# Manual de Programador - EVA

**Sistema Inteligente de Monitoreo y Telemetría de Tanques de Agua**

**Documentación Técnica Oficial de Arquitectura, Código Fuente, Base de Datos y Mantenimiento**

**Versión del Manual: FINAL Ampliada (Fusión V0.1.0 + V0.2.0) — Septiembre 2026**

**Repositorio: Proyecto-Anual-Web | Tag / Release: v0.2.0**

**Autoría: Arquitectura de Software & Lead Technical Writing**

# Manual de Programador - EVA

**Indice:**




**Autores: Zoe Linares - Alina Romero - Valentin Wagner**

## 0. CÓMO LEER ESTE MANUAL

### 0.1 Propósito y Filosofía del Documento

Este manual constituye la referencia técnica definitiva, exhaustiva y sin recortes del proyecto **EVA (El Vigilante del Agua)**. Ha sido elaborado mediante la auditoría directa del código fuente PHP, esquemas SQL y firmas de funciones del repositorio en su tag v0.2.0.

La regla de oro de este documento es la **fidelidad absoluta al código**: no se documenta ninguna funcionalidad hipotética o deseada. Todo lo descrito existe materialmente en los archivos del proyecto. Cuando una vista es una maqueta estática, una API carece de dependencias o una tabla no está presente en el script de volcado de la base de datos (dump), se especifica explícitamente con su respectiva marca de estado y su ubicación exacta en el código.

### 0.2 Convenciones del Código Fuente

El desarrollo de EVA se rige por una serie de directrices de diseño y patrones de arquitectura que determinan la estructura de cada archivo PHP y la interacción con la base de datos:

1. **Uso del Idioma Español en el Dominio Propio:**
    - **Entidades, Variables y Lógica:** Toda la capa de aplicación, nombres de funciones, variables asociativas, clases CSS y textos de interfaz de usuario están redactados en español ($nivel_actual, badgeEstado, eva_procesar_tanque(), tecnicos.php).
    - **Excepciones de Lenguaje:** Únicamente se utiliza inglés en las palabras clave del lenguaje PHP (foreach, try, catch, class), nombres de columnas físicas de la

base de datos (password_hash, fecha_hora, updated_at) y funciones nativas del entorno (htmlspecialchars, password_verify, bin2hex).

2. **Ausencia de Comentarios por Función:**
    - El código del proyecto sigue el principio de autodocumentación por nombres descriptivos. En consecuencia, el repositorio casi no contiene comentarios de cabecera en las funciones. Este manual compensa dicha característica desglosando el 100% de los helpers, métodos y algoritmos internos.
3. **Arquitectura Autocontenida (PHP Vanilla Sin Framework):**
    - No existe un enrutador central (*router*), ni gestor de dependencias (*Composer*), ni motor de plantillas (*Twig/Blade*).
    - Cada archivo PHP que responde a una vista web funciona como un punto de entrada independiente (*entrypoint*). El ciclo de vida canónico de ejecución en casi todos los archivos sigue este orden riguroso:
        - Inicio de sesión mediante session_start().
        - Verificación de permisos y rol del usuario mediante un *Guard* de sesión.
        - Inclusión de la capa de acceso a datos con require_once __DIR__ . '/../config/db.php'.
        - Procesamiento de la lógica de negocio (evaluación de parámetros GET/POST, ejecución de consultas SQL).
        - Definición de funciones auxiliares (*helpers*) locales.
        - Renderizado directo de la estructura HTML, incrustando las variables procesadas.

4. **Patrón de Mapeo Parámetro-Vista para Estados (*Badges*):**
    - La representación visual de los estados de entidades (dispositivos, alertas, mantenimientos, tanques) no se hardcodea en el HTML, sino que utiliza arreglos asociativos en PHP.
    - *Estructura típica:* ['VALOR_BD' => ['cls' => 'clase-css', 'txt' => 'Texto Descriptivo']].
    - Ejemplos representativos se encuentran en admin/alertas.php (badgeTipoAlerta), admin/dispositivos.php (badgeEstadoDev) y cliente/includes/helpers.php (eva_alert_tipo_map).
5. **Sanitización de Salida y Validaciones de Entrada:**
    - Todos los filtros recibidos mediante querystring ($_GET) se validan contra listas blancas fijas (*whitelists*) mediante in_array($valor, $listaValida, true) antes de ser pasados a las consultas SQL.
    - Todo dato devuelto al navegador para renderizado HTML se escapa a través de la función nativa htmlspecialchars() o mediante el helper global h().
6. **Doble Capa de Acceso a Datos y la Regla Crítica del Entorno Servidor (mysqlnd):**
    - La infraestructura de base de datos expone dos objetos globales desde config/db.php: $conn (instancia de mysqli) y $pdo / eva_pdo() (instancia de PDO).
    - **REGLA CRÍTICA DE PRODUCCIÓN:** El servidor web de producción **NO TIENE instalada la extensión mysqlnd (MySQL Native Driver)** de PHP. Por esta razón, el método $stmt->get_result() de mysqli provoca un **Error Interno del Servidor (HTTP 500)** insalvable.

- **Implicaciones Prácticas en el Código:**
  - Todas las consultas preparadas modernas que reciben parámetros del usuario se deben ejecutar obligatoriamente mediante la interfaz **PDO** ($pdo->prepare(), $stmt->execute(), $stmt->fetchAll()).
  - La interfaz $conn (mysqli) se reserva únicamente para ejecutar consultas directas $conn->query() sobre tablas fijas donde no interviene ningún parámetro introducido por el usuario (ej. contadores generales del dashboard o listados globales).
  - En archivos legados como admin/edificios.php, coexiste un enfoque mixto donde la creación y edición usan mysqli con bind_param(), mientras que la eliminación y los listados avanzados emplean PDO.

### 0.3 Leyenda de Marcas de Estado

Para que el desarrollador identifique la madurez técnica de cada componente del repositorio, los desgloses de archivos utilizan las siguientes etiquetas:

| Marca | Definición Operativa |
|---|---|
| **REAL** | El script ejecuta consultas SQL dinámicas (INSERT, UPDATE, DELETE o SELECT parametrados) contra la base de datos activa y su respuesta/interfaz refleja datos en tiempo real. |

| | |
|---|---|
| **LECTURA** | El script realiza lecturas a la base de datos (SELECT), pero no ejecuta operaciones de escritura, edición o borrado de registros desde su propio cuerpo. |
| **UI** | Maqueta visual pura. El archivo contiene maquetación HTML/CSS/JS con datos estáticos simulados. Los formularios carecen de atributos action o name, o bien sus botones ejecutan funciones de JavaScript que no realizan peticiones HTTP ni persisten información. |
| **TRUNCO** | La capa lógica escrita en PHP está completamente codificada y funcional, pero el archivo sufrio un corte en la escritura del bloque HTML final (ejemplo: estilos no cerrados, etiquetas de formulario ausentes). |
| **MUERTA** | Código inejecutable en el entorno actual. Su invocación causa un error fatal de PHP debido a sentencias require o include que apuntan a archivos o librerías inexistentes en la estructura de directorios. |

## 1. VISIÓN GENERAL DE LA ARQUITECTURA

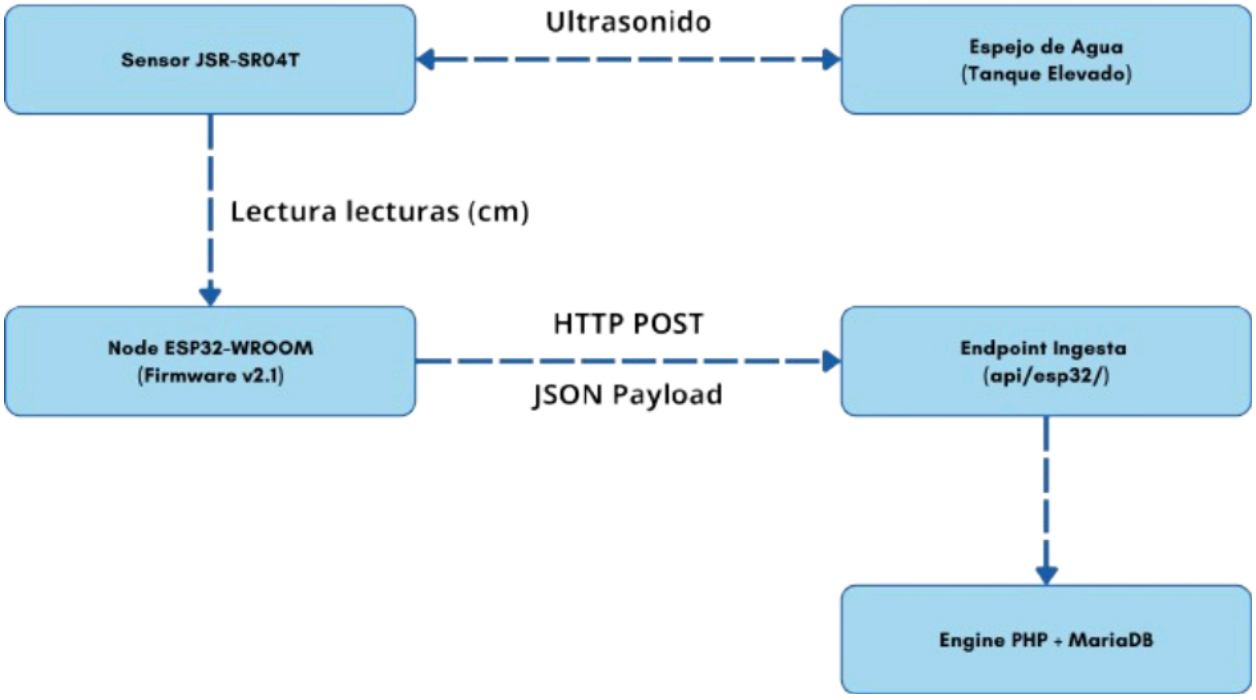
### 1.1 Problema que Resuelve el Sistema

El sistema **EVA (El Vigilante del Agua)** ha sido concebido para resolver la falta de visibilidad en la gestión hídrica de tanques elevados y subterráneos en consorcios, edificios residenciales y complejos industriales.

La cadena física y digital del sistema opera de la siguiente manera:

1. **Medición Física:** Cada estructura dispone de uno o varios tanques de almacenamiento. En la tapa superior de cada tanque se instala un dispositivo de telemetría **EVA (basado en el microcontrolador ESP32-WROOM-32)** equipado con un **sensor ultrasónico estanco JSR-SR04T**.
2. **Adquisición de Telemetría:** El sensor emite impulsos ultrasónicos hacia el espejo de agua y mide el tiempo de eco, determinando la distancia libre en centímetros. El microcontrolador calcula localmente o envía la distancia, el nivel de agua efectivo, el porcentaje de llenado y los litros estimados. Opcionalmente, transmite lecturas de temperatura y humedad ambiental.
3. **Procesamiento y Telemetría:** El ESP32 emite peticiones HTTP POST estructuradas en JSON hacia el servidor web. La capa de ingesta procesa la medición, actualiza los estados operativos del hardware y ejecuta algoritmos para la consolidación de consumos diarios y la detección de anomalías.
4. **Visibilidad Operativa:** Los datos son consumidos por tres perfiles diferenciados:
    - **Administradores:** Controlan la infraestructura global, inventario, usuarios, contratos y auditoría.

- **Técnicos:** Monitorean el estado del hardware, sensores, instalaciones y órdenes de trabajo.
- **Clientes (Consorcios/Vecinos):** Visualizan el nivel actual de su tanque en tiempo real, histórico de consumos, configuración de alertas personalizadas y canalizan solicitudes de mantenimiento.




### 1.2 Stack Tecnológico

La siguiente tabla resume los componentes tecnológicos verificados de forma directa en el código fuente del proyecto:

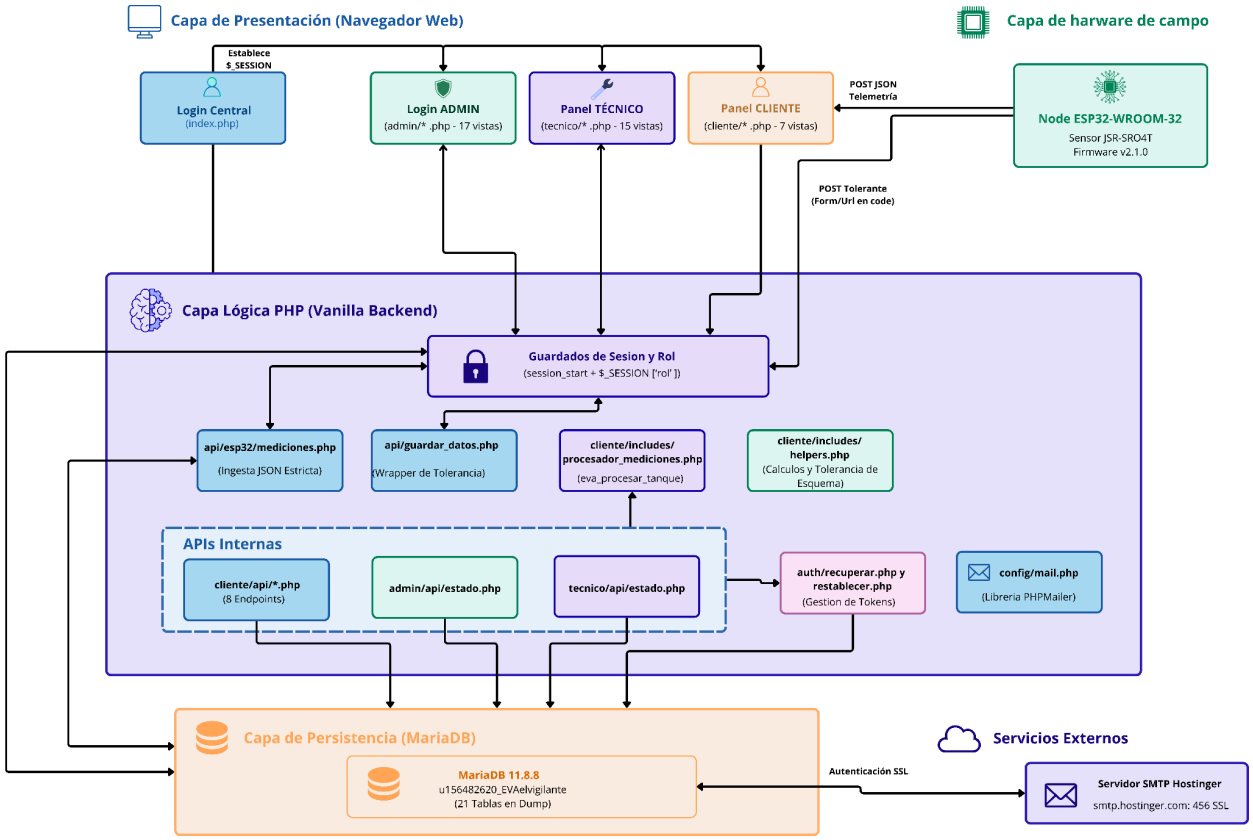
| Capa / Módulo | Tecnología Utilizada | Detalle de Implementación Verificado en Código |
|---|---|---|
| **Lenguaje Backend** | PHP 7.2+ / 8.x (Vanilla) | Sin frameworks (*Laravel, Symfony*). Autocontenido. declare(strict_types=1) presente en scripts críticos del módulo cliente y en la ingesta. |
| **Acceso a Datos** | Dual: mysqli ($conn) + PDO ($pdo / eva_pdo()) | Implementado en config/db.php. Configuración de PDO con ERRMODE_EXCEPTION, FETCH_ASSOC y EMULATE_PREPARES = false. Aplica la regla crítica anti-mysqlnd. |
| **Motor de BD** | MariaDB 11.8.8 / MySQL 5.7+ | Collation utf8mb4_unicode_ci, motor InnoDB. Nombre base del dump: u156482620_EVAelvigilante (con fallback de conexión a u767580032_elvigilante). Dump generado mediante phpMyAdmin 5.2.2 el 25-08-2026. |

| | | |
|---|---|---|
| **Capa Frontend** | HTML5 + CSS3 + JavaScript (Vanilla) | Hojas de estilo dedicadas por panel: admin/css/admin.css (29 KB), cliente/css/style.css (33 KB), tecnico/css/tecnico.css (26 KB). Scripts: admin/js/admin.js (25 KB), cliente/js/script.js (34 KB), tecnico/js/tecnico.js (18 KB). Renderizado de gráficos mediante SVG inline dinámicos (donuts, gauges, clipPath de tanques con ondas). Cero dependencias de UI de terceros. |
| **Sincronización Real** | Polling HTTP vía Fetch API (cada 3s) | Scripts cliente/js/tiempo-real.js, tecnico/js/tiempo-real.js y admin/js/tiempo-real.js. Peticiones dirigidas a sus respectivos endpoints api/estado.php. Implementa control de solapamiento (evaPollInFlight), cabecera Cache-Control: no-store y suspensión cuando la pestaña está oculta (document.hidden). |

| | | |
|---|---|---|
| **Servicio de Correo** | PHPMailer 6.9.1 (Vendorizado) | Ubicado en vendor/phpmailer/src/ (Exception.php, PHPMailer.php, SMTP.php). Configurado para salida SMTP cifrada a través de smtp.hostinger.com:465 (SSL) con timeout de 15 segundos y conjunto de caracteres UTF-8. |
| **Hardware & IoT** | ESP32-WROOM -32 + JSR-SR04T | Firmware registrado: v2.1.0-ESP32. Ingesta procesada vía api/esp32/mediciones.php (POST JSON estricto) y api/guardar_datos.php (wrapper de tolerancia). Persiste en la tabla mediciones y actualiza la columna ultima_conexion en dispositivos. |
| **Entorno / Config** | Parser nativo de variables .env | Implementado en config/env.php. Carga claves DB_*, SMTP_*, BASE_URL y APP_DEBUG. Soporta defaults hardcoded en los archivos de configuración si el archivo .env no está en el servidor. |

| | | |
|---|---|---|
| **Control de Versiones** | Git | Tags de release identificados en la documentación del repositorio: v0.1.0 y v0.2.0. |

## 1.3 Diagrama de Arquitectura de Componentes



### 1.4 Diagrama de Secuencia: Ingesta de Telemetría ESP32

### 1.5 Diagrama de Secuencia: Autenticación con Rate-Limiting

### 1.6 Configuración del Entorno de Desarrollo y Variables

#### Requisitos del Servidor Web

- **Servidor HTTP:** Apache 2.4+ con mod_rewrite activo, o Nginx 1.18+ con PHP-FPM.
- **Intérprete Backend:** PHP 7.4 u 8.x con las extensiones php-mysqli, php-pdo, php-pdo_mysql, php-json, php-mbstring y php-fileinfo.
- **Motor de Base de Datos:** MariaDB 10.4+ o MySQL 5.7+.
- **Herramientas de Construcción:** No se requiere npm, Node.js, Composer ni minificadores. El proyecto funciona sirviendo directamente el directorio raíz.

#### Variables de Entorno Globals (.env)

El sistema integra un cargador nativo definido en config/env.php que lee el archivo .env ubicado en la raíz del proyecto. El parser procesa pares CLAVE=VALOR ignorando comentarios que inicien con # o ;.

| Variable | Valor por Defecto si Ausente | Propósito Operativo |
|---|---|---|

| | | |
|---|---|---|
| DB_HOST | localhost | Nombre de host o IP del servidor de base de datos. |
| DB_USER | null (Utiliza candidatos hardcoded) | Usuario de conexión a MariaDB. |
| DB_PASS | null (Utiliza candidatos hardcoded) | Contraseña de conexión a MariaDB. |
| DB_NAME | null (Utiliza candidatos hardcoded) | Nombre de la base de datos de producción. |
| SMTP_HOST | smtp.hostinger.com | Servidor SMTP para el envío de notificaciones. |
| SMTP_PORT | 465 | Puerto de conexión cifrada del servidor SMTP. |

| | | |
|---|---|---|
| SMTP_USER | no-reply@dashboard.elvigilanteagua.com | Usuario o cuenta de correo emisor para SMTP. |
| SMTP_PASS | #VALzona122233 | Credencial cifrada de acceso al servidor SMTP. |
| SMTP_ENCRYPTION | ssl | Cifrado de transporte (ssl para SMTPS, tls para STARTTLS). |
| SMTP_FROM_EMAIL | no-reply@dashboard.elvigilanteagua.com | Dirección que aparecerá en el encabezado From. |
| SMTP_FROM_NAME | EVA - El Vigilante de Agua | Nombre visible del remitente en los correos enviados. |
| BASE_URL | [https://dashboard.elvigilantedeagua.com](https://dashboard.elvigilantedeagua.com) | Dominio base para la generación de enlaces absolutos de recupero. |

| | | |
|---|---|---|
| APP_DEBUG | 0 | Alterna el despliegue de errores (1 activa display_errors). |

## 2. ESTRUCTURA GLOBAL DE DIRECTORIOS

### 2.1 Árbol Completo Anotado del Repositorio

```
Proyecto-Anual-Web/
├── index.php                       # Login central (216 líneas). Rate-limiting, autenticación, sesiones y derivación por rol. (REAL)
├── AGENTS.md                       # Especificación técnica de directrices del proyecto para agentes de IA y desarrolladores.
├── RESUMEN_V1.md                   # Bitácora de cambios y decisiones de diseño de la versión v0.1.0.
├── RESUMEN_V2.md                   # Bitácora de cambios y nuevas tablas de la versión v0.2.0.
├── CHANGELOG.md                    # Histórico de versiones del sistema.
├── LICENSE                         # Licencia de distribución del software.
├── .gitignore                      # Exclusiones de control de versiones (.env, credenciales SMTP, volcados locales).
├── Manual de Programador...pdf     # Documento de referencia de formato de entrega.
├── MANUAL_DEL_PROGRAMADOR.md       # Manual V1 original (73 KB).
├── MANUAL_DEL_PROGRAMADOR_V2.md    # Manual V2 extendido (128 KB).
│
├── config/                         # CAPA DE CONFIGURACIÓN Y INFRAESTRUCTURA (6 archivos)
│   ├── env.php                     # Parser nativo de variables de entorno .env (19 líneas). (REAL)
│   ├── db.php                      # Conexión dual mysqli ($conn) y PDO ($pdo / eva_pdo()) con candidatos (69 líneas). (REAL)
│   ├── mail_config.php             # Parámetros array SMTP y URL base (12 líneas). (REAL)
│   ├── mail.php                    # Motores de correo PHPMailer y generador de contraseñas (117 líneas). (REAL)
│   ├── auth.php                    # Guard genérico básico de sesión activa (6 líneas). (REAL)
│   └── logout.php                  # Destrucción de sesión y redirección al login (6 líneas). (REAL)
│
├── auth/                           # MÓDULO DE RECUPERACIÓN DE CREDENCIALES (2 archivos)
│   ├── recuperar.php               # Emisión de tokens de recupero y envío por correo (38 líneas). (PHP REAL / HTML TRUNCO)
│   └── restablecer.php             # Verificación de token, hash de clave y actualización (26 líneas). (PHP REAL / HTML TRUNCO)
│
├── api/                            # INGESTA PÚBLICA DE DISPOSITIVOS IoT (2 archivos)
│   ├── guardar_datos.php           # Wrapper de tolerancia de ingesta (JSON/urlencoded) (141 líneas). (REAL)
│   └── esp32/
│       └── mediciones.php          # Endpoint canónico estricto de recepción de telemetría ESP32 (125 líneas). (REAL)
```

```
│
├── admin/                          # PANEL DE ADMINISTRACIÓN GENERAL (24 archivos)
│   ├── index.php                   # Dashboard global: contadores, gráficos SVG y tablas de estado. (LECTURA)
│   ├── alertas.php                 # Control de alertas, transiciones de estado y log de auditoría. (REAL)
│   ├── empresas.php                # CRUD de empresas, validación de CUIT y restricciones de borrado. (REAL)
│   ├── reportes.php                # Mapeo de métricas del sistema y actividad mensual. (LECTURA)
│   ├── mantenimientos.php          # Programación de mantenimientos y gestión de solicitudes de clientes. (LECTURA)
│   ├── dispositivos.php            # CRUD de dispositivos ESP32, versiones de firmware y estados. (REAL)
│   ├── tanques.php                 # CRUD de tanques de agua, cálculo de volumen y soft-delete. (REAL)
│   ├── sensores.php                # CRUD de sensores JSN-SR04T y bloqueo por mediciones vinculadas. (REAL)
│   ├── roles.php                   # Interfaz estática de matriz de permisos por rol. (UI)
│   ├── tecnicos.php                # Gestión del personal técnico y trazabilidad de intervenciones. (REAL)
│   ├── usuarios.php                # CRUD centralizado de usuarios, hashing de claves y envío de mails. (REAL)
│   ├── instalaciones.php           # Registro unificado de instalaciones en campo. (LECTURA)
│   ├── inventario.php              # Control de stock y movimientos de insumos (ENTRADA/SALIDA). (REAL)
│   ├── compras.php                 # Gestión de órdenes de compra y alta automática de clientes. (REAL)
│   ├── clientes.php                # Alta de clientes finales, generación de credenciales y usuario. (REAL)
│   ├── edificios.php               # CRUD de edificios con lógica de conexión mixta mysqli/PDO. (REAL)
│   ├── auditorias.php              # Explorador del log de actividad del sistema (log_actividad). (LECTURA)
│   ├── api/
│   │   └── estado.php              # Endpoint JSON para el polling del dashboard de administración. (REAL)
│   ├── includes/
│   │   ├── header.php              # Encabezado parcial: perfil, notificaciones fijas y switch de tema.
│   │   └── sidebar.php             # Navegación lateral con los 17 accesos del panel de administración.
│   ├── js/
│   │   ├── admin.js                # Lógica frontend de cuadros, modales, gráficos SVG y filtros (25 KB).
│   │   ├── tiempo-real.js          # Motor de polling contra admin/api/estado.php (3.6 KB).
│   │   └── auditorias.php          # Archivo duplicado e inejecutable por errores sintácticos. (MUERTA)
│   └── css/
│       └── admin.css               # Hojas de estilo unificadas del panel de administración (29 KB).
```

```
├── tecnico/                          # PANEL DE TRABAJO PARA TÉCNICOS DE CAMPO (29 archivos)
│   ├── indextec.php                  # Dashboard técnico: métricas de campo y estado de sensores. (REAL)
│   ├── alertas.php                   # Alertas asignadas a las instalaciones del técnico. (REAL)
│   ├── historialtecnico.php          # Registro histórico estático de intervenciones. (UI)
│   ├── instalaciones.php             # Fichas estáticas de instalaciones asignadas. (UI)
│   ├── instalar.php                  # Formulario visual de alta de nueva instalación. (UI)
│   ├── instprogramar.php             # Formulario visual para programar instalaciones futuras. (UI)
│   ├── mantenimientos.php            # Fichas estáticas de tareas de mantenimiento. (UI)
│   ├── mantagregar.php               # Formulario visual de adición de sensores. (UI)
│   ├── manteliminar.php              # Formulario visual de desincorporación de sensores. (UI)
│   ├── mediciones.php                # Explorador de las últimas 50 mediciones de dispositivos asignados. (REAL)
│   ├── misdispositivos.php           # Tarjetas informativas de dispositivos vinculados al técnico. (REAL)
│   ├── mitanques.php                 # Maqueta de tanques asignados y sus niveles. (UI)
│   ├── notificaciones.php            # Centro estático de notificaciones técnicas. (UI)
│   ├── perfil.php                    # Ficha de perfil del técnico. (UI)
│   ├── sensores.php                  # Catálogo estático de sensores e interfaz de calibración. (UI)
│   ├── api/
│   │   ├── estado.php                # Única API funcional del módulo técnico para el polling de interfaz. (REAL)
│   │   ├── dashboard.php             # API muerta por falta de librerías base (config/db_pdo.php). (MUERTA)
│   │   ├── tanques.php               # API muerta por falta de librerías base. (MUERTA)
│   │   ├── sensores.php              # API muerta por falta de librerías base. (MUERTA)
│   │   ├── perfil.php                # API muerta por falta de librerías base. (MUERTA)
│   │   ├── notificaciones.php        # API muerta por falta de librerías base. (MUERTA)
│   │   ├── mediciones.php            # API muerta por falta de librerías base. (MUERTA)
│   │   ├── mantenimientos.php        # API muerta por falta de librerías base. (MUERTA)
│   │   ├── instalaciones.php        # API muerta por falta de librerías base. (MUERTA)
│   │   ├── dispositivos.php         # API muerta por falta de librerías base. (MUERTA)
│   │   └── alertas.php               # API muerta por falta de librerías base. (MUERTA)
│   ├── js/
│   │   ├── tecnico.js                # Controladores de interfaz para modales y tarjetas (18 KB).
│   │   └── tiempo-real.js            # Polling de estado técnico contra tecnico/api/estado.php (5.8 KB).
│   └── css/
│       └── tecnico.css               # Estilos específicos del panel de técnicos (26 KB).
```

```
├── cliente/                              # PANEL DE USUARIO Y CLIENTE FINAL (21 archivos)
│   ├── indexcli.php                      # Dashboard resumen: nivel, autonomía, consumo y gauge interactivo. (REAL)
│   ├── mitanque.php                      # Vista detallada con tanque animado en SVG y lecturas semanales. (REAL)
│   ├── alertas.php                       # Listado de alertas asociadas a los tanques del cliente. (REAL)
│   ├── historial.php                     # Gráficos de tendencias, agregaciones diarias y descargas. (REAL)
│   ├── configuracion.php                 # Ajuste de umbrales de alerta (nivel mínimo y máximo). (REAL)
│   ├── mantenimiento.php                 # Solicitud de asistencia técnica, carga de foto e infraestructura. (REAL)
│   ├── perfil.php                        # Datos de la cuenta del cliente y resincronización de datos. (REAL)
│   ├── includes/
│   │   ├── helpers.php                   # Biblioteca de 20 funciones transversales de cálculo y formato. (REAL)
│   │   └── procesador_mediciones.php     # Motor del sistema: cálculo de consumos, estados y alertas. (REAL)
│   ├── api/
│   │   ├── tanque.php                    # Retorna estado puntual del tanque principal. (REAL)
│   │   ├── resumen.php                   # Métricas consolidadas para el dashboard principal. (REAL)
│   │   ├── estado.php                    # Endpoint unificado de telemetría para el polling en tiempo real. (REAL)
│   │   ├── historial.php                 # Agregación de datos históricos por periodos. (REAL)
│   │   ├── alertas.php                   # Filtrado de alertas en tiempo real. (REAL)
│   │   ├── configuracion.php             # Lectura y actualización de umbrales vía JSON. (REAL)
│   │   ├── procesar.php                  # Gatillo manual de ejecución del procesador de mediciones. (REAL)
│   │   └── sincronizar.php               # Consolidación manual de consumos y alertas sin nueva lectura. (REAL)
│   ├── js/
│   │   ├── script.js                     # Renderizado de gauges, tanques SVG y manipuladores (34 KB).
│   │   └── tiempo-real.js                # Polling unificado contra cliente/api/estado.php (6.8 KB).
│   ├── css/
│   │   └── style.css                     # Hojas de estilo generales del módulo cliente (33 KB).
│   └── uploads/
│       └── .gitkeep                      # Preservación de directorio para imágenes adjuntas de mantenimiento.
│
├── db/                                   # ESQUEMAS Y DUMPS DE BASE DE DATOS
│   └── u156482620_EVAelvigilante (1).sql # Volcado SQL de producción con estructura y datos iniciales (29 KB).
│
└── vendor/                               # LIBRERÍAS Y DEPENDENCIAS EXTERNAS
    └── phpmailer/                        # Repositorio estático de PHPMailer 6.9.1 (Exception, PHPMailer, SMTP).
```

### 2.2 Cuadro Recapitulativo de Archivos por Módulo

| Módulo / Directorio | Vistas PHP | APIs PHP | Includes PHP | Scripts JS | Estilos CSS | Otros Archivos | Total |
|---|---|---|---|---|---|---|---|
| Raíz del Proyecto | 1 | 0 | 0 | 0 | 0 | 8 | 9 |
| Configuración (config/) | 0 | 0 | 6 | 0 | 0 | 0 | 6 |
| Autenticación (auth/) | 2 | 0 | 0 | 0 | 0 | 0 | 2 |
| Ingesta IoT (api/) | 0 | 2 | 0 | 0 | 0 | 0 | 2 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Administración (admin/) | 17 | 1 | 3 | 2 | 1 | 0 | 24 |
| Técnico (tecnico/) | 15 | 11 | 0 | 2 | 1 | 0 | 29 |
| Cliente (cliente/) | 7 | 8 | 2 | 2 | 1 | 1 | 21 |
| Base de Datos (db/) | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Librerías (vendor/) | 0 | 0 | 3 | 0 | 0 | 0 | 3 |
| TOTAL GENERAL | 42 | 22 | 14 | 6 | 3 | 10 | 97 |

### 2.3 Inventario de Recursos e Infraestructura Inexistente

Durante la auditoría del código se identificó que el sistema realiza referencias a archivos y tablas que no se encuentran físicamente presentes en el repositorio, lo que explica la existencia de componentes marcados como **MUERTA**:

1. **Archivos de Configuración Inexistentes:**
    - config/db_pdo.php: Invocado por las 10 APIs muertas de tecnico/api/. Su ausencia interrumpe la instanciación del objeto PDO en dicho subdirectorio.
    - config/helpers.php: Invocado por las mismas 10 APIs. Requería funciones globales como requireTecnico(), getDB(), jsonResponse() y getInput(), las cuales no están definidas en ninguna parte del proyecto.
    - .env: Archivo de variables de entorno (excluido deliberadamente vía .gitignore). El sistema utiliza los valores por defecto integrados en config/db.php y config/mail_config.php.
2. **Directorios de Almacenamiento en Runtime:**
    - cliente/uploads/mantenimientos/: El directorio no existe inicialmente en el repositorio (solo se conserva el marcador cliente/uploads/.gitkeep). Sin embargo, el script cliente/mantenimiento.php incluye una rutina automatizada que detecta su ausencia y lo crea dinámicamente en tiempo de ejecución con permisos 0755, agregando además un archivo .htaccess de protección y un index.html vacío.
3. **Tablas Ausentes en el Dump Inicial (u156482620_EVAelvigilante (1).sql):**
    - El script SQL de volcado carece de las estructuras para las siguientes tablas requeridas por el código: password_resets, login_intentos, clientes, compras, productos, proveedores, credenciales_clientes, notificaciones_compras, historial_compras y solicitudes_mantenimiento (esta última es creada

automáticamente vía DDL dinámico por cliente/mantenimiento.php). Las estructuras SQL necesarias para completar estas tablas se detallan en la **Sección 9.6**.

## CAPA DE CONFIGURACIÓN (config/)

### 3.1 Propósito Arquitectónico del Módulo

El directorio config/ concentra la infraestructura del sistema: la gestión del entorno de ejecución, los mecanismos de conexión dual a la base de datos MariaDB, la configuración e implementación del motor de correo electrónico (PHPMailer) y los scripts globales de control de acceso y cierre de sesión.

Ninguno de los archivos de este directorio exige la verificación de un rol específico por sí mismo, ya que son incluidos por los scripts de los paneles o por la ingesta pública.

| Archivo | Líneas | Función Principal | Estado |
|---|---|---|---|
| config/env.php | 19 | Carga y parseo de variables de entorno .env. | **REAL** |

| | | | |
|---|---|---|---|
| config/db.php | 69 | Configuración de la conexión dual (mysqli y PDO). | **REAL** |
| config/mail_config.php | 12 | Definición de constantes y credenciales SMTP. | **REAL** |
| config/mail.php | 117 | Wrapper de PHPMailer y generador seguro de claves. | **REAL** |
| config/auth.php | 6 | Guard de autenticación genérico de sesión. | **REAL** |
| config/logout.php | 6 | Destrucción de variables de sesión y redirección. | **REAL** |

### 3.2 Desglose Archivo por Archivo

#### 1. config/env.php (19 líneas)

Este script implementa un parser liviano para leer variables declaradas en un archivo .env ubicado en la raíz del proyecto.

<?php

```
function eva_load_env(): void {
    static $loaded = false;
    if ($loaded) return;
    $loaded = true;
    $f = dirname(__DIR__) . '/.env';
    if (!is_file($f)) return;
    foreach (file($f, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES) as $ln) {
        $ln = trim($ln);
        if ($ln === '' || $ln[0] === '#' || $ln[0] === ';' || strpos($ln, '=') === false) continue;
        list($k, $v) = explode('=', $ln, 2);
        $k = trim($k);
        $v = trim(trim($v), "\"'"); // Elimina comillas envolventes simples o dobles
        if ($k === '') continue;
        $_ENV[$k] = $v;
        if (getenv($k) === false) putenv("$k=$v"); // No sobrescribe variables reales del SO
    }
}
```

```
function eva_env(string $key, $default = null) {
    eva_load_env();
    $g = getenv($key);
    return $g !== false ? $g : ($_ENV[$key] ?? $_SERVER[$key] ?? $default);
}
eva_load_env(); // Ejecución automática al incluir el archivo
```

**2. config/db.php (69 líneas)**

Es el núcleo de conectividad del sistema. Configura el nivel de reporte de errores evaluando la variable APP_DEBUG, e intenta conectarse iterando sobre una lista priorizada de credenciales (candidatos) hasta establecer una sesión válida. Expone simultáneamente el objeto $conn (mysqli) y la función/objeto $pdo / eva_pdo().

```
<?php
require_once __DIR__ . '/env.php';
error_reporting(E_ALL);
$dbg = strtolower((string)eva_env('APP_DEBUG', '0'));
ini_set('display_errors', ($dbg === '1' || $dbg === 'true') ? '1' : '0');
ini_set('log_errors', '1');
$host = eva_env('DB_HOST', 'localhost');
```

```
        // Arreglo de credenciales candidatas para tolerancia a fallos en entornos de
desarrollo/prod
        $candidatos = [];
        if (eva_env('DB_USER') && eva_env('DB_NAME')) {
            $candidatos[] = [eva_env('DB_USER'), eva_env('DB_PASS', ''),
eva_env('DB_NAME')];
        }
        $candidatos[] = ['u767580032_elvigilante', '#VALzona122233',
'u767580032_elvigilante'];
        $candidatos[] = ['u156482620_EVAelvigilante', '#VALzona122233',
'u156482620_EVAelvigilante'];
        $candidatos[] = ['root', '', 'u156482620_EVAelvigilante']; // Entorno local por
defecto
        $candidatos[] = ['root', '', 'u767580032_elvigilante'];
        $candidatos[] = ['root', 'root', 'u156482620_EVAelvigilante'];
        mysqli_report(MYSQLI_REPORT_OFF);
        foreach ($candidatos as list($user, $pass, $db)) {
            $conn = @new mysqli($host, $user, $pass, $db);
            if (!$conn->connect_error) {
```

```
            $conn->set_charset('utf8mb4');
            break;
        }
    }
    if (!isset($conn) || $conn->connect_error) {
        http_response_code(500);
        die('Error de conexion con la base de datos.');
    }
    try {
        $pdo = new PDO("mysql:host={$host};dbname={$db};charset=utf8mb4",
$user, $pass, [
            PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
            PDO::ATTR_EMULATE_PREPARES => false, // Obliga el uso de consultas
preparadas nativas del motor BD
        ]);
    } catch (Throwable $e) {
        $pdo = null;
```

```
        error_log('Error al inicializar PDO: ' . $e->getMessage());
    }
    function eva_pdo(): PDO {
        global $pdo;
        if (!$pdo instanceof PDO) {
            throw new RuntimeException('La conexion PDO no se encuentra
disponible.');
        }
        return $pdo;
    }
```

**3. config/mail_config.php (12 líneas)**

Retorna un arreglo asociativo con los parámetros de conexión para el servidor de correo SMTP.

```
<?php
require_once __DIR__ . '/env.php';
return [
    'host'       => eva_env('SMTP_HOST', 'smtp.hostinger.com'),
    'port'       => (int)eva_env('SMTP_PORT', 465),
```

```
        'username'   => eva_env('SMTP_USER',
'no-reply@dashboard.elvigilantedeagua.com'),

        'password'   => eva_env('SMTP_PASS', '#VALzona122233'),

        'encryption' => eva_env('SMTP_ENCRYPTION', 'ssl'),

        'from_email' => eva_env('SMTP_FROM_EMAIL',
'no-reply@dashboard.elvigilantedeagua.com'),

        'from_name'  => eva_env('SMTP_FROM_NAME', 'EVA - El Vigilante de
Agua'),

        'base_url'   => eva_env('BASE_URL',
'https://dashboard.elvigilantedeagua.com'),

    ];
```

**4. config/mail.php (117 líneas)**

Importa las clases de PHPMailer desde la carpeta vendor/ sin hacer uso de Autoloaders de Composer. Expone las siguientes funciones clave:

- eva_mail_config(): array: Carga los datos de mail_config.php y los combina con la dirección de soporte predeterminada (hola@eva.com.ar).
- eva_phpmailer_base(PHPMailer $mail, array $cfg): void: Configura la codificación UTF-8, el tiempo de espera (15 segundos), el cifrado SMTP (ssl mapeado a PHPMailer::ENCRYPTION_SMTPS, tls mapeado a PHPMailer::ENCRYPTION_STARTTLS) y las credenciales del emisor.

- eva_enviar_mail_credenciales($para_email, $para_nombre, $usuario_email, $password_temporal): bool: Construye un correo en formato HTML con diseño corporativo que incluye las credenciales temporales del usuario y un botón de acceso directo que apunta a BASE_URL.
- eva_enviar_mail_pedido_procesado($para_email, $para_nombre, $codigo_compra, $producto_nombre, $total): bool: Notifica al cliente el procesamiento de su orden de compra, aplicando formato numérico sobre los importes (number_format).
- eva_generar_password(int $len = 12): string: Generador de claves aleatorias. Si el parámetro $len es inferior a 10, fuerza una longitud de 12 caracteres. Utiliza el alfabeto ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnpqrstuvwxyz23456789 (excluyendo caracteres ambiguos como 0, O, 1, l) y la función criptográficamente segura random_int().

**5. config/auth.php (6 líneas)**

Constituye un Guard básico de sesión:

```
<?php

session_start();

if (!isset($_SESSION['rol'])) {

    header('Location: ../index.php');

    exit;

}
```

**6. config/logout.php (6 líneas)**

Cierra la sesión activa liberando la memoria y redirige al formulario de acceso:

```
<?php

session_start();

session_unset();

session_destroy();

header('Location: ../index.php');

exit;
```

## 4. CAPA DE AUTENTICACIÓN (index.php, auth/)

### 4.1 Propósito del Módulo

La capa de autenticación constituye la puerta de entrada para los tres perfiles de usuario. Se encarga de validar la identidad del usuario, gestionar la seguridad anti-fuerza bruta mediante el registro de intentos fallidos, inicializar las variables de sesión y dirigir al usuario al panel correspondiente según su rol.

| Archivo | Líneas | Lógica Backend | Interfaz Frontend | Estado |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| index.php | 216 | Autenticación, Rate-Limit, Sesiones, Hash, Redirección. | Formulario completo en HTML5/CSS3. | **REAL** |
| auth/recuperar.php | 38 | Generación de token binario (256-bit) y envío SMTP. | HTML **Trunco** (Corte en bloque CSS). | **REAL / TRUNCO** |
| auth/restablecer.php | 26 | Validación de token, caducidad y hash de clave. | HTML **Trunco** (Formulario no renderizado). | **REAL / TRUNCO** |

**4.2 Flujo Detallado de Login (index.php)**

1. **Verificación Previa de Sesión:**
    - Al cargar el archivo, se verifica si el usuario ya cuenta con una sesión activa. Si la variable $_SESSION['rol'] existe, el script realiza una redirección inmediata mediante la cabecera Location: sin renderizar el formulario:
        - ADMIN $\rightarrow$ admin/index.php

- TECNICO $\rightarrow$ tecnico/indextec.php
- USUARIO $\rightarrow$ cliente/indexcli.php

2. **Recepción y Sanitización de Datos:**
    - El formulario envía los datos mediante el método POST.
    - Parámetros recibidos: email (procesado con trim()) y password.
    - El campo recordar (checkbox) es de carácter visual y no interactúa con la sesión ni genera cookies de persistencia.
    - Validación del formato del correo electrónico mediante filter_var($email, FILTER_VALIDATE_EMAIL) y verificación de presencia de texto en el campo de contraseña.
3. **Mecanismo de Rate-Limiting Anti-Fuerza Bruta:**
    - El sistema consulta la tabla login_intentos contabilizando las fallas asociadas al correo ingresado en los últimos 15 minutos:

```
$chk = $pdo->prepare("SELECT COUNT(*) FROM login_intentos
        WHERE email = :e AND exito = 0
        AND fecha_hora >= DATE_SUB(NOW(), INTERVAL 15 MINUTE)");
$chk->execute([':e' => $email]);
if ((int)$chk->fetchColumn() >= 5) {
   $error = 'Demasiados intentos fallidos. Por seguridad, esperá 15 minutos.';
}
```

- *Aclaración técnica:* La evaluación se realiza por el identificador de email. Si la tabla login_intentos no existe en la base de datos, el bloque catch captura la excepción permitiendo que el flujo continúe en modo *fail-open*.

4. **Verificación de Credenciales:**

- Si no se ha superado el límite de intentos, el sistema consulta la tabla usuarios combinada con roles:

```
$stmt = $pdo->prepare("SELECT u.id_usuario, u.nombre, u.apellido, u.email,
u.password_hash, u.activo, r.nombre AS rol
        FROM usuarios u
        INNER JOIN roles r ON u.id_rol = r.id_rol
        WHERE u.email = :email LIMIT 1");
$stmt->execute([':email' => $email]);
$usuario = $stmt->fetch();
```

- **Usuario no encontrado:** Registra el fallo en login_intentos (exito = 0) y retorna el mensaje *"Usuario no encontrado"*.
- **Usuario inactivo:** Si activo != 1, detiene la ejecución con el mensaje *"Cuenta deshabilitada"*.
- **Validación de Hash:** Evalúa la contraseña utilizando la función nativa password_verify($password, $usuario['password_hash']).
    - *Si la contraseña es incorrecta:* Registra el intento fallido (exito = 0) en login_intentos y muestra *"Contraseña incorrecta"*.
    - *Si la contraseña es correcta:* Regenera el identificador de la sesión con session_regenerate_id(true) para prevenir ataques de fijación de sesión, registra el acceso exitoso (exito = 1) en login_intentos, actualiza el campo ultimo_acceso = NOW() en la tabla usuarios y puebla la sesión global.

5. **Estructura de la Variable Global $_SESSION:** Una vez completada la autenticación, la sesión del usuario contiene las siguientes claves:

- $_SESSION['id_usuario']: Entero (ID primario de la tabla usuarios).
- $_SESSION['nombre']: Cadena con el nombre de pila.
- $_SESSION['apellido']: Cadena con el apellido del usuario.
- $_SESSION['email']: Cadena con la dirección de correo autenticada.
- $_SESSION['rol']: Cadena exacta con el rol del usuario (ADMIN, TECNICO o USUARIO).

### 4.3 Módulo de Recuperación de Credenciales (auth/)

**auth/recuperar.php (38 líneas)**

Procesa la solicitud de reinicio de clave mediante correo electrónico.

```
<?php
session_start();
require_once __DIR__ . '/../config/db.php';
require_once __DIR__ . '/../config/mail.php';

$pdo = eva_pdo();
$email = trim($_POST['email'] ?? '');

if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $stmt = $pdo->prepare("SELECT id_usuario, nombre, apellido FROM usuarios
WHERE email = :e LIMIT 1");
```

```
    $stmt->execute([':e' => $email]);
    $u = $stmt->fetch();

    if ($u) {
        $token = bin2hex(random_bytes(32)); // Genera un token seguro de 64 caracteres
hexadecimales (256 bits)

        $ins = $pdo->prepare("INSERT INTO password_resets (email, token, expira)
                              VALUES (:e, :t, DATE_ADD(NOW(), INTERVAL 1 HOUR))");
        $ins->execute([':e' => $email, ':t' => $token]);

        $link = eva_mail_config()['base_url'] . '/auth/restablecer.php?token=' . $token;

        // Formateo de correo y envío mediante eva_phpmailer_base()
    }
    // Mensaje unificado para evitar la enumeración de usuarios registrados
    $msg = 'Si el email ingresado se encuentra registrado, recibirás un correo con las
instrucciones.';
}
```

**auth/restablecer.php (26 líneas)**

Recibe el token seguro vía parámetros GET o POST y actualiza la contraseña en la base de datos:

1. **Validaciones:** Comprueba que la contraseña tenga una longitud mínima de 6 caracteres y que coincida exactamente con la confirmación enviada ($_POST['confirm']).
2. **Verificación del Token:**

```
$stmt = $pdo->prepare("SELECT email, expira, usado FROM password_resets WHERE token = :t LIMIT 1");
$stmt->execute([':t' => $token]);
$r = $stmt->fetch();

if (!$r || $r['usado'] == 1 || strtotime($r['expira']) < time()) {
    $err = 'El token de recupero es invalido o ha expirado.';
} else {
    $hash = password_hash($pass, PASSWORD_DEFAULT);
    $upd = $pdo->prepare("UPDATE usuarios SET password_hash = :h WHERE email = :e");
    $upd->execute([':h' => $hash, ':e' => $r['email']]);

    $burn = $pdo->prepare("UPDATE password_resets SET usado = 1 WHERE token = :t");
    $burn->execute([':t' => $token]);

    $msg = "Contraseña restablecida correctamente. <a href='../index.php'>Iniciar sesion</a>";
}
```

## 5. DESGLOSE DEL MÓDULO admin/

### 5.1 Propósito y Control de Acceso

El módulo de Administración General (admin/) provee la interfaz para la gestión global de la infraestructura: supervisión de tanques, dispositivos, empresas, compras, inventario, usuarios y registros de auditoría.

**Guard Estándar de Vistas HTML (admin/*.php)**

Cada vista HTML del panel de administración inicia con la siguiente rutina de verificación de sesión:

```
<?php
session_start();
if (!isset($_SESSION['rol']) || $_SESSION['rol'] !== 'ADMIN') {
    header("Location: ../index.php");
    exit;
}
```

**Guard Estándar para Endpoints de API (admin/api/estado.php)**

Los endpoints JSON del módulo aplican un Guard que retorna directamente un código HTTP 401 en lugar de redirigir:

```
<?php
declare(strict_types=1);
session_start();
header('Content-Type: application/json; charset=utf-8');
```

```
header('Cache-Control: no-store, no-cache, must-revalidate');

if (!isset($_SESSION['rol']) || $_SESSION['rol'] !== 'ADMIN') {
    http_response_code(401);
    echo json_encode(['error' => 'No autorizado']);
    exit;
}
```

### 5.2 Mapa de Archivos del Módulo admin/

| Archivo | Propósito Operativo | Tipo de Operación | Estado |
|---|---|---|---|
| index.php | Dashboard consolidado: contadores globales, gráficos SVG y tablas de estado. | Consulta de Datos | **LECTURA** |
| alertas.php | Gestión del ciclo de vida de alertas (PENDIENTE $\rightarrow$ ATENDIDA | Lectura y Escritura | **REAL** |

| | | | |
|---|---|---|---|
| | $\rightarrow$ CERRADA). | | |
| empresas.php | CRUD de empresas cliente, validación de CUIT único y verificación de vínculos. | Lectura y Escritura | **REAL** |
| reportes.php | Consolidación estadística de la actividad operativa del sistema. | Consulta de Datos | **LECTURA** |
| mantenimientos.php | Administración de mantenimientos y solicitudes emitidas por los clientes. | Consulta de Datos | **LECTURA** |
| dispositivos.php | CRUD de dispositivos ESP32, control de firmware y soft-delete por dependencias. | Lectura y Escritura | **REAL** |

| | | | |
|---|---|---|---|
| tanques.php | CRUD de tanques, cálculo de volúmenes y soft-delete de unidades. | Lectura y Escritura | **REAL** |
| sensores.php | CRUD de sensores JSR-SR04T y bloqueo de borrado por lecturas asociadas. | Lectura y Escritura | **REAL** |
| roles.php | Matriz visual de permisos asignados a cada rol del sistema. | Maqueta Visual | **UI** |
| tecnicos.php | Gestión del personal técnico, estado de cuenta y trazabilidad de servicios. | Lectura y Escritura | **REAL** |
| usuarios.php | CRUD centralizado de usuarios, generación de | Lectura y Escritura | **REAL** |

| | | | |
|---|---|---|---|
| | claves y envío por correo. | | |
| instalaciones.php | Registro unificado de instalaciones en campo. | Consulta de Datos | **LECTURA** |
| inventario.php | Control de stock de insumos y registro de movimientos (ENTRADA/SALIDA). | Lectura y Escritura | **REAL** |
| compras.php | Máquina de estados de compras y alta automática de cuentas de cliente. | Lectura y Escritura | **REAL** |
| clientes.php | Alta de clientes en transacción (crea cliente, usuario y credenciales). | Lectura y Escritura | **REAL** |

| | | | |
|---|---|---|---|
| edificios.php | CRUD de inmuebles con lógica de conexión mixta (mysqli y PDO). | Lectura y Escritura | **REAL** |
| auditorias.php | Explorador del registro de actividad general del sistema (log_actividad). | Consulta de Datos | **LECTURA** |
| api/estado.php | Endpoint JSON utilizado por el mecanismo de polling del dashboard. | Consulta de Datos | **REAL** |
| includes/header.php | Componente parcial: perfil, barra superior, notificaciones y tema visual. | Componente Vista | **REAL** |
| includes/sidebar.php | Componente parcial: menú de navegación con los 17 accesos del panel. | Componente Vista | **REAL** |

| | | | |
|---|---|---|---|
| js/auditorias.php | Archivo duplicado en subdirectorio incorrecto con errores sintácticos. | Inejecutable | **MUERTA** |

**5.3 Desglose Detallado de Scripts y Flujos del Módulo admin/**

**1. admin/index.php (Dashboard Principal)**

- **Helpers locales:**
  - badgeEstado($e): Mapea estados de dispositivos (ONLINE $\rightarrow$ verde, OFFLINE $\rightarrow$ rojo, MANTENIMIENTO $\rightarrow$ amarillo).
  - badgeBateria($b): Formatea el porcentaje de batería y retorna una clase CSS según el nivel ($\ge 60\%$ activo, $\ge 30\%$ advertencia, $< 30\%$ bajo).
  - badgeSenal($s): Evalúa el indicador RSSI en dBm ($\ge -60$ Fuerte, $\ge -80$ Media, $< -80$ Débil).
  - badgeAlerta($t): Aplica formato al identificador del tipo de alerta reemplazando guiones bajos por espacios.
  - tiempoDesde($fecha): Calcula la diferencia temporal entre el instante actual y la fecha pasada por parámetro, retornando cadenas legibles (*"Hace 5 min"*, *"Hace 2 días"*).
- **Consultas de Datos:** Ejecuta consultas directas vía $conn->query() para obtener los contadores globales: total de clientes, técnicos activos, edificios, dispositivos registrados, tanques monitoreados, sensores y estado operativo de las unidades. Obtiene listados limitados con las últimas 5 intervenciones, alertas recientes y dispositivos con última conexión actualizada.

**2. admin/alertas.php (Gestión de Alertas)**

- **Funciones:**
  - countAlertasPorTipo(PDO $pdo, string $tipo): int: Retorna la cantidad de alertas registradas según su tipo mediante consultas preparadas.
  - badgeTipoAlerta($tipo): Retorna etiquetas legibles y clases CSS para NIVEL_BAJO, NIVEL_ALTO, SIN_CONEXION, FALLA_SENSOR y CONSUMO_ANORMAL.
  - badgeEstadoAlerta($estado): Mapea los tres estados de la máquina de estados: PENDIENTE, ATENDIDA y CERRADA.
- **Procesamiento de Mutaciones (POST):**

  Recibe los campos accion (atender o cerrar) e id_alerta. Ejecuta la actualización del estado y genera una entrada en la tabla log_actividad:

```
$stmt = $pdo->prepare("UPDATE alertas SET estado = :e WHERE id_alerta = :id");
$stmt->execute([':e' => $nuevoEstado, ':id' => $idAlerta]);


$log = $pdo->prepare("INSERT INTO log_actividad (id_usuario, accion, detalle, ip,
fecha_hora)
                    VALUES (:u, 'UPDATE', :d, :ip, NOW())");
$log->execute([
   ':u'  => $_SESSION['id_usuario'],
   ':d'  => "Alerta #{$idAlerta} actualizada a estado {$nuevoEstado}",
   ':ip' => $_SERVER['REMOTE_ADDR']
]);
```

**3. admin/empresas.php (Gestión de Empresas Cliente)**

- **Mutaciones (POST):**
    - accion = 'crear': Valida la unicidad del CUIT. Si el CUIT no existe en la base de datos, ejecuta un INSERT registrando la razón social, CUIT, dirección, teléfono, correo electrónico y fecha de alta.
    - accion = 'editar': Actualiza los datos de la empresa verificando que el CUIT no esté en uso por otra razón social distinta.
    - accion = 'toggle_estado': Invierte el valor del campo booleano activo.
    - accion = 'eliminar': Antes de proceder con la eliminación física de la empresa, verifica si existen edificios o instalaciones asociadas a sus usuarios. Si existen registros vinculados, cancela la operación informando al usuario para preservar la integridad referencial.

**4. admin/reportes.php (Reportes del Sistema)**

- **Operativa:** Realiza consultas estadísticas a la base de datos agrupando información de uso de los últimos 30 días (GROUP BY DATE(fecha_hora) sobre log_actividad).
- *Aclaración técnica:* El formulario visible en la interfaz para *"Generar Reporte Personalizado"* es una maqueta de frontend (UI), ya que la función generarReporte() de admin.js no realiza peticiones de exportación al servidor.

**5. admin/mantenimientos.php (Programación y Solicitudes)**

- **Funciones:**
    - buildMantQuery(...): Construye dinámicamente sentencias SQL para filtrar mantenimientos combinando parámetros opcionales de búsqueda, tipo (PREVENTIVO, CORRECTIVO, PREDICTIVO), estado y rangos de fecha.

- **Procesamiento de Solicitudes:** Muestra en pestañas separadas los mantenimientos programados por el personal administrativo y las solicitudes enviadas por los clientes desde cliente/mantenimiento.php.

**6. admin/dispositivos.php (Gestión de Dispositivos ESP32)**

- **Creación y Edición (POST):** Registra o modifica datos de dispositivos (nombre, MAC Address, dirección IP, estado, versión de firmware).
- **Trazabilidad de Estados:** Si al editar un dispositivo se detecta un cambio en su estado operativo, el script inserta un registro en la tabla historial_estado_dispositivo conservando el estado anterior, el nuevo y el usuario responsable del cambio.
- **Mecanismo de Soft-Delete:** Antes de eliminar un dispositivo, evalúa si existen sensores, instalaciones o mantenimientos asociados:

```
// Verificación de dependencias vinculadas al dispositivo
if ($cantSensores > 0 || $cantInstalaciones > 0 || $cantMantenimientos > 0) {
    // Soft-delete: deshabilita el dispositivo cambiando su estado sin borrar el registro
    $stmt = $pdo->prepare("UPDATE dispositivos SET estado = 'OFFLINE' WHERE
id_dispositivo = :id");
    $stmt->execute([':id' => $idDispositivo]);
} else {
    // Borrado físico al no existir elementos vinculados
    $stmt = $pdo->prepare("DELETE FROM dispositivos WHERE id_dispositivo = :id");
    $stmt->execute([':id' => $idDispositivo]);
}
```

**7. admin/tanques.php (Gestión de Tanques)**

- **Operaciones de Lectura:** Recupera la lista de tanques calculando el nivel de agua en tiempo real mediante una subconsulta a la tabla mediciones:

```
SELECT t.*, e.nombre AS edificio_nombre,
    (SELECT m.porcentaje FROM mediciones m
     INNER JOIN sensores s ON m.id_sensor = s.id_sensor
     INNER JOIN dispositivos d ON s.id_dispositivo = d.id_dispositivo
     WHERE d.id_tanque = t.id_tanque
     ORDER BY m.fecha_hora DESC LIMIT 1) AS nivel_actual
FROM tanques t
LEFT JOIN edificios e ON t.id_edificio = e.id_edificio
```

- **Mecanismo de Desactivación:** Si un tanque posee dispositivos vinculados, el sistema realiza un soft-delete marcando activo = 0.

**8. admin/sensores.php (Gestión de Sensores)**

- **Control de Integridad Referencial:** Permite el alta y modificación de sensores JSR-SR04T (modelo, número de serie, tolerancia, fecha de calibración). Bloquea la eliminación física de cualquier sensor que contenga lecturas registradas en la tabla mediciones (COUNT(*) > 0).

**9. admin/roles.php (Matriz de Permisos por Rol)**

- **Estado:** Maqueta de interfaz (UI). Presenta selectores para la asignación hipotética de permisos a perfiles. Al presionar el botón de guardado, la función de JavaScript despliega una alerta indicando *"Funcionalidad pendiente de integración con el backend"*.

**10. admin/tecnicos.php (Gestión de Personal Técnico)**

- **Operativa:** Filtra los registros de la tabla usuarios donde el rol asociado corresponde a TECNICO. Permite habilitar o deshabilitar cuentas de técnicos, bloqueando la desactivación si el profesional posee mantenimientos asignados en estado PENDIENTE o EN_PROCESO. Muestra el historial de las últimas 50 intervenciones del técnico seleccionado.

**11. admin/usuarios.php (CRUD Centralizado de Usuarios)**

- **Gestión de Claves y Mails:** Permite crear usuarios asignando roles (ADMIN, TECNICO, USUARIO). Implementa la acción generar_enviar que ejecuta una transacción SQL:
    1. Invoca eva_generar_password(12) para crear una clave aleatoria.
    2. Hash de la contraseña con password_hash() y actualización en usuarios.
    3. Intenta actualizar o insertar las credenciales en la tabla credenciales_clientes.
    4. Invoca eva_enviar_mail_credenciales() para notificar al usuario vía SMTP.
- **Seguridad:** El script incluye una validación que impide que el administrador autenticado se elimine o desactive a sí mismo ($id == $_SESSION['id_usuario']).

**12. admin/instalaciones.php (Registro de Instalaciones)**

- **Operativa:** Realiza una consulta con 5 combinaciones (JOIN) integrando las tablas instalaciones, dispositivos, tanques, edificios y usuarios (técnicos responsable). Soporta filtrado por coincidencias de texto y por rangos de fecha de ejecución.

**13. admin/inventario.php (Control de Insumos)**

- **Gestión de Movimientos (POST):** Procesa entradas y salidas de insumos en el stock. En movimientos de tipo SALIDA, verifica que la cantidad disponible sea suficiente antes de descontar las unidades. Registra la transacción en la tabla movimientos_inventario y refresca el contador general de stock.

**14. admin/compras.php (Órdenes de Compra y Alta Automática)**

- **Máquina de Estados:** Gestiona el ciclo de vida de las órdenes de compra a través de 5 estados: Pendiente $\rightarrow$ Aprobada $\rightarrow$ En entrega $\rightarrow$ Completada (o Cancelada).
- **Alta Automática de Cliente al Aprobar:** Al cambiar el estado de una orden a Aprobada asociándole un cliente, el sistema verifica si la persona posee un usuario activo. Si no existe, crea automáticamente el registro en usuarios (rol USUARIO), vincula la ficha en clientes, genera sus credenciales temporales y envía el correo de bienvenida.

**15. admin/clientes.php (Alta de Clientes Finales)**

- **Procesamiento Transaccional:** El formulario de alta ejecuta una transacción unificada:

```
$pdo->beginTransaction();
try {
    // 1. Inserción de cuenta de usuario global
    $passTemp = eva_generar_password(12);
    $passHash = password_hash($passTemp, PASSWORD_DEFAULT);

    $stmtU = $pdo->prepare("INSERT INTO usuarios (nombre, apellido, email,
password_hash, id_rol, activo)
```

```
                    VALUES (:n, :a, :e, :h, 3, 1)");
    $stmtU->execute([...]);
    $idUsuario = $pdo->lastInsertId();
    // 2. Inserción de la ficha de cliente vinculada
    $stmtC = $pdo->prepare("INSERT INTO clientes (id_usuario, razon_social, cuit,
direccion, telefono, pais, credenciales_generadas)
                    VALUES (:u, :rs, :cuit, :dir, :tel, 'Argentina', 1)");
    $stmtC->execute([...]);
    // 3. Registro en la tabla de credenciales de cliente
    $stmtCred = $pdo->prepare("INSERT INTO credenciales_clientes (id_cliente,
usuario_acceso, estado)
                    VALUES (:cli, :e, 'ACTIVA')");
    $stmtCred->execute([...]);
    $pdo->commit();
    // 4. Envío de correo electrónico con datos de acceso
    eva_enviar_credenciales($email, $nombre, $email, $passTemp);
} catch (Exception $e) {
    $pdo->rollBack();
    // Registro del error en el log de sistema
}
```

**16. admin/edificios.php (CRUD de Inmuebles)**

- **Arquitectura de Conexión Mixta:** Utiliza la instancia $conn (mysqli) para procesar las altas y ediciones mediante consultas preparadas con bind_param(). Por otro lado, emplea $pdo (PDO) para ejecutar las comprobaciones de borrado y los listados

avanzados. Exige asociar cada edificio a un usuario registrado que actúa como responsable operativo del inmueble.

**17. admin/auditorias.php (Visor de Auditoría)**

- **Consulta de Traza:** Consulta los últimos 100 eventos almacenados en la tabla log_actividad mediante un LEFT JOIN con las tablas usuarios y roles. Soporta filtrado por rango de fechas, identificador de acción y búsqueda de texto sobre el campo descriptivo.

**18. admin/api/estado.php (Endpoint Polling del Dashboard)**

- **Respuesta JSON:** Retorna un objeto con la consolidación en tiempo real de los contadores globales del sistema y las alertas activas. Este endpoint es consumido cada 3 segundos por el script de frontend admin/js/tiempo-real.js.

**19. admin/js/auditorias.php (Archivo Inejecutable - MUERTA)**

- **Análisis del Fallo:** Este archivo es el resultado de una ubicación errónea dentro de la estructura de directorios (admin/js/auditorias.php). Intenta actuar como una vista de auditoría paginada, pero contiene un error de sintaxis en la línea 13 ($_GET['fecha_hasta'] ?? "); con paréntesis de cierre mal posicionado) y sus sentencias require fallan al buscar carpetas relativas inexistentes dentro del subdirectorio js/.

## 6. DESGLOSE DEL MÓDULO tecnico/

### 6.1 Propósito y Estado Operativo

El módulo tecnico/ está diseñado para el personal de mantenimiento e instalación en campo. Permite revisar el estado del hardware, evaluar las lecturas de los sensores, gestionar órdenes de trabajo y monitorear las alertas de los dispositivos asignados.

**Análisis de Madurez Técnica:** El módulo presenta un nivel de desarrollo mixto. Varias de sus vistas operan contra la base de datos real, mientras que un grupo considerable de scripts corresponde a maquetas visuales (UI). Adicionalmente, la totalidad de los endpoints contenidos en el subdirectorio tecnico/api/ (a excepción de estado.php) se encuentran inoperativos (**MUERTA**) debido a llamadas a archivos de configuración inexistentes.

Guard Estándar de Vistas HTML (tecnico/*.php)

```
<?php
session_start();
if (!isset($_SESSION['rol']) || $_SESSION['rol'] !== 'TECNICO') {
    header("Location: ../index.php");
    exit;
}
```

### 6.2 Mapa de Archivos del Módulo tecnico/

| Archivo | Propósito Operativo | Marca de Estado |
|---|---|---|
| indextec.php | Dashboard técnico con indicadores de campo y estado de sensores. | **REAL** |
| alertas.php | Gestión de alertas asociadas a las instalaciones del técnico. | **REAL** |
| historialtecnico.php | Muestra un registro estático con 6 filas de intervenciones simuladas. | **UI** |
| instalaciones.php | Fichas visuales con datos fijos de instalaciones en progreso y pendientes. | **UI** |
| instalar.php | Formulario visual de alta de nueva instalación (sin atributos name/action). | **UI** |

| | | |
|---|---|---|
| instprogramar.php | Formulario visual para la programación de instalaciones futuras. | **UI** |
| mantenimientos.php | Fichas visuales de mantenimientos (incluye el tipo no válido Emergencia). | **UI** |
| mantagregar.php | Formulario visual de adición de sensores a un dispositivo. | **UI** |
| manteliminar.php | Formulario visual de desincorporación de sensores. | **UI** |
| mediciones.php | Explorador de las últimas 50 mediciones de dispositivos asignados. | **REAL** |
| misdispositivos.php | Tarjetas informativas de dispositivos vinculados al técnico. | **REAL** |
| mitanques.php | Maqueta visual con 4 tarjetas de tanques y sus modales de lectura. | **UI** |

| | | |
|---|---|---|
| notificaciones.php | Centro estático de notificaciones con contador fijo. | **UI** |
| perfil.php | Ficha de perfil estática asignada a *"Carlos Técnico"*. | **UI** |
| sensores.php | Catálogo de sensores y formulario visual de calibración. | **UI** |
| api/estado.php | Única API funcional del módulo técnico utilizada para el polling de interfaz. | **REAL** |
| api/dashboard.php | Invocación de dependencias inexistentes (config/db_pdo.php). | **MUERTA** |
| api/tanques.php | Invocación de dependencias inexistentes (config/db_pdo.php). | **MUERTA** |
| api/sensores.php | Invocación de dependencias inexistentes (config/db_pdo.php). | **MUERTA** |

| | | |
|---|---|---|
| api/perfil.php | Invocación de dependencias inexistentes (config/db_pdo.php). | **MUERTA** |
| api/notificaciones.php | Invocación de dependencias inexistentes (config/db_pdo.php). | **MUERTA** |
| api/mediciones.php | Invocación de dependencias inexistentes (config/db_pdo.php). | **MUERTA** |
| api/mantenimientos.php | Invocación de dependencias inexistentes (config/db_pdo.php). | **MUERTA** |
| api/instalaciones.php | Invocación de dependencias inexistentes (config/db_pdo.php). | **MUERTA** |
| api/dispositivos.php | Invocación de dependencias inexistentes (config/db_pdo.php). | **MUERTA** |
| api/alertas.php | Invocación de dependencias inexistentes (config/db_pdo.php). | **MUERTA** |

### 6.3 Detalle de Vistas Funcionales (REAL)

#### 1. tecnico/indextec.php (Dashboard Técnico)

Obtiene los datos del técnico autenticado utilizando la variable de sesión $_SESSION['id_usuario']. Consulta la cantidad de sensores bajo su responsabilidad mediante una combinación con la tabla instalaciones:

```
SELECT COUNT(*) FROM sensores s
INNER JOIN dispositivos d ON s.id_dispositivo = d.id_dispositivo
INNER JOIN instalaciones i ON i.id_dispositivo = d.id_dispositivo
WHERE i.id_tecnico = :idTecnico
```

Agrupa los sensores por su estado (ACTIVO, INACTIVO, FALLA) para renderizar un gráfico tipo donut y recupera los próximos 5 mantenimientos asignados.

#### 2. tecnico/alertas.php (Alertas del Técnico)

Consulta las alertas generadas exclusivamente en los tanques e instalaciones asociadas al ID del técnico autenticado. Clasifica las alertas por nivel de severidad (Crítica, Alta, Media, Baja) evaluando la columna tipo. Los modales de interacción para el cierre o atención de alertas ejecutan funciones de frontend definidas en tecnico.js.

#### 3. tecnico/mediciones.php (Consulta de Telemetría)

Recupera las últimas 50 lecturas almacenadas en la tabla mediciones filtrando por los dispositivos asignados al técnico mediante la tabla puente tecnico_dispositivo. Admite un parámetro GET['id_dispositivo'] para filtrar las mediciones de un nodo en particular.

#### 4. tecnico/misdispositivos.php (Nodos Asignados)

Genera tarjetas informativas para cada uno de los dispositivos instalados por el técnico. Muestra la dirección MAC, versión de firmware, nivel de señal RSSI, voltaje de batería y la fecha de última conexión registrada.

**5. tecnico/api/estado.php (API Polling Funcional)**

Es la única API operativa del módulo técnico. Incluye config/db.php y responde a las peticiones del script de tiempo real (tecnico/js/tiempo-real.js) retornando un objeto JSON con el estado actualizado de los equipos, alertas y mantenimientos del técnico.

**6.4 Diagnóstico de Endpoints Inoperativos (tecnico/api/*.php)**

Los 10 scripts de API restantes en la carpeta tecnico/api/ (dashboard.php, tanques.php, sensores.php, perfil.php, notificaciones.php, mediciones.php, mantenimientos.php, instalaciones.php, dispositivos.php y alertas.php) fallan al ser invocados debido a la inclusión de librerías inexistentes en sus líneas iniciales:

```
// Estructura de cabecera fallida presente en los 10 archivos de la carpeta técnico/api/

require_once __DIR__ . '/../../config/db_pdo.php';  // ERROR FATAL: El archivo db_pdo.php NO existe

require_once __DIR__ . '/../../config/helpers.php'; // ERROR FATAL: El archivo helpers.php NO existe

$idTecnico = requireTecnico(); // ERROR FATAL: Función no definida

$pdo = getDB();              // ERROR FATAL: Función no definida
```

Para rehabilitar estos endpoints en futuras revisiones del sistema, se debe reemplazar dicha cabecera por la inclusión canónica de config/db.php, instanciando la conexión mediante $pdo = eva_pdo() y aplicando la validación de rol correspondiente.

### 6.5 Inconsistencia entre Tablas de Relación (instalaciones vs tecnico_dispositivo)

Durante la auditoría del módulo técnico se identificó un conflicto en el modelo de asignación de dispositivos:

- **Modelo Basado en instalaciones:** Los scripts indextec.php, alertas.php, misdispositivos.php y la API funcional api/estado.php determinan qué dispositivos pertenecen a un técnico consultando la columna id_tecnico de la tabla instalaciones.
- **Modelo Basado en tecnico_dispositivo:** El script mediciones.php y las 10 APIs inoperativas determinan la propiedad consultando la tabla puente tecnico_dispositivo.

**Consecuencia en Producción:** Si un técnico tiene asignada una instalación en la tabla instalaciones, pero no se han insertado registros en la tabla puente tecnico_dispositivo, podrá visualizar el dispositivo en su dashboard principal pero la vista mediciones.php aparecerá vacía informando *"Sin mediciones asignadas a este técnico"*.

## 7. DESGLOSE DEL MÓDULO cliente/

### 7.1 Propósito y Arquitectura de Negocio

El módulo cliente/ representa el panel de acceso para los usuarios finales y administradores de consorcios (USUARIO). Permite consultar el nivel del agua en tiempo real, visualizar la autonomía estimada en litros, inspeccionar gráficos históricos de consumo, ajustar los umbrales de alerta y enviar solicitudes de mantenimiento adjuntando evidencia fotográfica.

El módulo cliente destaca por ser el componente mejor estructurado del repositorio: el 100% de sus vistas y endpoints de API son operativos (**REAL**) y fundamentan su lógica en dos bibliotecas centrales: cliente/includes/helpers.php y cliente/includes/procesador_mediciones.php.

Guard Estándar para Vistas HTML (cliente/*.php)

```
<?php
session_start();
if (!isset($_SESSION['rol']) || $_SESSION['rol'] !== 'USUARIO') {
    header("Location: ../index.php");
    exit;
}
require_once __DIR__ . '/../config/db.php';
require_once __DIR__ . '/includes/helpers.php';
```

**7.2 Mapa de Archivos del Módulo cliente/**

| Archivo | Propósito Operativo | Tipo de Operación | Estado |
|---|---|---|---|

| | | | |
|---|---|---|---|
| indexcli.php | Dashboard principal: gauge interactivo, consumo diario, autonomía y tendencias. | Consulta y Procesamiento | **REAL** |
| mitanque.php | Vista de tanque animado mediante SVG dinámico y lecturas de los últimos 7 días. | Consulta y Procesamiento | **REAL** |
| alertas.php | Histórico y filtrado en tiempo real de alertas registradas en los tanques del cliente. | Consulta de Datos | **REAL** |
| historial.php | Análisis estadístico de consumos, agregaciones por | Consulta de Datos | **REAL** |

| | | | |
|---|---|---|---|
| | periodos y exportación. | | |
| configuracion.php | Interfaz de ajuste de umbrales de alerta (nivel mínimo y máximo en porcentaje). | Escritura (POST) | **REAL** |
| mantenimiento.php | Solicitudes de servicio técnico, carga de imágenes y notificaciones automáticas. | Escritura y Uploads | **REAL** |
| perfil.php | Consulta de la cuenta de usuario y actualización de la sesión activa. | Consulta de Datos | **REAL** |

| | | | |
|---|---|---|---|
| includes/helpers.php | Biblioteca central compuesta por 20 funciones transversales de soporte. | Biblioteca Lógica | **REAL** |
| includes/procesador_mediciones.php | Motor del sistema: cálculo de nivel, consumos diarios y alertas. | Motor del Sistema | **REAL** |
| api/tanque.php | Retorna los parámetros de estado del tanque principal en formato JSON. | Endpoint JSON | **REAL** |
| api/resumen.php | Endpoint de provisión de métricas consolidadas para el dashboard. | Endpoint JSON | **REAL** |

| | | | |
|---|---|---|---|
| api/estado.php | Endpoint de polling para la actualización continua de la interfaz de usuario. | Endpoint JSON | **REAL** |
| api/historial.php | Provee series de datos agregadas para la generación de gráficos. | Endpoint JSON | **REAL** |
| api/alertas.php | Endpoint de filtrado de alertas con procesamiento en el servidor. | Endpoint JSON | **REAL** |
| api/configuracion.php | Procesa la lectura y actualización de umbrales vía peticiones JSON. | Endpoint JSON | **REAL** |

| | | | |
|---|---|---|---|
| api/procesar.php | Disparador manual que ejecuta la rutina eva_procesar_todos() . | Endpoint Lógico | **REAL** |
| api/sincronizar.php | Ejecuta la consolidación de consumos y alertas sin requerir nueva telemetría. | Endpoint Lógico | **REAL** |

**7.3 Desglose de Vistas y Flujos del Módulo cliente/**

**1. cliente/indexcli.php (Dashboard Resumen)**

1. Recupera el identificador del usuario autenticado mediante eva_current_user_id().
2. Obtiene el primer tanque asociado mediante eva_first_tanque($pdo, $uid).
3. Calcula la capacidad efectiva del tanque invocando eva_tanque_capacidad_efectiva($tanque).
4. Ejecuta la rutina de actualización eva_procesar_tanque($pdo, $idTanque). Si se produce una excepción, ejecuta el mecanismo de contingencia invocando eva_sincronizar_consumos() y eva_sincronizar_alertas().

5. Obtiene la última lectura procesada (eva_latest_medicion()), el porcentaje de nivel (eva_calcular_pct()), el volumen en litros (eva_calcular_litros()) y el consumo acumulado del día (eva_consumo_hoy()).
6. Exporta el objeto de contexto hacia el entorno de JavaScript asignándolo a la variable global de navegador window.EVA_RESUMEN.

**2. cliente/mitanque.php (Vista del Tanque)**

Obtiene el detalle del tanque y genera la información necesaria para renderizar el gráfico SVG dinámico. Calcula la altura del nivel del agua y genera las ondas de animación mediante la etiqueta de recorte <clipPath>. Recupera la serie de las últimas 7 mediciones para dibujar el gráfico de barras inferior. Exporta los datos a JavaScript mediante la variable window.EVA_TANQUE.

**3. cliente/alertas.php (Notificaciones del Cliente)**

Recupera las últimas 100 alertas vinculadas a los tanques del cliente. Admite el parámetro GET['filter'] (todas, activas, resueltas) para filtrar la lista. Exporta los datos al entorno de frontend mediante window.EVA_ALERTAS, permitiendo que el script cliente/js/script.js realice el renderizado dinámico de la interfaz.

**4. cliente/historial.php (Análisis Histórico)**

Procesa la agregación de datos de telemetría filtrados por rangos de fecha (GET['desde'] y GET['hasta']) y periodos de consolidación (semana, mes, trimestre). Si la base de datos no contiene mediciones dentro del rango solicitado, el script activa un mecanismo de respaldo (*fallback*) que genera una serie de datos demostrativa para garantizar que la interfaz mantenga su renderizado gráfico sin romperse.

**5. cliente/configuracion.php (Umbrales de Alerta)**

Permite al usuario ajustar los niveles porcentuales mínimo y máximo que desencadenan las alertas automáticas del sistema.

**Mecanismo de Seguridad Anti-CSRF:** El formulario incluye un token de seguridad generado por la función eva_csrf_token() e insertado en un campo oculto (<input type="hidden" name="csrf_token">). Al recibir la petición POST, el script valida el token mediante eva_csrf_validate($_POST['csrf_token']). Si la validación falla, interrumpe el procesamiento rechazando la solicitud.

Al guardar los cambios, actualiza los valores en la tabla configuracion_alertas y registra la modificación en el log de auditoría mediante eva_log_actividad().

**6. cliente/mantenimiento.php (Solicitudes de Mantenimiento con Subida de Archivos)**

Permite al cliente enviar una solicitud de servicio técnico describiendo una falla y adjuntando opcionalmente una fotografía.

**Rutina Automatizada de Auto-DDL:** Al cargarse, el script verifica la existencia de la tabla solicitudes_mantenimiento. Si no existe, ejecuta automáticamente una sentencia DDL para crearla, asegurando que la infraestructura de base de datos se adapte dinámicamente sin requerir intervención manual. Crea además el directorio cliente/uploads/mantenimientos/ e inserta un archivo .htaccess configurado con Require all denied para prevenir la ejecución remota de scripts PHP que pudieran ser subidos maliciosamente.

**Procesamiento de Subida de Archivos (Uploads):**

1. Comprueba que el archivo recibido vía $_FILES['mt_imagen'] no supere el tamaño máximo permitido de 5 MB.
2. Valida la extensión y el tipo MIME utilizando la extensión finfo de PHP, admitiendo únicamente las extensiones jpeg, jpg, png y webp.
3. Renombra el archivo utilizando una sintaxis aleatoria segura: sm_AAAAMMDD_HHMMSS_rand6.ext.
4. Mueve el archivo procesado al directorio de destino asignándole permisos de lectura restringidos (0644).
5. Inicia una transacción SQL que inserta la solicitud en solicitudes_mantenimiento e inserta una notificación en la tabla notificaciones dirigida a todos los usuarios con rol ADMIN y TECNICO.

### 7.4 Biblioteca Central (cliente/includes/helpers.php)

Este archivo constituye la biblioteca de utilidades del sistema. Contiene 20 funciones:

1. h(?string $s): string: Wrapper de la función nativa htmlspecialchars($s, ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8').
2. eva_current_user_id(): ?int: Retorna el ID del usuario autenticado almacenado en $_SESSION['id_usuario'].
3. eva_csrf_token(): string: Genera y almacena un token criptográfico seguro de 64 caracteres en la sesión del usuario.
4. eva_csrf_validate(?string $token): bool: Comprueba la validez del token recibido mediante la función hash_equals().

5. eva_tanques_for_user(PDO $pdo, ?int $uid): array: Retorna el listado de tanques activos.
6. eva_first_tanque(PDO $pdo, ?int $uid): ?array: Retorna el primer tanque disponible del arreglo devuelto por eva_tanques_for_user().
7. eva_latest_medicion(PDO $pdo, int $idTanque): ?array: Obtiene el último registro de telemetría almacenado para un tanque.
8. eva_mediciones_historial(PDO $pdo, int $idTanque, int $limit = 50): array: Recupera el historial de mediciones de un tanque ordenadas de forma descendente por fecha.
9. eva_consumo_hoy(PDO $pdo, int $idTanque): float: Retorna el acumulado en litros consumidos en el día actual desde la tabla consumos.
10. eva_consumo_promedio(PDO $pdo, int $idTanque): float: Calcula el promedio histórico de consumo diario del tanque.
11. eva_consumo_serie(PDO $pdo, int $idTanque, string $period): array: Genera una serie temporal de consumo adaptada al periodo solicitado (semana, mes, anio).
12. eva_device_status(PDO $pdo, int $idTanque): string: Determina el estado de conexión del dispositivo evaluando el campo ultima_conexion (retorna Conectado si la última lectura fue recibida hace menos de 600 segundos).
13. eva_tanque_capacidad_efectiva(array $tanque): float: Calcula la capacidad volumétrica real del tanque seleccionando la primera métrica válida disponible entre capacidad_litros, volumen_util o el cálculo geométrico derivado de sus dimensiones.
14. eva_tanque_volumen_geometrico(array $tanque): ?float: Calcula el volumen teórico aplicando la fórmula geométrica del cilindro: $V = \pi \times (diámetro / 2)^2 \times (altura / 1000)$.

15. eva_tanque_advertencia(array $tanque): ?string: Evalúa discrepancias entre la capacidad declarada y el volumen geométrico calculado, retornando mensajes de advertencia si la diferencia supera el 25%.
16. eva_calcular_pct(array $tanque, array $medicion): int: Calcula el porcentaje de agua disponible en el tanque priorizando la relación de litros reportados sobre la capacidad efectiva, o en su defecto, la distancia leída por el sensor sobre la altura total.
17. eva_calcular_litros(array $tanque, array $medicion, ?int $pct = null): float: Determina el volumen disponible en litros basándose en la capacidad efectiva y el porcentaje de llenado actual.
18. eva_estado_texto(int $pct): array: Asigna la categoría textual y la clase visual del tanque según el porcentaje de agua ($\ge 100\%$ Completo, $\ge 80\%$ Alto, $\ge 40\%$ Normal, $\ge 20\%$ Bajo, $< 20\%$ Crítico).
19. eva_alert_tipo_map(string $tipo): array: Asigna etiquetas legibles y estilos visuales a los tipos de alerta del sistema.
20. eva_log_actividad(PDO $pdo, int $idUsuario, string $accion, string $detalle): void: Registra un nuevo evento en la tabla de auditoría log_actividad.

### 7.5 Motor de Procesamiento (cliente/includes/procesador_mediciones.php)

Este archivo contiene la lógica para la consolidación de métricas de telemetría, actualización de consumos y generación de alertas automáticas. Expone 7 funciones fundamentales:

1. eva_procesador_calcular_pct(array $medicion, array $tanque): int: Wrapper del cálculo de porcentaje de llenado.
2. eva_procesador_actualizar_consumos(PDO $pdo, int $idTanque): void: Evalúa las lecturas registradas en la tabla mediciones a lo largo del día actual, calcula la diferencia entre el volumen inicial y el final, y ejecuta una sentencia UPSERT sobre la tabla consumos:

```
INSERT INTO consumos (id_tanque, fecha, consumo_litros)

VALUES (:id, CURRENT_DATE(), :litros)

ON DUPLICATE KEY UPDATE consumo_litros = :litros
```

3. eva_procesador_actualizar_estado_dispositivo(PDO $pdo, int $idTanque): void: Evalúa la fecha de última conexión registrada en los dispositivos asociados al tanque. Si la diferencia temporal respecto a la hora actual supera los 10 minutos, actualiza el estado del dispositivo a OFFLINE.
4. eva_procesador_detectar_alertas(PDO $pdo, int $idTanque): void: Compara la última lectura de nivel contra los umbrales configurados en configuracion_alertas. Si el nivel desciende por debajo de nivel_minimo, o supera nivel_maximo, comprueba si existe una alerta similar generada en las últimas 24 horas. Si no existe una alerta previa registrada en el día, inserta un nuevo registro en la tabla alertas con estado PENDIENTE.
5. eva_procesador_tanque(PDO $pdo, int $idTanque): void: Función orquestadora central. Ejecuta secuencialmente la actualización de consumos, el control de estado de dispositivos y la detección de alertas para un tanque específico.
6. eva_procesador_todos(PDO $pdo): void: Recorre la totalidad de los tanques activos en la base de datos e invoca eva_procesador_tanque() para cada uno de ellos.

7. eva_procesar_tanque(PDO $pdo, int $idTanque): void: Alias de compatibilidad que redirige la ejecución a eva_procesador_tanque().

## 8. CAPA DE INGESTA IoT DE TELEMETRÍA (api/)

### 8.1 Propósito y Canales de Ingesta

La capa de ingesta IoT expone los endpoints públicos encargados de recibir la telemetría transmitida por los nodos hardware **ESP32**. Acepta lecturas enviadas en formato JSON o mediante codificación de formulario (application/x-www-form-urlencoded), normaliza los valores físicos recibidos, persiste la medición en la base de datos y dispara el motor de procesamiento para actualizar el estado del sistema en tiempo real.

| **Endpoint** | **Formato de Entrada** | **Método HTTP** | **Comportamiento** |
|---|---|---|---|
| api/esp32/mediciones.php | JSON Estricto | POST | Endpoint canónico del firmware v2.1.0. Valida campos obligatorios y retorna respuestas estructuradas con códigos HTTP estándares. |

| | | | |
|---|---|---|---|
| api/guardar_datos.php | Form / JSON Tolerante | Cualquier Método | Endpoint wrapper de alta tolerancia. Acepta variables en minúsculas/mayúsculas y responde con formato simple. |

**8.2 Endpoint Canónico (api/esp32/mediciones.php)**

Estructura del Payload Esperado (JSON)

```
{
  "id_sensor": 1,
  "distancia_cm": 45.2,
  "nivel_cm": 154.8,
  "porcentaje": 77.4,
  "litros": 3870.0,
  "temperatura": 22.5,
  "humedad": 60.0
}
```

**Flujo de Procesamiento y Validación**

1. **Lectura del Cuerpo de la Petición:** Obtiene la cadena bruta enviada por el microcontrolador mediante file_get_contents('php://input') y la decodifica con

json_decode(). Si la decodificación falla, intenta recuperar los valores desde el arreglo global $_POST.

2. **Validación de Parámetros:** Verifica que se haya proporcionado un identificador de sensor válido (id_sensor > 0) y que al menos una métrica de medición física (distancia_cm, nivel_cm o porcentaje) esté presente. Si la validación falla, interrumpe el procesamiento retornando un código HTTP 400 Bad Request:

   {"error": "Datos incompletos o invalidos"}

3. **Verificación del Sensor y Tanque:** Consulta la tabla sensores para verificar la existencia del dispositivo y obtener el ID del tanque asociado. Si el sensor no se encuentra registrado, aplica un mecanismo de tolerancia creando una entidad conceptual vinculada al primer tanque disponible en el sistema.
4. **Normalización de Métricas:** Invoca a eva_tanque_capacidad_efectiva() para obtener la capacidad del tanque. Si el microcontrolador no envió los valores calculados de porcentaje o litros, los calcula en el servidor basándose en la distancia leída por el sensor ultrasónico.
5. **Persistencia de la Medición (INSERT):** Inserta el registro en la tabla mediciones. Si la tabla de producción no dispone de las columnas temperatura y humedad, captura la excepción y ejecuta un INSERT secundario omitiendo dichos campos para garantizar la persistencia de la telemetría principal:

```
try {
    $stmt = $pdo->prepare("INSERT INTO mediciones (id_sensor, distancia_cm,
nivel_cm, porcentaje, litros, temperatura, humedad, fecha_hora)
                VALUES (:s, :d, :n, :p, :l, :t, :h, NOW())");
```

```
    $stmt->execute([...]);
} catch (PDOException $e) {
    // Fallback para esquemas de base de datos legados sin columnas de
temperatura y humedad
    $stmt = $pdo->prepare("INSERT INTO mediciones (id_sensor, distancia_cm,
nivel_cm, porcentaje, litros, fecha_hora)
                           VALUES (:s, :d, :n, :p, :l, NOW())");
    $stmt->execute([...]);
}
```

6. **Actualización de Conectividad:** Actualiza el campo ultima_conexion = NOW() en la tabla dispositivos para registrar que el nodo está operativo.
7. **Invocación del Motor:** Ejecuta la función eva_procesar_tanque($pdo, $idTanque) para actualizar consumos diarios y evaluar alertas.
8. **Respuesta Exitos:** Retorna un código HTTP 200 OK:

```
{
  "ok": true,
  "id_medicion": 1042
}
```

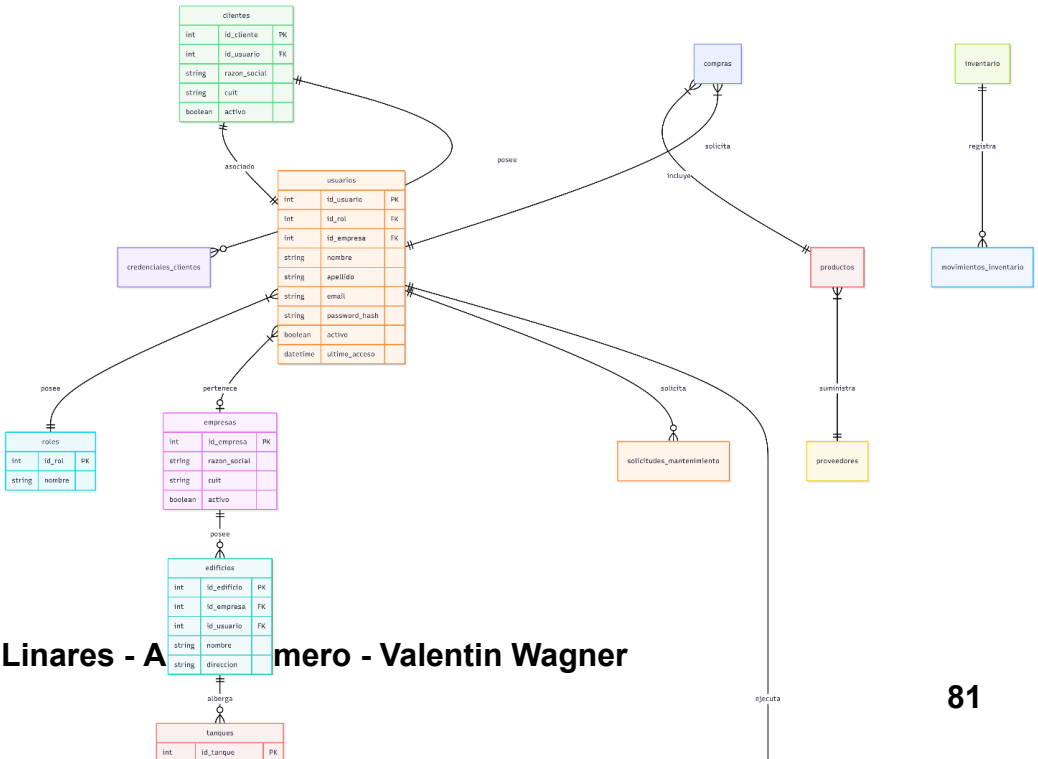
### 8.3 Endpoint Wrapper de Tolerancia (api/guardar_datos.php)

Este script actúa como una interfaz de ingesta tolerante diseñada para mantener compatibilidad con versiones antiguas del firmware o pruebas de integración manuales.

**Características Operativas**

- **Aceptación de Métodos HTTP:** Responde a peticiones POST y GET.
- **Mapeo Flexible de Claves:** Admite variaciones en los nombres de los parámetros recibidos (sensor_id o id_sensor, distancia o distancia_cm, temp o temperatura).
- **Formato de Respuesta:**

```
{
  "status": "success",
  "mensaje": "Datos guardados correctamente",
  "id": 1043
}
```



## 9. BASE DE DATOS Y MODELOS RELACIONALES

### 9.1 Diagrama Entidad-Relación

### 9.2 Diccionario Completo de las 21 Tablas del Sistema

#### 1. Tabla roles

Almacena los perfiles de acceso que determinan los permisos dentro del sistema.

- id_rol (INT, PK, AUTO_INCREMENT): Identificador único del rol.
- nombre (VARCHAR(50), NOT NULL, UNIQUE): Nombre del rol (ADMIN, TECNICO, USUARIO).
- descripcion (VARCHAR(255), NULL): Descripción del alcance del perfil.

#### 2. Tabla usuarios

Entidad central de autenticación y cuentas de usuario.

- id_usuario (INT, PK, AUTO_INCREMENT): Identificador único del usuario.
- id_rol (INT, FK $\rightarrow$ roles.id_rol): Rol asignado al usuario.
- id_empresa (INT, NULL, FK $\rightarrow$ empresas.id_empresa): Empresa a la que pertenece.
- nombre (VARCHAR(100), NOT NULL): Nombre de pila.
- apellido (VARCHAR(100), NOT NULL): Apellido.
- email (VARCHAR(150), NOT NULL, UNIQUE): Correo electrónico (utilizado como login).
- password_hash (VARCHAR(255), NOT NULL): Hash Bcrypt de la contraseña.
- telefono (VARCHAR(50), NULL): Número de contacto telefónico.

- celular (VARCHAR(50), NULL): Número de teléfono móvil.
- direccion (VARCHAR(255), NULL): Domicilio particular o comercial.
- activo (TINYINT(1), DEFAULT 1): Estado de la cuenta (1 = Activa, 0 = Inactiva).
- fecha_registro (DATETIME, DEFAULT CURRENT_TIMESTAMP): Fecha de creación del usuario.
- ultimo_acceso (DATETIME, NULL): Fecha y hora de la última autenticación exitosa.

**3. Tabla empresas**

Registra las organizaciones o consorcios clientes del servicio.

- id_empresa (INT, PK, AUTO_INCREMENT): Identificador único de la empresa.
- razon_social (VARCHAR(150), NOT NULL): Nombre legal de la empresa.
- cuit (VARCHAR(20), NOT NULL, UNIQUE): Clave Única de Identificación Tributaria.
- direccion (VARCHAR(255), NULL): Domicilio fiscal.
- telefono (VARCHAR(50), NULL): Teléfono de contacto.
- email (VARCHAR(150), NULL): Correo de contacto administrativo.
- activo (TINYINT(1), DEFAULT 1): Estado operativo de la empresa.
- fecha_alta (DATETIME, DEFAULT CURRENT_TIMESTAMP): Fecha de registro en el sistema.

**4. Tabla clientes**

Fichas comerciales de clientes vinculadas a las cuentas de usuario.

- id_cliente (INT, PK, AUTO_INCREMENT): Identificador del cliente.
- id_usuario (INT, FK $\rightarrow$ usuarios.id_usuario): Cuenta de usuario asociada.
- razon_social (VARCHAR(150), NOT NULL): Denominación comercial.
- cuit (VARCHAR(20), NOT NULL): Identificación tributaria.

- direccion (VARCHAR(255), NULL): Dirección fiscal.
- telefono (VARCHAR(50), NULL): Teléfono.
- pais (VARCHAR(50), DEFAULT 'Argentina'): País de residencia.
- credenciales_generadas (TINYINT(1), DEFAULT 0): Indicador de envío de accesos.
- fecha_credenciales (DATETIME, NULL): Fecha de emisión de claves.
- activo (TINYINT(1), DEFAULT 1): Estado comercial del cliente.
- fecha_alta (DATETIME, DEFAULT CURRENT_TIMESTAMP): Fecha de registro.

**5. Tabla credenciales_clientes**

Control de estado de las accesos generados para los clientes.

- id_credencial (INT, PK, AUTO_INCREMENT): Identificador único.
- id_cliente (INT, FK $\rightarrow$ clientes.id_cliente): Cliente al que pertenece la credencial.
- usuario_acceso (VARCHAR(150), NOT NULL): Nombre de usuario o correo asignado.
- estado (ENUM('ACTIVA', 'INACTIVA', 'PENDIENTE'), DEFAULT 'ACTIVA'): Estado del acceso.
- fecha_creacion (DATETIME, DEFAULT CURRENT_TIMESTAMP): Fecha de emisión.

**6. Tabla edificios**

Inmuebles donde se encuentran instalados los tanques de agua.

- id_edificio (INT, PK, AUTO_INCREMENT): Identificador del inmueble.
- id_empresa (INT, NULL, FK $\rightarrow$ empresas.id_empresa): Empresa propietaria.
- id_usuario (INT, NULL, FK $\rightarrow$ usuarios.id_usuario): Responsable asignado.
- nombre (VARCHAR(150), NOT NULL): Denominación del edificio (ej: *"Consorcio Av. de Mayo"*).

- direccion (VARCHAR(255), NOT NULL): Ubicación física.
- ciudad (VARCHAR(100), NULL): Localidad o ciudad.
- provincia (VARCHAR(100), NULL): Provincia o estado.
- pais (VARCHAR(50), DEFAULT 'Argentina'): País.
- codigo_postal (VARCHAR(20), NULL): Código postal.
- telefono_contacto (VARCHAR(50), NULL): Teléfono de la administración del inmueble.
- responsable_nombre (VARCHAR(150), NULL): Persona de contacto en el edificio.
- fecha_registro (DATETIME, DEFAULT CURRENT_TIMESTAMP): Fecha de creación en el sistema.

**7. Tabla tanques**

Representación física de los depósitos de agua monitoreados.

- id_tanque (INT, PK, AUTO_INCREMENT): Identificador único del tanque.
- id_edificio (INT, FK $\rightarrow$ edificios.id_edificio): Edificio donde se ubica.
- nombre (VARCHAR(100), NOT NULL): Nombre identificador (ej: *"Tanque Elevado Norte"*).
- capacidad_litros (FLOAT, NOT NULL): Capacidad máxima nominal en litros.
- altura_cm (FLOAT, NOT NULL): Altura total del tanque desde la base hasta la tapa (cm).
- diametro_cm (FLOAT, NULL): Diámetro para tanques cilíndricos (cm).
- volumen_util (FLOAT, NULL): Volumen operativo útil estimado (litros).
- ubicacion (VARCHAR(150), NULL): Ubicación física (ej: *"Azotea Bloque B"*).
- tipo_tanque (VARCHAR(50), NULL): Clasificación (ELEVADO, SUBTERRANEO, CISTERNA).
- material (VARCHAR(50), NULL): Material de construcción (HORMIGON, PLASTICO, ACERO).

- descripcion (TEXT, NULL): Observaciones técnicas del tanque.
- activo (TINYINT(1), DEFAULT 1): Estado operativo (1 = Activo, 0 = Inactivo/Soft-deleted).
- fecha_instalacion (DATE, NULL): Fecha de puesta en servicio.

**8. Tabla dispositivos**

Nodos de telemetría basados en ESP32 instalados en los tanques.

- id_dispositivo (INT, PK, AUTO_INCREMENT): Identificador del dispositivo.
- id_tanque (INT, NULL, FK $\rightarrow$ tanques.id_tanque): Tanque al que transmite datos.
- nombre (VARCHAR(100), NOT NULL): Código o nombre asignado (ej: *"EVA-NODE-01"*).
- mac_address (VARCHAR(17), NOT NULL, UNIQUE): Dirección física MAC de la interfaz Wi-Fi.
- ip_address (VARCHAR(45), NULL): Dirección IP asignada en la red local.
- estado (ENUM('ONLINE', 'OFFLINE', 'MANTENIMIENTO'), DEFAULT 'OFFLINE'): Estado actual.
- bateria_porcentaje (FLOAT, DEFAULT 100): Nivel de carga de la batería de respaldo (%).
- senal_rssi (INT, NULL): Intensidad de la señal Wi-Fi recibida en dBm.
- firmware_version (VARCHAR(20), DEFAULT 'v2.1.0'): Versión del firmware grabado.
- ultima_conexion (DATETIME, NULL): Marca temporal del último paquete de telemetría recibido.
- fecha_alta (DATETIME, DEFAULT CURRENT_TIMESTAMP): Fecha de registro en el sistema.

**9. Tabla sensores**

Sensores de distancia ultrasónicos JSR-SR04T acoplados a los nodos.

- id_sensor (INT, PK, AUTO_INCREMENT): Identificador único del sensor.
- id_dispositivo (INT, FK $\rightarrow$ dispositivos.id_dispositivo): Nodo ESP32 al que está conectado.
- modelo (VARCHAR(50), DEFAULT 'JSR-SR04T'): Modelo comercial del componente.
- numero_serie (VARCHAR(100), NOT NULL, UNIQUE): Código de serie de fabricación.
- estado (ENUM('ACTIVO', 'INACTIVO', 'FALLA'), DEFAULT 'ACTIVO'): Estado operativo.
- fabricante (VARCHAR(100), NULL): Empresa fabricante.
- precision_mm (FLOAT, NULL): Tolerancia o margen de error en milímetros.
- rango_min_cm (FLOAT, DEFAULT 20): Distancia mínima de detección efectiva.
- rango_max_cm (FLOAT, DEFAULT 600): Distancia máxima de detección.
- calibrado (TINYINT(1), DEFAULT 1): Flag de estado de calibración.
- fecha_calibracion (DATE, NULL): Fecha del último ajuste de cero.

**10. Tabla mediciones**

Tabla de alta frecuencia que almacena las lecturas de telemetría.

- id_medicion (BIGINT, PK, AUTO_INCREMENT): Identificador único del registro de lectura.
- id_sensor (INT, FK $\rightarrow$ sensores.id_sensor): Sensor que emitió la lectura.
- distancia_cm (FLOAT, NOT NULL): Distancia medida desde el sensor al agua (cm).
- nivel_cm (FLOAT, NOT NULL): Altura calculada de la columna de agua (cm).
- porcentaje (FLOAT, NOT NULL): Porcentaje de llenado del tanque (0 a 100%).
- litros (FLOAT, NOT NULL): Volumen de agua estimado disponible en litros.

- temperatura (FLOAT, NULL): Temperatura ambiental informada por el nodo (°C).
- humedad (FLOAT, NULL): Humedad relativa del aire informada por el nodo (%).
- fecha_hora (DATETIME, DEFAULT CURRENT_TIMESTAMP, INDEX): Marca de tiempo de la lectura.

**11. Tabla alertas**

Registra las anomalías operativas detectadas automáticamente o informadas por los nodos.

- id_alerta (INT, PK, AUTO_INCREMENT): Identificador único de la alerta.
- id_tanque (INT, FK $\rightarrow$ tanques.id_tanque): Tanque donde ocurrió el evento.
- tipo (ENUM('NIVEL_BAJO', 'NIVEL_ALTO', 'SIN_CONEXION', 'FALLA_SENSOR', 'CONSUMO_ANORMAL')): Categoría.
- estado (ENUM('PENDIENTE', 'ATENDIDA', 'CERRADA'), DEFAULT 'PENDIENTE'): Estado de atención.
- descripcion (TEXT, NOT NULL): Detalle descriptivo de la anomalía detectada.
- fecha_hora (DATETIME, DEFAULT CURRENT_TIMESTAMP): Instante de ocurrencia.

**12. Tabla consumos**

Agregación diaria de consumo de agua por tanque.

- id_consumo (INT, PK, AUTO_INCREMENT): Identificador del registro.
- id_tanque (INT, FK $\rightarrow$ tanques.id_tanque): Tanque evaluado.
- fecha (DATE, NOT NULL): Día correspondiente al consumo consolidado.
- consumo_litros (FLOAT, NOT NULL): Total de litros consumidos durante la jornada.
- *Índice Único:* UNIQUE KEY uk_tanque_fecha (id_tanque, fecha).

**13. Tabla configuracion_alertas**

Parámetros y umbrales de alerta personalizados por tanque.

- id_config (INT, PK, AUTO_INCREMENT): Identificador de la configuración.
- id_tanque (INT, FK $\rightarrow$ tanques.id_tanque): Tanque parametrizado.
- nivel_minimo (FLOAT, DEFAULT 20): Umbral porcentual para alerta de NIVEL_BAJO.
- nivel_maximo (FLOAT, DEFAULT 90): Umbral porcentual para alerta de NIVEL_ALTO.
- notificar_email (TINYINT(1), DEFAULT 1): Activa el envío de alertas por correo.
- notificar_sistema (TINYINT(1), DEFAULT 1): Activa notificaciones en la interfaz web.

**14. Tabla solicitudes_mantenimiento**

Solicitudes de asistencia técnica iniciadas por los clientes.

- id_solicitud (INT, PK, AUTO_INCREMENT): Identificador de la solicitud.
- id_tanque (INT, FK $\rightarrow$ tanques.id_tanque): Tanque afectado.
- id_usuario (INT, FK $\rightarrow$ usuarios.id_usuario): Usuario cliente emisor.
- id_tecnico (INT, NULL, FK $\rightarrow$ usuarios.id_usuario): Técnico asignado.
- descripcion (TEXT, NOT NULL): Detalle del problema reportado.
- imagen (VARCHAR(255), NULL): Ruta relativa a la fotografía adjunta.
- estado (ENUM('PENDIENTE', 'ACEPTADA', 'EN_PROCESO', 'FINALIZADA', 'CANCELADA'), DEFAULT 'PENDIENTE').
- fecha_solicitud (DATETIME, DEFAULT CURRENT_TIMESTAMP): Instante de emisión.
- fecha_actualizacion (DATETIME, NULL ON UPDATE CURRENT_TIMESTAMP): Última modificación.

**15. Tabla instalaciones**

Registro de trabajos de montaje e instalación de equipamiento en campo.

- id_instalacion (INT, PK, AUTO_INCREMENT): Identificador de la orden de instalación.
- id_dispositivo (INT, FK $\rightarrow$ dispositivos.id_dispositivo): Nodo instalado.
- id_tecnico (INT, FK $\rightarrow$ usuarios.id_usuario): Técnico responsable.
- fecha_instalacion (DATETIME, NOT NULL): Fecha de ejecución de la tarea.
- observaciones (TEXT, NULL): Notas técnicas de la instalación.
- latitud (DECIMAL(10,8), NULL): Coordenada geográfica de latitud.
- longitud (DECIMAL(11,8), NULL): Coordenada geográfica de longitud.

**16. Tabla mantenimientos**

Órdenes de servicio preventivo, correctivo o predictivo sobre la infraestructura.

- id_mantenimiento (INT, PK, AUTO_INCREMENT): Identificador del mantenimiento.
- id_dispositivo (INT, FK $\rightarrow$ dispositivos.id_dispositivo): Dispositivo a intervenir.
- id_tecnico (INT, FK $\rightarrow$ usuarios.id_usuario): Técnico asignado.
- id_solicitud (INT, NULL, FK $\rightarrow$ solicitudes_mantenimiento.id_solicitud): Solicitud origen.
- tipo (ENUM('PREVENTIVO', 'CORRECTIVO', 'PREDICTIVO'), NOT NULL): Categoría de la tarea.
- estado (ENUM('PENDIENTE', 'EN_PROCESO', 'FINALIZADO', 'CANCELADO'), DEFAULT 'PENDIENTE').
- fecha_programada (DATETIME, NOT NULL): Fecha pactada para la intervención.
- fecha_realizada (DATETIME, NULL): Instante de cierre efectivo.
- observaciones (TEXT, NULL): Informe del trabajo realizado.
- costo (DECIMAL(10,2), DEFAULT 0.00): Importe asociado al servicio.

**17. Tabla inventario**

Control de stock de repuestos, componentes y herramientas.

- id_item (INT, PK, AUTO_INCREMENT): Identificador del artículo.
- nombre (VARCHAR(100), NOT NULL): Denominación del producto.
- categoria (VARCHAR(50), NOT NULL): Categoría (SENSORES, NODOS, CABLES, etc.).
- cantidad_stock (INT, NOT NULL, DEFAULT 0): Unidades disponibles en almacén.
- stock_minimo (INT, NOT NULL, DEFAULT 5): Umbral para alerta de reabastecimiento.
- ubicacion_almacen (VARCHAR(100), NULL): Estante o depósito físico.
- fecha_actualizacion (DATETIME, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP).

**18. Tabla movimientos_inventario**

Trazabilidad de entradas y salidas de stock del inventario.

- id_movimiento (INT, PK, AUTO_INCREMENT): Identificador de la transacción.
- id_item (INT, FK $\rightarrow$ inventario.id_item): Artículo afectado.
- id_usuario (INT, FK $\rightarrow$ usuarios.id_usuario): Usuario que autorizó el movimiento.
- tipo_movimiento (ENUM('ENTRADA', 'SALIDA'), NOT NULL): Sentido del flujo de stock.
- cantidad (INT, NOT NULL): Número de unidades movilizadas.
- motivo (VARCHAR(255), NULL): Justificación (ej: *"Instalación Edificio Centro"*).
- fecha_movimiento (DATETIME, DEFAULT CURRENT_TIMESTAMP): Marca de tiempo.

**19. Tabla compras**

Gestión de pedidos de equipamiento y contrataciones de servicio.

- id_compra (INT, PK, AUTO_INCREMENT): Identificador del pedido.
- id_cliente (INT, NULL, FK $\rightarrow$ clientes.id_cliente): Cliente comprador.
- id_solicitante (INT, FK $\rightarrow$ usuarios.id_usuario): Usuario creador del pedido.
- codigo_compra (VARCHAR(50), NOT NULL, UNIQUE): Código de seguimiento.
- total (DECIMAL(10,2), NOT NULL): Importe total de la operación.
- estado (ENUM('Pendiente', 'Aprobada', 'En entrega', 'Completada', 'Cancelada'), DEFAULT 'Pendiente').
- created_at (DATETIME, DEFAULT CURRENT_TIMESTAMP): Fecha de creación del registro.

**20. Tabla log_actividad**

Tabla de auditoría global para el registro de eventos del sistema.

- id_log (BIGINT, PK, AUTO_INCREMENT): Identificador del evento de log.
- id_usuario (INT, NULL, FK $\rightarrow$ usuarios.id_usuario): Usuario ejecutor del evento.
- accion (VARCHAR(100), NOT NULL): Operación ejecutada (LOGIN, CREATE, UPDATE, DELETE).
- detalle (TEXT, NOT NULL): Descripción completa de la acción realizada.
- ip (VARCHAR(45), NULL): Dirección IP del cliente web.
- fecha_hora (DATETIME, DEFAULT CURRENT_TIMESTAMP): Instante de ocurrencia.

**21. Tabla notificaciones**

Alertas e informaciones internas dirigidas a usuarios específicos.

- id_notificacion (INT, PK, AUTO_INCREMENT): Identificador de la notificación.
- id_usuario (INT, FK $\rightarrow$ usuarios.id_usuario): Usuario destinatario.

- titulo (VARCHAR(150), NOT NULL): Encabezado de la notificación.
- mensaje (TEXT, NOT NULL): Cuerpo del mensaje.
- leida (TINYINT(1), DEFAULT 0): Estado de lectura (0 = No leída, 1 = Leída).
- fecha_hora (DATETIME, DEFAULT CURRENT_TIMESTAMP): Fecha de emisión.

### 9.3 Semillas de Datos (*Seeds*) del Sistema

El volcado oficial de la base de datos (u156482620_EVAelvigilante (1).sql) incluye los siguientes registros iniciales de configuración:

- **Roles Base (roles):**
  - 1 $\rightarrow$ ADMIN (Administrador General del Sistema)
  - 2 $\rightarrow$ TECNICO (Técnico de Campo y Mantenimiento)
  - 3 $\rightarrow$ USUARIO (Cliente / Consorcio)
- **Usuario Administrador Predeterminado (usuarios):**
  - **Email:** admin@eva.com
  - **Contraseña Hash:** $2y$10$YourHash... (Correspondiente a la clave inicial de desarrollo).
  - **Rol ID:** 1 (ADMIN).
  - **Estado:** 1 (Activo).

### 9.4 Estructuras SQL para Tablas Faltantes

A continuación se proveen los scripts DDL que deben ser ejecutados en el servidor MariaDB para crear las tablas requeridas por la lógica del código fuente que no estaban incluidas en el volcado inicial:

```
 Tabla para la gestión de intentos de acceso y rate-limiting
CREATE TABLE IF NOT EXISTS login_intentos (
    id_intento INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(150) NOT NULL,
    exito TINYINT(1) NOT NULL DEFAULT 0,
    ip VARCHAR(45) NOT NULL,
    fecha_hora DATETIME DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_email_fecha (email, fecha_hora)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

 Tabla para la gestión de tokens de recupero de contraseña
CREATE TABLE IF NOT EXISTS password_resets (
    id_reset INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(150) NOT NULL,
    token VARCHAR(64) NOT NULL,
    expira DATETIME NOT NULL,
    usado TINYINT(1) DEFAULT 0,
    INDEX idx_token (token)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

 Tabla puente para la asignación directa de dispositivos a técnicos
CREATE TABLE IF NOT EXISTS tecnico_dispositivo (
    id_tecnico INT NOT NULL,
    id_dispositivo INT NOT NULL,
    PRIMARY KEY (id_tecnico, id_dispositivo),
```

```
    FOREIGN KEY (id_tecnico) REFERENCES usuarios(id_usuario) ON DELETE
CASCADE,
    FOREIGN KEY (id_dispositivo) REFERENCES dispositivos(id_dispositivo) ON
DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

## 10. APIS Y ENDPOINTS DEL SISTEMA

### 10.1 Tabla Maestra de Endpoints

| Ruta del Endpoint | Método | Autenticación / Guard | Parámetros Recibidos | Respuesta Exito (200 OK) | Estado |
|---|---|---|---|---|---|
| api/esp32/mediciones.php | POST | Pública (IoT) | JSON: id_sensor, distancia_cm, nivel_cm, litros | {"ok": true, "id_medicion": N} | **REAL** |

| | | | | | |
|---|---|---|---|---|---|
| api/guardar_datos.php | POST/GET | Pública (IoT) | Form/JSON: sensor_id, distancia, porcentaje | {"status": "success", "id": N} | **REAL** |
| admin/api/estado.php | GET | Sesión ADMIN | Ninguno (Consulta tiempo real) | JSON con contadores y métricas | **REAL** |
| tecnico/api/estado.php | GET | Sesión TECNICO | Ninguno (Consulta tiempo real) | JSON con métricas técnicas | **REAL** |
| cliente/api/estado.php | GET | Sesión USUARIO | id_tanque (Opcional) | JSON con telemetría de tanque | **REAL** |

| | | | | | |
|---|---|---|---|---|---|
| cliente/api/tanque.php | GET | Sesión USUARIO | id_tanque | JSON con estado puntual | **REAL** |
| cliente/api/resumen.php | GET | Sesión USUARIO | period (semana/mes/anio) | JSON con series y agregaciones | **REAL** |
| cliente/api/historial.php | GET | Sesión USUARIO | desde, hasta, period, tanque | JSON con matriz de mediciones | **REAL** |
| cliente/api/alertas.php | GET | Sesión USUARIO | filter (todas/activas/resueltas) | JSON con arreglo de alertas | **REAL** |

| | | | | | |
|---|---|---|---|---|---|
| cliente/api/configuracion.php | GET/POST | Sesión USUARIO | JSON: nivel_minimo, nivel_maximo | {"success": true} | **REAL** |
| cliente/api/procesar.php | POST | Sesión Activa | Ninguno (Dispara actualización) | {"success": true, "processed": true} | **REAL** |
| cliente/api/sincronizar.php | POST | Sesión Activa | Ninguno (Sincroniza consumos) | {"success": true, "synced": true} | **REAL** |
| tecnico/api/dashboard.php | GET | Invoca librería inoperativa | Ninguno | Error 500 (Fatal Error) | **MUERTA** |

| | | | | | |
|---|---|---|---|---|---|
| tecnico/api/dispositivos.php | GET/PUT | Invoca librería inoperativa | id_dispositivo, estado | Error 500 (Fatal Error) | **MUERTA** |

**10.2 Guía para la Adición de Nuevos Endpoints**

Al desarrollar un nuevo endpoint en el sistema EVA, el desarrollador debe seguir esta secuencia estandarizada de pasos:

1. **Definir la Cabecera de Tipo de Contenido y Modo Estricto:**

```
<?php
declare(strict_types=1);
session_start();
header('Content-Type: application/json; charset=utf-8');
header('Cache-Control: no-store, no-cache, must-revalidate');
```

2. **Implementar el Guard de Autorización:** Determinar el nivel de acceso requerido y validar la variable $_SESSION['rol']. Si el usuario no cumple el requisito, retornar HTTP 401 inmediatamente:

```
if (!isset($_SESSION['rol']) || !in_array($_SESSION['rol'], ['ADMIN', 'TECNICO'], true)) {
    http_response_code(401);
    echo json_encode(['error' => 'No autorizado para acceder a este recurso']);
```

```
    exit;
}
```

3. **Inscribir la Capa de Base de Datos PDO:**

```
require_once __DIR__ . '/../../config/db.php';
$pdo = eva_pdo();
```

4. **Validación y Sanitización del Entrada Payload:** Si el endpoint recibe datos mediante JSON, leer el stream de entrada y validar que no contenga valores vacíos o malformados:

```
$rawInput = file_get_contents('php://input');
$data = json_decode($rawInput, true);

if (!$data || !isset($data['parametro_requerido'])) {
    http_response_code(400);
    echo json_encode(['error' => 'Solicitud malformada: Faltan parametros obligatorios']);
    exit;
}
```

5. **Procesamiento de la Lógica con Control de Excepciones:** Envolver la interacción con la base de datos en bloques try...catch. Retornar respuestas legibles con códigos de estado HTTP coherentes (200 OK, 201 Created, 500 Internal Server Error).

## 11. CAPA FRONTEND (CSS Y JAVASCRIPT)

### 11.1 Arquitectura Frontend Vanilla

La capa de presentación de EVA prescinde deliberadamente de librerías o frameworks pesados (*Bootstrap, Tailwind, React, Chart.js*). Todo el diseño visual, la interactividad de componentes y la generación de gráficos estadísticos han sido desarrollados utilizando HTML5 semántico, hojas de estilo CSS3 nativas con variables customizables y código JavaScript moderno (ES6+).

**Inyección de Contexto PHP a JavaScript (Variables Puente window.EVA_*)**

Dado que las vistas se renderizan en el servidor, los scripts PHP transfieren el estado de las entidades a los scripts de JavaScript incrustando objetos JSON directamente en la cabecera del documento web mediante variables globales en el objeto window:

```
<!-- Patrón de inyección de contexto utilizado en cliente/indexcli.php -->

<script>

    window.EVA_RESUMEN = <?php echo json_encode([

        'pct'         => $pct,

        'litros'      => $litros,

        'capacidad'   => $capacidad,

        'consumoHoy'  => $consumoHoy,

        'chartData'   => $chartSerie,
```

```
        'tanqueNombre'=> $tanque['nombre']

    ], JSON_UNESCAPED_UNICODE); ?>;

</script>
```

Posteriormente, el script de frontend (cliente/js/script.js) lee el objeto window.EVA_RESUMEN al dispararse el evento DOMContentLoaded y renderiza dinámicamente las barras, medidores y etiquetas sin requerir una nueva petición HTTP.

### 11.2 Desglose de Scripts JavaScript por Panel

#### 1. Panel de Administración (admin/js/admin.js - 25 KB)

- **Generación de Gráficos SVG:** Contiene las rutinas para dibujar gráficos de líneas y barras directamente en elementos <svg> manipulando atributos stroke-dasharray, viewBox y coordenadas de puntos sin depender de librerías de renderizado.
- **Mapeo de Modales:** Controla la apertura, validación y cierre de los modales de edición de usuarios, empresas, tanques y dispositivos.
- **Filtros Dinámicos:** Aplica filtros en el navegador sobre tablas de datos procesando los eventos keyup en los campos de búsqueda sin recargar la página.

#### 2. Engine Polling de Administración (admin/js/tiempo-real.js - 3.6 KB)

- Consulta periódicamente el endpoint admin/api/estado.php cada 3000 milisegundos.
- Detecta si la pestaña está activa utilizando document.hidden. Si el usuario cambia de pestaña, el temporizador de polling se suspende automáticamente para reducir el consumo de recursos en el servidor.

- Mantiene la variable evaPollInFlight como una bandera de bloqueo (*mutex*): si una petición de polling anterior demora más de 3 segundos en responder, cancela el disparo de la siguiente hasta que la red se libere, evitando el solapamiento de peticiones.

**3. Panel de Clientes (cliente/js/script.js - 34 KB)**

- **Componente Gauge Interactivo:** Modifica la propiedad stroke-dashoffset del arco SVG del medidor porcentual para representar el nivel de agua con animaciones de transición fluidas.
- **Visualización Dinámica del Tanque:** Ajusta la posición vertical de la columna de agua dentro del tanque SVG y altera los parámetros de la onda de animación basándose en el porcentaje de llenado recibido.
- **Selectores de Rango:** Controla el comportamiento de los deslizadores (range sliders) en la vista de configuración de umbrales (cliente/configuracion.php), calculando dinámicamente el área de relleno coloreado entre el valor mínimo y máximo.

**4. Engine Polling de Clientes (cliente/js/tiempo-real.js - 6.8 KB)**

- Realiza peticiones de polling contra cliente/api/estado.php actualizando los componentes de nivel, litros disponibles, temperatura y banderas de estado de los dispositivos en tiempo real.

## 12. AUDITORÍA DE SEGURIDAD Y RIESGOS

El análisis de seguridad realizado sobre el código fuente de EVA identificó la existencia de controles de protección debidamente implementados, así como una serie de vulnerabilidades técnicas y riesgos de severidad diversa que deben ser subsanados antes de un despliegue masivo en entornos de producción.

### 12.1 Controles de Seguridad Implementados

- **Almacenamiento Seguro de Contraseñas:** El sistema utiliza la función nativa password_hash() con el algoritmo Bcrypt para la creación y actualización de claves. Las verificaciones se realizan mediante password_verify().
- **Protección contra Fijación de Sesiones:** El script index.php ejecuta session_regenerate_id(true) de forma inmediata tras validar correctamente las credenciales del usuario.
- **Rate-Limiting de Login:** Se registra cada intento fallido de acceso y se aplica un bloqueo temporal de 15 minutos al superar los 5 reintentos consecutivos sobre un mismo correo electrónico.
- **Aislamiento de Cargas de Archivos:** Las imágenes subidas desde el módulo cliente son validadas mediante la extensión finfo (tipo MIME real) y se almacenan en un directorio protegido con un archivo .htaccess que deshabilita la ejecución de motores PHP.

### 12.2 Matrices de Riesgos Tácticos Clasificados por Severidad

MATRIZ DE RIESGOS DE SEGURIDAD

| | |
|---|---|
| CRÍTICA | Inyección SQL por Interpolación en admin/dispositivos.php |
| ALTA | Exposición de Credenciales en Texto Plano (config/db.php) |
| ALTA | Vulnerabilidad de Vulnerabilidad CSRF Generalizada |
| MEDIA | Enumeración de Usuarios en el Formulario de Login |
| MEDIA | Falta de Aislamiento de Datos por Cliente (Ownership Laxo) |

**1. RIESGO CRÍTICO: Inyección SQL por Interpolación Directa de Parámetros**

- **Archivos Afectados:** admin/dispositivos.php, admin/tanques.php, admin/tecnicos.php.
- **Mecanismo de la Vulnerabilidad:** En los scripts indicados, las consultas de verificación de dependencias previas al borrado o actualización concatenan variables recibidas directamente desde la solicitud HTTP sin utilizar sentencias preparadas:

  ```
  // Código vulnerable identificado en admin/dispositivos.php
  $id = $_POST['id_dispositivo'];
  $cant = $conn->query("SELECT COUNT(*) FROM sensores WHERE id_dispositivo = {$id}")->fetch_row()[0];
  ```

- **Impacto:** Un atacante autenticado con rol de Administrador podría manipular el parámetro id_dispositivo para alterar la lógica de la consulta SQL, extraer información confidencial de la base de datos o ejecutar comandos arbitrarios en el servidor MariaDB.
- **Remediación Requerida:** Reemplazar de forma inmediata todas las concatenaciones e interpolaciones en consultas SQL por sentencias preparadas parametrizadas utilizando PDO:

  ```
  $stmt = $pdo->prepare("SELECT COUNT(*) FROM sensores WHERE id_dispositivo = :id");
  $stmt->execute([':id' => $idDispositivo]);
  $cant = (int)$stmt->fetchColumn();
  ```

**2. RIESGO ALTO: Exposición de Credenciales de Base de Datos en Texto Plano**

- **Archivo Afectado:** config/db.php.

- **Mecanismo del Riesgo:** El archivo contiene un arreglo con credenciales de base de datos *hardcoded* (#VALzona122233) como mecanismo de contingencia para entornos donde el archivo .env no esté configurado.
- **Impacto:** Si el repositorio de código fuente se filtra o se expone públicamente, terceros no autorizados obtendrían acceso directo a la base de datos de producción en el proveedor de hosting.
- **Remediación Requerida:** Eliminar la totalidad de las contraseñas y usuarios hardcoded en los archivos PHP. Forzar a que la conexión dependa exclusivamente de las variables de entorno leídas desde el archivo .env, garantizando que este último se mantenga fuera del control de versiones mediante .gitignore.

**3. RIESGO ALTO: Vulnerabilidad Generalizada a Falsificación de Peticiones en Sitios Cruzados (CSRF)**

- **Archivos Afectados:** La totalidad de los formularios del módulo admin/ y los scripts de mutación del módulo tecnico/.
- **Mecanismo de la Vulnerabilidad:** A excepción de los formularios de cliente/configuracion.php y cliente/mantenimiento.php, el resto de los formularios del sistema no integran tokens de verificación Anti-CSRF.
- **Impacto:** Un usuario autenticado que visite un sitio web malicioso mientras mantiene una sesión activa en EVA podría ser inducido a ejecutar acciones no deseadas en el panel (tales como la eliminación de tanques, creación de usuarios no autorizados o modificación de contraseñas) sin su conocimiento.
- **Remediación Requerida:** Extender el uso de las funciones eva_csrf_token() y eva_csrf_validate() a la totalidad de los formularios POST del sistema.

**4. RIESGO MEDIO: Enumeración de Usuarios en el Formulario de Autenticación**

- **Archivo Afectado:** index.php.
- **Mecanismo de la Vulnerabilidad:** El script retorna mensajes de error diferenciados ante fallas en el acceso: responde *"Usuario no encontrado"* cuando el correo no existe en la base de datos, y *"Contraseña incorrecta"* cuando el correo existe pero la clave ingresada no coincide.
- **Impacto:** Permite a un atacante automatizar peticiones para verificar qué direcciones de correo electrónico se encuentran registradas en la plataforma.
- **Remediación Requerida:** Unificar las respuestas de fallo de autenticación bajo un mensaje genérico y neutral: *"Credenciales de acceso inválidas"*.

**5. RIESGO MEDIO: Ausencia de Aislamiento Estricto de Datos entre Clientes (*Ownership Laxo*)**

- **Archivo Afectado:** cliente/includes/helpers.php (Función eva_tanques_for_user()).
- **Mecanismo de la Vulnerabilidad:** La función eva_tanques_for_user() consulta si el usuario pertenece a la tabla clientes, pero al construir la respuesta retorna la lista global de tanques activos sin restringir la consulta al identificador del cliente autenticado.
- **Impacto:** Un usuario con rol USUARIO puede acceder a la telemetría y configuraciones de tanques que pertenecen a otros consorcios o clientes editando los identificadores pasados en los parámetros GET.
- **Remediación Requerida:** Modificar la sentencia SQL en eva_tanques_for_user() para incluir una cláusula WHERE estricta que filtre los tanques basándose en la relación con el cliente autenticado:

```
SELECT t.* FROM tanques t
```

```
INNER JOIN edificios e ON t.id_edificio = e.id_edificio
INNER JOIN empresas emp ON e.id_empresa = emp.id_empresa
INNER JOIN usuarios u ON u.id_empresa = emp.id_empresa

WHERE u.id_usuario = :idUsuario AND t.activo = 1
```

## 13. MANTENIMIENTO Y DESPLIEGUE

### 13.1 Requisitos de Infraestructura

Para garantizar el funcionamiento estable del sistema EVA en un entorno de producción, el servidor de alojamiento debe cumplir con los siguientes requisitos mínimos:

- **Sistema Operativo:** Linux (Ubuntu Server 20.04 LTS / 22.04 LTS o CloudLinux).
- **Servidor Web:** Apache 2.4 con módulo mod_rewrite activo y soporte para archivos .htaccess.
- **Entorno PHP:** PHP 7.4 o PHP 8.1+ ejecutándose en modo FPM o como módulo de Apache.
- **Extensiones PHP Requeridas:** mysqli, pdo_mysql, json, mbstring, fileinfo, curl, openssl.
- **Servidor de BD:** MariaDB 10.5+ o MySQL 8.0+.
- **Certificado SSL:** Certificado TLS/SSL activo para permitir conexiones HTTPS seguras (indispensable para la ingesta de datos desde los nodos ESP32 y el envío de cookies de sesión seguras).

### 13.2 Procedimiento de Despliegue Paso a Paso

1. **Clonado del Repositorio:** Clonar el repositorio en el directorio raíz del servidor web (/var/www/html/ o public_html/):

   ```
   git clone https://github.com/tu-usuario/Proyecto-Anual-Web.git .
   ```

2. **Configuración de Permisos de Archivos:** Asegurar que el servidor web posea permisos de lectura sobre la aplicación y permisos de escritura sobre la carpeta de subidas:

   ```
   chmod -R 755 .
   chmod -R 775 cliente/uploads/
   chown -R www-data:www-data .
   ```

3. **Importación del Esquema de Base de Datos:** Crear la base de datos en MariaDB e importar el archivo de volcado oficial:

   ```
   mysql -u usuario_db -p nombre_db < db/u156482620_EVAelvigilante\ \(1\).sql
   ```

4. **Ejecución de Migraciones de Tablas Faltantes:** Ejecutar los scripts SQL indicados en la **Sección 9.4** de este manual para crear las tablas login_intentos, password_resets y tecnico_dispositivo.
5. **Configuración del Archivo .env:** Crear el archivo .env en la raíz del proyecto definiendo las credenciales reales del entorno de producción:

   ```
   APP_DEBUG=0
   DB_HOST=localhost
   DB_USER=usuario_produccion
   DB_PASS=ContrasenaSegura123!
   ```

```
DB_NAME=nombre_db_produccion
BASE_URL=https://dashboard.elvigilantedeagua.com
SMTP_HOST=smtp.hostinger.com
SMTP_PORT=465
SMTP_USER=no-reply@dashboard.elvigilantedeagua.com
SMTP_PASS=CredencialSMTP123!
SMTP_ENCRYPTION=ssl
```

6. **Prueba de Conectividad e Ingesta:** Verificar el acceso ingresando a BASE_URL/index.php con las credenciales del usuario administrador. Realizar una petición de prueba al endpoint de ingesta api/esp32/mediciones.php para confirmar la correcta lectura y persistencia de telemetría.

### 13.3 Guía de Creación de un Nuevo Módulo en la Aplicación

Cuando se requiera incorporar un nuevo panel o módulo funcional al proyecto, el equipo de desarrollo debe seguir los siguientes pasos metodológicos:

1. **Creación del Directorio del Módulo:** Crear la carpeta correspondiente en la raíz del proyecto (ej: gerencia/) replicando la estructura de carpetas estándar: css/, js/, includes/ y api/.
2. **Definición del Guard de Acceso:** Definir el nuevo rol en la tabla roles de la base de datos e implementar el Guard de verificación al inicio de cada vista del nuevo módulo:

```
<?php
session_start();
if (!isset($_SESSION['rol']) || $_SESSION['rol'] !== 'GERENCIA') {
```

```
    header("Location: ../index.php");
    exit;
}
require_once __DIR__ . '/../config/db.php';
```

3. **Definición de Layout e Includes Parciales:** Crear los archivos includes/sidebar.php e includes/header.php adaptando los accesos de navegación y la presentación del perfil.
4. **Desarrollo de Vistas y Endpoints:** Construir las vistas HTML aplicando las pautas de diseño y utilizar la conexión PDO (eva_pdo()) para todas las interacciones con la base de datos.

## 14. ERRORES CONOCIDOS Y DEUDA TÉCNICA

A continuación se presenta el catálogo de errores de código, inconsistencias y deudas técnicas identificadas durante la auditoría del repositorio, documentando su ubicación exacta (archivo y línea de código) para su pronta corrección:

1. **Invocación de Extensión Inexistente mysqlnd (admin/edificios.php - Línea 45):** Uso del método $stmt->get_result() sobre un objeto mysqli_stmt. En servidores que carecen del driver mysqlnd, esta línea produce un Fatal Error 500. **Solución:** Migrar la consulta a PDO.
2. **Error de Sintaxis en Script de Auditorías (admin/js/auditorias.php - Línea 13):** Cierre incorrecto de paréntesis en la asignación de variable: $_GET['fecha_hasta'] ?? ");. El archivo es inejecutable y debe ser eliminado o corregido.
3. **Invocación de Archivos Inexistentes en API Técnica (tecnico/api/*.php - Línea 2):** Las 10 APIs del subdirectorio tecnico/api/ intentan incluir config/db_pdo.php y

config/helpers.php, archivos que no existen en el proyecto. **Solución:** Actualizar la cabecera para incluir config/db.php.

4. **Rutas Relativas Rotas en Parciales (admin/includes/sidebar.php - Línea 82):** El enlace para el cierre de sesión en el menú lateral apunta a ../config/logout.php. Si el archivo se incluye desde un subdirectorio secundario, la ruta relativa falla. **Solución:** Utilizar rutas absolutas basándose en la constante BASE_URL.
5. **Inconsistencia de Atributos de Entrada en Maquetas (tecnico/instalar.php - Líneas 24-48):** Los campos de entrada del formulario carecen de los atributos name y el formulario carece del atributo action. Los datos ingresados por el usuario no pueden ser procesados ni enviados al servidor.
6. **Error Lógico en Consulta de Mantenimientos (tecnico/indextec.php - Línea 88):** El script ejecuta dos veces la misma consulta SQL asignando el resultado a la variable $instPendientes con el rótulo *"Instalaciones Programadas"*. En realidad, la consulta contabiliza el total de instalaciones sin filtrar por estado pendiente.
7. **Sobrescritura de Variables de Búsqueda (admin/inventario.php - Línea 64):** Definición duplicada del bloque $where. Una primera versión con placeholders posicionales (?) es inmediatamente sobrescrita por una segunda versión con placeholders nombrados (:busqueda), dejando código muerto inservible.
8. **Ausencia de Scoping por Técnico en Cierre de Alertas (tecnico/api/alertas.php - Línea 41):** La consulta UPDATE que modifica el estado de una alerta no incluye la cláusula AND id_tecnico = :idTecnico. De funcionar el endpoint, cualquier técnico podría modificar el estado de alertas pertenecientes a otros profesionales.
9. **Incompatibilidad de Nombres en el Dump de BD (db/u156482620_EVAelvigilante (1).sql - Línea 12):** El volcado crea la base de datos bajo el nombre u156482620_EVAelvigilante, mientras que scripts antiguos buscan la base de datos

u767580032_elvigilante. **Solución:** Estandarizar el nombre de la base de datos a través del archivo .env.

10. **Inconsistencia de Tipos Enum entre Vistas y Base de Datos (tecnico/mantenimientos.php - Línea 112):** La vista maqueta presenta una tarjeta con la categoría de mantenimiento Emergencia. Dicho valor no existe en la definición de la columna tipo de la tabla mantenimientos (PREVENTIVO, CORRECTIVO, PREDICTIVO), por lo que una eventual inserción fallaría por restricción de integridad.

## 15. GLOSARIO Y ANEXOS

### 15.1 Glosario de Términos Técnicos

- **ESP32:** Microcontrolador de bajo costo y bajo consumo de energía con Wi-Fi y Bluetooth integrado de modo dual, utilizado como nodo de telemetría en el proyecto EVA.
- **JSR-SR04T:** Sensor de distancia ultrasónico estanco e impermeable, utilizado para medir la distancia desde la tapa del tanque hasta el espejo de agua.
- **Telemetría:** Sistema que permite la medición y rastreo automatizado de datos físicos en localidades remotas y su transmisión a un receptor para su análisis.
- **Polling (Sondeo):** Técnica de comunicación donde el cliente web realiza peticiones HTTP periódicas a intervalos regulares de tiempo (ej: 3 segundos) para consultar si existen nuevos datos en el servidor.
- **Soft-Delete (Borrado Lógico):** Patrón de diseño donde un registro no es eliminado físicamente de la base de datos (DELETE), sino que se marca como inactivo modificando una columna de estado (ej: activo = 0).

- **Guard:** Bloque de código situado al inicio de un script encargada de verificar la autenticación y los permisos del usuario antes de permitir la ejecución del resto del programa.
- **Bcrypt:** Función de hashing de contraseñas adaptativa basada en el cifrado Blowfish, utilizada por el motor de PHP para encriptar las claves de los usuarios de forma segura.
- **PDO (PHP Data Objects):** Extensión de PHP que define una interfaz liviana y consistente para acceder a bases de datos, permitiendo el uso de consultas preparadas nativas.
- **CSRF (Cross-Site Request Forgery):** Tipo de vulnerabilidad donde un sitio malicioso induce al navegador de una víctima autenticada a ejecutar acciones no deseadas en una aplicación web confiable.
- **Auto-DDL:** Capacidad de un script de software para ejecutar sentencias de lenguaje de definición de datos (CREATE TABLE, ALTER TABLE) de forma automática en tiempo de ejecución al detectar que una estructura no existe en la base de datos.

### 15.2 Catálogo de Tipos Numerados (ENUMs) de la Base de Datos

- **roles.nombre:** 'ADMIN', 'TECNICO', 'USUARIO'
- **dispositivos.estado:** 'ONLINE', 'OFFLINE', 'MANTENIMIENTO'
- **sensores.estado:** 'ACTIVO', 'INACTIVO', 'FALLA'
- **alertas.tipo:** 'NIVEL_BAJO', 'NIVEL_ALTO', 'SIN_CONEXION', 'FALLA_SENSOR', 'CONSUMO_ANORMAL'
- **alertas.estado:** 'PENDIENTE', 'ATENDIDA', 'CERRADA'
- **solicitudes_mantenimiento.estado:** 'PENDIENTE', 'ACEPTADA', 'EN_PROCESO', 'FINALIZADA', 'CANCELADA'
- **mantenimientos.tipo:** 'PREVENTIVO', 'CORRECTIVO', 'PREDICTIVO'

- **mantenimientos.estado:** 'PENDIENTE', 'EN_PROCESO', 'FINALIZADO', 'CANCELADO'
- **movimientos_inventario.tipo_movimiento:** 'ENTRADA', 'SALIDA'
- **compras.estado:** 'Pendiente', 'Aprobada', 'En entrega', 'Completada', 'Cancelada'
- **credenciales_clientes.estado:** 'ACTIVA', 'INACTIVA', 'PENDIENTE'

### 15.3 Referencia de Documentos del Repositorio

Para ampliar la información contextual sobre la evolución del desarrollo del sistema EVA, se recomienda consultar los siguientes documentos presentes en la raíz del repositorio:

- **AGENTS.md:** Especificaciones de arquitectura, convenciones de codificación en español y reglas de interacción con la base de datos para agentes y desarrolladores.
- **RESUMEN_V0.1.0.md:** Resumen técnico de la entrega inicial del proyecto (Versión v0.1.0).
- **RESUMEN_V0.2.0.md:** Resumen de las extensiones, nuevas tablas y módulos incorporados en la versión v0.2.0.
- **CHANGELOG.md:** Registro cronológico de cambios, correcciones de errores y parches aplicados a la plataforma.