

**GRUPO STUDIUM**

**Instituto Técnico de Grado Superior en Informática**

**Promoción 2024/2025**

**Proyecto: Aplicación Web de Calendario de  
Gestión de eventos**

**PROYECTO INTEGRADO PARA OBTENER EL GRADO DE:  
Desarrollo de Aplicaciones Web**

**Presenta: José María Ortiz Román**

**Tutor de Proyecto: Mariano**

**Sevilla, junio de 2025**

<b>1. Estudio preliminar del problema.....</b>	<b>3</b>
1.1. Descripción preliminar del sistema.....	3
1.2. Funcionalidades básicas.....	3
1.3. Viabilidad del sistema.....	4
1.4. Estimación (inicial) del coste del sistema.....	4
1.4.1. Recursos humanos.....	4
1.4.2. Recursos hardware.....	4
1.4.3. Recursos software.....	5
<b>2. Análisis y diseño de la aplicación.....</b>	<b>5</b>
2.1. Datos.....	5
2.1.1. Esquema/s entidad-relación.....	6
2.1.2. Traslado a un esquema relacional normalizado.....	6
2.1.3. Elección justificada de un SGBD relacional concreto.....	7
2.1.4. Traslado a un esquema físico relacional.....	7
2.1.5. Restricciones no implementables en el modelo de datos que deberá contemplar la aplicación.....	7
2.2. Funciones.....	8
2.2.1. Diagramas de flujo de datos.....	8
2.2.2. Diccionario de datos.....	11
2.2.3. Descripción de Clases, Métodos, Atributos, .....	13
<b>4. Manuales.....</b>	<b>18</b>
4.1. Manual técnico.....	18
4.2. Manual de instalación.....	20
4.3. Manual de usuario.....	21
4.4. Manual de administración.....	23
<b>5. Conclusiones.....</b>	<b>27</b>
5.1. Grado de consecución de los objetivos fijados.....	27
5.2. Posibles mejoras o ampliaciones para implementar en el futuro.....	27
<b>6. Bibliografía.....</b>	<b>27</b>

# 1. Estudio preliminar del problema

## 1.1. Descripción preliminar del sistema

Aplicación web que publica eventos. Los usuarios pueden visitar la página y consultar los días del mes que tienen registros, se les mostrará información relevante. Contará con una página de inicio de sesión para que los administradores, usando sus credenciales, accedan a la vista de gestión donde podrán hacer las funciones CRUD a través de un menú intuitivo. Los datos que se guardarán de los eventos son el título, descripción, fecha e imagen.

Se usarán las siguientes tecnologías: HTML, CSS, Bootstrap, jQuery, PHP y AJAX.

- HTML, CSS y Bootstrap: estas tres herramientas se utilizarán para construir y diseñar la interfaz visual de la aplicación. HTML se encargará de la estructura del contenido, CSS dará estilo y apariencia a los elementos, y Bootstrap permite un diseño responsive y adaptable a cualquier dispositivo.
- jQuery: se utilizará para modificar los elementos del DOM, gestionar eventos y realizar peticiones AJAX. Permitirá que la interfaz reaccione de forma dinámica sin recargar la página.
- AJAX: se usará junto con jQuery para enviar y recibir peticiones de datos desde el servidor de forma asíncrona. Esto hace posible consultar eventos, añadirlos o editarlos sin necesidad de recargar toda la página.
- PHP: será el lenguaje de programación del servidor. Se encargará de procesar las solicitudes enviadas por AJAX, conectarse a la base de datos, ejecutar las operaciones CRUD (crear, leer, actualizar y eliminar eventos) y gestionar las sesiones de usuario (por ejemplo, el control del rol de administrador).

## 1.2. Funcionalidades básicas

Calendario mensual donde el usuario puede navegar y donde los días con eventos registrados se marcarán para destacar. Cuando se quiera consultar un día con registros se desplegará un listado de eventos, junto a sus títulos, descripciones e imágenes. Se hace una distinción entre visitantes y administradores, donde los primeros sólo podrán consultar los registros; mientras, los segundos tienen un panel solo accesible a ellos para crear eventos, actualizarlos y borrarlos. Cuenta con sistema de inicio de sesión y de registro. La actualización de la información se hace siempre sin recargar la página usando peticiones AJAX.

### 1.3. Viabilidad del sistema

El proyecto es totalmente viable ya que usa tecnologías web con un amplio recorrido y que por lo tanto tienen soporte y a futuro seguirán siendo utilizadas. PHP 8, MySQL, JavaScript + jQuery y Bootstrap son tecnologías aceptadas por las páginas de hosting. No se necesita ninguna licencia de pago ni suscripción ya que todo el contenido es libre y gratuito. Lo único que podría costar dinero es el dominio y el hosting.

### 1.4. Estimación (inicial) del coste del sistema

Para hacer un cálculo estimado del coste inicial tendremos en cuenta: el coste humano, el coste de hardware y el de software.

Recursos humanos: estimaremos unas 80 horas de desarrollo, usando la estimación de que de media un junior cobra en torno a 13 euros la hora, aproximadamente se gastarán 1040 euros.

Recursos de hardware: no tendremos en cuenta el equipo en el que se trabaja porque se da por hecho que ya se tiene. Contaremos el gasto de un VPS básico con estas características: 1 vCPU, 1 GB RAM, 25 GB SSD. Son suficientes para un tráfico moderado y así hacemos los backups automáticos. Esto resultaría en unos 120 euros anuales.

Recursos de software: debido a que todo el stack usado es open source no gastaremos dinero en las tecnologías. Tan solo contaremos el pago anual del dominio, teniendo en cuenta que un .es cuesta en torno a 12 euros.

En total, el precio por el desarrollo y mantenimiento estaría en 1172 euros.

#### 1.4.1. Recursos humanos

Para hacer esta aplicación web solo se ha necesitado a un programador junior. Una sola persona ha llevado la creación de base de datos, desarrollo de la aplicación tanto en el front como en el back. También ha escogido el stack.

#### 1.4.2. Recursos hardware

Procesador: Intel® Core™ i7-3770 CPU a 3.40 GHz, con 4 núcleos físicos.

Memoria RAM: 16 GB de memoria física instalada.

Almacenamiento: Disco duro 704GB(HDD).

Sistema operativo: Microsoft Windows 11 Pro, versión 10.0.26100.

Plataforma: PC de escritorio HP Compaq Pro 6300 SFF.

### 1.4.3. Recursos software

Sistema operativo: Microsoft Windows 11 Pro.

Entorno de desarrollo: Visual Studio Code.

Gestión de versiones: GitHub Desktop.

Base de datos: MySql en phpMyAdmin.

## 2. Análisis y diseño de la aplicación

### 2.1. Datos

#### **Entidades y Atributos.**

##### **Tabla usuarios**

id (ID de usuario) – Clave primaria, autoincremental.

nombre (nombre del usuario)

telefono (teléfono de contacto)

correo (correo electrónico del usuario)

contrasena (contraseña cifrada del usuario)

rol (rol asignado al usuario: puede ser admin o usuario)

##### **Tabla agenda**

id (ID del evento) – Clave primaria, autoincremental.

titulo (título del evento)

descripcion (descripción del evento)

enlace (URL opcional relacionada con el evento)

fecha\_evento (fecha y hora del evento)

imagenes (ruta de la imagen asociada al evento)

created\_at (fecha y hora de creación del evento)

updated\_at (fecha y hora de la última modificación)

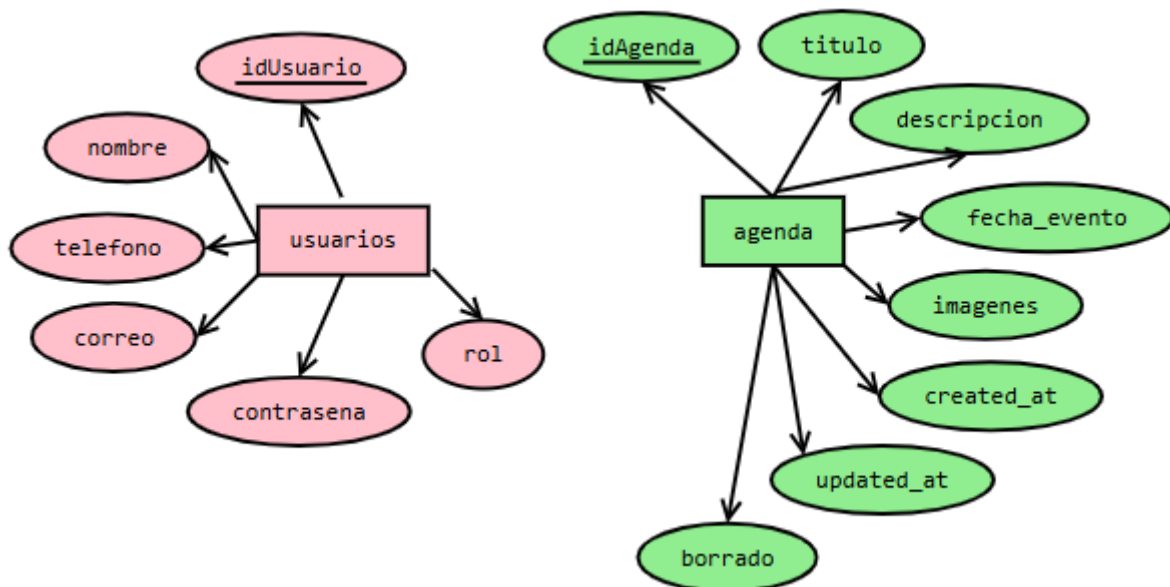
borrado (valor lógico para borrar, solo se desactiva el evento 0 = activo, 1 = eliminado)

Relaciones.

La relación es una relación de uno a muchos entre usuarios y agenda, en la que un administrador puede registrar múltiples eventos en la agenda, pero cada evento está gestionado por un único administrador.

La tabla agenda no contiene una clave foránea directa y solo los usuarios con rol admin pueden hacer las funciones CRUD sobre ella.

### 2.1.1. Esquema/s entidad-relación



### 2.1.2. Traslado a un esquema relacional normalizado

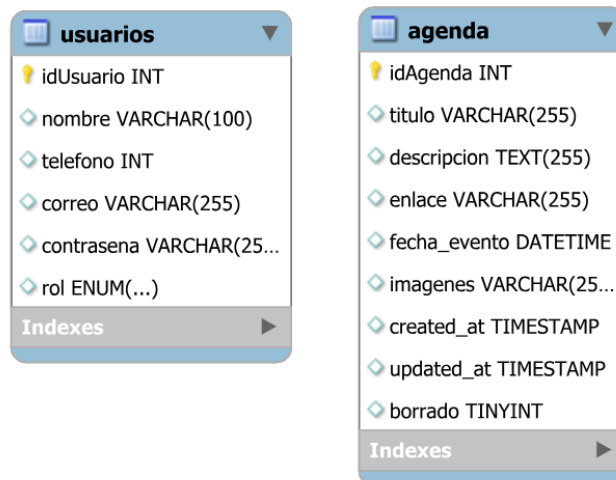
1. usuarios (#idUsuario, nombre, telefono, correo, contraseña, rol)
2. agenda (#idAgenda, titulo, descripcion, enlace, fecha\_evento, imagenes, created\_at, updated\_at, borrado)

La base de datos cumple la tercera forma normal (3FN), cada campo depende de su Primary Key y no hay dependencia entre campos que no son clave.

### 2.1.3. Elección justificada de un SGBD relacional concreto

Para mi proyecto he usado MySQL como sistema de gestión de bases de datos relacional. Lo he controlado a través de phpMyAdmin ya que es una interfaz muy sencilla de usar y que viene integrada en XAMPP. Junto a PHP y AJAX usar MySQL a través de phpMyAdmin es una opción muy viable, cómoda y sencilla.

### 2.1.4. Traslado a un esquema físico relacional



### 2.1.5. Restricciones no implementables en el modelo de datos que deberá contemplar la aplicación

Las acciones que no se pueden controlar a nivel de base de datos y por lo tanto se deben controlar con la aplicación son las siguientes:

Formato de contraseña: en la base de datos se guarda el hash de la contraseña por lo que no puede obligar a cumplir la restricción de mínimo 8 caracteres, mayúsculas, minúsculas, número y símbolo, esto lo hacemos desde el formulario en JQuery y PHP.

Confirmación de contraseña: se valida en la vista ya que la base de datos no puede saberlo.

Campo correo: aunque tenga el unique se valida en el código con una consulta antes de insertarlo, así se evitan duplicaciones. Imágenes: guardamos la ruta pero no podemos modificar la foto, esto se hace desde CSS.

## 2.2. Funciones

Los usuarios pueden consultar un calendario dinámico que señala los días que tienen eventos registrados junto a información sobre ellos. El administrador tiene dos pantallas para hacer las funciones CRUD, por un lado tiene la vista de gestión donde puede modificar los eventos y eliminar (es un borrado lógico ya que no se borra el registro sino que se marca como borrado, se desactiva) y también tiene la vista de añadir, donde puede registrar eventos en la base de datos y se reflejarán al usuario.

### 2.2.1. Diagramas de flujo de datos

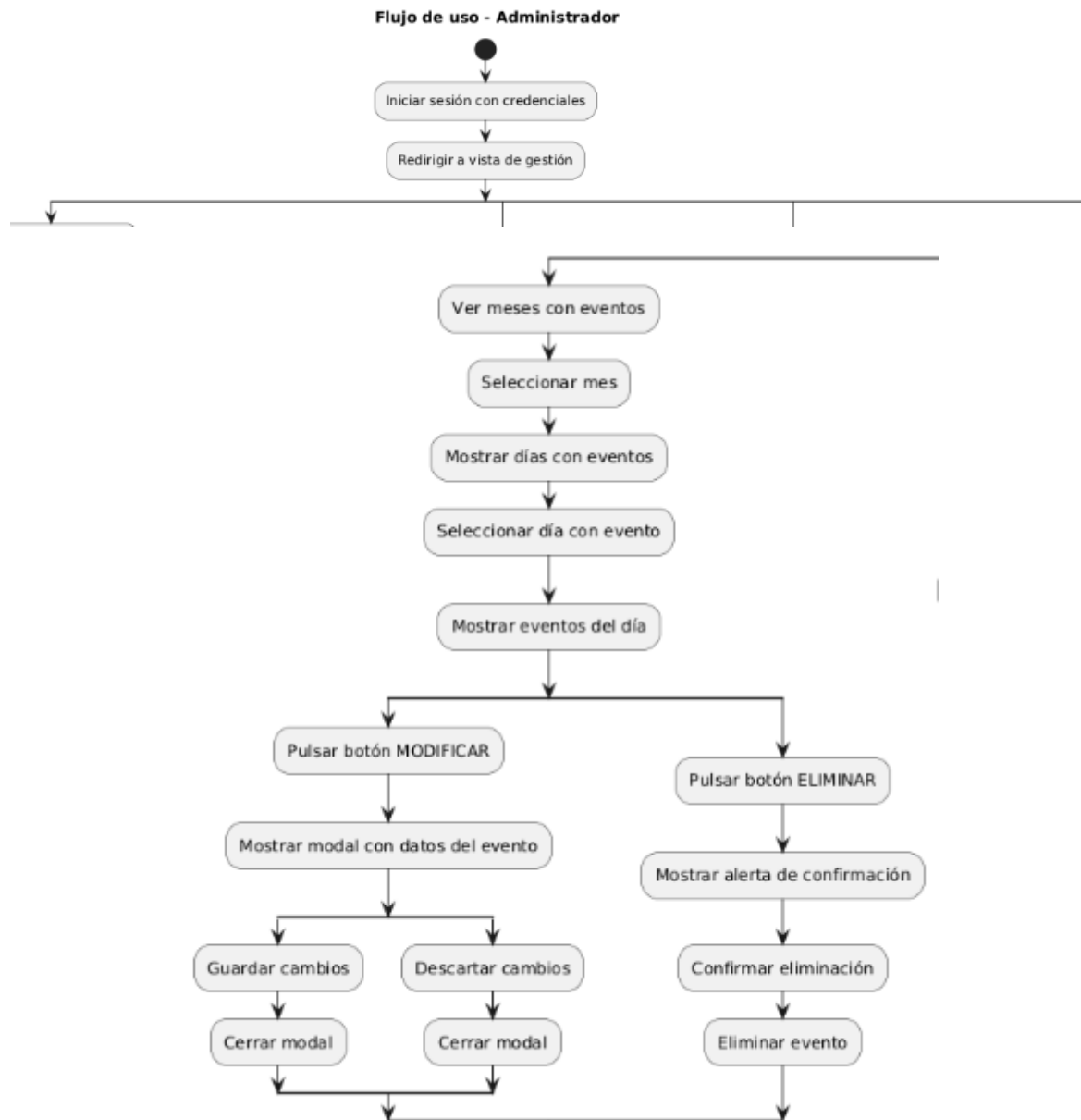
Este es el flujo que siguen los usuarios, solo pueden hacer la función de consulta del CRUD.

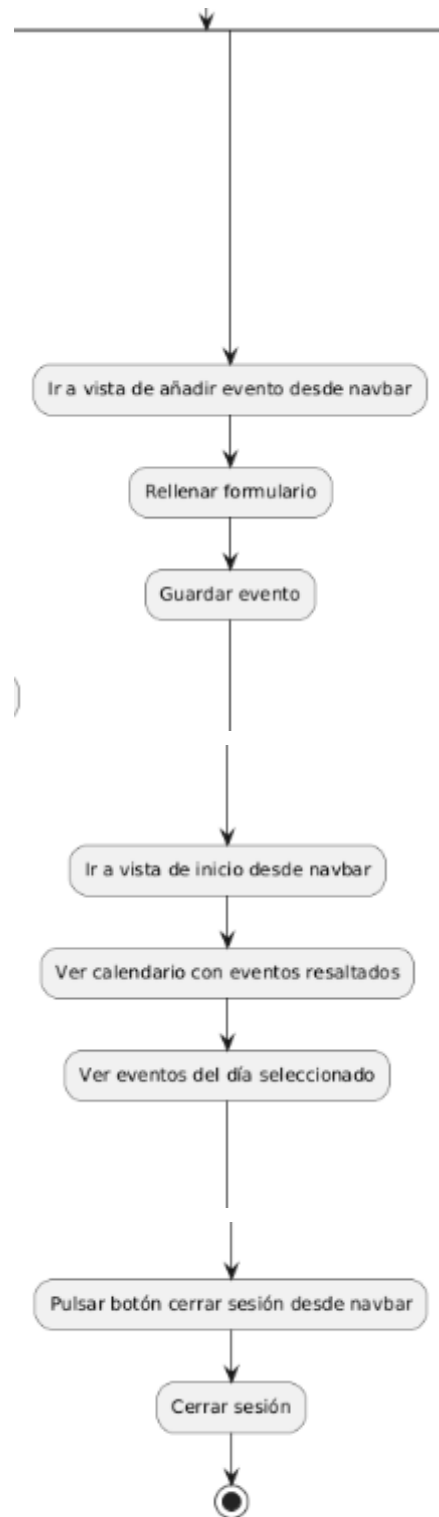




## José María Ortiz Román | Proyecto integrado

El flujo del administrador se parte en cuatro ramas que he pegado en imágenes separadas para que se pueda ver más claro. Gestión de eventos con las funciones UPDATE y DELETE, añadir eventos con INSERT INTO, página de inicio y cierre de sesión son las cuatro ramas respectivamente.





## 2.2.2. Diccionario de datos

### **Tabla: agenda**

id

Tipo de dato: entero (int)

Restricciones: primary key, autoincrementable, no nulo.

titulo

Tipo de dato: cadena de caracteres (string (varchar 255))

Restricciones: no nulo.

descripcion

Tipo de dato: texto largo (text)

Restricciones: puede ser nulo.

enlace

Tipo de dato: cadena de caracteres (string (varchar 255))

Restricciones: puede ser nulo. Se espera que contenga una URL válida.

fecha\_evento

Tipo de dato: fecha y hora (datetime)

Restricciones: no nulo.

imagenes

Tipo de dato: cadena de caracteres (string (varchar 255))

Restricciones: no nulo. Almacena la ruta relativa a la imagen del evento.

created\_at

Tipo de dato: timestamp

Restricciones: se asigna automáticamente la fecha y hora actual al crear el registro.

updated\_at

Tipo de dato: timestamp

Restricciones: se actualiza automáticamente cada vez que se modifica el registro.

borrado

Tipo de dato: entero (tinyint 1)

Restricciones: no nulo. Por defecto 0 (no borrado); 1 indica borrado lógico (no se borra el registro sino que pasa a estar desactivado).

### **Tabla: usuarios**

id

Tipo de dato: entero (int)

Restricciones: clave primaria, autoincrementable, no nulo.

nombre

Tipo de dato: cadena de caracteres (string(varchar 100))

Restricciones: No nulo.

telefono

Tipo de dato: cadena de caracteres (string(varchar 20))

Restricciones: no nulo. Se controla que solo pueda registrar dígitos.

correo

Tipo de dato: cadena de caracteres (string (varchar 255))

Restricciones: no nulo. Debe tener un formato de correo electrónico válido y ser único.

contrasena

Tipo de dato: cadena de caracteres (string (varchar 255))

Restricciones: no nulo. Almacenada en formato cifrado.

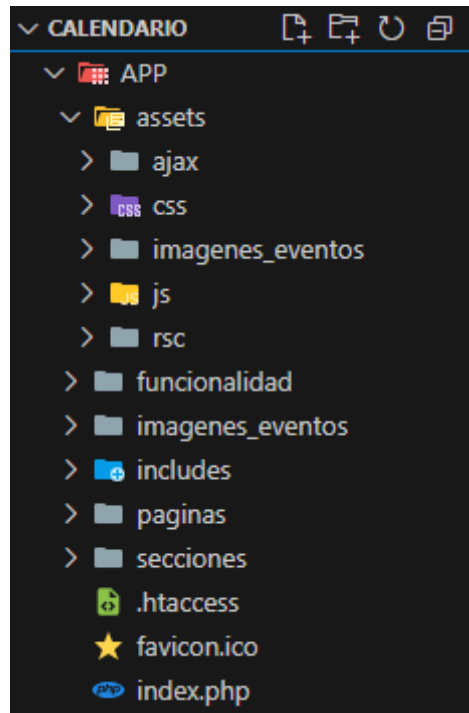
rol

Tipo de dato: enumerado (admin, usuario)

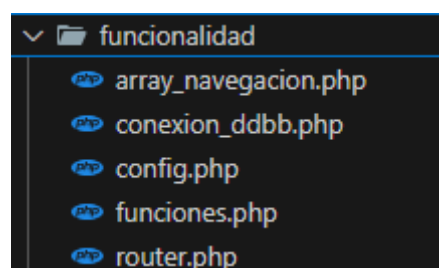
Restricciones: no nulo. Por defecto usuario. Controla los permisos de acceso a funcionalidades.

### 2.2.3. Descripción de Clases, Métodos, Atributos, ...

Estructura de carpetas del proyecto. El proyecto se divide en la carpeta assets donde están los ficheros AJAX, CSS, la carpeta donde guardamos las imágenes de los eventos de forma física, carpeta de JQuery y rsc con imágenes que usamos en la página.

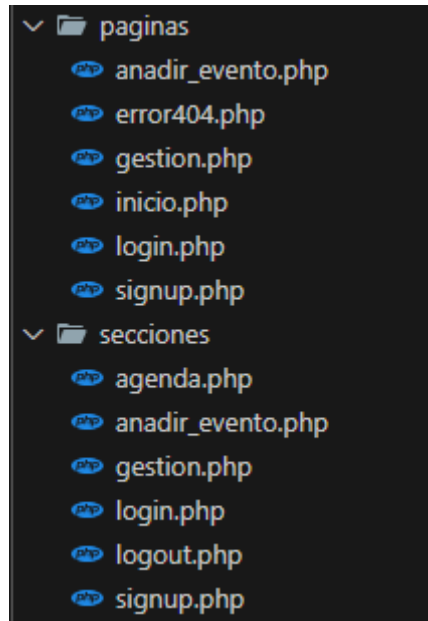


En la carpeta de funcionalidad están archivos que iremos cargando en otras clases con funcionalidades cómo el sistema de rutas, conexión a base de datos, configuraciones y el router que también participa para gestionar el sistema de rutas amigable.



## José María Ortiz Román | Proyecto integrado

En las carpetas de páginas y secciones tenemos la parte de la vista. En las páginas incluimos las secciones que es donde viene verdaderamente el contenido que vamos a ver por pantalla como usuario.



### Secciones

agenda.php: vista principal de los usuarios, se muestra el calendario dinámico. Dependiendo si el usuario tiene rol admin tendrá navbar de navegación.

gestion.php: vista para hacer el UPDATE y DELETE de eventos. Solo los usuarios con rol admin pueden acceder.

añadir evento.php: formulario para hacer el INSERT INTO de los eventos en la base de datos. De nuevo, sólo los usuarios con rol admin tienen acceso.

login.php: formulario de inicio de sesión.

signup.php: formulario de registro. Valida que la contraseña tenga cierta complejidad con pattern.

logout.php: cierre de sesión. Cuando se llama a esta clase borra la sesión.

### Js

En la carpeta de los Jquery tenemos por un lado los archivos de librerías como Bootstrap, Jquery, las cookies y SweetAlert, por otro lado, tenemos los archivos que se encargan de controlar los eventos de los botones.

agenda.js: controla la lógica del calendario dinámico.

actualizar\_calendario(): consulta mediante AJAX (actualizar\_calendario.ajax.php) para actualizar el nuevo año, mes y cuadrícula; se llama al cargar página.

## José María Ortiz Román | Proyecto integrado

`clickear_dia()`: entiende el clic en un día (con o sin evento), actualiza la fecha seleccionada y vuelve a llamar a `actualizar_calendario()`.

`clickear_navegadorfecha()`: gestiona los botones de desplazamiento de año y mes, para esto llama a las funciones `ajustarAnio()` y `ajustarMes()` que refrescan el calendario.

`gestion_eventos.js`: script de la vista `gestion.php` para hacer SELECT, UPDATE y DELETE desde la vida de gestión.

`obtenerMeses()`: pide meses con eventos y los muestra en `#meses_eventos`.

Listener `.mes`: llama a AJAX para imprimir por pantalla `#dias_eventos`.

Listener `.dia`: AJAX `obtenerEventosDia`; lista eventos con botones Modificar y Eliminar.

Listener `.eliminar-evento`: SweetAlert de confirmación.

Listener `.modificar-evento`: carga datos en `modalEditarEvento`, gestiona imagen actual y envía cambios con `FormData` (`modificarEvento`).

`anadir_evento.js`: maneja el formulario `añadir_evento.php`.

Submit de `#form_nuevo_evento`: manda todo en `FormData` (incluida imagen), lo envía a `anadir_evento.ajax.php` y muestra SweetAlert de resultado. Al guardar limpia el formulario y, si existe la función `obtenerMeses()`, refresca la lista de meses del admin.

`mostrar_eventos_usuario.js`: carga y muestra los eventos del día seleccionado en la vista pública.

`cargarEventos(fechaISO)`: AJAX `obtenerEventosDia`, imprime la lista en `#contenido_eventos_dia`.

Listener `.evento_d`: llama a `cargarEventos()`.

Listener `.evento_clickable`: rellena y abre el modal `#modalContenidoEvento` con título, descripción e imagen. Al iniciar, llama a `cargarEventos()` con la fecha actual para mostrar los eventos de hoy.

`registro_usuarios.js`: valida y envía el formulario de registro. Comprueba que contraseña y confirmación coinciden; si son correcta, por POST pasa a `registro_usuarios.ajax.php`; SweetAlert según resultado y te redirige a `/LOGIN`.

`login_usuarios.js`: inicio de sesión sin recargar página. Envía `formLogin` por AJAX a

`script.js`: utilidades globales. Detección de móvil y formato de pantalla.

## José María Ortiz Román | Proyecto integrado

Preloader: oculta #preloader y #pantalla\_carga al terminar de cargar.

Botón #btnLogout: POST a /logout.php y redirige a /LOGIN.

Botón #btnInicio: vuelve a agenda.php.

### **AJAX**

actualizar\_calendario.ajax.php: prepara los datos del calendario que necesita agenda.js. Valida la fecha recibida por POST, actualiza la sesión, calcula cantidadDias y marcadoresDias, obtiene los eventos del mes (SELECT agenda WHERE borrado 0) y responde en JSON.

anadir\_evento.ajax.php: inserta un nuevo evento. Comprueba que lleguen título y fecha, procesa la imagen (crea carpeta si falta, genera nombre único, move\_uploaded\_file), guarda ruta en la columna imagenes y hace el INSERT INTO agenda (titulo descripcion fecha\_evento imagenes created\_at updated\_at borrado=0). Devuelve JSON de estado final.

gestion\_eventos.ajax.php: backend de la vista de gestión. Recibe acción por POST y usa un switch.

obtenerMeses: SELECT DISTINCT MONTH YEAR de agenda borrado 0 y devuelve meses.

obtenerDias: SELECT DISTINCT DAY para el mes y año pedido.

obtenerEventosDia: SELECT id titulo descripcion imagenes para la fecha.

eliminarEvento: UPDATE agenda SET borrado 1 updated\_at NOW WHERE id.

modificarEvento: UPDATE agenda con título descripción fecha e imagen. Si llega nueva imagen la sube y borra la antigua; si el admin marca eliminar\_imagen borra el archivo y pone NULL en BD.

obtenerEventoPorId: SELECT id titulo descripcion imagenes fecha\_evento por id.

Responde siempre en JSON success/error.

login\_usuarios.ajax.php: valida correo y contraseña. SELECT id nombre contrasena rol FROM usuarios WHERE correo, compara con password\_verify. Si coincide inicia sesión (\$\_SESSION id nombre rol) y devuelve success nombre rol; si no, mensaje de error en JSON.

registro\_usuarios.ajax.php: registra un usuario nuevo. Comprueba que el correo no exista, hashea la contraseña con password\_hash y hace INSERT INTO usuarios (nombre telefono correo contrasena rol). Devuelve JSON success o error.

cabeceras.ajax.php: define los métodos permitidos, fija la zona horaria a la de Madrid.



## **Funcionalidad**

conexion\_dbbb.php: gestiona la conexión a MySQL mediante mysqli, define host, usuario, contraseña y base de datos, pone la colación de caracteres en UTF-8 y para la ejecución si da error.

config.php: agrupa variables globales de la aplicación; nombre\_session\_app para \$\_SESSION, URLs base de la web, más las credenciales de la BD para reutilizarlas en cualquier otro archivo.

funciones.php: librería de utilidades genéricas.

no\_vacio(\$valor\_campo): devuelve true si la cadena, tras trim(), tiene longitud > 0.

router.php: clase Router que interpreta la URL y decide qué vista cargar.

setDirectorioBase(\$url): guarda la URL raíz y calcula la profundidad de directorios.

navegar(\$url): procesa \$\_SERVER['REQUEST\_URI'], compara con las rutas registradas y, si no hay coincidencia, redirige.

registrarRuta(\$ruta,\$vista): añade una ruta con su vista asociada al array interno.

setVista(\$ruta): método privado; recorre las rutas registradas, resuelve parámetros id, y fija \$this->vista cuando encuentra coincidencia exacta.

getVista(): devuelve la vista seleccionada por el router.

redirigir(\$url): método privado; hace header("Location") y exit.

array\_navegacion.php: mapa de rutas amigables.

datosNav: array con nombre, href y fichero; sirve al Router para asociar cada ruta (LOGIN, GESTION, etc.) con el PHP de vista al que va enlazado y así construir la navegación sin necesidad de escribir la ruta completa.

## 4. Manuales

### 4.1. Manual técnico

La aplicación permite a usuarios registrados visualizar eventos en un calendario dinámico; los administradores, además, pueden crear, modificar y eliminar eventos. El frontend se ejecuta en el navegador (HTML + Bootstrap + JavaScript/jQuery). La lógica reside en PHP (servidor Apache/PHP) y la persistencia en MySQL. Toda comunicación cliente-servidor se realiza mediante peticiones AJAX sobre HTTP; el formato de intercambio es JSON.

#### **Flujo de datos entre cliente y servidor:**

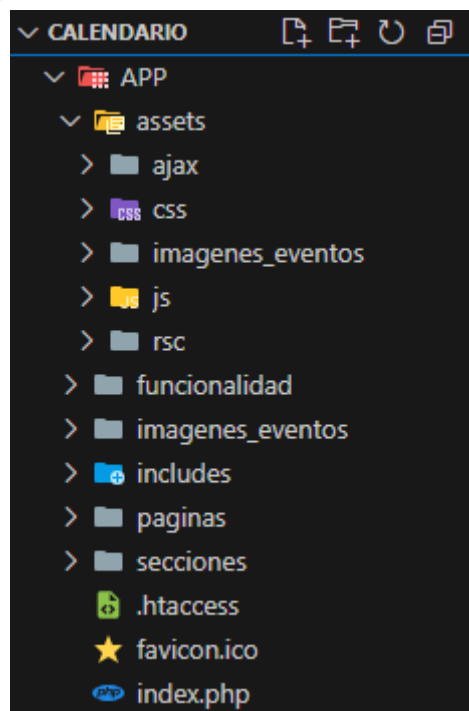
Petición AJAX: el JS del navegador envía POST al endpoint PHP correspondiente (por ejemplo, gestion\_eventos.ajax.php).

Validación y lógica: el script PHP valida datos, ejecuta las sentencias SQL, formatea la respuesta JSON y la devuelve.

Actualización dinámica: el JS recibe el JSON, actualiza el DOM (calendario, listados, modales) sin recargar la página.

Gestión de sesiones: PHP usa \$\_SESSION para conservar identidad y rol; el frontend ajusta la interfaz según si el usuario es admin o usuario.

#### **Estructura del Código**



## **Clases/archivos clave:**

agenda.php: vista pública con calendario.

gestion.php: vista admin para UPDATE/DELETE.

anadir\_evento.php: formulario INSERT.

login.php / signup.php / logout.php: autenticación.

router.php: resuelve URLs amigables.

conexion\_ddbb.php: abre conexión MySQL (mysqli).

config.php: constantes globales (URLs, nombre de sesión y credenciales de la base de datos).

## **JS principales:**

agenda.js: actualiza calendario, controla navegación de mes/año.

gestion\_eventos.js: CRUD vía AJAX, modales y SweetAlert.

mostrar\_eventos\_usuario.js: carga eventos del día y muestra modal detalle.

anadir\_evento.js, registro\_usuarios.js, login\_usuarios.js: manejan los respectivos formularios.

## **Detalles de implementación**

Gestión de sesiones y roles: login\_usuarios.ajax.php verifica contraseña (password\_verify) y guarda \$\_SESSION['rol'] ('admin' | 'usuario'). Las vistas protegen secciones con if (!isset(\$\_SESSION['rol']) || \$\_SESSION['rol'] !== 'admin').

Acceso a datos (MySQL): todos los scripts AJAX usan sentencias preparadas (prepare, bind\_param) para prevenir que se inyecten sentencias SQL malignas. Las fechas se almacenan en agenda.fecha\_evento tipo DATE; la columna borrado (tinyint) implementa borrado lógico (no borra de la base y solo desactiva).

Subida y gestión de imágenes: anadir\_evento.ajax.php y la rama modificarEvento de gestion\_eventos.ajax.php guardan archivos en assets/imagenes\_eventos/, generan nombre único con uniqid(), y persisten la ruta relativa en agenda.imagenes. Al reemplazar o eliminar la imagen se borra físicamente con unlink.

Router personalizado y amigable: permite URLs limpias del estilo /CALENDARIO/APP/CALENDARIO/GESTION. Los parámetros capturados quedan en \$router->parametros['id\_form'].

Calendario dinámico: actualizar\_calendario.ajax.php calcula el primer día de mes (DateTime->format('N')), el número de días (->format('t')), el array diasEventos (SELECT DAY(fecha\_evento)). Devuelve JSON y agenda.js imprime la cuadrícula a través del DOM.

Seguridad centrada en datos: cualquier texto recibido pasa por `htmlspecialchars(addslashes(trim()))` o `mysqli->real_escape_string`. Las contraseñas se almacenan hasheadas con `PASSWORD_DEFAULT`.

### **Dependencias y librerías**

Extensiones nativas PHP: `mysqli`, `intl`, `session`.

SweetAlert: alertas dinámicas y personalizables con estilos más visuales.

Bootstrap: maquetación responsive.

jQuery y AJAX simplificado y gestión del DOM.

### **Consideraciones de seguridad**

Todas las sentencias SQL son preparadas; sin concatenación de variables. Roles: las acciones de gestión de eventos sólo son accesibles con `$_SESSION['rol']==='admin'`, si cerramos sesión desde la vista admin y queremos volver atrás en la página nos redirige al login ya que una vez se cierra sesión se elimina el caché. Subidas de imagen: se valida `$_FILES['error']`, extensión y tamaño; los nombres se generan de forma aleatoria para evitar duplicaciones. En `cabeceras.ajax.php` define CORS básico (acceder a recursos en otros archivos desde el DOM). Las sesiones usan un nombre exclusivo (`CALENDARIO_EVENTOS`) para aislarse de otros ficheros.

## 4.2. Manual de instalación

No es necesaria la instalación de ningún software específico por parte del usuario para usar la aplicación, solo necesita tener cualquier navegador ya que las tecnologías usadas son todas ampliamente compatibles. Teniendo en cuenta lo anterior, cualquier dispositivo o equipo es capaz de entrar a la web y navegar sin problema.

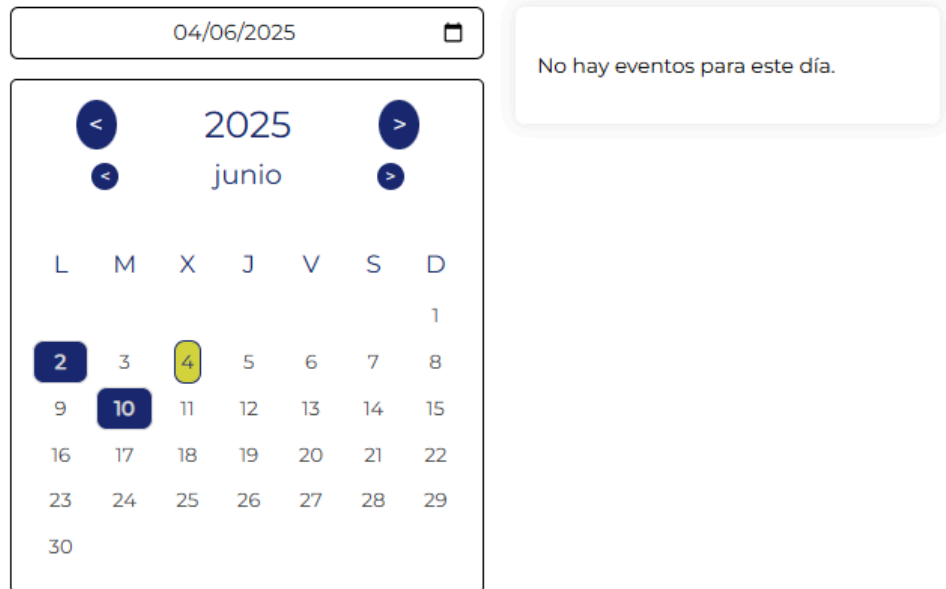
Para la instalación local se necesitaría instalar un servidor local, en mi caso he usado XAMPP, la app está preparada para funcionar con PHP 8.1 y MySQL como sistema de gestión de base de datos, preferiblemente se usaría phpMyAdmin al estar integrado en el servidor local. Lo siguiente sería clonar el repositorio y ubicarlo en la carpeta `htdocs` (si estamos usando XAMPP).

En resumen, para poder usar la web como usuario solo necesitaríamos un navegador moderno y para poder hacer la instalación local bastaría con XAMPP y clonar el repositorio.

### 4.3. Manual de usuario

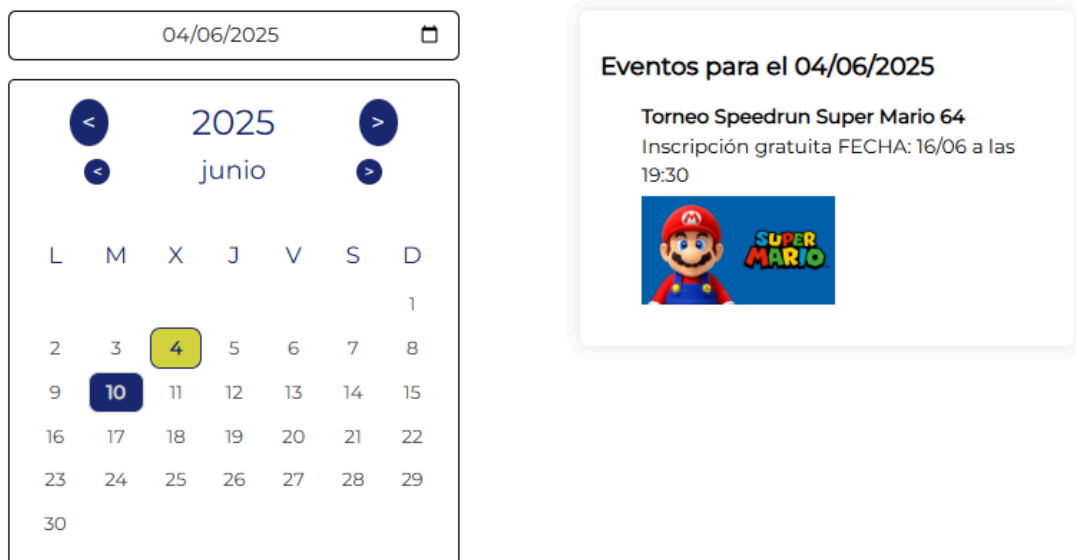
Lo primero que ve el usuario al entrar a la página es el calendario mostrando si hay algún evento en el día en el que nos encontramos.

## CALENDARIO DE EVENTOS



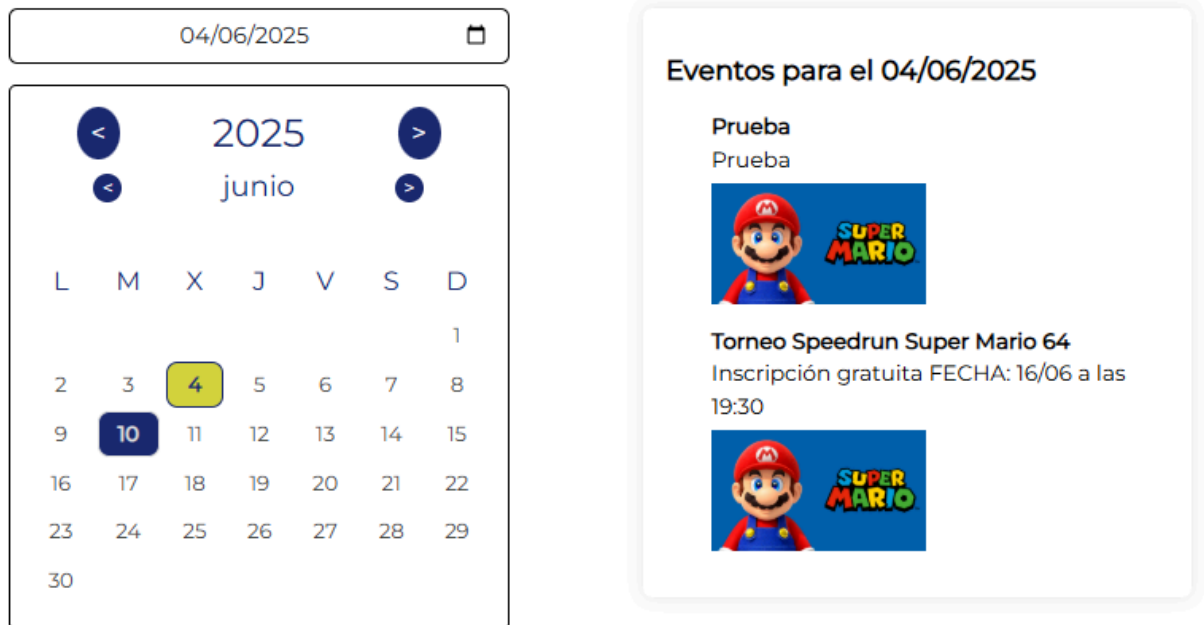
En el caso de haber evento se vería así.

## CALENDARIO DE EVENTOS



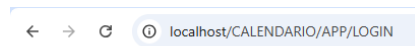
En la vista de usuario tan solo se pueden consultar los días con eventos y se mostrará tal y cómo se ve en la imagen superior. Si hay varios eventos se mostrarían así.

## CALENDARIO DE EVENTOS



### 4.4. Manual de administración

Para acceder a la vista de administrador debes iniciar sesión con nuestras credenciales, estas deben ser insertadas en la base de datos ya que la página de registro está limitada a usuarios con el rol admin para así poder crear más usuarios que puedan hacer las funciones CRUD.



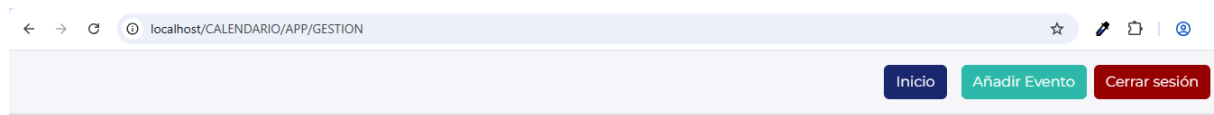
#### Inicio de sesión

Correo electrónico

Contraseña

Una vez iniciada sesión la página nos redirige a la vista de gestión. Aquí podemos ver los meses que tienen eventos y el año.

# José María Ortiz Román | Proyecto integrado



## Gestión de Eventos

Meses con eventos

12/1212 12/1313 06/2025

Si damos clic en el mes se despliegan los días.

## Gestión de Eventos

Meses con eventos

12/1212 12/1313 06/2025

Días con eventos

Día 2 Día 10

Al pulsar en los días se nos muestran los eventos con el título, descripción e imagen. También en este punto tenemos los botones de modificar y eliminar.

## Gestión de Eventos

Meses con eventos


12/1212 12/1313 06/2025

Días con eventos

Día 2 Día 10

Eventos para 02/06/2025

**Torneo Speedrun Super Mario 64**  
Inscripción gratuita FECHA: 01/06 a las 19:30

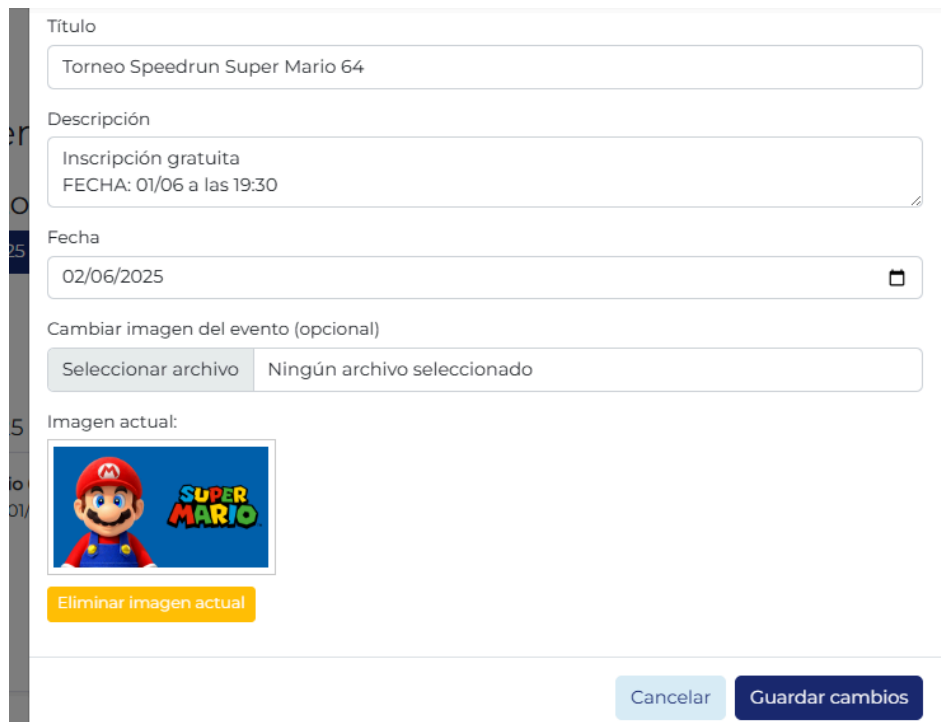


Modificar Eliminar



## José María Ortiz Román | Proyecto integrado

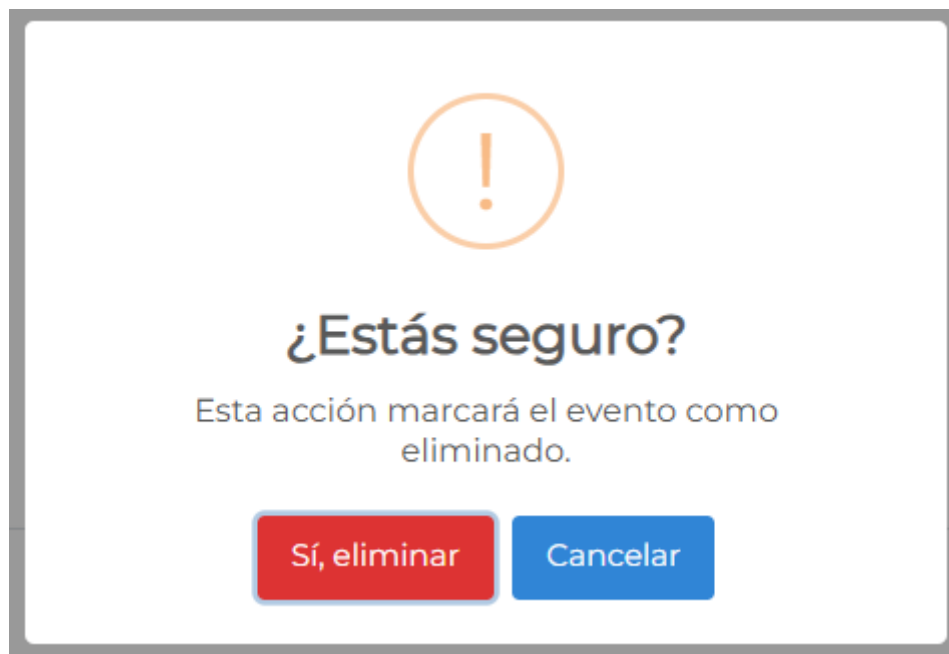
Si pulsamos el modificar se abre un modal con todos los datos sobre el evento mencionados anteriormente, aquí podremos modificar y guardar o cancelar usando los botones.



The screenshot shows a modal for editing an event. It contains the following fields and elements:

- Título:** Torneo Speedrun Super Mario 64
- Descripción:** Inscripción gratuita  
FECHA: 01/06 a las 19:30
- Fecha:** 02/06/2025
- Cambiar imagen del evento (opcional):** A button labeled "Seleccionar archivo" and a text field showing "Ningún archivo seleccionado".
- Imagen actual:** A small image of Super Mario with the text "SUPER MARIO". Below it is a yellow button labeled "Eliminar imagen actual".
- Botones de acción:** "Cancelar" and "Guardar cambios" at the bottom right.

Por último, tenemos el botón de eliminar el cual mandará una alerta para asegurarnos.



## José María Ortiz Román | Proyecto integrado

Para navegar en la vista de administrador tenemos el navbar que desde la página de gestión podemos ir al inicio, en esta vista tenemos las mismas funciones que el usuario normal, quitando el hecho de que tenemos el navbar para navegar entre las funciones de administrador. También mencionar el botón de cerrar sesión, una vez lo pulsemos nos redirigirá a la página de inicio de sesión, en caso de volver atrás usando la flecha del navegador nos redirigirá constantemente al inicio de sesión.



En la vista de añadir nuevo evento tenemos un formulario básico el cual rellenaremos con los datos deseados y cuando lo guardemos se registrará en la base de datos y estará disponible para ser consultado en la página de inicio.

### Añadir nuevo evento

Título del evento:

Descripción:

Imagen del evento:

Seleccionar archivo

Ningún archivo seleccionado

Fecha:

dd/mm/aaaa

Guardar evento

## 5. Conclusiones

### 5.1. Grado de consecución de los objetivos fijados

Se ha creado una app web de calendario donde se pueden consultar los días con eventos y a parte tiene la vista de admin donde se gestiona, por lo tanto, se han cumplido con los objetivos fijados.

### 5.2. Posibles mejoras o ampliaciones para implementar en el futuro.

Una mejora que podría implementarse sería un sistema de recuperación de contraseña usando PHPMailer y Gmail, esto serviría para que los administradores sin acceso a la base de datos pudiesen recuperar sus credenciales. También sería interesante hacer un sistema de filtrado y solo se muestren eventos de una categoría, para ello habría que crear un nuevo campo en la base de datos. Otra funcionalidad interesante sería un sistema de notificaciones por correo para avisar a los usuarios de eventos o de categorías de evento que ellos marquen.

## 6. Bibliografía

OpenAI. (2025). ChatGPT – Conversational AI for everyone. Recuperado de <https://openai.com/chatgpt>

Google. (2025). Gemini: multimodal generative AI by Google DeepMind. Recuperado de <https://deepmind.google/technologies/gemini/>

The Bootstrap Authors. (2025). Bootstrap 5.3 Documentation. Recuperado de <https://getbootstrap.com/docs/5.3/getting-started/introduction/>

W3Schools. (2025). jQuery . Recuperado de <https://www.w3schools.com/jquery/>

W3Schools. (2025). AJAX Tutorial – Introduction to Asynchronous JavaScript and XML. Recuperado de [https://www.w3schools.com/xml/ajax\\_intro.asp](https://www.w3schools.com/xml/ajax_intro.asp)

W3Schools. (2025). CSS Reference & Advanced Guides. Recuperado de <https://www.w3schools.com/css/>