SISTEMA DE RESERVAS DE EVENTOS

Guía de instalación

Para el departamento técnico

Versión 3.19

Propietario y desarrollador
Eduardo García Gómez
info@calmalogic.com

calmalogic ®

La parte de programación ha sido gestionada por Claude (Anthropic)

Para quién es esta guía

Este documento es para quien sube la aplicación a un servidor: un informático, el departamento de sistemas, o quien administre el alojamiento web. No es necesario para el uso diario; para eso está el *Manual del administrador*.

Requisitos del servidor

Requisito	Detalle
PHP	Versión 7.4 o superior (probado en 8.0). Es lo único imprescindible.
Base de datos	Ninguna. Los datos se guardan en archivos JSON.
Escritura	La carpeta de datos debe tener permisos de escritura (755).
Correo saliente	Una cuenta SMTP para los correos (opcional pero recomendado).
Tareas programadas	Cron, solo si se quieren envíos automáticos.
HTTPS	Muy recomendable. Necesario para el menú de compartir en móvil.

Arquitectura, en una frase

Una aplicación de una sola página (`index.html`) que habla con un backend PHP (`api.php`) por peticiones AJAX. Los datos son archivos JSON en una carpeta protegida. Sin base de datos, sin framework, sin dependencias que instalar. Se despliega copiando archivos.

1. Los archivos

El ZIP contiene estos archivos. Todos van en la misma carpeta (la raíz de la aplicación), salvo los de datos:

Archivo	Qué es
index.html	La aplicación completa (HTML + CSS + JS en un solo archivo).
api.php	El backend: lee y escribe los JSON, valida reservas, envía correos.
tour.php	Genera la vista previa (Open Graph) al compartir un evento en redes.
imagen.php	Sirve las imágenes (guardadas en base64) como archivos reales, para las vistas previas.
vcard.php	Genera una tarjeta de contacto .vcf con el teléfono de Bizum.
cron.php	Los envíos automáticos. Lo ejecuta la tarea programada.
datos/	Carpeta con los JSON y el .htaccess que la protege.

Los cuatro JSON (config, tours, reservas, historico) se crean solos en la primera visita si no existen. No hay que crearlos a mano.

2. Instalación paso a paso

1

Subir los archivos

Copie todo el contenido del ZIP a la carpeta pública deseada (por FTP, SFTP o gestor de archivos del panel). Respete la estructura: los archivos sueltos en la raíz, y la carpeta `datos/` con su contenido dentro.

2

Verificar el .htaccess de datos

En `datos/` debe existir `.htaccess` con `Deny from all` (o equivalente). Muchos clientes FTP ocultan los archivos que empiezan por punto: si no aparece, suba el `htaccess_RENOMBRAR_A_PUNTO_HTACCESS.txt` incluido y renómbrelo.

3

Permisos de escritura

La carpeta `datos/` necesita permisos 755 (o 775 según el servidor). Si el proceso PHP no puede escribir, la aplicación cargará pero no guardará nada.

4

Caso WordPress

Si la aplicación se instala en un dominio con WordPress en la raíz, las reglas de reescritura de WordPress interceptan las subcarpetas. Suba el `htaccess_SOLO_SI_HAY_WORDPRESS.txt` a la raíz de la aplicación, renombrado a `.htaccess`. Contiene `RewriteEngine Off` para aislar la carpeta.

5 Comprobar el estado

Abra en el navegador la dirección de la aplicación con este parámetro:

```
/api.php?action=version
```

Devuelve un JSON con la versión, la ruta de datos, si la carpeta existe y si se puede escribir (`sePuedeEscribir: true`). Es la comprobación más rápida de que todo está bien.

6 Comprobar la protección de datos

Abra `/datos/config.json` en el navegador. Debe devolver 403 (Forbidden). Si muestra el contenido, el .htaccess no está activo: es crítico, porque ese archivo contiene la contraseña SMTP.

3. Cambiar la contraseña de administrador

La contraseña no viaja al navegador: se guarda como hash bcrypt en `api.php` y se verifica en el servidor, con un pequeño retardo por intento. La inicial es `xzentrik`. Para cambiarla:

1. Genere el hash de la nueva clave (por línea de comandos, o cualquier consola PHP):

```
php -r "echo password_hash('NUEVA_CLAVE', PASSWORD_DEFAULT);"
```

2. Abra `api.php`, busque la línea:

```
$ADMIN_PASSWORD_HASH = '...';
```

3. Sustituya el hash entre comillas por el nuevo. Guarde y suba el archivo.

> Cada instalación tiene su propia contraseña. Si hay varias instancias del producto en carpetas distintas, cada una lleva su hash independiente.

4. Correo saliente (SMTP)

El envío de correo se configura desde el panel (Configuración), no en el código. Pero conviene que el técnico conozca el funcionamiento:

- El cliente SMTP está implementado a mano por sockets en PHP (STARTTLS en el puerto 587, SSL directo en el 465). No usa librerías externas.
- Las credenciales se guardan en `datos/config.json`, protegido por el .htaccess. De ahí la importancia del paso 6 de la instalación.
- Gmail y Outlook requieren "contraseña de aplicación", no la del usuario.
- El botón "Verificar Conexión" del panel hace un handshake real y devuelve el diálogo SMTP línea a línea, útil para diagnosticar.

5. Envíos automáticos (cron)

Los correos automáticos (recordatorio, agradecimiento, propuesta de pago) los envía `cron.php`, que debe ejecutarse periódicamente. Sin la tarea programada, los interruptores del panel no hacen nada.

Seguridad

`cron.php` exige una clave secreta, para que nadie pueda dispararlo desde fuera. Está definida al principio del archivo:

```
define('CRON_CLAVE', '...');
```

Puede cambiarla por otra; si lo hace, actualice también la URL de la tarea programada.

Configurar la tarea

Se ejecuta cada 15 minutos. La forma depende del panel:

Panel	Cómo
Plesk	Sitios web y dominios > Tareas programadas > Añadir. Tipo "Obtener una URL". URL: la de cron.php con la clave. Programación estilo cron: `*/15 * * * *`
cPanel	Cron Jobs. Comando: `wget -q -O /dev/null "URL"` o `php /ruta/cron.php CLAVE`. Cada 15 minutos.
Línea de comandos	`*/15 * * * * php /ruta/cron.php CLAVE` en el crontab.

URL de ejemplo (por web):

```
https://dominio.com/carpeta/cron.php?clave=LA_CLAVE
```

Por línea de comandos, la clave es el primer argumento: `php cron.php LA_CLAVE`

Cómo trabaja

- En cada ejecución recorre las reservas y calcula, para cada una, si toca algún envío según la fecha/hora del evento y los ajustes de automatización.
- Anota en cada reserva qué se ha enviado, de modo que nunca duplica. Es idempotente: se puede ejecutar muchas veces sin efectos secundarios.
- Ignora objetivos de más de 72 horas atrás, para no reenviar en masa si el cron estuvo caído.
- Ejecutarlo manualmente por web devuelve un informe de texto con lo enviado. Útil para probar.

6. Actualizar a una versión nueva

Las actualizaciones se entregan como archivos sueltos. El procedimiento habitual:

- Casi siempre basta con reemplazar `index.html`. A veces también `api.php` u otros. El changelog de cada versión indica qué archivos cambian.
- **Nunca se toca la carpeta datos/** al actualizar. Los eventos, reservas y configuración se conservan.
- Tras subir, forzar recarga sin caché (Ctrl+F5). Verificar la versión en el pie de la página.

Caché de servidor. Algunos alojamientos (LiteSpeed, Varnish, plugins de WordPress) cachean el HTML. Si tras actualizar sigue viéndose la versión antigua pese al Ctrl+F5, purgar la caché del servidor.

Copias de seguridad

El propio panel permite descargar y restaurar una copia (Configuración > Copia de seguridad). A nivel de servidor, basta con respaldar la carpeta `datos/`: contiene todo el estado. La aplicación además guarda copias automáticas en `datos/copias/` antes de cada cambio importante (conserva las 10 últimas).

7. Varias instancias

El producto se puede desplegar en varias carpetas del mismo dominio, cada una independiente: sus propios datos, configuración y contraseña. No comparten nada. Es el modelo habitual para dar una instancia a cada cliente.

Cada instancia necesita su propia tarea cron si quiere envíos automáticos, apuntando a su propio `cron.php`.

8. Diagnóstico rápido

Síntoma	Causa probable / solución
La página carga vacía	Revisar consola del navegador. A menudo, config.json corrupto o carpeta datos sin permisos. Comprobar con ?action=version.
No guarda nada	Permisos de escritura en datos/ (755/775). Verificar sePuedeEscribir en ?action=version.
config.json accesible por URL	.htaccess de datos ausente o inactivo. Crítico: contiene credenciales.
No se ve la app (hay WordPress)	Falta el .htaccess con RewriteEngine Off en la raíz de la aplicación.
Los automáticos no salen	Tarea cron no configurada, mal la clave, o SMTP desactivado. Ejecutar cron.php a mano por web para ver el informe.
Versión antigua tras actualizar	Caché de servidor. Purgar (LiteSpeed / Varnish / plugin).

Soporte técnico

Eduardo García Gómez
info@calmalogic.com
calmalogic ®

Sistema de Reservas de Eventos, versión 3.19
La parte de programación ha sido gestionada por Claude (Anthropic)