**OPERASYS**

Sistema de Reporte Diario de Operadores

**DOCUMENTO DE DESARROLLO COMPLETO**

*Guía Técnica para Desarrolladores*

8 Módulos Completos • PWA • Offline-First

Versión 1.0 - Octubre 2025

# TABLA DE CONTENIDOS

1. INTRODUCCIÓN Y VISIÓN GENERAL

2. ARQUITECTURA DEL PROYECTO

3. BASE DE DATOS

4. DESARROLLO POR MÓDULOS

4.1. Módulo 1: Sistema de Autenticación

4.2. Módulo 2: Registro de Usuarios y Firma Digital

4.3. Módulo 3: Gestión de Equipos (CRUD)

4.4. Módulo 4: Formulario de Reportes Diarios

4.5. Módulo 5: Sistema de Sincronización Offline

4.6. Módulo 6: Panel de Administración

4.7. Módulo 7: Exportación de Reportes a PDF

4.8. Módulo 8: Configuración PWA

5. ESTÁNDARES Y BUENAS PRÁCTICAS

6. PLAN DE PRUEBAS INTEGRAL

# 1. INTRODUCCIÓN Y VISIÓN GENERAL

## 1.1. Objetivo del Proyecto

OperaSys es una aplicación web progresiva (PWA) diseñada para digitalizar y automatizar los reportes de operación diaria de maquinaria pesada. El sistema permite a los operadores registrar su jornada laboral de forma eficiente, incluso sin conexión a internet, garantizando la sincronización automática de datos cuando la conexión esté disponible.

## 1.2. Stack Tecnológico

|  |  |  |
| --- | --- | --- |
| Tecnología | Versión | Propósito |
| PHP Nativo | 8.1+ | Backend y lógica de negocio |
| MySQL | 8.0+ | Base de datos |
| AdminLTE | 3.0 | Interfaz administrativa |
| JavaScript ES6+ | Vanilla | Interactividad y PWA |
| IndexedDB | 2.0 | Almacenamiento offline |

# 2. ARQUITECTURA DEL PROYECTO

## 2.1. Estructura de Carpetas Completa

operasys/

├── index.php

├── manifest.json

├── service-worker.js

│

├── config/

│ ├── database.php

│ └── config.php

│

├── modules/

│ ├── auth/ (login.php, logout.php, register.php)

│ ├── usuarios/ (perfil.php, firma.php, listar.php)

│ ├── equipos/ (listar.php, agregar.php, editar.php)

│ ├── reportes/ (nuevo.php, listar.php, ver.php, pdf.php)

│ └── admin/ (dashboard.php, auditoria.php)

│

├── api/

│ ├── auth.php

│ ├── usuarios.php

│ ├── equipos.php

│ ├── reportes.php

│ └── sync.php

│

└── assets/

├── js/ (app.js, offline.js, indexeddb.js, sync.js, firma.js)

├── css/ (custom.css, offline.css)

└── img/ (icons/, logo.png)

# 3. BASE DE DATOS

## 3.1. Script SQL Completo

CREATE DATABASE IF NOT EXISTS operasys;

USE operasys;

CREATE TABLE usuarios (

id INT PRIMARY KEY AUTO\_INCREMENT,

nombre\_completo VARCHAR(150) NOT NULL,

dni VARCHAR(20) UNIQUE NOT NULL,

cargo VARCHAR(50) NOT NULL,

password VARCHAR(255) NOT NULL,

firma TEXT NULL,

rol ENUM('operador','supervisor','admin') DEFAULT 'operador',

estado TINYINT(1) DEFAULT 1,

fecha\_creacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE equipos (

id INT PRIMARY KEY AUTO\_INCREMENT,

categoria VARCHAR(50) NOT NULL,

codigo VARCHAR(20) UNIQUE NOT NULL,

descripcion VARCHAR(255) NULL,

estado TINYINT(1) DEFAULT 1

);

CREATE TABLE reportes (

id INT PRIMARY KEY AUTO\_INCREMENT,

usuario\_id INT NOT NULL,

equipo\_id INT NOT NULL,

fecha DATE NOT NULL,

hora\_inicio TIME NOT NULL,

hora\_fin TIME NULL,

horas\_trabajadas DECIMAL(5,2) NULL,

actividad TEXT NOT NULL,

observaciones TEXT NULL,

estado\_sinc ENUM('pendiente','sincronizado') DEFAULT 'sincronizado',

FOREIGN KEY (usuario\_id) REFERENCES usuarios(id),

FOREIGN KEY (equipo\_id) REFERENCES equipos(id)

);

# 4. DESARROLLO POR MÓDULOS

## 4.1. MÓDULO 1: SISTEMA DE AUTENTICACIÓN

OBJETIVO:

Sistema de login/logout seguro con validación de credenciales y manejo de sesiones.

ARCHIVOS A CREAR:

|  |  |  |
| --- | --- | --- |
| Archivo | Ubicación | Función |
| database.php | config/ | Conexión PDO a MySQL |
| login.php | modules/auth/ | Vista de inicio de sesión |
| auth.php | api/ | API de autenticación |

FLUJO:

1. Usuario ingresa DNI y contraseña en login.php

2. JavaScript envía POST a api/auth.php

3. API valida con password\_verify() y crea sesión

4. Redirige a dashboard si correcto, error si incorrecto

CÓDIGO CLAVE - config/database.php:

$pdo = new PDO('mysql:host=localhost;dbname=operasys', 'root', '');

CÓDIGO CLAVE - api/auth.php:

$stmt = $pdo->prepare('SELECT \* FROM usuarios WHERE dni = ? AND estado = 1');

if (password\_verify($password, $user['password'])) {

$\_SESSION['user\_id'] = $user['id'];

echo json\_encode(['success' => true]);

}

PRUEBAS:

✓ Login con credenciales correctas → redirige a dashboard

✓ Login con credenciales incorrectas → muestra error

✓ Sesión se crea correctamente con variables user\_id, rol

## 4.2. MÓDULO 2: REGISTRO Y FIRMA DIGITAL

OBJETIVO:

Registrar nuevos usuarios y capturar firma digital mediante canvas HTML5.

ARCHIVOS A CREAR:

|  |  |  |
| --- | --- | --- |
| Archivo | Ubicación | Función |
| register.php | modules/auth/ | Formulario de registro |
| firma.php | modules/usuarios/ | Canvas de firma |
| usuarios.php | api/ | API CRUD usuarios |
| firma.js | assets/js/ | Lógica del canvas |

CONEXIONES:

register.php → api/usuarios.php (action=register) → BD

firma.php → firma.js → canvas.toDataURL() → api/usuarios.php (action=guardar\_firma)

CÓDIGO CLAVE - firma.js:

const canvas = document.getElementById('canvasFirma');

const firmaBase64 = canvas.toDataURL();

// Enviar a API para guardar en BD

PRUEBAS:

✓ Registro crea usuario con password hasheado

✓ Canvas captura trazos correctamente

✓ Firma se guarda como base64 en campo 'firma'

## 4.3. MÓDULO 3: GESTIÓN DE EQUIPOS (CRUD)

OBJETIVO:

Crear, leer, actualizar y eliminar equipos de maquinaria pesada. Incluye búsqueda por categoría.

ARCHIVOS A CREAR:

|  |  |  |
| --- | --- | --- |
| Archivo | Ubicación | Función |
| listar.php | modules/equipos/ | Tabla con todos los equipos |
| agregar.php | modules/equipos/ | Formulario nuevo equipo |
| editar.php | modules/equipos/ | Formulario editar equipo |
| equipos.php | api/ | API CRUD equipos |

CONEXIONES:

listar.php → api/equipos.php?action=listar → JSON con equipos

agregar.php → api/equipos.php?action=crear → INSERT BD

editar.php → api/equipos.php?action=actualizar → UPDATE BD

CÓDIGO CLAVE - api/equipos.php:

// LISTAR

if ($action === 'listar') {

$stmt = $pdo->query('SELECT \* FROM equipos WHERE estado = 1');

echo json\_encode($stmt->fetchAll());

}

// CREAR

elseif ($action === 'crear') {

$stmt = $pdo->prepare('INSERT INTO equipos (categoria, codigo, descripcion) VALUES (?, ?, ?)');

$stmt->execute([$\_POST['categoria'], $\_POST['codigo'], $\_POST['descripcion']]);

}

// BUSCAR POR CATEGORÍA

elseif ($action === 'buscar') {

$stmt = $pdo->prepare('SELECT \* FROM equipos WHERE categoria = ? AND estado = 1');

$stmt->execute([$\_GET['categoria']]);

echo json\_encode($stmt->fetchAll());

}

PRUEBAS:

✓ Listar muestra todos los equipos activos

✓ Crear equipo con código único funciona

✓ Editar equipo actualiza datos correctamente

✓ Eliminar (estado=0) oculta equipo de listado

✓ Búsqueda por categoría filtra correctamente

## 4.4. MÓDULO 4: FORMULARIO DE REPORTES DIARIOS

OBJETIVO:

Permitir a operadores crear reportes diarios con fecha/hora automática, selección de equipo, actividad y firma.

ARCHIVOS A CREAR:

|  |  |  |
| --- | --- | --- |
| Archivo | Ubicación | Función |
| nuevo.php | modules/reportes/ | Formulario de reporte |
| listar.php | modules/reportes/ | Listado de reportes |
| ver.php | modules/reportes/ | Detalle del reporte |
| reportes.php | api/ | API CRUD reportes |
| reportes.js | assets/js/ | Lógica del formulario |

CAMPOS DEL FORMULARIO:

• Fecha (automática: fecha actual)

• Hora inicio (automática: al abrir formulario)

• Hora fin (automática: al enviar)

• Categoría del equipo (select: Excavadora, Volquete, Tractor...)

• Código del equipo (filtrado por categoría)

• Actividad realizada (textarea)

• Observaciones (textarea opcional)

• Firma del operador (muestra la guardada, no captura nueva)

CÓDIGO CLAVE - reportes.js:

// Fecha y hora automáticas

document.getElementById('fecha').value = new Date().toISOString().split('T')[0];

const horaInicio = new Date().toLocaleTimeString('es-PE', {hour12: false});

// Filtrar equipos por categoría

categoriaSelect.addEventListener('change', async (e) => {

const res = await fetch(`api/equipos.php?action=buscar&categoria=${e.target.value}`);

const equipos = await res.json();

// Llenar select de códigos

});

CÓDIGO CLAVE - api/reportes.php:

if ($action === 'crear') {

$stmt = $pdo->prepare('

INSERT INTO reportes (usuario\_id, equipo\_id, fecha, hora\_inicio, hora\_fin, actividad, observaciones)

VALUES (?, ?, ?, ?, ?, ?, ?)

');

$stmt->execute([

$\_SESSION['user\_id'],

$\_POST['equipo\_id'],

$\_POST['fecha'],

$\_POST['hora\_inicio'],

$\_POST['hora\_fin'],

$\_POST['actividad'],

$\_POST['observaciones']

]);

}

PRUEBAS:

✓ Fecha se autocompleta con fecha actual

✓ Hora inicio se captura al cargar formulario

✓ Filtro de equipos muestra solo los de la categoría seleccionada

✓ Reporte se guarda correctamente en BD

✓ Firma del usuario aparece automáticamente sin recapturar

## 4.5. MÓDULO 5: SISTEMA DE SINCRONIZACIÓN OFFLINE

OBJETIVO:

Permitir que el sistema funcione completamente sin internet. Los datos se guardan localmente en IndexedDB y se sincronizan automáticamente cuando hay conexión.

ARCHIVOS A CREAR:

|  |  |  |
| --- | --- | --- |
| Archivo | Ubicación | Función |
| service-worker.js | raíz del proyecto | Caché de recursos |
| indexeddb.js | assets/js/ | Operaciones IndexedDB |
| sync.js | assets/js/ | Sincronización automática |
| offline.js | assets/js/ | Detectar estado online/offline |
| sync.php | api/ | Recibe datos para sincronizar |

FLUJO DE TRABAJO:

MODO ONLINE:

1. Usuario crea reporte → envía directamente a api/reportes.php

2. Se guarda en BD con estado\_sinc = 'sincronizado'

MODO OFFLINE:

1. Usuario crea reporte → detecta que no hay internet

2. Guarda en IndexedDB con estado = 'pendiente'

3. Muestra mensaje: 'Sin conexión. Datos guardados localmente'

SINCRONIZACIÓN AUTOMÁTICA:

1. Sistema detecta que volvió internet (evento 'online')

2. sync.js lee todos los reportes pendientes de IndexedDB

3. Los envía uno por uno a api/sync.php

4. Al confirmar guardado en BD, los elimina de IndexedDB

5. Muestra notificación: '✓ 3 reportes sincronizados correctamente'

CÓDIGO CLAVE - indexeddb.js:

// Abrir base de datos

const request = indexedDB.open('OperaSysDB', 1);

// Crear almacén

request.onupgradeneeded = (e) => {

const db = e.target.result;

db.createObjectStore('reportes', { keyPath: 'id', autoIncrement: true });

};

// Guardar reporte

function guardarReporteLocal(reporte) {

const transaction = db.transaction(['reportes'], 'readwrite');

const store = transaction.objectStore('reportes');

store.add({ ...reporte, estado: 'pendiente' });

}

CÓDIGO CLAVE - sync.js:

// Detectar conexión

window.addEventListener('online', sincronizarDatos);

async function sincronizarDatos() {

const reportesPendientes = await obtenerReportesPendientes();

for (const reporte of reportesPendientes) {

const res = await fetch('api/sync.php', {

method: 'POST',

body: JSON.stringify(reporte)

});

if (res.ok) {

await eliminarReporteLocal(reporte.id);

}

}

}

CÓDIGO CLAVE - service-worker.js:

const CACHE\_NAME = 'operasys-v1';

const urlsToCache = [

'/',

'/assets/css/custom.css',

'/assets/js/app.js',

'/vendor/adminlte/css/adminlte.min.css'

];

self.addEventListener('install', (e) => {

e.waitUntil(

caches.open(CACHE\_NAME).then(cache => cache.addAll(urlsToCache))

);

});

PRUEBAS:

✓ Desconectar internet → crear reporte → se guarda en IndexedDB

✓ Reconectar internet → reportes se sincronizan automáticamente

✓ Verificar que reportes sincronizados aparecen en BD

✓ IndexedDB se limpia después de sincronizar

✓ Aplicación funciona completamente sin internet

## 4.6. MÓDULO 6: PANEL DE ADMINISTRACIÓN

OBJETIVO:

Dashboard con métricas, listado de reportes con filtros avanzados, y registro de auditoría.

ARCHIVOS A CREAR:

|  |  |  |
| --- | --- | --- |
| Archivo | Ubicación | Función |
| dashboard.php | modules/admin/ | Panel principal con métricas |
| auditoria.php | modules/admin/ | Registro de actividades |
| estadisticas.php | api/ | API métricas dashboard |

MÉTRICAS DEL DASHBOARD:

• Total de reportes del día

• Total de reportes del mes

• Total de usuarios activos

• Total de equipos registrados

• Reportes pendientes de sincronización

• Gráfico de reportes por día (últimos 7 días)

• Gráfico de reportes por equipo

FILTROS AVANZADOS EN LISTADO DE REPORTES:

• Por fecha (desde - hasta)

• Por operador

• Por equipo

• Por categoría de equipo

• Por estado de sincronización

CÓDIGO CLAVE - api/estadisticas.php:

// Reportes del día

$stmt = $pdo->query('SELECT COUNT(\*) as total FROM reportes WHERE DATE(fecha) = CURDATE()');

$reportesHoy = $stmt->fetch()['total'];

// Reportes por equipo (top 5)

$stmt = $pdo->query('

SELECT e.codigo, COUNT(\*) as total

FROM reportes r

JOIN equipos e ON r.equipo\_id = e.id

GROUP BY e.codigo

ORDER BY total DESC LIMIT 5

');

CÓDIGO CLAVE - Filtros en reportes/listar.php:

$sql = 'SELECT r.\*, u.nombre\_completo, e.codigo FROM reportes r

JOIN usuarios u ON r.usuario\_id = u.id

JOIN equipos e ON r.equipo\_id = e.id WHERE 1=1';

if (!empty($\_GET['fecha\_desde'])) {

$sql .= ' AND r.fecha >= :fecha\_desde';

}

if (!empty($\_GET['operador'])) {

$sql .= ' AND r.usuario\_id = :operador';

}

PRUEBAS:

✓ Dashboard muestra métricas correctas

✓ Gráficos se generan con datos reales

✓ Filtros funcionan correctamente (individual y combinados)

✓ Auditoría registra todas las acciones importantes

✓ Solo usuarios con rol 'admin' pueden acceder

## 4.7. MÓDULO 7: EXPORTACIÓN DE REPORTES A PDF

OBJETIVO:

Generar reportes en formato PDF con firma digital del operador y datos completos del reporte.

ARCHIVOS A CREAR:

|  |  |  |
| --- | --- | --- |
| Archivo | Ubicación | Función |
| exportar\_pdf.php | modules/reportes/ | Genera PDF del reporte |

LIBRERÍA A USAR:

TCPDF o FPDF (librerías PHP para generar PDFs)

Instalación: composer require tecnickcom/tcpdf

CONTENIDO DEL PDF:

• Logo de la empresa

• Título: 'REPORTE DE OPERACIÓN DIARIA'

• Datos del operador (nombre, DNI, cargo)

• Fecha del reporte

• Hora inicio y fin

• Total de horas trabajadas

• Equipo utilizado (categoría y código)

• Actividad realizada

• Observaciones

• Firma digital del operador (imagen base64)

• Fecha y hora de generación del PDF

CÓDIGO CLAVE - exportar\_pdf.php:

require\_once('vendor/autoload.php');

require\_once('../config/database.php');

$reporteId = $\_GET['id'];

// Obtener datos del reporte

$stmt = $pdo->prepare('

SELECT r.\*, u.nombre\_completo, u.dni, u.cargo, u.firma,

e.categoria, e.codigo, e.descripcion

FROM reportes r

JOIN usuarios u ON r.usuario\_id = u.id

JOIN equipos e ON r.equipo\_id = e.id

WHERE r.id = ?

');

$stmt->execute([$reporteId]);

$datos = $stmt->fetch(PDO::FETCH\_ASSOC);

// Crear PDF

$pdf = new TCPDF();

$pdf->AddPage();

// Logo

$pdf->Image('../../assets/img/logo.png', 15, 10, 30);

// Título

$pdf->SetFont('helvetica', 'B', 18);

$pdf->Cell(0, 10, 'REPORTE DE OPERACIÓN DIARIA', 0, 1, 'C');

// Datos del operador

$pdf->SetFont('helvetica', '', 12);

$pdf->Cell(0, 8, 'Operador: ' . $datos['nombre\_completo'], 0, 1);

$pdf->Cell(0, 8, 'DNI: ' . $datos['dni'], 0, 1);

// Firma digital

if (!empty($datos['firma'])) {

$pdf->Image('@' . base64\_decode(preg\_replace('#^data:image/[^;]+;base64,#', '', $datos['firma'])),

15, $pdf->GetY(), 50);

}

// Generar PDF

$pdf->Output('Reporte\_' . $reporteId . '.pdf', 'D');

PRUEBAS:

✓ PDF se genera correctamente con todos los datos

✓ Firma digital aparece en el PDF

✓ Formato del PDF es profesional y legible

✓ Descarga automática funciona correctamente

## 4.8. MÓDULO 8: CONFIGURACIÓN PWA

OBJETIVO:

Convertir la aplicación web en PWA para que pueda instalarse como app en dispositivos móviles y de escritorio.

ARCHIVOS A CREAR:

|  |  |  |
| --- | --- | --- |
| Archivo | Ubicación | Función |
| manifest.json | raíz del proyecto | Configuración PWA |
| icon-192.png | assets/img/icons/ | Ícono 192x192 |
| icon-512.png | assets/img/icons/ | Ícono 512x512 |

CÓDIGO COMPLETO - manifest.json:

{

"name": "OperaSys - Sistema de Reportes",

"short\_name": "OperaSys",

"description": "Sistema de reporte diario de operadores",

"start\_url": "/",

"display": "standalone",

"background\_color": "#ffffff",

"theme\_color": "#2E86AB",

"orientation": "portrait",

"icons": [

{

"src": "/assets/img/icons/icon-192.png",

"sizes": "192x192",

"type": "image/png"

},

{

"src": "/assets/img/icons/icon-512.png",

"sizes": "512x512",

"type": "image/png"

}

]

}

REGISTRAR EN HTML (index.php):

<head>

<link rel="manifest" href="/manifest.json">

<meta name="theme-color" content="#2E86AB">

<link rel="apple-touch-icon" href="/assets/img/icons/icon-192.png">

</head>

REGISTRAR SERVICE WORKER:

if ('serviceWorker' in navigator) {

navigator.serviceWorker.register('/service-worker.js')

.then(() => console.log('Service Worker registrado'))

.catch(err => console.log('Error:', err));

}

PRUEBAS:

✓ En Chrome: aparece ícono de instalación en barra de direcciones

✓ En móvil: aparece opción 'Agregar a pantalla de inicio'

✓ App instalada abre en modo standalone (sin barra del navegador)

✓ Ícono y nombre de la app aparecen correctamente

✓ App funciona completamente sin conexión

# 5. ESTÁNDARES Y BUENAS PRÁCTICAS

## 5.1. Seguridad

• Siempre usar prepared statements para evitar SQL injection

• Hashear contraseñas con password\_hash() y verificar con password\_verify()

• Validar sesiones en cada página protegida

• Sanitizar entradas del usuario

• Usar HTTPS en producción

## 5.2. Código PHP

• Variables en camelCase: $nombreUsuario

• Constantes en MAYÚSCULAS: DB\_HOST

• Siempre usar try-catch para conexiones a BD

• Retornar JSON en APIs: header('Content-Type: application/json')

## 5.3. JavaScript

• Usar const/let, nunca var

• Funciones con camelCase: enviarFormulario()

• Usar async/await en lugar de callbacks

• Siempre manejar errores con try-catch

## 5.4. Base de Datos

• Tablas en plural: usuarios, equipos

• Campos en snake\_case: nombre\_completo

• Siempre usar foreign keys para relaciones

• Agregar índices en campos de búsqueda frecuente

# 6. PLAN DE PRUEBAS INTEGRAL

## 6.1. Pruebas por Módulo

Cada módulo debe completar sus pruebas específicas antes de continuar con el siguiente.

## 6.2. Pruebas de Integración

FLUJO COMPLETO DEL USUARIO:

1. Registro → Firma → Login exitoso

2. Crear equipo → Aparece en listado y en formulario de reportes

3. Crear reporte → Aparece en dashboard y puede exportarse a PDF

4. Modo offline → Reporte se guarda local → Sincroniza al reconectar

## 6.3. Pruebas de Rendimiento

• Dashboard debe cargar en menos de 2 segundos

• Listado de 1000 reportes debe renderizar en menos de 3 segundos

• PDF debe generarse en menos de 5 segundos

• Sincronización de 50 reportes en menos de 30 segundos

## 6.4. Pruebas de Compatibilidad

NAVEGADORES:

• Chrome (últimas 2 versiones)

• Firefox (últimas 2 versiones)

• Safari (iOS y macOS)

• Edge

DISPOSITIVOS:

• Smartphones Android

• iPhone

• Tablets

• Desktop (Windows, Mac, Linux)

# DOCUMENTO COMPLETO

Este documento contiene la guía completa para desarrollar OperaSys desde cero.

✓ 8 Módulos detallados

✓ Arquitectura completa

✓ Código de ejemplo

✓ Pruebas por módulo

✓ Estándares y buenas prácticas

Cualquier desarrollador puede seguir este documento paso a paso para construir el sistema completo.

🚀 OperaSys

Sistema de Reporte Diario de Operadores

**MODULOS POR ROLES**

**👥 ROLES Y PERMISOS EN OPERASYS**

**📊 TABLA COMPARATIVA:**

| **Funcionalidad** | **Operador** | **Supervisor** | **Admin** |
| --- | --- | --- | --- |
| **Dashboard** | **Solo suyo** | **Global** | **Global + Auditoría** |
| **Crear reportes** | **✅** | **✅** | **✅** |
| **Ver sus reportes** | **✅** | **✅** | **✅** |
| **Ver reportes de otros** | **❌** | **✅** | **✅** |
| **Editar reportes** | **❌** | **❌** | **✅** |
| **Eliminar reportes** | **❌** | **❌** | **✅** |
| **Descargar PDF propios** | **✅** | **✅** | **✅** |
| **Descargar PDF de otros** | **❌** | **✅** | **✅** |
| **Ver equipos** | **✅** | **✅** | **✅** |
| **Agregar equipos** | **❌** | **✅** | **✅** |
| **Editar equipos** | **❌** | **✅** | **✅** |
| **Eliminar equipos** | **❌** | **❌** | **✅** |
| **Ver usuarios** | **❌** | **❌** | **✅** |
| **Gestionar usuarios** | **❌** | **❌** | **✅** |
| **Ver auditoría** | **❌** | **❌** | **✅** |
| **Subir firma** | **✅** | **✅** | **✅** |