el anexo #1 script de la creación de la base de datos.

```php
<?php
$servidor ='localhost';
$user='root';
$contra='';
$bd='riegosem';

$cnx=mysqli_connect($servidor,$user,$contra,$bd);

?>
```

Llamado a la base de datos.

Al estar el archivo conexión.php al mismo nivel de las carpetas separadas por entidades esta debería llamarse con require como en el siguiente ejemplo, si el llamado es en el mismo nivel solo hace falta escribir ‘conexión.php’ aunque la recomendación es crear una carpeta en php para ser más ordenados.

```php
<?php
//Llamada a la conexion
require '../conexion.php';
```

Login

En el login manejos 4 procesos en total recibir los datos, consultar si existen, validar la contraseña y redirigir según el rol además aquí usaremos session_start() para que nos permita mantener los valores que obtengamos para mostarlos y usarlos en los demás módulos.

Obtener datos

Aquí obtenemos los datos y con el uso de if validamos si este llego para proceder a almacenarlos en variables ya que esto es una constante en los demás códigos de crud no se volverá a presentar.

```php
require '../conexion.php';
session_start();

if (isset($_POST['user']) && isset($_POST['contra'])) {
    $usuario = $_POST['user'];
    $contra = $_POST['contra'];
```

Consultar si existe el usuario y pedir sus datos

En este caso se hace una consulta a la tabla personal si el usuario existe se hará una consulta donde obtendremos el id el nombre y su rol esto para diferentes CRUD así como mostrar en la interfaz vamos a crear dos variables rol y hash_guardado con el fin de comparar la contraseña y validar si esta es la misma.

.

```php
 // Consulta para obtener el hash y el rol en una sola consulta
    $stmt = $cnx->prepare("SELECT idPer,contra,rol,idpro FROM personal WHERE usuario = ?");
    $stmt->bind_param("s", $usuario);
    $stmt->execute();
    $resultado = $stmt->get_result();
    //valida si la consulta devolvio un valor
    if ($resultado->num_rows === 1) {
        $fila = $resultado->fetch_assoc(); // obtener array asociativo
        $hash_guardado = $fila['contra'];
        $rol = $fila['rol'];

```

Validar contraseña y redirigir según dependiendo del rol

Aunque estos procesos son diferentes se validan en un mismo if aquí mismo guardamos los datos que sacamos en el punto anterior dependiendo del rol el sistema mandara a un index diferente en este caso tenemos 3 superadministrador, administrador y usuario.

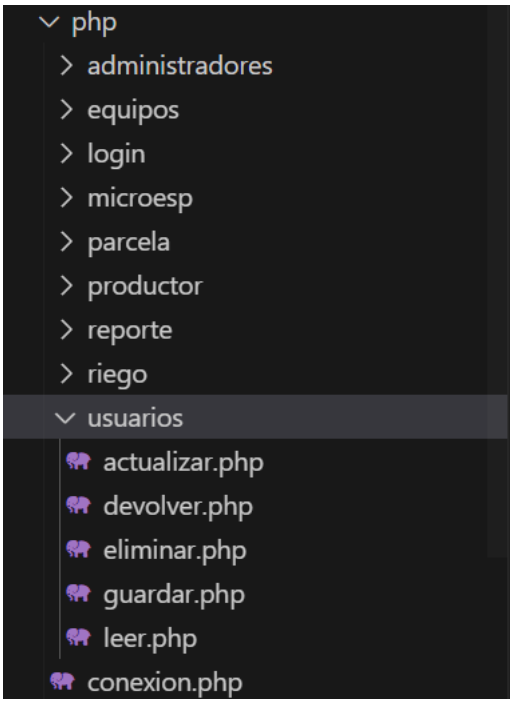
```
  // Verificar la contraseña ingresada
        if (password_verify($contra, $hash_guardado)) {
              $_SESSION['productor'] = $fila['idpro'];
              $_SESSION['rol'] = $fila['rol'];
               $_SESSION['id'] = $fila['idPer'];
            if ($rol === 'Administrador') {
                header("Location: ../../index.php");
                exit();
            } elseif ($rol=== 'Usuario') {
                header("Location: ../../indexUsu.php");
                exit();
            } else {
             header("Location: ../../indexSA.php");
                exit();
            }
```

El código completo lo encontraremos en el anexo 2 de esta guía

Crud para los módulos.

Si bien cada módulo tiene sus datos propios consultas para obtener datos de otros tipos de datos para su guardado o actualización estos siguen la misma lógica por tanto aquí se mostrará el de personal ya que este lleva más cantidad de datos a diferencia de riego, pero para el mostraremos una sección propia.

Hay que recordar que el manejo de carpetas que se presento es respetando cada entidad adentro de nada una se creara leer, guardar, actualizar, eliminar, devolver y otros según sean necesarios con el fin de tener organización.



Leer

Leer como se llamó este archivo php tiene el objetivo de hacer una consulta a la base de datos en este caso a personal estos datos se enviarán al js con el mismo nombre y estructura esto se verá en el siguiente punto llamado Java Script , en este archivo llamamos a la conexión usamos session_start para un filtrado, hacemos la consulta comprábamos si devolvió datos y con el uso de un while mostramos los datos que necesitamos imprimiendo con un echo código html celdas de una tabla y dentro agregamos dos botones de acción editar y eliminar.

```
session_start();

$leer = mysqli_query($cnx, "SELECT idPer,nomPer,apePer,rol,usuario
FROM personal WHERE idpro='".$_SESSION['productor']."' ");

if (mysqli_num_rows($leer) > 0) {
    while ($row = mysqli_fetch_assoc($leer)) {
        echo "<tr><td>{$row['idPer']}</td><td>{$row['nomPer']}
{$row['apePer']}</td>
            <td>{$row['rol']}</td><td>{$row['usuario']}</td>
```

```
            <td><button class='editar-btn' data-
id='{$row['idPer']}' type='button'><i class='fas fa-
edit'></i>Editar</button>
            <button class='eliminar-btn' data-id='{$row['idPer']}'
type='button' ><i class='fas fa-trash'></i>Eliminar</button></td>
            </tr>
        ";
    }
```

En el anexo 3 se muestra el código completo donde se ve el else y la llamada a la conexión.

Guardar

Por el nombre se supone que no hay que imaginar mucho que hace este archivo php siguiendo la misma lógica llamamos a la conexión obtenemos los datos y los almacenamos en variables este archivo tiene una peculiaridad y es la trasformación de la contraseña para encriptarla con el fin de que los datos no se vean comprometidos este es propio de esta tabla y solo aquí haremos esto así como la creación del usuario este es en base al nombre así que primero consultamos si ya existe si es el caso este se crear en base a un número randomico del 1 al 1000 con el fin de que este no se duplique.

```
 $contra = password_hash($usuario, PASSWORD_DEFAULT);//la
contraseña se encripta

        //crear usuarios unicos
    while (mysqli_num_rows(mysqli_query($cnx, "SELECT usuario FROM
personal WHERE usuario='$usuario'")) > 0) {
        $numero = rand(1, 1000); // Número entre 1 y 1000
        $usuario = strtolower($usuario . $numero);
    }
```

Siguiente a esto procedemos a guardar los datos usamos un prepare con el fin de evitar inyecciones sql si el guardado es exitoso devuelve “1” si no el error esto va al js.

```php
 //Guardar datos
      $stmt = $cnx->prepare("INSERT INTO personal VALUES(null,?,?,?,?,?,?,?) ");
     $stmt->bind_param("ssssssi",$nombre,$apellido,$rol,$correo,$usuario,$contra,$productor);
         if ( $stmt->execute()) {
             echo "1"; // Devuelve un entero si se guardo correctmente
         }else {
              echo "Error SQL: " . $stmt->error;
         }
         $stmt->close();
```

Devolver

Aunque su nombre es algo confuso devolver es esencial para nuestro proyecto si más bien no es ninguna función del CRUD si esta enlazada a uno directamente actualizar, entonces ¿Que hace? Mediante los botones de acción al presionar editar los datos regresan mediante una consulta al formulario con el fin de cambiar estos datos para esto hacemos una consulta solo tomando el id y devolvemos con json

```php
//obtener id para devolver datos
if (isset($_GET['id'])) {
    $id = $_GET['id'];

    $query = mysqli_query($cnx, "SELECT * FROM personal WHERE idPer = '$id'");
    if ($row = mysqli_fetch_assoc($query)) {
        echo json_encode([
            'success' => true,
            'personal' => $row
        ]);
```

En el anexo 4 devolver se muestra de manera completa.

Actualizar

Como su nombre lo dice actualizar hace honor a su nombre actualiza los datos y devuelve una booleano un true o el error en el caso de que los datos no fueron actualizados obviamente antes llamamos a la conexión y obtenemos los datos

```
//actualizar datos
     $actualizar=mysqli_query($cnx,"UPDATE personal SET
    nomPer='$nombre',
    apePer='$apellido',
    rol='$rol',
    corrPer='$correo'
    WHERE idPer='$id'");
   if ($actualizar) {
       echo json_encode(['success' => true]);
    } else {
       echo json_encode(['success' => false, 'message' => 'No se
pudo actualizar']);
    }
```

En el anexo 5 actualizar se encuentra el código completo

Eliminar

Aquí ya no tenemos mucha ciencia eliminar elimina un dato tomando desde el apartado acciones de la tabla con el id.

```
$eliminar=mysqli_query($cnx,"DELETE FROM personal WHERE
idPer='$id'");
     if ($eliminar) {
        echo "1";
```

En el anexo 5 eliminar se muestra todo el código para la compresión si hace falta.

Otras acciones importantes

Aquí tenemos otras funciones importantes como por ejemplo filtrar datos para ponerlos en select cerrar sesión o hacer búsquedas en reportes como venimos trabajando estos se verán en el anexo 7,8 y 9 de manera completa respectivamente.

Filtrar parcelas

En este filtramos para mostrar parcelas usando session start e imprimiendo directamente codigo html con un while.

```
$consulta=mysqli_query($cnx,"SELECT idPar, nomPar FROM parcela
WHERE idpro='".$_SESSION['productor']."' ");
if (mysqli_num_rows($consulta)>0) {
    while ($row=mysqli_fetch_assoc($consulta)) {

        echo '<option value="' . $row['idPar'] . '">' .
$row['nomPar'] . '</option>';
    }
```

Buscar riegos

Aquí buscamos los riegos para el apartado reporte mediante el uso de una vista y usamos el mismo para la búsqueda es decir reutilizamos el mismo.

```
$sql = "
SELECT idRie,
tipo,
fecha,
hora,
duracion,
humedad,
parcela.nomPar as parcela ,
personal.nomPer as personal
from riego
INNER JOIN equipo ON
riego.idequi=equipo.idEqui
INNER JOIN parcela ON
equipo.idpar=parcela.idPar
INNER JOIN personal ON
riego.idusu=personal.idPer
INNER JOIN productor ON
parcela.idpro=productor.idPro
where productor.idPro='".$_SESSION['productor']."'
        ORDER BY fecha DESC, hora DESC";

$result = mysqli_query($cnx, $sql);

$reportes = [];
while ($row = $result->fetch_assoc()) {
    $reportes[] = $row;
}
```

```
•
•   echo json_encode($reportes);
•
```

Cerrar sesión

No hay que buscarle mucha logica a este archivo hace lo mismo que su nombre destruye y nos lleva al login serrando sesión.

```
•   <?php
•   session_start();
•   session_unset();           // Limpia todas las variables de la
    sesión
•   session_destroy();         // Destruye la sesión del servidor
•   header('Location: ../../login.php'); // envia nuevamente al login
•   exit();
•   ?>
```

Manejo del riego y envió de datos a esp32

Ya vimos como guardar datos en la base de datos entonces ese tema no lo tocaremos aquí lo que nosotros vamos a ver es como se envía esos datos al esp32 o como hace el microprocesador para guardarlos en la base de datos para esto recapitulemos.

Primero creamos una carpeta en php llamda microesp y dentro de esta manejaremos el guardado de los tipos de riego.

El módulo riego posee un tab con 3 opciones estas son Riego programado, automático y riego manual.

- **Programado**

Aquí veremos un guardado un poco más extenso debido a la elección del usuario si es por dia o es por semana dentro de un while.

```
    // Validación de fechas

    if ($inicio > $fin) {
        $response['message'] = 'La fecha de inicio no puede ser mayor que la fecha final.';
        echo json_encode($response);
        exit;
    }

    // Transformar a HH:MM:SS
    $duracion = sprintf('%02d:%02d:%02d', $h, $m, $s);

    // Preparar el insert
    $stmt = $cnx->prepare("INSERT INTO riego VALUES (null,'programado', ?, ?, ?, ?, ?, ?)");

    if (!$stmt) {
        $response['message'] = 'Error al preparar consulta: ' . $cnx->error;
        echo json_encode($response);
        exit;
    }

    // Crear fechas desde inicio a fin
    $fechas = [];
    $actual = new DateTime($inicio);
    $fechaFin = new DateTime($fin);
    $fechaFin->modify('+1 day'); // para incluir el día final

    while ($actual < $fechaFin) {
        $fechas[] = $actual->format('Y-m-d');
        $actual->modify('+1 day');
    }

    // Ejecutar insert por cada fecha
    foreach ($fechas as $fecha) {
        $stmt->bind_param("sssiii", $fecha, $hora, $duracion, $humedad, $equipo, $usuario);
        if (!$stmt->execute()) {
            $response['message'] = "Error al guardar riego del $fecha: " . $stmt->error;
            echo json_encode($response);
            exit;
        }
    }
```

- **Automático**

Aquí el usuario podrá elegir el umbral del riego es decir si el usuario coloca que la humedad del suelo debe ser mínimo 30 y el sensor conectado al esp32 devuelve una humedad de 29%, 28% etc se regara por el tiempo que el usuario eligió para que el suelo siga con la humedad elevada y posterior se guardara en la base de datos para que el usuario lleve el control del riego en sus reportes.

Una vez comprendido hacemos lo que ya hemos hecho en otros módulos con pasos extras recibimos los datos validamos que no estén vacíos hacemos una consulta mediante el dato que envía el esp32 del equipo, guardamos fecha y hora actual para el registro y se hace el guardado a la base .

```
$fecha = date("Y-m-d");
$hora = date("H:i:s");

// Buscar el idEqui en la tabla equipo
$res = mysqli_query($cnx,"SELECT idEqui FROM equipo WHERE codEqui='".$equipo."' LIMIT 1");
if ($fila = mysqli_fetch_assoc($res)) {
    $idequi = intval($fila['idEqui']);
} else {
    echo "Error: No se encontró el equipo con codEqui=".$equipo;
    exit;
}
```

¿Pero? Donde se encuentra la humedad y tiempo que el usuario eligió bueno esto se presenta en el siguiente código en el archivo llamado prueba aquí creamos un txt al mismo nivel llamado confg_auto.txt el mismo que el esp32 guarda en su interior esto se verá en el código del mismo.

```
 if ($humedad !== null && $tiempo !== null && $humedad > 0 && $tiempo > 0) {
        $datos = [
            "humedad_minima" => $humedad,
```

```
            "tiempo_riego" => $tiempo
        ];

        $archivo = "config_auto.txt";
        file_put_contents($archivo, json_encode($datos));
        echo "Configuración guardada correctamente.";

```

- **Manual**

El riego manual como su propio nombre lo dice al hacer un riego directo este se guarda puede ser usado para pruebas o demostraciones en php recibe crea y guarda una variable para hora y fecha actual, guarda en la base de datos y renvía a un txt que debemos crear estado_riego.txt donde envía el estado On y el tiempo este recibidos en la base de datos.

- Estado de riego tendrá esto

```
    {"estado":"ON","tiempo":14400000}
```

- Riego manual tendrá esto

```
 $datos = [
        "estado" => $estado,
        "tiempo" => intval($tiempo)
    ];

    //Guardar en la base
    $stmt=$cnx->prepare("INSERT INTO riego
VALUES(null,'manual',?,?,?,?,?,?)") ;
    $stmt-
>bind_param('siisii',$fecha,$hora,$tiempo,$humedad,$equipo,$idusu);
     if ( $stmt->execute()) {
            echo "1"; // Devuelve un entero si se guardo
correctmente
        }else {
             echo "Error SQL: " . $stmt->error;
        }

    file_put_contents("estado_riego.txt", json_encode($datos));
    echo "Estado actualizado: " . json_encode($datos);
```

Otros archivos importantes.

Cancelar riego

Aquí podremos cancelar un riego manual que se este ejecutando este se enviara al txt el mismo que el esp32 lee para saber

```php
<?php
// Cancela el riego cambiando estado a OFF
$datos = [
    "estado" => "OFF",
    "tiempo" => 0
];
file_put_contents("estado_riego.txt", json_encode($datos));
echo "Riego cancelado correctamente.";
?>
```

Borrar estado

Este se encarga de cambiar el estado del riego al esp32 cambiándolo con off para evitar que se riego ilimitadamente igual que los casos anteriores trabaja con el txt estado_riego

```php
<?php
file_put_contents('estado_riego.txt', "OFF");
echo "Estado borrado";
?>
```

Devolver humedad

Este archivo nos permite devolver la humedad para mostrara consulta y la guarda para cuando no está cargando para poder verla en el index.

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (isset($_POST['humedad'])) {
        $humedad = intval($_POST['humedad']);

        // Guardar en archivo simple (puedes usar BD si lo deseas)
        file_put_contents("ultimo_valor.txt", $humedad);
```

```
•
•           echo json_encode(['humedad' => $humedad]);
•       } else {
•           echo json_encode(['error' => 'No se recibió el valor de
    humedad']);
•       }
•   } elseif ($_SERVER['REQUEST_METHOD'] === 'GET') {
•       // Leer valor desde el archivo
•       if (file_exists("ultimo_valor.txt")) {
•           $humedad = intval(file_get_contents("ultimo_valor.txt"));
•           echo json_encode(['humedad' => $humedad]);
```

Reportes

En esta sección se generan y visualizan informes detallados sobre el estado y el funcionamiento del sistema de riego. Los reportes incluyen datos históricos de humedad del suelo, horarios y duraciones de riego, así como estadísticas que permiten analizar el consumo de agua y la eficiencia del riego. Esta información ayuda al usuario a tomar decisiones informadas para optimizar el uso de recursos y mejorar el cuidado de los cultivos. Además, los reportes pueden exportarse en formatos accesibles para su impresión o almacenamiento.

```
•   // Generar PDF
•   $dompdf = new Dompdf();
•   $dompdf->set_option('isRemoteEnabled', true);
•   $dompdf->loadHtml($html);
•   $dompdf->setPaper('A4', 'landscape');
•   $dompdf->render();
•   $dompdf->stream("reporte_riegos_{$mesSeleccionado}.pdf",
    ["Attachment" => true]);
•   ?>
```

Código del esp32 (C++)

Aquí vamos a ver el codigo del esp32 para cada función que ya vimos antes desde su conexión a internet la clave, la ip de la computadora ya que al ser simulador no está en un servidor.

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include "time.h"


const char* ssid = "departamentotena"; //Red de internet
const char* password = "#megustaTena";  //Contraseña
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = -5 * 3600;   // Ecuador: GMT-5
const int daylightOffset_sec = 0;       // Sin horario de verano

// Pines
const int pinSensor = 34;        // Analógico del sensor YL-69
const int pinRiego = 23;         // Pin que controla el relé
// Pines para LEDs
const int LED_VERDE = 18;   // LED para riego activo
const int LED_ROJO = 19;    // LED para riego inactivo
const int LED_WIFI = 2;     // LED interno ESP32 para WiFi

bool riegoActivo = false;     // Control para que no riegue
repetido
unsigned long finRiego = 0;   // Marca cuándo termina el riego
programado

// Umbral de humedad (%)
const int umbralHumedad = 10;

void verificarRiegoProgramado() {
  if (riegoActivo) {
    // Si ya está regando, chequeamos si terminó el tiempo
    if (millis() >= finRiego) {
      digitalWrite(pinRiego, HIGH); // Apagar relé
      digitalWrite(LED_VERDE, LOW);   // LED verde apaga
      digitalWrite(LED_ROJO, HIGH);   // LED rojo prende
      riegoActivo = false;
      Serial.println("🔲 Riego programado finalizado.");
    }
    return; // Mientras riega no vuelve a hacer request
  }
```

Código Front-End (HTML, CSS, JS)

HTML

Para el código de las interfaces no vamos a profundizar ya que es código html en su totalidad y css presentamos el ejemplo del índex y en anexo 13 el css

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="css/login.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
</head>
<body>
    <div class="login-container">
        <div class="login-header">
            <div class="logo">
                <i class="fas fa-tint" style="font-size: 40px; margin-bottom: 10px;"></i>
            </div>
            <h1>Riego Inteligente</h1>
            <p>Sistema de control semiautomatizado</p>
        </div>
        //formulario de inicio de session
        <div class="login-body">
            <form  method="POST"  action="php/login/ingresar.php" id="formlogin" >
                <div class="input-group">
                    <label for="username">Usuario</label>
                    <input type="text" id="user" name="user" placeholder="Ingrese su usuario" required>
                    <i class="fas fa-user"></i>
                </div>

                <div class="input-group">
                    <label for="password">Contraseña</label>
                    <input type="password" id="contra" name="contra" placeholder="Ingrese su contraseña" required>
                    <i class="fas fa-lock"></i>
                </div>

                <button type="submit" class="login-button">Iniciar Sesión</button>
                <div id="mensajeError" style="color: red; margin-top: 10px;"></div>
            </form>
        </div>
    </div>
</body>
</html>
```

JavaScrip

En este apartado igual que en el crud veremos solo los códigos de parcelas que sirven para guadar editar devolver leer y eliminar estos serán creados de igual manera que php dentro de la carpeta js y dentro de sus entidades en este caso parcela.

Los scripts deben ir al final del html antes de cerrar el cuerpo (body) recomendamos colocar leer primero.

```
<!--scrpits -->
<script src="js/parcela/leer.js" ></script>
<script src="js/mostrar.js" ></script>
<script src="js/parcela/guardar.js" ></script>
<script src="js/parcela/devolver.js" ></script>
<script src="js/parcela/eliminar.js" ></script>

```

Guardar

Aquí recibimos los datos de la interfaz y los enviamos mediante fech al php tenemos validaciones de datos vacíos y el model que se mostrara para los mansajes después de guardar la tabla se actualiza asi también en este caso único los tabs .

```
  fetch('php/parcela/guardar.php', {
        method: 'POST',
        body: datos
      })
      .then(response => response.text())
      .then(data => {
   if (data == "1") {
     Swal.fire({
       title: 'Éxito',
       text: 'Parcela guardada correctamente',
       icon: 'success',
       confirmButtonText: 'Aceptar'
     });
     cargarCards();
     leerpar();
     formulario.reset();
```

Leer

Esta función toma los datos de php y los regresa al cuerpo de la tabla en html.

```
function leerpar() {
  fetch('php/parcela/leer.php')
    .then(response => response.text())
    .then(data => {
      document.querySelector("#tabla-par tbody").innerHTML = data;
    })
    .catch(error => {
      console.error("Error al cargar los datos:", error);
    });
}

```

Devolver

Este js devuelve directamente los valores al formulario oculta guardar y muestra actualizar además cambia el texto de limpiar a cancelar.

```
 const formulario = document.getElementById('formpar');

document.addEventListener("click", async (e) => {
  if (e.target.classList.contains("editar-btn")) {
    const id = e.target.getAttribute("data-id");

    try {
      const res = await fetch("php/parcela/devolver.php?id=" + id);
      const datos = await res.json();

      if (datos.success) {
        document.getElementById("id").value = datos.parcela.idPar;
        document.getElementById("productor").value =
datos.parcela.idpro;
          document.getElementById("nombre").value =
datos.parcela.nomPar;
        document.getElementById("tamano").value =
datos.parcela.tamaño;
        document.getElementById("cultivo").value =
datos.parcela.cultivo;
       // document.getElementById("img").value =
datos.parcela.foto;
```

```
        document.getElementById("btnActualizar").hidden=false;
         document.getElementById("btnGuardar").hidden=true;

         const btnLimpiar = document.getElementById("limpiar");
         btnLimpiar.textContent = "Cancelar";

         btnLimpiar.onclick = function () {
           formulario.reset();
          btnLimpiar.innerHTML = '<i class="fas fa-broom"></i>
Limpiar';
          document.getElementById("btnActualizar").hidden=true;
         document.getElementById("btnGuardar").hidden=false;
         };

       } else {
         alert("No se encontraron datos.");
       }
     } catch (error) {
       console.error("Error:", error);
       alert("Error al obtener los datos.");
     }
   }
});
```

Actualizar

Este archivo es sencillo usa la misma lógica que guardar recibe los datos y los envía a actualizar php.

```
fetch("php/parcela/actualizar.php", {
            method: "POST",
            body: datos //  no se pone headers, el navegador los
gestiona
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                Swal.fire({
                    title: 'Éxito',
                    text: 'Parcela actualizada correctamente',
                    icon: 'success',
                    confirmButtonText: 'Aceptar'
                });
                cargarCards();
                leerpar();
                document.getElementById("formpar").reset();
                document.getElementById("btnGuardar").hidden =
false;
```

```
                    document.getElementById("btnActualizar").hidden =
true;
                    document.getElementById("formpar").reset();
                     document.getElementById("imgparcela").hidden =
true;
                    document.getElementById("limpiar").textContent =
"Limpiar";
```

Eliminar

No hay mucho que explicar al igual que los anteriores elimina actualiza la tabla y muestra los mensajes de confirmación.

```
const form=document.getElementById('formpar');
document.addEventListener("DOMContentLoaded", function () {
  document.querySelector("#tabla").addEventListener("click",
function (e) {
    if (e.target.classList.contains("eliminar-btn")) {
      const id = e.target.getAttribute("data-id");

      if (confirm("¿Estás seguro de eliminar esta parcela?")) {
        fetch(`php/parcela/eliminar.php?id=${id}`, {
          method: 'GET',
          cache: 'no-cache'
        })
        .then(response => response.text())
        .then(data => {
          const respuesta = data.trim();
          if (respuesta === "1") {
          leerpar();
           cargarCards();
          form.reset();
          } else {
            alert ("error por"+respuesta);
          }
        })
        .catch(error => {
          console.error("Error al eliminar:", error);

        });
      }
    }
  });
});
```

Otros códigos importantes:

Integración de API de Clima

El sistema incorpora una API de clima para mostrar en tiempo real datos meteorológicos en el panel principal, incluyendo temperatura, humedad y descripción del clima. Esta información ayuda al usuario a tomar decisiones de riego más eficientes.

El script encargado de obtener estos datos se encuentra en la carpeta /js bajo el nombre clima.js y se carga al final del documento HTML para garantizar que todos los elementos estén disponibles en el DOM.

```
fetch(url)
  .then(response => response.json())
  .then(data => {
    let temp = Math.round(data.main.temp);
    temperatura.textContent = temp + "°C";
    let desc = data.weather[0].description;
    cielo.textContent = desc.toUpperCase();
    ubicacion.textContent = data.name;
    let icono_pro = data.weather[0].icon;
    icono.src =
`http://openweathermap.org/img/wn/${icono_pro}@2x.png`;
    humedad.textContent = data.main.humidity + "% de humedad";
  });
```

- Contactos de soporte:
- josue.gonzalez@est.itstena.edu.ec
- max.obando@est.itstena.edu.ec

Anexos

Script completo SQL de creación de base de datos

```
CREATE DATABASE riegosem

CREATE TABLE productor (
```

```
idPro int AUTO_INCREMENT PRIMARY KEY,
   tipPro varchar(30),
   nombre varchar (100),
   cuidad varchar(50)
);
CREATE TABLE personal(
idPer int PRIMARY KEY AUTO_INCREMENT,
nomPer varchar (50),
apePer varchar (50),
rol varchar (30),
corrPer varchar (100),
usuario varchar(50),
contra varchar (255)
idpro int,
  FOREIGN KEY (idpro) REFERENCES productor(idPro)

);
CREATE TABLE parcela (
idPar int PRIMARY KEY AUTO_INCREMENT,
idpro int,
nomPar varchar (75),
tamaño double,
tipo varchar (50),
foto varchar(255),
   FOREIGN KEY (idpro) REFERENCES productor(idPro)
);
CREATE TABLE equipo (
idEqui int AUTO_INCREMENT PRIMARY KEY,
```

```
codEqui varchar (50),
nomEqui varchar (50),
idpar int,
FOREIGN KEY (idpar) REFERENCES parcela(idPar)
);
CREATE TABLE riego (
idRie int PRIMARY KEY AUTO_INCREMENT,
tipo varchar (30),
fecha date,
hora time,
duracion time,
humedad varchar (30),
idequi int ,
idusu int,
FOREIGN KEY (idequi) REFERENCES equipo(idEqui)
)
```

Script login

```php
<?php
require '../conexion.php';
session_start();

if (isset($_POST['user']) && isset($_POST['contra'])) {
    $usuario = $_POST['user'];
    $contra = $_POST['contra'];

    // Consulta para obtener el hash y el rol en una sola consulta
    $stmt = $cnx->prepare("SELECT idPer,contra,rol,idpro FROM personal WHERE usuario = ?");
    $stmt->bind_param("s", $usuario);
    $stmt->execute();
    $resultado = $stmt->get_result();
    //valida si la consulta devolvio un valor
    if ($resultado->num_rows === 1) {
        $fila = $resultado->fetch_assoc(); // obtener array asociativo
```

```
            $hash_guardado = $fila['contra'];
            $rol = $fila['rol'];

            // Verificar la contraseña ingresada
            if (password_verify($contra, $hash_guardado)) {
                  $_SESSION['productor'] = $fila['idpro'];
                  $_SESSION['rol'] = $fila['rol'];
                   $_SESSION['id'] = $fila['idPer'];
                if ($rol === 'Administrador') {
                    header("Location: ../../index.php");
                    exit();
                } elseif ($rol=== 'Usuario') {
                    header("Location: ../../indexUsu.php");
                    exit();
                } else {
                 header("Location: ../../indexSA.php");
                    exit();
                }
            } else {
                 echo 'Contraseña incorrecta';
            }
        } else {
            echo '<h2>Usuario no encontrad</h2>';
        }
    }
    ?>
```

Leer

```
    <?php
    require '../conexion.php';
    session_start();

    $leer = mysqli_query($cnx, "SELECT idPer,nomPer,apePer,rol,usuario
    FROM personal WHERE idpro='".$_SESSION['productor']."' ");

    if (mysqli_num_rows($leer) > 0) {
        while ($row = mysqli_fetch_assoc($leer)) {
            echo "<tr><td>{$row['idPer']}</td><td>{$row['nomPer']}
    {$row['apePer']}</td>
                <td>{$row['rol']}</td><td>{$row['usuario']}</td>
                <td><button class='editar-btn' data-
    id='{$row['idPer']}' type='button'><i class='fas fa-
    edit'></i>Editar</button>
                <button class='eliminar-btn' data-id='{$row['idPer']}'
    type='button' ><i class='fas fa-trash'></i>Eliminar</button></td>
                </tr>
```

```
            ";
        }
    } else {
        echo "<tr><td colspan='2'>No hay usuarios
    registrados</td></tr>";
    }
    ?>
```

Devolver

```
<?php
require '../conexion.php';
//obtener id para devolver datos
if (isset($_GET['id'])) {
    $id = $_GET['id'];

    $query = mysqli_query($cnx, "SELECT * FROM personal WHERE idPer
= '$id'");
    if ($row = mysqli_fetch_assoc($query)) {
        echo json_encode([
            'success' => true,
            'personal' => $row
        ]);
    } else {
        echo json_encode(['success' => false, 'message' => 'No
encontrado']);
    }
} else {
    echo json_encode(['success' => false, 'message' => 'ID no
proporcionado']);
}
?>
```

Actualizar

```
<?php
require '../conexion.php';
if (
    isset( $_POST['idper'])  && isset($_POST['nombre']) &&
isset($_POST['apellido']) && isset($_POST['correo'])
&&  isset($_POST['rol'])
     ) {
        $id=$_POST['idper'];
    $nombre = $_POST['nombre'];
    $apellido = $_POST['apellido'];
    $correo = $_POST['correo'];
```

```
    $rol = $_POST['rol'];
    //$productor = $_POST['productor'];

    //actualizar datos
     $actualizar=mysqli_query($cnx,"UPDATE personal SET
    nomPer='$nombre',
    apePer='$apellido',
    rol='$rol',
    corrPer='$correo'
    WHERE idPer='$id'");
   if ($actualizar) {
       echo json_encode(['success' => true]);
    } else {
       echo json_encode(['success' => false, 'message' => 'No se
pudo actualizar']);
    }
}else {
    //Respuesta en caso de datos incompletos
    echo json_encode(['success' => false, 'message' => 'Datos
incompletos']);
}

?>
```

Eliminar

```
<?php
require '../conexion.php';
if (isset($_GET['id'])) {
     $id = $_GET['id'];

     $eliminar=mysqli_query($cnx,"DELETE FROM personal WHERE
idPer='$id'");
     if ($eliminar) {
        echo "1";
    } else {
        echo "0";
    }
}
?>
```

Filtrar datos

```
<?php
require '../conexion.php';
session_start();

```

```php
$consulta=mysqli_query($cnx,"SELECT idPar, nomPar FROM parcela
WHERE idpro='".$_SESSION['productor']."' ");
if (mysqli_num_rows($consulta)>0) {
    while ($row=mysqli_fetch_assoc($consulta)) {

        echo '<option value="' . $row['idPar'] . '">' .
$row['nomPar'] . '</option>';
    }
}else {
    echo 'No hay parcelas que mostar';
}
?>
```

Busqueda y filtrado

```php
<?php
header('Content-Type: application/json');
require '../conexion.php';
session_start();
$sql = "
SELECT idRie,
tipo,
fecha,
hora,
duracion,
humedad,
parcela.nomPar as parcela ,
personal.nomPer as personal
from riego
INNER JOIN equipo ON
riego.idequi=equipo.idEqui
INNER JOIN parcela ON
equipo.idpar=parcela.idPar
INNER JOIN personal ON
riego.idusu=personal.idPer
INNER JOIN productor ON
parcela.idpro=productor.idPro
where productor.idPro='".$_SESSION['productor']."'
        ORDER BY fecha DESC, hora DESC";

$result = mysqli_query($cnx, $sql);

$reportes = [];
while ($row = $result->fetch_assoc()) {
    $reportes[] = $row;
}

echo json_encode($reportes);
```

```php
mysqli_close($cnx);
?>
```

Programado

```php
<?php
require '../conexion.php';
session_start();
header('Content-Type: application/json'); // Para que JS sepa que es JSON

$response = ['status' => 'error', 'message' => 'Error desconocido'];

if (
    isset($_POST['inicio']) && isset($_POST['fin']) &&
    isset($_POST['hora']) && isset($_POST['h']) &&
    isset($_POST['m']) && isset($_POST['s']) &&
    isset($_POST['equipo'])
) {
    $inicio = $_POST['inicio'];
    $fin = $_POST['fin'];
    $hora = $_POST['hora'];
    $h = $_POST['h'];
    $m = $_POST['m'];
    $s = $_POST['s'];
    $equipo = $_POST['equipo'];
    $humedad = 0;
    $usuario = $_SESSION['id'];

    // Validación de fechas
    if ($inicio > $fin) {
        $response['message'] = 'La fecha de inicio no puede ser mayor que la fecha final.';
        echo json_encode($response);
        exit;
    }

    // Transformar a HH:MM:SS
    $duracion = sprintf('%02d:%02d:%02d', $h, $m, $s);

    // Preparar el insert
    $stmt = $cnx->prepare("INSERT INTO riego VALUES (null,'programado', ?, ?, ?, ?, ?, ?)");

    if (!$stmt) {
        $response['message'] = 'Error al preparar consulta: ' . $cnx->error;
```

```
            echo json_encode($response);
            exit;
        }

        // Crear fechas desde inicio a fin
        $fechas = [];
        $actual = new DateTime($inicio);
        $fechaFin = new DateTime($fin);
        $fechaFin->modify('+1 day'); // para incluir el día final

        while ($actual < $fechaFin) {
            $fechas[] = $actual->format('Y-m-d');
            $actual->modify('+1 day');
        }

        // Ejecutar insert por cada fecha
        foreach ($fechas as $fecha) {
            $stmt->bind_param("sssiii", $fecha, $hora, $duracion,
$humedad, $equipo, $usuario);
            if (!$stmt->execute()) {
                $response['message'] = "Error al guardar riego del
$fecha: " . $stmt->error;
                echo json_encode($response);
                exit;
            }
        }

        $stmt->close();
        $response = ['status' => 'success', 'message' => 'Todos los
riegos se guardaron correctamente'];
    } else {
        $response['message'] = 'Faltan datos requeridos';
    }

    echo json_encode($response);
```

Automático

```
    <?php
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $humedad = isset($_POST['humedad']) ? intval($_POST['humedad'])
    : null;
        $tiempo = isset($_POST['tiempo']) ? intval($_POST['tiempo']) :
    null;

        if ($humedad !== null && $tiempo !== null && $humedad > 0 &&
    $tiempo > 0) {
            $datos = [
```

```
            "humedad_minima" => $humedad,
            "tiempo_riego" => $tiempo
        ];

        $archivo = "config_auto.txt";
        file_put_contents($archivo, json_encode($datos));
        echo "Configuración guardada correctamente.";
    } else {
        echo "Datos inválidos.";
    }
} else {
    echo "Método no permitido.";
}
?>
```

Manual

```
<?php
require '../conexion.php';
session_start();
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $estado = $_POST['estado'] ?? 'OFF';
    $tiempo = $_POST['tiempo'] ?? 0;
    $equipo = $_POST['equipo'];
    $idusu=$_SESSION['id'];
    $fecha=date("Y-m-d");
    $hora=date("H:i:s");
    $humedad='0';

    $datos = [
        "estado" => $estado,
        "tiempo" => intval($tiempo)
    ];

    //Guardar en la base
    $stmt=$cnx->prepare("INSERT INTO riego
VALUES(null,'manual',?,?,?,?,?,?)") ;
    $stmt-
>bind_param('siisii',$fecha,$hora,$tiempo,$humedad,$equipo,$idusu);
     if ( $stmt->execute()) {
            echo "1"; // Devuelve un entero si se guardo
correctmente
        }else {
             echo "Error SQL: " . $stmt->error;
        }

    file_put_contents("estado_riego.txt", json_encode($datos));
    echo "Estado actualizado: " . json_encode($datos);
```

```
} else {
    echo "Método no permitido";
}
?>
```

Codigo del esp32

```
#include <WiFi.h>          // Librería para conectar ESP32 a WiFi
#include <HTTPClient.h>    // Librería para hacer solicitudes HTTP (GET, POST)
#include <ArduinoJson.h>   // Librería para manejar JSON en Arduino
#include "time.h"          // Librería para manejo de tiempo y sincronización con NTP

// --- Configuración WiFi ---
const char* ssid = "departamentotena";  // Nombre de la red WiFi
const char* password = "#megustaTena";  // Contraseña del WiFi

// --- Configuración para obtener la hora vía NTP ---
const char* ntpServer = "pool.ntp.org";         // Servidor NTP público
const long gmtOffset_sec = -5 * 3600;           // GMT -5 horas (Ecuador)
const int daylightOffset_sec = 0;               // No hay horario de verano

// --- Pines utilizados ---
const int pinSensor = 34;      // Pin analógico para sensor humedad YL-69
const int pinRiego = 23;       // Pin para controlar el relé (bomba de riego)

// Pines para LEDs indicadores
const int LED_VERDE = 18;      // LED indica que el riego está activo
const int LED_ROJO = 19;       // LED indica que el riego está inactivo
const int LED_WIFI = 2;        // LED interno del ESP32 para estado de conexión WiFi

// --- Variables globales ---
bool riegoActivo = false;      // Control para evitar múltiples activaciones simultáneas de riego
unsigned long finRiego = 0;    // Momento (en millis) en que debe finalizar el riego programado

// Umbral de humedad mínima para activar el riego automático (en porcentaje)
const int umbralHumedad = 10;

// --- Función para controlar el riego programado recibido desde el servidor ---
void verificarRiegoProgramado() {
```

```
    if (riegoActivo) {
      // Si ya se está regando, revisar si ya pasó el tiempo programado
      if (millis() >= finRiego) {
        // Detener el riego
        digitalWrite(pinRiego, HIGH); // Apaga el relé (relé activo con LOW)
        digitalWrite(LED_VERDE, LOW); // Apaga LED verde
        digitalWrite(LED_ROJO, HIGH); // Enciende LED rojo (riego apagado)
        riegoActivo = false;
        Serial.println("🟦 Riego programado finalizado.");
      }
      // Mientras se está regando, no hacer nuevas solicitudes
      return;
    }

    // Crear objeto HTTPClient para hacer solicitud GET
    HTTPClient http;
    // URL para obtener el estado del riego programado del equipo 7

http.begin("http://192.168.100.77/dashboard/riegoS/php/microesp/obtener_riego_programado.php?equipo=7");
    int code = http.GET();  // Hacer petición GET

    if (code == 200) {  // Si respuesta HTTP fue exitosa
      String respuesta = http.getString();  // Obtener texto de respuesta

      Serial.print("Respuesta completa recibida: ");
      Serial.println(respuesta);  // Mostrar la respuesta para depuración

      // Parsear el JSON recibido usando ArduinoJson
      DynamicJsonDocument doc(2048);
      DeserializationError error = deserializeJson(doc, respuesta);
      if (error) {
        Serial.print("❌ Error parseando JSON: ");
        Serial.println(error.c_str());
        http.end();
        return;
      }

      // Leer el campo "status" del JSON para saber si hay riego activo
      String status = doc["status"];
      Serial.print("📡 Estado recibido: ");
      Serial.println(status);

      if (status == "regando") {
        // Si el estado es "regando", iniciar el riego programado
        int duracionSeg = doc["duracion_total"]; // Duración en segundos

```

```
        Serial.print("⏰ ¡Iniciando riego programado por ");
        Serial.print(duracionSeg);
        Serial.println(" segundos!");

        digitalWrite(pinRiego, LOW);   // Activar relé (bomba ON)
        digitalWrite(LED_VERDE, HIGH); // Encender LED verde
        digitalWrite(LED_ROJO, LOW);   // Apagar LED rojo
        riegoActivo = true;
        // Calcular el tiempo final del riego sumando duración en ms a millis actual
        finRiego = millis() + (duracionSeg * 1000UL);
      } else {
        Serial.println("⏳ No hay riego programado activo ahora.");
      }
    } else {
      // Si hubo error en la petición HTTP, mostrar código
      Serial.print("❌ Error HTTP: ");
      Serial.println(code);
    }
    http.end(); // Finalizar conexión HTTP
  }

  // --- Función para reconectar WiFi si la conexión se pierde ---
  void reconectarWiFi() {
    Serial.println("🔄 WiFi desconectado, intentando reconectar...");

    WiFi.disconnect(true);
    delay(500);
    WiFi.begin(ssid, password);

    unsigned long inicio = millis();
    const unsigned long tiempoEspera = 15000;  // Máximo 15 segundos para
  reconectar

    // Parpadeo del LED WiFi mientras intenta conectar
    while (WiFi.status() != WL_CONNECTED && millis() - inicio < tiempoEspera) {
      digitalWrite(LED_WIFI, !digitalRead(LED_WIFI));  // Parpadea LED WiFi
      delay(500);
      Serial.print(".");
    }

    if (WiFi.status() == WL_CONNECTED) {
      Serial.println("\n✅ WiFi reconectado correctamente.");
      digitalWrite(LED_WIFI, HIGH); // LED fijo al conectarse
    } else {
      Serial.println("\n❌ Falló la reconexión WiFi.");
      digitalWrite(LED_WIFI, LOW);  // Apagar LED si no conecta
    }
```

```
}

// --- Función setup se ejecuta una vez al iniciar ---
void setup() {
  Serial.begin(115200);           // Inicializar puerto serie para debug
  pinMode(pinSensor, INPUT);      // Pin sensor como entrada
  pinMode(pinRiego, OUTPUT);      // Pin relé como salida

  // Configurar LEDs como salidas
  pinMode(LED_VERDE, OUTPUT);
  pinMode(LED_ROJO, OUTPUT);
  pinMode(LED_WIFI, OUTPUT);

  digitalWrite(pinRiego, HIGH);   // Apagar bomba inicialmente (relé activo en LOW)

  digitalWrite(LED_VERDE, LOW);   // Apagar LED verde
  digitalWrite(LED_ROJO, HIGH);   // Encender LED rojo (riego apagado)
  digitalWrite(LED_WIFI, LOW);    // Apagar LED WiFi

  // Iniciar conexión WiFi
  WiFi.begin(ssid, password);
  // Esperar hasta conectarse
  while(WiFi.status() != WL_CONNECTED) {
    digitalWrite(LED_WIFI, HIGH);
    delay(1000);
    digitalWrite(LED_WIFI, LOW);
    delay(1000);
    Serial.println("Conectando a WiFi...");
  }
  Serial.println("WiFi conectado");
  digitalWrite(LED_WIFI, HIGH);  // LED WiFi fijo indicando conexión exitosa

  // Configurar hora local mediante servidor NTP
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("❌ No se pudo obtener la hora del servidor NTP");
    return;
  }
  Serial.print("🕒 Hora sincronizada: ");
  Serial.println(&timeinfo, "%Y-%m-%d %H:%M:%S");
}

// --- Función loop se ejecuta repetidamente ---
void loop() {
```

```
  // Verificar conexión WiFi y reconectar si está desconectado
  if (WiFi.status() != WL_CONNECTED) {
    reconectarWiFi();
    if (WiFi.status() != WL_CONNECTED) {
      Serial.println(" ⚠️ WiFi aún no disponible. Saltando ciclo.");
      return; // No continuar si no hay WiFi
    }
  }
  digitalWrite(LED_WIFI, HIGH);  // LED WiFi encendido fijo

  // Leer el valor analógico del sensor de humedad
  int lecturaBruta = analogRead(pinSensor);
  // Mapear el valor analógico a porcentaje de humedad (0% seco, 100% húmedo)
  int humedad = map(lecturaBruta, 0, 4095, 100, 0);

  Serial.print("Lectura bruta: ");
  Serial.print(lecturaBruta);
  Serial.print("  →  Humedad: ");
  Serial.print(humedad);
  Serial.println("%");

  // ===== RIEGO AUTOMÁTICO =====
  // Leer configuración automática desde archivo en servidor
  HTTPClient httpConfig;

  httpConfig.begin("http://192.168.100.77/dashboard/riegoS/php/microesp/config_auto.txt");
  int code = httpConfig.GET();

  if (code == 200) {
    String payload = httpConfig.getString();
    payload.trim();

    // Buscar los parámetros necesarios dentro del texto
    int humedadIndex = payload.indexOf("\"humedad_minima\":");
    int tiempoIndex = payload.indexOf("\"tiempo_riego\":");

    if (humedadIndex >= 0 && tiempoIndex >= 0) {
      // Extraer valores numéricos de umbral y tiempo
      int startHum = humedadIndex + strlen("\"humedad_minima\":");
      int endHum = payload.indexOf(",", startHum);
      int umbral = payload.substring(startHum, endHum).toInt();

      int startTiempo = tiempoIndex + strlen("\"tiempo_riego\":");
      int endTiempo = payload.indexOf("}", startTiempo);
      int tiempoRiego = payload.substring(startTiempo, endTiempo).toInt();
```

```
•
•       Serial.print(" 🌡 Umbral lectura bruta configurado: ");
•       Serial.println(umbral);
•       Serial.print("⏱️ Tiempo de riego (ms): ");
•       Serial.println(tiempoRiego);
•
•       // Si humedad está por debajo del umbral y umbral es válido, iniciar riego
•       if (lecturaBruta > umbral && umbral > 0) {
•         int humedadInicial = humedad;  // Guardar humedad que activó riego
•         Serial.println(" 🌱 Humedad baja, iniciando riego automático...");
•         digitalWrite(pinRiego, LOW);      // Activar bomba
•         digitalWrite(LED_VERDE, HIGH);    // LED verde encendido
•         digitalWrite(LED_ROJO, LOW);      // LED rojo apagado
•
•         unsigned long inicioRiego = millis();
•
•         // Mantener riego hasta que humedad supere umbral
•         while (true) {
•           int lectura = analogRead(pinSensor);
•           int humedadActual = map(lectura, 0, 4095, 100, 0);
•           Serial.print(" 💧 Humedad actual: ");
•           Serial.println(humedadActual);
•
•           if (lectura <= umbral) {
•             Serial.println(" ✅ Humedad suficiente, deteniendo riego automático.");
•             break;
•           }
•           delay(2000); // Esperar 2 segundos antes de siguiente lectura
•         }
•
•         // Apagar riego y LEDs
•         digitalWrite(pinRiego, HIGH);
•         digitalWrite(LED_VERDE, LOW);
•         digitalWrite(LED_ROJO, HIGH);
•
•         // Registrar riego en base de datos vía POST HTTP
•         HTTPClient httpReg;
•
    httpReg.begin("http://192.168.100.77/dashboard/riegoS/php/microesp/registrar_riego.php");
•         httpReg.addHeader("Content-Type", "application/x-www-form-urlencoded");
•
•         unsigned long duracionSeg = (millis() - inicioRiego) / 1000;
•         String datosRiego = "tipo=Automatico&duracion=" + String(duracionSeg) +
•                             "&humedad=" + String(humedadInicial) +
•                             "&equipo=1" + "&usuario=1";
```

```
•
•         int respuestaReg = httpReg.POST(datosRiego);
•         Serial.print("Registro HTTP: ");
•         Serial.println(respuestaReg);
•
•         httpReg.end();
•        } else {
•         Serial.println("Humedad suficiente o parámetros inválidos.");
•        }
•      } else {
•        Serial.println("❌ Error al interpretar parámetros del archivo.");
•      }
•     } else {
•      Serial.print("❌ No se pudo obtener config_auto.txt. Código: ");
•      Serial.println(code);
•     }
•     httpConfig.end();
•
•     // ===== RIEGO MANUAL =====
•     // Consultar estado del riego manual desde archivo en servidor
•     HTTPClient httpManual;
•
    httpManual.begin("http://192.168.100.77/dashboard/riegoS/php/microesp/est
    ado_riego.txt");
•     int codigo = httpManual.GET();
•
•     if (codigo == 200) {
•      String cuerpo = httpManual.getString();
•      cuerpo.trim();
•
•      // Extraer estado y tiempo desde el texto recibido
•      int estadoIndex = cuerpo.indexOf("\"estado\":\"");
•      int tiempoIndex = cuerpo.indexOf("\"tiempo\":");
•
•      if (estadoIndex >= 0 && tiempoIndex >= 0) {
•       String estado = cuerpo.substring(estadoIndex + 10, cuerpo.indexOf("\"",
    estadoIndex + 10));
•       int tiempo = cuerpo.substring(tiempoIndex + 9).toInt();
•
•       // Si el estado indica que riego manual está ON y el tiempo es válido
•       if (estado == "ON" && tiempo > 0) {
•         Serial.println("Riego manual detectado desde página web");
•
•         // Activar bomba y LEDs
•         digitalWrite(pinRiego, LOW);
•         digitalWrite(LED_VERDE, HIGH);
•         digitalWrite(LED_ROJO, LOW);
```

```
•
•         unsigned long inicio = millis();
•
•         // Mantener riego manual mientras dure el tiempo o hasta que se cancele
•         while (millis() - inicio < tiempo) {
•           delay(500);  // Esperar 0.5 segundos antes de consultar nuevo estado
•
•           // Consultar nuevamente estado para detectar cancelación
•           HTTPClient checkCancel;
•
    checkCancel.begin("http://192.168.100.77/dashboard/riegoS/php/microesp/es
    tado_riego.txt");
•           int cod = checkCancel.GET();
•           if (cod == 200) {
•             String nuevo = checkCancel.getString();
•             nuevo.trim();
•             int nuevoEstadoIndex = nuevo.indexOf("\"estado\":\"");
•             if (nuevoEstadoIndex >= 0) {
•               String nuevoEstado = nuevo.substring(nuevoEstadoIndex + 10,
    nuevo.indexOf("\"", nuevoEstadoIndex + 10));
•               if (nuevoEstado == "OFF") {
•                 Serial.println("Riego manual cancelado desde la web");
•                 break;  // Salir del ciclo si cancelan el riego manual
•               }
•             }
•           }
•           checkCancel.end();
•         }
•
•         // Apagar riego y actualizar LEDs
•         digitalWrite(pinRiego, HIGH);
•         digitalWrite(LED_VERDE, LOW);
•         digitalWrite(LED_ROJO, HIGH);
•
•         // Borrar archivo para indicar que ya no hay riego manual activo
•         HTTPClient httpBorrar;
•
    httpBorrar.begin("http://192.168.100.77/dashboard/riegoS/php/microesp/borr
    ar_estado.php");
•         httpBorrar.GET();
•         httpBorrar.end();
•       }
•     }
•   }
•   httpManual.end();
•
•   // ===== ENVÍO DE DATOS DE HUMEDAD AL SERVIDOR =====
```

```
  HTTPClient http;

http.begin("http://192.168.100.77/dashboard/riegoS/php/microesp/prueba.php");
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");

  String datos = "humedad=" + String(humedad);
  int respuesta = http.POST(datos);

  Serial.print("Código HTTP: ");
  Serial.println(respuesta);
  http.end();

  // Revisar si hay riego programado para activar
  verificarRiegoProgramado();

  delay(5000);  // Esperar 5 segundos antes del próximo ciclo
}
```

CSS Index

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}
/* Tarjetas de información */
.card {
    background-color: white;
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.05);
    margin-bottom: 20px;
    width: 300px;
}

.card h2 {
    color: #334155;
    margin-bottom: 15px;
    display: flex;
    align-items: center;
}

.card h2 i {
    margin-right: 10px;
    color: #3b82f6;
}
```

```css

/* Sección de clima */
.clima {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 20px;
}

.clima-info {
    display: flex;
    flex-direction: column;
    align-items: center;
}

#temperatura-clima {
    font-size: 36px;
    font-weight: bold;
    color: #3b82f6;
    margin: 10px 0;
}

#ubicacion {
    color: #64748b;
    margin-bottom: 15px;
}

.clima-detalle {
    display: flex;
    align-items: center;
    margin-bottom: 10px;
}

#icono {
    width: 50px;
    height: 50px;
    margin-right: 10px;
}

#humedad {
    color: #64748b;
    font-size: 14px;
}

/* Estadísticas */
.estadisticas {
    display: flex;
    flex-direction: column;
    gap: 15px;
}
```

```css

.estadistica-item {
    display: flex;
    align-items: center;
    padding: 10px;
    background-color: #f8fafc;
    border-radius: 8px;
}

.estadistica-item i {
    color: #3b82f6;
    margin-right: 10px;
    width: 20px;
    text-align: center;
}

.estadistica-item span {
    color: #334155;
}

.estadistica-item strong {
    color: #3b82f6;
}

/* Diseño para tarjetas en fila */
.container-clima {
       display: flex;
    flex-wrap: nowrap; /* fuerza fila única */
    justify-content: space-between; /* separa con espacio entre */
    margin: 20px;
    gap: 0;
}

.card-clima {
     flex: 0 0 27%; /* 28% del ancho del contenedor, deja espacio para gaps */
    max-width: 27%; /* aseguras que no crezca más */
    min-width: 200px; /* un mínimo para que no queden demasiado chicos */
    background-color: white;
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.05);
    box-sizing: border-box;
}

/* Diseño para el section */
section {
    background-color: #f1f5f9;
```

```
        padding: 30px;
        margin: 30px;
        border-radius: 12px;
        box-shadow: 0 2px 4px rgba(0,0,0,0.05);
        color: #334155;
        font-size: 16px;
        line-height: 1.6;
    }

    section h3 {
        color: #1e3a8a;
        margin-bottom: 15px;
    }

    /* Responsive */
    @media (max-width: 768px) {
        .menu-izquierdo {
            width: 100%;
            height: auto;
            position: relative;
        }

        .contenido {
            margin-left: 0;
            width: 100%;
        }

        .clima {
            grid-template-columns: 1fr;
        }

        .container-clima {
            flex-direction: column;
            align-items: center;
        }
    }
```

Integración de la API de clima (clima.js)

```
    // Escucha cuando la ventana (página) termina de cargar
    completamente
    window.addEventListener('load', () => {
        // Declarar variables para longitud y latitud
        let lon;
        let lat;
```

```
    // Obtener referencias a los elementos HTML donde se mostrará la información del clima
    let ubicacion = document.getElementById('ubicacion');
    let cielo = document.getElementById('cielo');
    let temperatura = document.getElementById('temperatura-clima');
    let icono = document.getElementById('icono');
    let humedad = document.getElementById('humedad');

    // Verificar si el navegador soporta geolocalización
    if (navigator.geolocation) {
        // Obtener la posición actual del usuario (latitud y longitud)
        navigator.geolocation.getCurrentPosition(posicion => {
            lat = posicion.coords.latitude;   // Latitud actual
            lon = posicion.coords.longitude;  // Longitud actual

            // Construir la URL para llamar a la API de OpenWeather con lat/lon, unidad métrica y en español
            const url = `https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=0fc4b64b316492953965685b83d36d7&units=metric&lang=es`;

            // Hacer la solicitud HTTP para obtener los datos del clima
            fetch(url)
            .then(response => {
                return response.json();  // Convertir la respuesta a formato JSON
            })
            .then(data => {
                // Redondear la temperatura y mostrarla en el elemento correspondiente
                let temp = Math.round(data.main.temp);
                temperatura.textContent = temp + "°C";

                // Obtener la descripción del clima y mostrarla en mayúsculas
                let desc = data.weather[0].description;
                cielo.textContent = desc.toUpperCase();

                // Mostrar el nombre de la ubicación (ciudad)
                ubicacion.textContent = data.name;

                // Obtener el código del ícono y asignar la imagen correspondiente
                let icono_pro = data.weather[0].icon;
                icono.src = `http://openweathermap.org/img/wn/${icono_pro}@2x.png`;

```

```
                // Mostrar el porcentaje de humedad
                humedad.textContent = data.main.humidity + "% de
humedad";

                // Mostrar los datos completos en consola para
depuración
                console.log(data);

            })
            .catch(error => {
                // En caso de error en la solicitud, mostrarlo en
consola
                console.log(error);
            });
        });
    }
});
```

Reportes

```
<?php
// Conexión a la base de datos
require '../conexion.php'; // Debe definir la variable $cnx con la
conexión
// Carga la librería Dompdf para generar PDFs
require_once '../dompdf/autoload.inc.php';
use Dompdf\Dompdf;

// Inicia la sesión para validar usuario
session_start();

// Validar que el productor esté autenticado y que se haya enviado
el mes a consultar
if (!isset($_SESSION['productor']) || !isset($_POST['mes'])) {
    echo "Acceso denegado o datos incompletos.";
    exit;
}

// Obtener el mes seleccionado en formato YYYY-MM desde el
formulario POST
$mesSeleccionado = $_POST['mes'];
// Separar año y mes para usarlos en la consulta
list($anio, $mes) = explode('-', $mesSeleccionado);

// Consulta para obtener el resumen de riegos por tipo en el mes
seleccionado
$sqlResumen = "SELECT tipo, COUNT(*) as total
    FROM riego
```

```
    INNER JOIN equipo ON riego.idequi = equipo.idEqui
    INNER JOIN parcela ON equipo.idpar = parcela.idPar
    INNER JOIN productor ON parcela.idpro = productor.idPro
    WHERE productor.idPro = '{$_SESSION['productor']}'
    AND MONTH(fecha) = {$mes} AND YEAR(fecha) = {$anio}
    GROUP BY tipo";

// Ejecutar la consulta de resumen
$resResumen = $cnx->query($sqlResumen);
$datos = [];

// Recorrer resultados y almacenar totales por tipo
while ($row = $resResumen->fetch_assoc()) {
    $datos[$row['tipo']] = $row['total'];
}

// Generar la URL para el gráfico de pastel usando QuickChart con los datos obtenidos
$chart_url = "https://quickchart.io/chart?c=" . urlencode(json_encode([
    "type" => "pie",
    "data" => [
        "labels" => array_keys($datos),  // Tipos de riego
        "datasets" => [[
            "data" => array_values($datos), // Cantidad por tipo
            "backgroundColor" => ["#36A2EB", "#FF6384", "#FFCE56"] // Colores del gráfico
        ]]
    ],
    "options" => [
        "title" => ["display" => true, "text" => "Riegos por tipo - $mesSeleccionado"]
    ]
]));

// Consulta detallada de todos los riegos realizados en el mes seleccionado
$sqlDetalle = "SELECT idRie, tipo, fecha, hora, duracion, humedad,
    parcela.nomPar as parcela, personal.nomPer as personal
    FROM riego
    INNER JOIN equipo ON riego.idequi = equipo.idEqui
    INNER JOIN parcela ON equipo.idpar = parcela.idPar
    INNER JOIN personal ON riego.idusu = personal.idPer
    INNER JOIN productor ON parcela.idpro = productor.idPro
    WHERE productor.idPro = '{$_SESSION['productor']}'
    AND MONTH(fecha) = {$mes} AND YEAR(fecha) = {$anio}
    ORDER BY fecha, hora";

// Ejecutar consulta detallada
```

```
$resDetalle = $cnx->query($sqlDetalle);

// Construcción del contenido HTML para el PDF con estilos incluidos
$html = "
<style>
  body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f8faf5;
    color: #334d1a;
    margin: 20px;
  }
  h1, h3 {
    text-align: center;
    color: #3a5a40;
    margin-bottom: 10px;
  }
  table {
    width: 100%;
    border-collapse: separate;
    border-spacing: 0 10px;
  }
  table th {
    background-color: #5a7d4c;
    color: #ffffff;
    padding: 10px;
    border-radius: 8px 8px 0 0;
  }
  table td {
    background-color: #e6f0d4;
    padding: 8px;
    text-align: center;
    border-bottom: 1px solid #a3b07e;
  }
  table tr:last-child td {
    border-radius: 0 0 8px 8px;
  }
  img {
    display: block;
    margin: 0 auto;
    border-radius: 15px;
    box-shadow: 0 0 10px #a3b07e;
  }
  p {
    font-size: 14px;
    text-align: center;
    color: #557a2f;
  }
</style>
```

```
•
•   <h1>Reporte de Riegos - $mesSeleccionado</h1>
•   <p><strong>Generado el:</strong> " . date('Y-m-d') . "</p>
•   ";
•
•   // Tabla resumen con totales por tipo de riego
•   $html .= "<h3>Resumen por tipo</h3><table border='1'
    cellpadding='5'><tr><th>Tipo</th><th>Total</th></tr>";
•   foreach ($datos as $tipo => $total) {
•       $html .= "<tr><td>$tipo</td><td>$total</td></tr>";
•   }
•   $html .= "</table><br>";
•
•   // Insertar imagen del gráfico de pastel generado por QuickChart
•   $html .= "<img src='{$chart_url}' width='400'><br><br>";
•
•   // Tabla detallada con cada riego realizado
•   $html .= "<h3>Listado de Riegos</h3><table border='1'
    cellpadding='5'>
•   <tr>
•
    <th>Tipo</th><th>Fecha</th><th>Hora</th><th>Duración</th><th>Humeda
    d</th><th>Parcela</th><th>Personal</th>
•   </tr>";
•
•   // Agregar fila por cada registro obtenido en la consulta detallada
•   while ($row = $resDetalle->fetch_assoc()) {
•       $html .= "<tr>
•           <td>{$row['tipo']}</td>
•           <td>{$row['fecha']}</td>
•           <td>{$row['hora']}</td>
•           <td>{$row['duracion']}</td>
•           <td>{$row['humedad']}</td>
•           <td>{$row['parcela']}</td>
•           <td>{$row['personal']}</td>
•       </tr>";
•   }
•   $html .= "</table>";
•
•   // Crear instancia de Dompdf para generar el PDF
•   $dompdf = new Dompdf();
•   $dompdf->set_option('isRemoteEnabled', true); // Permitir cargar
    imágenes remotas (el gráfico)
•   $dompdf->loadHtml($html); // Cargar el contenido HTML para
    renderizar
•   $dompdf->setPaper('A4', 'landscape'); // Configurar tamaño y
    orientación del papel
•   $dompdf->render(); // Renderizar el PDF
```

```
// Enviar el PDF al navegador para descarga con un nombre basado en el mes seleccionado
$dompdf->stream("reporte_riegos_{$mesSeleccionado}.pdf", ["Attachment" => true]);
?>
```

Guardar (JS)

```
// Espera a que todo el contenido del DOM esté cargado antes de ejecutar el script
document.addEventListener('DOMContentLoaded', function () {
  // Obtener referencia al formulario con id 'formpar'
  const formulario = document.getElementById('formpar');
  // Obtener referencia al botón de guardar con id 'btnGuardar'
  const btnGuardar = document.getElementById('btnGuardar');

  // Escuchar el evento click sobre el botón Guardar
  btnGuardar.addEventListener('click', function () {
    // Obtener y limpiar los valores ingresados en los campos del formulario
    const nombre = document.getElementById("nombre").value.trim();
    const tamano = document.getElementById("tamano").value.trim();
    const cultivo = document.getElementById("cultivo").value.trim();
    const imagen = document.getElementById("imagen").files[0]; // Archivo de imagen seleccionado
    const productor = document.getElementById("productor").value;

    // Validar que ninguno de los campos obligatorios esté vacío
    if (!nombre || !tamano || !cultivo  || !productor) {
      // Mostrar mensaje de advertencia usando SweetAlert
      Swal.fire({
        title: '¡Aviso!',
        text: 'Los campos no pueden estar vacíos.',
        icon: 'warning',
        timer: 2000,
        timerProgressBar: true,
        showConfirmButton: false
      });
    } else {
      // Si la validación pasa, crear un objeto FormData para enviar los datos
      const datos = new FormData();
      datos.append("nombre", nombre);
      datos.append("cultivo", cultivo);
      datos.append("tamaño", tamano);
```

```
        datos.append("foto", imagen);
        datos.append("productor", productor);

        // Enviar los datos al servidor con fetch usando método POST
        fetch('php/parcela/guardar.php', {
          method: 'POST',
          body: datos
        })
        .then(response => response.text())  // Leer la respuesta como texto
        .then(data => {
          // Evaluar la respuesta del servidor
          if (data == "1") {  // Si respuesta es "1" indica éxito
            Swal.fire({
              title: 'Éxito',
              text: 'Parcela guardada correctamente',
              icon: 'success',
              confirmButtonText: 'Aceptar'
            });
            // Actualizar las cards (listas) y recargar datos, funciones definidas en otro lado
            cargarCards();
            leerpar();
            // Reiniciar el formulario para dejarlo limpio
            formulario.reset();
          } else {
            // Si hubo error, mostrar mensaje con detalles del error recibido
            Swal.fire({
              title: 'Error',
              text: 'Error al guardar en la base de datos.\nDetalles: ' + data,
              icon: 'error',
              confirmButtonText: 'Aceptar'
            });
          }
        });
      }
    });
  });
```