

# PROYECTO

# Plataforma Agrodoc



*PROYECTO FINAL DE CURSO DAW 2025*

ALUMNA: María Tíscar Amador Moreno

# **ÍNDICE:**

<b>INTRODUCCIÓN</b>	<b>3</b>
OBJETIVOS DEL PROYECTO	3
<b>DESPLIEGUE DE SERVIDORES</b>	<b>4</b>
<b>BASE DE DATOS</b>	<b>6</b>
FUNCIONALIDAD	6
MODELO ENTIDAD/RELACIÓN	7
RELACIONES:	7
TABLAS:	8
<b>GESTIÓN DEL BACK-END</b>	<b>10</b>
CODIFICACIÓN	10
ARCHIVO LOGIN.PHP	10
MÉTODOS CRUD	12
<b>GESTIÓN DEL FRONT-END</b>	<b>16</b>
VALIDACIÓN DE LOS CAMPOS DE FORMULARIOS	16
CREACIÓN DE ELEMENTOS DINÁMICOS USANDO ACCESO AL DOM	19
PETICIONES ASÍNCRONAS	19
<b>INTERFACES WEB, USABILIDAD Y ACCESIBILIDAD</b>	<b>21</b>
RESPONSIVE:	24
OPTIMIZACIÓN DE IMÁGENES:	25

# INTRODUCCIÓN

La agricultura y las explotaciones agrícolas requieren de una gran cantidad de mano de obra, especialmente en temporadas de cosecha o actividades intensivas. Actualmente, cooperativas agrícolas y grandes empresas enfrentan dificultades para encontrar personal cualificado en períodos críticos, así como para gestionar la subcontratación de empresas especializadas en mano de obra agrícola. Por esta razón, propongo el desarrollo de una plataforma digital que facilite el contacto entre cooperativas, explotaciones agrícolas, agricultores y empresas proveedoras de mano de obra. La plataforma tiene como objetivo optimizar la contratación y gestión de personal, agilizando el proceso y fomentando un entorno de confianza y colaboración.

La aplicación consiste en una plataforma web de gestión y subcontratación de mano de obra agrícola, donde cooperativas agrícolas, grandes empresas con explotaciones y agricultores podrán registrarse para subcontratar servicios de mano de obra. De igual manera, empresas especializadas en el suministro de mano de obra pueden registrarse y ofrecer sus servicios a través de la plataforma.

Público Objetivo:

- Cooperativas agrícolas: Necesitan mano de obra estacional o temporal.
- Empresas agrícolas: Buscan gestionar mejor la subcontratación de personal.
- Agricultores independientes: Requieren apoyo de mano de obra para ciertos trabajos o temporadas.
- Empresas proveedoras de mano de obra agrícola.

## OBJETIVOS DEL PROYECTO

Los objetivos del proyecto son:

- Facilitar la comunicación y subcontratación de mano de obra agrícola.
- Optimizar la gestión de servicios ofrecidos y contratados.
- Centralizar la información y generar un espacio confiable para ambas partes.

La estructura del sistema va a consistir en:

- **Frontend:** La interfaz visual permite a los usuarios interactuar con la plataforma. En el desarrollo se ha utilizado tecnologías como HTML, CSS y JavaScript y Jquery. Tiene un diseño responsive que permite a los usuarios acceder desde distintos dispositivos, con una navegación sencilla y filtros de

búsqueda.

- **Backend:** Gestiona la lógica del negocio y conexión a la base de datos. He utilizado php para manejar las solicitudes y las rutas de la aplicación.
- **Base de Datos:** MySQL para almacenar información de usuarios, proyectos, trabajadores...

## DESPLIEGUE DE SERVIDORES

He optado por utilizar contenedores Docker para garantizar la portabilidad del entorno de desarrollo y producción. Esto me permite levantar la aplicación completa (servidor web, base de datos y herramientas de administración) mediante contenedores con docker-compose. El despliegue está configurado en tres archivos principales:

- docker-compose.yml: define los servicios y cómo se comunican.
- Dockerfile: define cómo construir el contenedor web personalizado.
- apache.conf: configura Apache dentro del contenedor web.

Archivo docker-compose.yml:

```
version: '3.8'
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
      container_name: agrodoc_web
    restart: always
    volumes:
      - /path/to/www/html
      - ./src:/var/www/src
      - ./conf:/var/www/config
      - ./documentos_trab:/var/www/documentos_trab
      - ./docker/apache/apache.conf:/etc/apache2/sites-available/000-default.conf
    ports:
      - "8080:80"
    depends_on:
      - db
  db:
    image: mysql:18.4
    container_name: agrodoc_db
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: bigagrodoc
      MYSQL_USER: maria
      MYSQL_PASSWORD: 1234
    ports:
      - "3307:3306"
    volumes:
      - ./database/mysql:/var/lib/mysql
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: agrodoc_phpmyadmin
    restart: always
    depends_on:
      - db
    environment:
      PMA_HOST: db
      PMA_USER: root
      PMA_PASSWORD: root
    ports:
      - "8081:80"
    volumes:
      db_data:
```

Los servicios desplegados son:

Servicio Web:

- Este contenedor se llama agro doc web.
- Usa el archivo Dockerfile
- Mapea el puerto 8080 del host al 80 del contenedor.
- Depende del contenedor db esté listo antes de iniciarse.

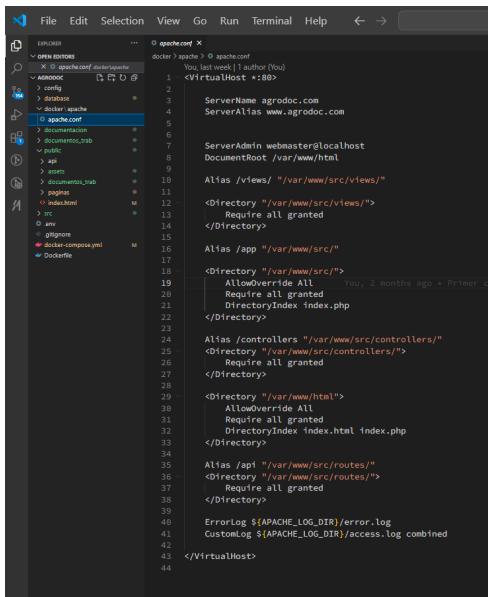
- En los volumes se montan carpetas locales como: código, configuración, documentos...

## Servicio db:

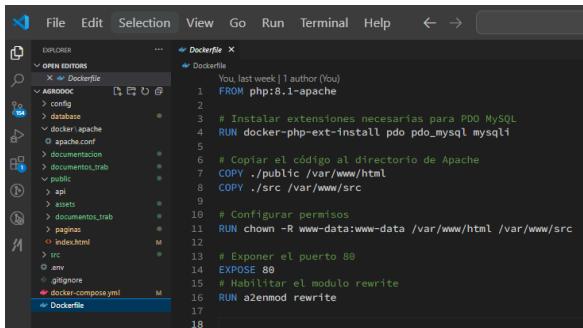
- Se llama agrodoc\_db.
  - Usa la imagen oficial de mysql:8.0.
  - Define usuario, contraseña y base de datos por variables de entorno.
  - ports: mapea el puerto 3306 como 3307 en el host.
  - volumes: Persiste la base de datos.

## Servicio Phpmyadmin:

- Se llama agrodoc\_phpmyadmin.
  - Interfaz gráfica para gestionar MySQL.
  - Utiliza variables de entorno como: PMA\_HOST, PMA\_USER, PMA\_PASSWORD
  - ports: Mapea el puerto 8081 del host al 81 del contenedor.



El archivo apache.conf que está en la carpeta docker, y tiene la configuración de cómo el servidor Apache gestiona las rutas y accesos dentro del contenedor, el dominio local www.agrodoc.com....



El archivo Dockerfile define cómo se construye la imagen personalizada del contenedor web de Apache + PHP . Se define la imagen de PHP que se utiliza, instala las extensiones de PHP necesarias para la conexión a bases de datos, copia los archivos locales, configura permisos....

Levantamos los contenedores con los comandos correspondientes de docker compose.  
Muestra de los contenedores levantados:

```
PS C:\Users\mtisc\Desktop\agrodoc> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
dbcbb379df3e agrodoc-web "docker-php-entrypoi..." 47 hours ago Up 47 hours 0.0.0.0:8080->80/tcp agrodoc_web
4675c8f453b8 phpmyadmin/phpmyadmin "/docker-entrypoint..." 47 hours ago Up 47 hours 0.0.0.0:8081->80/tcp agrodoc_phpmyadmin
c31197fffc8ea mysql:8.0 "docker-entrypoint.s..." 47 hours ago Up 47 hours 3306/tcp, 0.0.0.0:3307->3306/tcp agrodoc_db
○ PS C:\Users\mtisc\Desktop\agrodoc>
```

Accesos de los servicios en el navegador. En mi caso, como he configurado el host local, puedo acceder a la aplicación en el navegador por el nombre de dominio de la aplicación: [www.agrodoc.com](http://www.agrodoc.com)

The screenshots show the Agrodoc website interface. The top one is from the live domain, and the bottom one is from the local development server. Both versions of the site are identical, featuring a green header bar with the Agrodoc logo, user authentication buttons ('Soy contratista', 'Soy proveedor'), support links ('Soporte'), a blog section ('Blog NEW'), and an 'Entrar' button. Below the header are navigation links for 'Inicio', 'Plataforma CAE', 'Software PRL', 'Mas soluciones', and 'Contacto'.

This screenshot shows the phpMyAdmin interface for the 'agrodoc' database. The left sidebar lists databases like 'agrodoc', 'contratistas', 'documentos', 'fincas', 'proveedores', 'proyectos', 'proyectos\_trabajadores', 'trabajadores', and 'usuarios'. The main panel displays the 'Configuraciones generales' (General configurations) section, which includes a dropdown for the connection to the server ('utf8mb4\_unicode\_ci') and a 'Configuraciones de apariencia' (Appearance configurations) section with language ('Español - Spanish') and theme ('pmahomme'). On the right, there are panels for 'Servidor de base de datos' (Database server) and 'Servidor web' (Web server), both of which are currently set to Apache 2.4.62 (Debian).

# BASE DE DATOS

## FUNCIONALIDAD

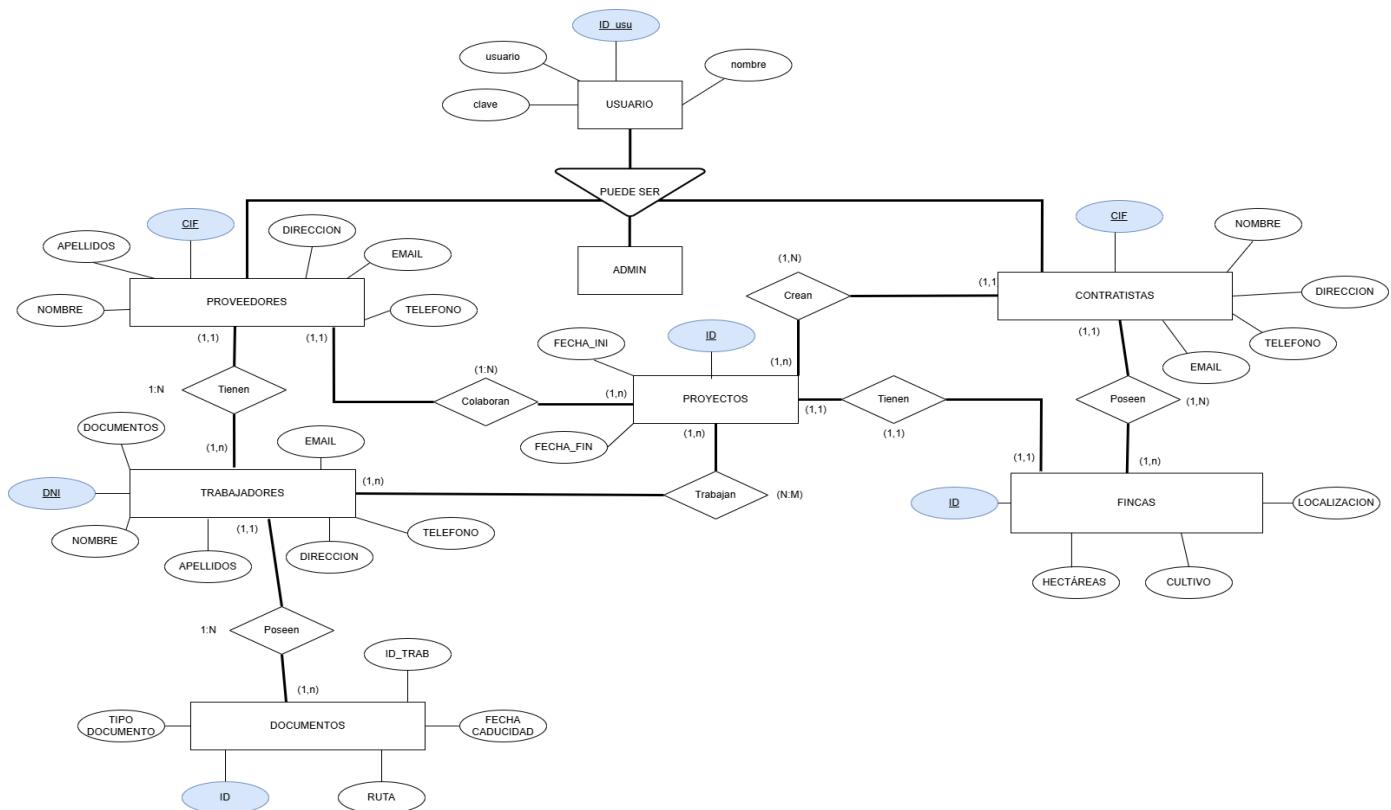
La base de datos de *Agrodoc* gestiona toda la información relacionada con los usuarios, las campañas agrícolas (proyectos), las fincas, los contratistas, proveedores,

trabajadores y documentos.

Su estructura responde a las siguientes funcionalidades:

- Registrar y autenticar usuarios (login).
- Relacionar contratistas con proveedores.
- Gestionar fincas y sus características (ubicación, cultivo, hectáreas).
- Crear proyectos asociados a contratistas, proveedores y fincas.
- Asociar trabajadores a campañas específicas.
- Subir la documentación de los trabajadores y validar su fecha de caducidad.
- Visualizar, modificar y eliminar esta información desde la plataforma.

## MODELO ENTIDAD/RELACIÓN



La base de datos contendrá tablas clave como Contratistas, Proveedores, Fincas, Trabajadores, Proyectos y Documentos. Se va a trabajar con un Modelo de Datos Relacional

## RELACIONES:

1. Usuarios : Pueden ser de tipo admin, contratista o proveedor.
2. Proveedores y Trabajadores: Un proveedor tiene uno o más trabajadores, y

cada trabajador pertenece a un sólo proveedor.

3. Trabajadores y Documentos: Un documento pertenece a un trabajador, y cada trabajador posee uno o más documentos.
4. Contratistas y Fincas: Un contratista puede tener una o varias fincas, pero cada finca pertenece a un sólo contratista.
5. Proveedores y Proyectos: Un proveedor puede participar en múltiples proyectos, y cada proyecto tiene un proveedor.
6. Contratistas y Proyectos: Un contratista puede hacer varios proyectos, y cada proyecto lo hace un contratista.
7. Fincas y Proyectos: Una finca tiene un proyecto, y cada proyecto tiene una finca.
8. Proyectos y Trabajadores: Cada proyecto puede involucrar varios trabajadores, y un trabajador puede estar asignado a varios proyectos. La tabla Proyectos\_trabajadores gestiona esta relación de muchos a muchos.

## TABLAS:

PROVEEDORES	
PK	<u>id_prov int AUTO_INCREMENT NOT NULL</u>
nombre	VARCHAR(20)
apellidos	VARCHAR(60)
cif	VARCHAR(20)
email	VARCHAR(50)
telefono	VARCHAR(20)
direccion	VARCHAR(100)

USUARIOS	
PK	<u>id_usu int AUTO_INCREMENT NOT NULL</u>
usuario	VARCHAR(30) UNIQUE NOT NULL
clave	VARCHAR(64) NOT NULL
nombre	VARCHAR(30)
tipo	ENUM('admin', 'contratista', 'proveedor') NOT NULL
FK	<u>id_cont</u>
FK	<u>id_prov</u>

CONTRATISTAS	
PK	<u>id_cont int AUTO_INCREMENT NOT NULL</u>
nombre	VARCHAR(20)
cif	VARCHAR(20)
email	VARCHAR(50)
telefono	VARCHAR(20)
direccion	VARCHAR(100)

TRABAJADORES	
PK	<u>id_trab int AUTO_INCREMENT NOT NULL</u>
nombre	VARCHAR(20)
apellidos	VARCHAR(60)
dni	VARCHAR(20)
email	VARCHAR(50)
telefono	VARCHAR(20)
direccion	VARCHAR(100)
documentos	BOOLEAN
FK	<u>id_prov</u>

PROYECTOS	
PK	<u>id_proyec int AUTO_INCREMENT NOT NULL</u>
trabajo	VARCHAR(100)
fecha_inicio	DATE
fecha_fin	DATE
FK	<u>id_cont</u>
FK	<u>id_prov</u>
FK	<u>id_finca</u>

FINCAS	
PK	<u>id_finca int AUTO_INCREMENT NOT NULL</u>
localizacion	VARCHAR(100)
cultivo	VARCHAR(50)
hectarea	int
FK	<u>id_cont int NOT NULL</u>

DOCUMENTOS	
PK	<u>id_doc int AUTO_INCREMENT NOT NULL</u>
tipo_documento	ENUM('dni', 'alta_ss', 'pri', 'reconocimiento_medico', 'aut_maquinaria')
ruta_archivo	VARCHAR(255) NOT NULL
fecha_caducidad	DATE
FK	<u>id_trab</u>

PROYECTOS_TRABAJADORES	
PK	<u>id_proyecto_trabajador int AUTO_INCREMENT NOT NULL</u>
FK1	<u>id_trab</u>
FK1	<u>id_proyec</u>

El archivo con el script para crear la base de datos se encuentra en el directorio database, en la carpeta mysql. El archivo se llama esquema.sql:

```

CREATE DATABASE IF NOT EXISTS bdagrodoc;
USE bdagrodoc;

-- Tabla de Contratistas
CREATE TABLE contratistas (
    id_cont INTEGER AUTO_INCREMENT NOT NULL,
    nombre VARCHAR(60),
    cif VARCHAR(20),
    email VARCHAR(50),
    telefono VARCHAR(20),
    direccion VARCHAR(100),
    PRIMARY KEY (id_cont)
);

-- Tabla de Fincas
CREATE TABLE fincas (
    id_finc INTEGER AUTO_INCREMENT NOT NULL,
    localizacion VARCHAR(100),
    cultivo VARCHAR(50),
    hectarea INTEGER,
    id_cont INTEGER NOT NULL,
    PRIMARY KEY (id_finc),
    FOREIGN KEY (id_cont) REFERENCES contratistas(id_cont) ON UPDATE CASCADE ON DELETE CASCADE
);

```

El fichero ‘config.ini’, se encuentra en la carpeta config, en el directorio raíz del proyecto, donde tiene la configuración del servidor, usuario, clave y base de datos:

```

[Configuración BD]
server=db
base=bdagrodoc
usu=mariá
pas=1234

```

Para el manejo de la BD he usado PDO como se requiere.

Para ello he creado el fichero base\_de\_datos.php, que se encuentra en la carpeta models del directorio src en la raíz del proyecto, y se encarga de hacer la conexión a la base BD

```

<?php
class basedatos {
    public $conn;

    public function __construct()
    {
        // Verificamos si el archivo config.ini existe
        $configFile = __DIR__ . '/../../config/config.ini';
        if (!file_exists($configFile)) {
            die("Error: El archivo config.ini no se encuentra en la ruta: $configFile");
        }

        // Cargamos la configuración desde config.ini
        $config = parse_ini_file($configFile);
        if (!$config) {
            die("Error: No se pudo leer el archivo config.ini");
        }

        // Intentamos la conexión a MySQL con PDO
        try {
            $dsn = "mysql:host={$config['server']};dbname={$config['base']};charset=utf8mb4";
            $this->conn = new PDO($dsn, $config['usu'], $config['pas'], [
                PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
                PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8mb4 COLLATE utf8mb4_unicode_ci"
            ]);
        } catch (PDOException $ex) {
            die("Error de conexión a MySQL: " . $ex->getMessage());
        }
    }

    public function ejecutarConsulta($sql)
    {
        try {
            return $this->conn->query($sql);
        } catch (PDOException $ex) {
            die("Error ejecutando consulta: " . $ex->getMessage());
        }
    }

    public function __destruct()
    {
    }
}

```

# GESTIÓN DEL BACK-END

## CODIFICACIÓN

El lenguaje utilizado para la gestión del back-end es PHP. Para evitar errores relacionados con caracteres especiales o codificaciones incorrectas, en el archivo basededatos.php se ha especificado el uso de la codificación utf8mb4 al establecer la conexión con la base de datos.

```
//Intentamos la conexión a MySQL con PDO
try {
    $dsn = "mysql:host={$config['server']};dbname={$config['base']};charset=utf8mb4";
    $this->conn = new PDO($dsn, $config['usu'], $config['pas'], [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8mb4 COLLATE utf8mb4_unicode_ci"
    ]);
} catch (PDOException $ex) {
    die("Error de conexión a MySQL: " . $ex->getMessage());
}
```

## ARCHIVO LOGIN.PHP

El archivo login.php es el que implementa el proceso de autenticación en la plataforma. Se encuentra en la carpeta *views* del directorio *src*.

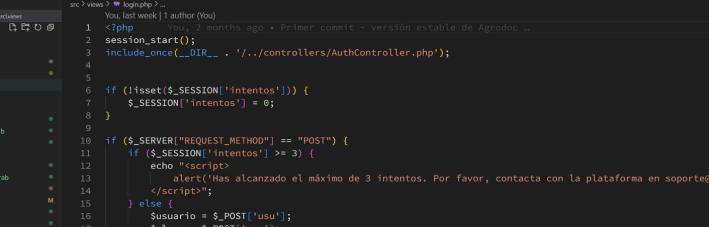
Valida las credenciales del usuario y controla los intentos fallidos.

Se comienza con la variable de sesión *session\_start()*, que se usan en todas las páginas de PHP.

Se inicia un contador para controlar el número de intentos de inicio de sesión fallidos. En caso de superar tres intentos, se bloquea temporalmente el acceso y se sugiere contactar con el soporte técnico.

El formulario se envía mediante el método POST, se recogen los valores introducidos por el usuario y se envían al controlador de autenticación AuthController.php, que gestiona la validación en la base de datos.

Si la autenticación ha sido exitosa, se guarda el objeto del usuario en la sesión utilizando *serialize()* y se redirige automáticamente a la página principal de la aplicación /app/. Si ha sido fallida, se incrementa el contador de intentos y se informa al usuario con un mensaje indicando cuántos intentos ha realizado. Tiene tres intentos para acceder y si los utiliza, se le bloquea los intentos y se le avisa que se ponga en contacto con soporte de la plataforma por un email.



The screenshot shows a code editor with the file `login.php` open. The code is a PHP script that handles user authentication. It includes session management, a counter for failed login attempts, and a maximum attempt limit of 3. If the attempt limit is reached, it displays a message encouraging users to contact support. The code uses `AuthController` to handle the login logic.

```
<?php
You, 2 months ago * Primer commit - versión estable de Agrodoc ...
session_start();
include_once(__DIR__ . '/../controllers/AuthController.php');

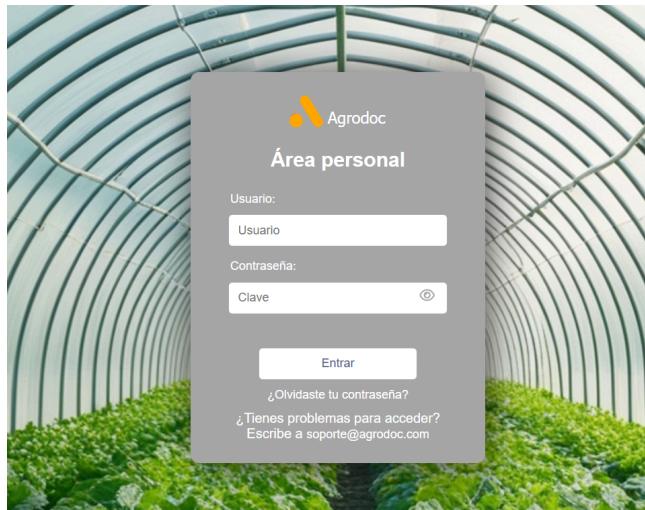
if (isset($_SESSION['intentos'])) {
    $_SESSION['intentos'] = 0;
}

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    if ($_SESSION['intentos'] >= 3) {
        echo "<script>\n            alert('Has alcanzado el m\u00f3ximo de 3 intentos. Por favor, contacta con la plataforma en soporte@agrodoc.com.')";
        </script>";
    } else {
        $usuario = $_POST['usu'];
        $clave = $_POST['pas'];

        $auth = new AuthController();
        $datosdeusuario = $auth->login($usuario, $clave);

        if (empty($datosdeusuario)) {
            $_SESSION['intentos'] = 0;
            $_SESSION['usuario'] = serialize($datosdeusuario);
            session_write_close();
            header("Location: /app/");
            exit;
        } else {
            $_SESSION['intentos]++;
            echo "<script>\n                alert('Usuario o contrase\u00f1a incorrectos. Intento (" . $_SESSION['intentos'] . ") de 3.'");
            </script>";
        }
    }
}
```

## Vista del formulario de validación:

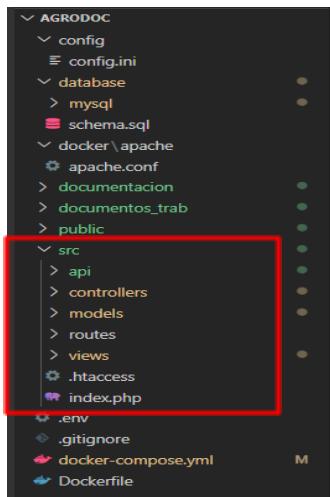


Mensaje en una ventana de alerta, de los intentos que ha utilizado en caso de validación errónea:



Como la aplicación es por suscripción, el usuario no tiene forma de registrarse. Se tiene que poner en contacto con la plataforma, a través del formulario de contacto o a través del email, y el administrador le da el alta como usuario, según el tipo que sea (contratista o proveedor). El usuario tipo administrador es el único que tiene acceso a la creación de usuario.

Toda la lógica se encuentra desarrollada en PHP. Su estructura está organizada en carpetas, siendo la más relevante para esta capa la carpeta src/controllers, donde se encuentran los controladores de cada entidad principal: contratistas, fincas, trabajadores, proveedores, proyectos, etc..



Cada controlador utiliza una clase específica y se conecta a la base de datos a través del archivo basededatos.php, ubicado en src/models. Este archivo implementa una clase que gestiona la conexión con MySQL usando PDO, estableciendo la codificación utf8mb4 para garantizar compatibilidad con caracteres especiales y acentos.

## MÉTODOS CRUD

Cada clase de controlador incluye métodos que implementan las operaciones CRUD (Crear, Leer, Actualizar, Eliminar). Por ejemplo, el archivo ProyecController.php, que gestiona los proyectos (campañas), incluye métodos como:

- `getProyectos()` — Recupera todos los proyectos.
- `getProyectosPorContratista($id)` — Devuelve proyectos filtrados por contratista.
- `getProyectosPorProveedor($id)` — Devuelve proyectos filtrados por proveedor.
- `crearProyecto($datos)` — Inserta un nuevo proyecto.
- `modificarProyecto($datos)` — Modifica un proyecto existente.
- `eliminarProyecto($id)` — Elimina un proyecto por ID.

Estos métodos reciben los datos desde llamadas AJAX, normalmente en formato JSON, y devuelven respuestas también en formato JSON que el front-end interpreta dinámicamente.

## Acceso por tipo de usuario

La plataforma distingue entre tres tipos de usuarios:

Tipo de Usuario	Acciones Permitidas
<b>Administrador</b>	Ver, crear, modificar y eliminar contratistas, proveedores, usuarios, fincas, etc.
<b>Contratista</b>	Ver y gestionar fincas, proyectos y asignación de proveedores.
<b>Proveedor</b>	Ver campañas (proyectos) asignadas, registrar trabajadores y subir documentación.

Esto se gestiona utilizando sesiones PHP (`$_SESSION['usuario']`) y, dependiendo del tipo (admin, contratista, proveedor), se cargan distintas vistas y se restringe el acceso a funciones.

## Ejemplo CRUD de Proyectos (Campañas)

### 1. Crear Proyecto

- En el front-end, el usuario completa un formulario que recoge nombre, fecha, contratista, proveedor y finca.
- Se envían vía fetch en formato JSON a la ruta `/api/proyectos?action=crearProyecto`.
- El método `crearProyecto($datos)` en `ProyecController.php` ejecuta una consulta `INSERT INTO proyectos`.

Solo el administrador y el contratista son los que pueden crear proyectos.

```
96 <?php if ($tipo === 'admin' || $tipo === 'contratista'): ?>
97     <div class="boton_crear">
98         <a href="javascript:cargar('#portada','/views/nuevo_proyec.php');">
99             <button>Nueva campaña</button>
100        </a>
101    </div>
102 <?php endif; ?>
103
```

## 2. Ver Proyectos

- Las tablas se generan dinámicamente usando JavaScript y datos obtenidos por AJAX (listarProyectos).
- El controlador ejecuta la consulta SELECT correspondiente y devuelve los proyectos en JSON.

Pueden verlos los tres usuarios, pero cada uno tiene unas vistas diferentes:

```
35 |     <thead>
36 |         <tr>
37 |             <th class="ocultar-sm">ID Campaña</th>
38 |             <th>Tipo de trabajo</th>
39 |             <th>Finca</th>
40 |             <th>Ver en mapa</th>
41 |             <th>Tipo de cultivo</th>
42 |             <?php if ($tipo === 'admin' || $tipo === 'proveedor'): ?>
43 |                 <th>Contratista</th>
44 |             <?php endif; ?>
45 |             <?php if ($tipo === 'admin' || $tipo === 'contratista'): ?>
46 |                 <th>Proveedor</th>
47 |                 <th>Trabajadores</th>
48 |             <?php endif; ?>
49 |             <th>Fecha Inicio</th>
50 |             <th>Fecha Fin</th>
51 |             <?php if ($tipo === 'admin' || $tipo === 'contratista'): ?>
52 |                 <th>Modificar</th>
53 |                 <th>Eliminar</th>
54 |             <?php endif; ?>
55 |         </tr>
56 |     </thead>
```

## 3. Modificar Proyecto

- Al hacer clic en "modificar", las celdas se convierten en campos editables.
- Al guardar, se envían los nuevos datos al endpoint modificarProyecto.
- El método en el controlador actualiza la base de datos con una consulta UPDATE.

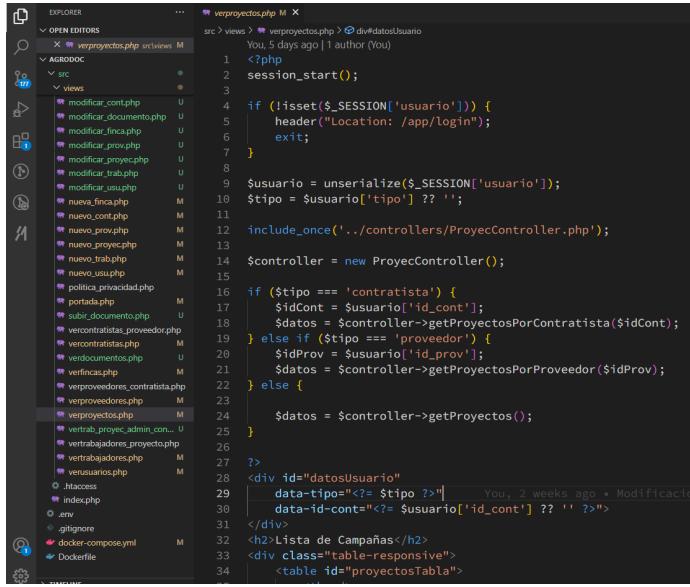
## 4. Eliminar Proyecto

- Al pulsar "eliminar", se muestra una confirmación y luego se llama a eliminarProyecto.
- Este método ejecuta un DELETE y devuelve un mensaje de éxito o error.

Solo los administradores y contratistas:

```
<?php if ($tipo === 'admin' || $tipo === 'contratista'): ?>
    <td>
        <a href="javascript:cargar('#portada','/views/modificar_proyec.php?id=<?= $proyecto['id_proyec'] ?>")">
            
        </a>
    </td>
    <td>
        <a href="javascript:eliminarProyecto(<?= $proyecto['id_proyec'] ?>)">
            
        </a>
    </td>
<?php endif; ?>
```

Otros métodos como getProyectosPorContratista(\$id) o getProyectosPorProveedor(\$id), son usados según el tipo de usuario.



```

<?php
if (!isset($_SESSION['usuario'])) {
    header("Location: /app/login");
    exit;
}

$usuario = unserialize($_SESSION['usuario']);
$tipo = $usuario['tipo'] ?? '';
include_once('../controllers/ProyecController.php');

if ($tipo === 'contratista') {
    $idCont = $usuario['id_cont'];
    $datos = $controller->getProyectosPorContratista($idCont);
} else if ($tipo === 'proveedor') {
    $idProv = $usuario['id_prov'];
    $datos = $controller->getProyectosPorProveedor($idProv);
} else {
    $datos = $controller->getProyectos();
}

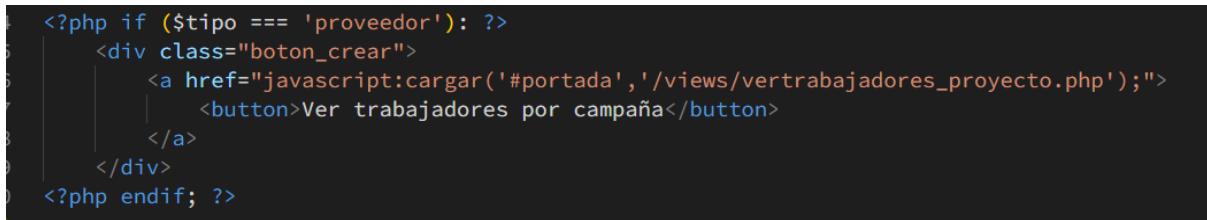


You, 5 days ago | author (You)
    

<div>-Lista de Campañas</h2>
        <div class="table-responsive">
            <table id="proyectosTabla">
                ...
            </table>
        </div>
    </div>


```

Los proveedores tendrán un enlace a otra página para ver trabajadores por proyecto:

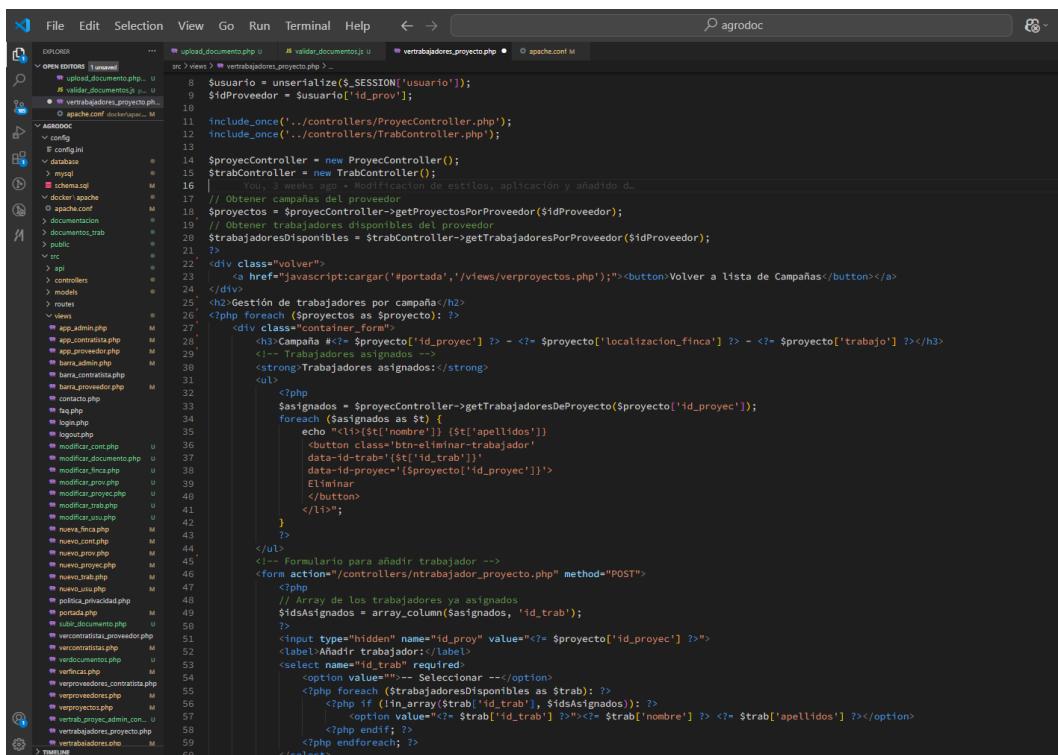


```

<?php if ($tipo === 'proveedor'): ?>
    <div class="boton_crear">
        <a href="javascript:cargar('#portada','/views/vertrabajadores_proyecto.php');">
            <button>Ver trabajadores por campaña</button>
        </a>
    </div>
<?php endif; ?>

```

En esta vista, puede ver los trabajadores por campaña activa, y añadir o eliminar trabajadores:



```

<?php
if ($tipo === 'proveedor'): ?>
    <div class="boton_crear">
        <a href="javascript:cargar('#portada','/views/vertrabajadores_proyecto.php');">
            <button>Ver trabajadores por campaña</button>
        </a>
    </div>
<?php endif; ?>

<?php
$usuario = unserialize($_SESSION['usuario']);
$idProveedor = $usuario['id_prov'];
$proveedorController = new ProveedorController();
$trabajadorController = new TrabController();
$trabajadoresDisponibles = $trabajadorController->getTrabajadoresPorProveedor($idProveedor);
$trabajadoresAsignados = $trabajadorController->getTrabajadoresDeProyecto($proveedorController->getProyecto($idProveedor));
$proyecto = $proveedorController->getProyectoPorProveedor($idProveedor);

// Obtener trabajadores disponibles del proveedor
$trabajadoresDisponibles = $trabajadorController->getTrabajadoresPorProveedor($idProveedor);

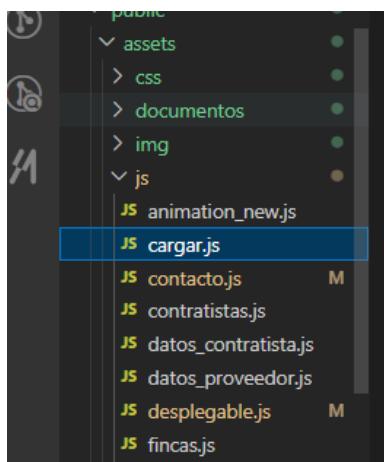
// Obtener trabajadores asignados del proveedor
$trabajadoresAsignados = $trabajadorController->getTrabajadoresDeProyecto($proveedorController->getProyecto($idProveedor));

// Formulario para añadir trabajador
<form action="/controllers/trabajador_proyecto.php" method="POST">
    <input type="hidden" name="id_proyec" value="=$proyecto['id_proyec']?" />
    <label>Añadir trabajador:</label>
    <select id="id_trab" required>
        <option value="">-- Seleccionar --</option>
        <?php foreach ($trabajadoresDisponibles as $trab): ?>
            <option value="=$trab['id_trab']?">=$trab['nombre'] . ' ' . $trab['apellidos']?"</option>
        <?php endforeach; ?>
    </select>
</form>

```

# GESTIÓN DEL FRONT-END

Toda la gestión del front-end está desarrollada en JavaScript, cumpliendo los requisitos del proyecto en cuanto al uso de métodos de acceso y manipulación del DOM.. Los archivos que contienen la codificación de JavaScript, están en el directorio publico > assets > js .

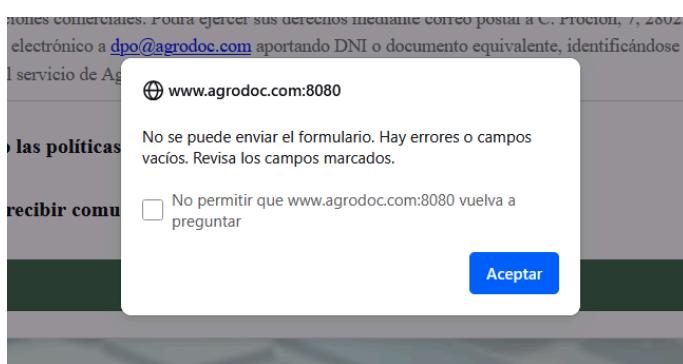


## VALIDACIÓN DE LOS CAMPOS DE FORMULARIOS

### FORMULARIO CONTACTO

La página de contacto, en la parte pública donde el usuario quiere ponerse en contacto con la plataforma, se validan todos los campos obligatorios, así como aquellos que deben cumplir formatos específicos mediante expresiones regulares (como el email, teléfono o CIF).

Si no está relleno un campo que es obligatorio o no es válido, primero sale una ventana de alerta para avisar que no se ha enviado el formulario por error de campos o por tener campos vacíos, y que se revisen los campos marcados.



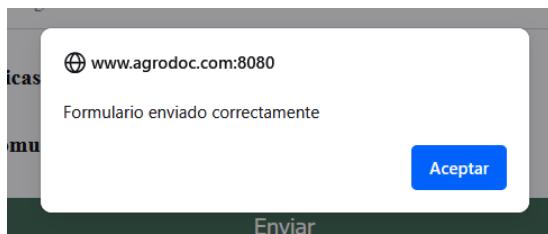
Los campos que son obligatorios se marcan con un borde rojo y sale el mensaje de que es un campo obligatorio, y si está erróneo también se indica con un mensaje, y el cursor se pone en el campo con el primer error.

The screenshot shows a contact form with three fields highlighted in red:

- Nombre de la empresa \***: The input field is empty, and a red border surrounds it. Below the field, the message "Este campo es obligatorio" is displayed.
- Correo corporativo \***: The input field contains "mtiscar@hotmail", and a red border surrounds it. Below the field, the message "Introduce un correo válido" is displayed.
- CIF de la empresa \***: The input field contains "B457118", and a red border surrounds it. Below the field, the message "Introduce un CIF válido, debe tener una letra y 7 dígitos" is displayed.

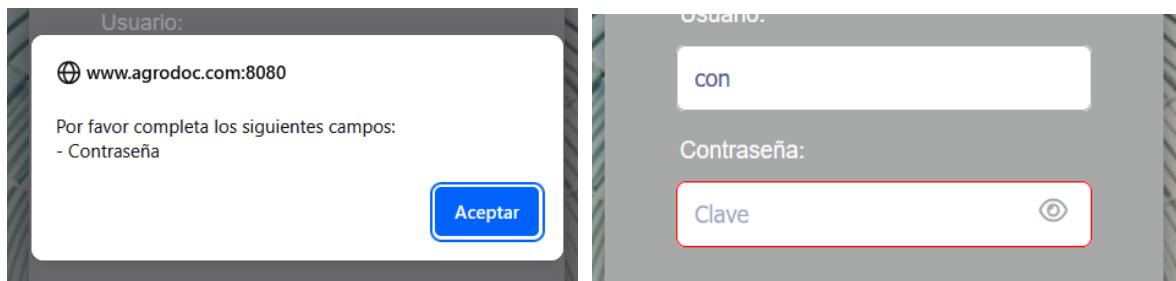
Archivo que lo valida se llama [contacto.js](#) la ruta donde se encuentran todos los archivos de JavaScript es: public > assets > js

Si el formulario ha sido completado correctamente, se muestra un mensaje de confirmación indicando que el envío se ha realizado con éxito.



### FORMULARIO LOGIN

También se valida el formulario de login. Si se deja algún campo vacío, se muestra un mensaje de advertencia indicando el campo afectado, el cual se marca con un borde rojo.



El archivo que valida este formulario se llama validacion\_login.js

## FORMULARIO WEBMINAR

Igual que en el formulario de contacto, si no está rellenado algún campo obligatorio se muestra un mensaje en ventana y una vez que aceptas se muestra los campos que se han quedado vacíos o si hay algún error en el correo.

The screenshot shows a modal dialog box with the following content:

Por favor, indica tu correo corporativo y el tema que te interesa.

**Correo corporativo \***

 Introduce un correo válido

**Tema que te interesa \***

 Este campo es obligatorio

**Aceptar**

**Enviar solicitud**

Below the modal, there is a message: "Hay errores o campos vacíos en el formulario."

El archivo que valida este formulario se llama form\_webinar.js

## FORMULARIO APP DE CREACIÓN Y MODIFICACIÓN DE REGISTROS

En todos los formularios de creación y modificación de registros se validan antes de ser mandados a la base de datos, para garantizar que todos los campos contienen datos válidos y con el formato esperado antes de ser enviados a la base de datos.

Si hay algún error, tanto por campos vacíos o por testear expresiones regulares no correctas, sale un mensaje debajo de cada campo, especificando el error.

Ejemplo de formulario de creación de nuevo usuario.

### Nuevo usuario

#### Crear nuevo usuario

Nombre de usuario:

El usuario es obligatorio.

Contraseña:

La contraseña debe tener al menos 8 caracteres, una mayúscula, una minúscula y un número.

Nombre completo:

El nombre es obligatorio.

Tipo de usuario:

 Debes seleccionar un tipo.

**Crear Usuario**

Todas las validaciones de estos formularios se hacen en cada archivo .js de nuevo y modificación de los diferentes registros en la base de datos.

## CREACIÓN DE ELEMENTOS DINÁMICOS USANDO ACCESO AL DOM

En varios puntos del sistema, se crean elementos dinámicamente mediante manipulación del DOM. Por ejemplo, en el formulario de contacto se generan contenedores para mostrar errores de validación, y en las tablas de registros (usuarios, contratistas, etc.) se insertan dinámicamente filas con datos obtenidos desde el servidor.

Archivo contacto.js:

```
// Función para crear los mensajes de error
function crearError(field, mensaje) {
    field.setAttribute('aria-invalid', 'true');
    field.style.border = '2px solid red';

    const errorDiv = document.createElement('div');
    errorDiv.className = 'error-dinamico';
    errorDiv.textContent = mensaje;
    errorDiv.style.color = 'red';
    errorDiv.style.fontSize = '0.9em';
    errorDiv.style.marginTop = '5px';

    field.parentNode.insertBefore(errorDiv, field.nextSibling);
}
```

Archivo contratista.js:

```
data.forEach(contratista => {
    let row = `
        <tr data-id="${contratista.id_cont}">
            <td class="ocultar-sm">${contratista.id_cont}</td>
            <td class="editable">${contratista.nombre}</td>
            <td class="editable">${contratista.cif}</td>
            <td class="editable">${contratista.email}</td>
            <td class="editable">${contratista.telefono}</td>
            <td class="editable">${contratista.direccion}</td>
            <td>
                <button onclick="editarContratista(${contratista.id_cont})">Modificar</button>
                <button style="display:none;" onclick="guardarContratista(${contratista.id_cont})">Guardar</button>
            </td>
            <td>
                <button onclick="eliminarContratista(${contratista.id_cont})">Eliminar</button>
            </td>
        </tr>
    `;
    tabla.innerHTML += row;
});
```

## PETICIONES ASÍNCRONAS

En el desarrollo del proyecto he utilizado los archivos JavaScript para la gestión dinámica de contenido, la interacción con el usuario y la comunicación con el servidor.

A lo largo del desarrollo se han implementado peticiones asíncronas al servidor utilizando la API fetch() de JavaScript. Estas peticiones permiten la creación,

modificación o eliminación de registros como contratistas, proveedores, proyectos, trabajadores, usuarios, fincas y documentos, sin necesidad de recargar la página.

Los scripts que utilizan fetch() son para realizar operaciones como la creación, modificación o eliminación de entidades (contratistas, proveedores, proyectos, trabajadores, usuarios, fincas y documentos). Estas acciones se ejecutan de forma asíncrona, y la información obtenida se utiliza para actualizar dinámicamente tablas o formularios en el DOM.

Archivos con fech(): [usuarios.js](#), [trabajadores.js](#), [contratistas.js](#), [proveedores.js](#), [fincas.js](#), [proyectos.js](#), nuevo\_usu.js, nuevo\_trab.js..., modificar\_usu.js, modificar\_trab.js....

Ejemplo de archivo con peticiones fetch(), de [usuario.js](#) para eliminar un usuario:

```
// Eliminar usuario
function eliminarUsuario(id) {
    if (confirm("¿Estás seguro de que deseas eliminar este usuario?")) {
        fetch(`controllers/UserController.php?action=eliminarUsuario&id=${id}`, {
            method: "GET",
        })
        .then((response) => response.json())
        .then((data) => {
            if (data.mensaje) {
                alert(data.mensaje);
                console.log(
                    "Usuario eliminado correctamente. Actualizando la lista..."
                );
                cargarUsuarios();
            } else {
                alert("Error al eliminar usuario: " + data.error);
            }
        })
        .catch((error) => console.error("Error al eliminar usuario:", error));
    }
}
```

En el script cargar.js se emplea la función \$.ajax() de jQuery para insertar fragmentos HTML de forma dinámica en la interfaz.

```
You, 4 weeks ago | 1 author (You)
1 function cargar(contenedor, url) {
2
3     $.ajax({
4         url: url,
5         type: 'GET',
6         success: function (data) {
7             $(contenedor).html(data);
8             console.log(`Contenido cargado correctamente desde ${url}`);
9         },
10        error: function (xhr, status, error) {
11            console.error(`Error al cargar ${url}:`, error);
12        }
13    });
14}
15
```

Además, se ha hecho uso de APIs públicas en el archivo ver\_mapa.js: una para obtener coordenadas geográficas a partir de direcciones de texto (DistanceMatrix.ai), y otra para obtener la dirección desde coordenadas (Nominatim de OpenStreetMap), integradas en un mapa interactivo mediante la librería Leaflet.

```
//Obtener dirección a partir de coordenadas
function obtenerDireccion(lat, lon) {
    fetch(`https://nominatim.openstreetmap.org/reverse?lat=${lat}&lon=${lon}&format=json`)
        .then(response => response.json())
        .then(data => {
            let direccion = data.display_name || "Ubicación desconocida";
            crearMarcador(lat, lon, direccion);
        })
        .catch(error => {
            console.error('Error al obtener la dirección:', error);
        });
}
```

```
//Obtener coordenadas desde la API
function obtenerCoordenadas(direccion) {
    const key = "hdKDIQjbYjuRZa73PzdkomXAvyIrlHmFR0REdd2nr0KDzdUR1xhMBQnV4IUIqAsh";

    console.log(`Buscando ubicación para: ${direccion}`);
    fetch(`https://api.distancematrix.ai/maps/api/geocode/json?address=${encodeURIComponent(direccion)}&key=${key}`)
        .then(response => response.json())
        .then(data => {
            console.log(`\u2708 Respuesta de la API:`, data);
            if (data.status === "OK" && data.result.length > 0) {
                let lat = data.result[0].geometry.location.lat;
                let lon = data.result[0].geometry.location.lng;
                let dir = direccion;
                crearMarcador(lat, lon, dir);
            } else {
                alert("No se encontraron coordenadas.");
            }
        })
        .catch(error => console.error('Error al obtener coordenadas:', error));
}
```

## INTERFACES WEB, USABILIDAD Y ACCESIBILIDAD

Las interfaces se han diseñado con HTML5 y CSS3, utilizando Flexbox como sistema principal de maquetación. Esto permite crear diseños flexibles y adaptativos, facilitando la distribución de los elementos en pantalla de manera eficiente.

Los archivos .css se encuentran en el directorio *public > assets > css*

✓ public	•	14
✓ assets	•	15
✓ css	•	16
# blog.css		17
# estilos_app_usuari... M		18
# estilos_app.css M		19
# estilos_tv_proveedor.css		20
# estilos_tv.css M		21
# faq.css		22
# form_login.css		23 }
# formulario.css M		24
# plataforma.css		25 b
# privacidad.css		26
# sobre_nosotros.css U		27
# software.css M		28
# soluciones.css		29
# tarifa.css		
# webminar.css M		

La organización del CSS está dividida por perfiles y funcionalidades: por ejemplo, estilos\_tv.css y estilos\_tv\_proveedor.css para la parte pública, y estilos\_app.css o estilos\_app\_usuarios.css para los usuarios registrados (administradores, contratistas, proveedores)..

Luego dependiendo del contenido de la página, he creado diferentes archivos css para poder usarlos independientemente de si están en la parte pública o de aplicación. Por ejemplo estilos para el formulario de contacto, para las tablas de tarifas, para el formulario de login....

jQuery se ha utilizado para proporcionar dinamismo adicional en la interfaz. Los diferentes sitios usados son:

1- Para la animación del aviso NEW, para destacar un enlace a una página con nuevo contenido, con un ligero movimiento para llamar la atención. El archivo se llama animation\_new.js.

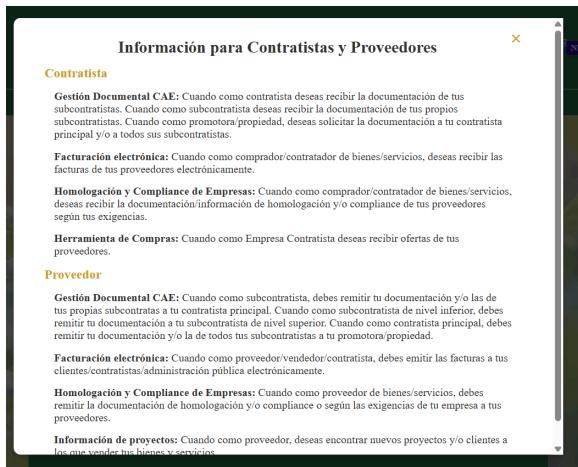
```
public > assets > js > js animation_new.js > ...
You, 2 months ago | 1 author (You)
1  function animateBadge() {
2    $('#newBadge').animate({
3      top: '55%'
4    }, 500).animate({
5      top: '50%'
6    }, 500, animateBadge);
7  }
8
9  animateBadge();
10
```



2- Para el icono de información de los diferentes perfiles de usuario. Cuando se pincha en el ícono, aparece una modal informativa.

```
public > assets > js > js icon_informacion.js > ...
You, 1 second ago | 1 author (You)
1 //Para abrir la modal cuando se hace clic en el ícono de información
2 ~ $('.informacion a').on('click', function(event) {
3   event.preventDefault();
4   $('#infoModal').fadeIn();
5 });
6 //Para cerrar la modal cuando se hace clic en la X
7 ~ $('.close').on('click', function() {
8   $('#infoModal').fadeOut();
9 });
10
11 //Para cerrar la modal si se hace clic fuera del contenido
12 ~ $(window).on('click', function(event) {
13   if ($(event.target).is('#infoModal')) {
14     $('#infoModal').fadeOut();
15   }
16});
```





3- Para el vídeo de conocer la plataforma en 2 minutos. Cuando se pincha en el enlace, aparece una modal en el centro de pantalla para ver el vídeo.

```
# estilos_tv.css M index.html M manual_uso.html M JS pestañas.js M
public > assets > js > JS pestañas.js > ...
You, 18 seconds ago | 1 author (You)
  1 $(document).ready(function () {
  2   $(".pestaña").on("click", function () {
  3     $(".pestaña").removeClass("active");
  4     $(".tab-content").removeClass("active");
  5
  6     $(this).addClass("active");
  7     const tabId = $(this).data("tab");
  8     $(`#tab-${tabId}`).addClass("active");
  9   });
 10 });
 11 |
```

**Tranquilidad en tu cadena de suministro y subcontratación**

Únete sin coste\* como contratista a la nube inteligente de soluciones de Agrodoc, y optimiza la relación con tus proveedores a lo largo de las fases de tu cadena de suministro y subcontratación.

**Conócenos en 2 minutos** **Solicita más información**



#### 4- Para el comportamiento del menú de navegación según el ancho de pantalla:

```
public > assets > js > menu.js > $() callback > $(window).resize() callback
You, 49 seconds ago | 1 author (You)
1  $(function(){
2    var anchura = $(this).width();
3    $('#menuHamburguesa').click(function(){
4      $('#nav').toggle();
5    });
6    $(window).resize(function(){
7      var anchura = $(window).width();
8      if(anchura > 767){
9        $('#nav').show();
10      }else {
11        $('#nav').hide();
12      }
13    });
14    if(anchura > 992){
15      // función para submenu del navegador
16      $('.dropdown').hover(function(){
17        $(this).find('.submenu').stop(true, true).slideDown(200);
18      }, function(){
19        $(this).find('.submenu').stop(true, true).slideUp(200);
20      });
21    }
22  });
23
```

#### 5- Para las preguntas de la página FAQ. Se muestra la respuesta si le das clic a la pregunta seleccionada:

```
$(function() {
  $("#p01").click( function() {
    $("#r01").slideToggle('slow');
  });
  $("#p02").click( function() {
    $("#r02").slideToggle('slow');
  });
  $("#p03").click( function() {
    $("#r03").slideToggle('slow');
  });
  $("#p04").click( function() {
    $("#r04").slideToggle('slow');
  });
  $("#p05").click( function() {
    $("#r05").slideToggle('slow');
  });
  $("#p06").click( function() {
    $("#r06").slideToggle('slow');
  });
});
```

The screenshot shows a section titled "Consulta nuestras preguntas frecuentes". Below it is a dropdown menu with the question "¿Qué es Agrodoc?". A tooltip or dropdown content below the menu says: "Agrodoc es una plataforma que conecta a empresas agrícolas y proveedores para facilitar el intercambio de documentación y servicios.".

## RESPONSIVE:

Para que las páginas sean responsive, he definido dos puntos de ruptura (*breakpoints*) mediante Media Queries: uno para pantallas medianas (576px a 768px) y otro para dispositivos pequeños (<576px), los cuales están detallados en la guía de estilos.

## **OPTIMIZACIÓN DE IMÁGENES:**

Aunque el sitio no contiene un gran número de imágenes, las que se utilizan (principalmente como fondos decorativos o apoyo visual) han sido optimizadas para mantener un peso reducido, no superior a 250 KB. Se ha utilizado un compresor de imágenes para reducir el tamaño sin pérdida perceptible de calidad.

Se han empleado formatos adecuados como SVG, JPG, PNG y JPEG, seleccionados según el tipo de contenido. Además, se han ajustado las dimensiones de las imágenes según su uso real en pantalla, evitando escalado innecesario y mejorando así el rendimiento general de carga..