**📚 Documentación Técnica SaaS - MensajeroPro**

**🏗️ Arquitectura General del Sistema**

**Stack Tecnológico**

* **Backend**: PHP 8.1+ (nativo)
* **Frontend**: AdminLTE 3.2
* **Base de Datos**: MySQL 5.7+
* **WhatsApp**: Node.js 16+ con @wppconnect-team/wppconnect
* **IA**: OpenAI API (GPT-3.5/4)
* **Pagos**: MercadoPago SDK + PayPal SDK
* **Email**: PHPMailer
* **Autenticación**: Google OAuth 2.0

**Flujo de Datos**

Usuario → Landing → Login → Panel Cliente → API PHP → MySQL

↓

Node.js Service

↓

WhatsApp + OpenAI API

**📁 Estructura de Directorios Completa**

/mensajeropro-saas/

├── /public/ # Única carpeta accesible desde web

│ ├── index.php # Landing page

│ ├── login.php # Login unificado

│ ├── registro.php # Registro nuevo usuario

│ ├── verificar-email.php # Verificación de email

│ ├── .htaccess # Redirecciones y seguridad

│ └── /assets/

│ ├── /css/

│ ├── /js/

│ ├── /img/

│ └── /uploads/ # Archivos subidos

│ ├── /catalogos/ # PDFs de catálogos

│ └── /logos/ # Logos de empresas

│

├── /app/ # Lógica de aplicación

│ ├── /cliente/ # Panel del cliente

│ │ ├── dashboard.php

│ │ ├── /modulos/

│ │ │ ├── contactos.php

│ │ │ ├── categorias.php

│ │ │ ├── mensajes.php

│ │ │ ├── bot-config.php

│ │ │ ├── negocio-config.php

│ │ │ └── conversaciones.php

│ │ └── /layouts/

│ │ ├── header.php

│ │ ├── sidebar.php

│ │ └── footer.php

│ │

│ ├── /superadmin/ # Panel SuperAdmin

│ │ ├── dashboard.php

│ │ ├── /modulos/

│ │ │ ├── empresas.php

│ │ │ ├── planes.php

│ │ │ ├── pagos.php

│ │ │ ├── configuracion.php

│ │ │ └── emails.php

│ │ └── /layouts/

│ │ ├── header.php

│ │ ├── sidebar.php

│ │ └── footer.php

│ │

│ └── /api/

│ ├── /v1/

│ │ ├── /auth/

│ │ │ ├── login.php

│ │ │ ├── registro.php

│ │ │ ├── google-oauth.php

│ │ │ └── logout.php

│ │ ├── /cliente/

│ │ │ ├── /contactos/

│ │ │ ├── /mensajes/

│ │ │ ├── /bot/

│ │ │ └── /negocio/

│ │ ├── /superadmin/

│ │ │ ├── /empresas/

│ │ │ ├── /planes/

│ │ │ └── /configuracion/

│ │ └── /webhooks/

│ │ ├── mercadopago.php

│ │ └── paypal.php

│ └── response.php # Clase helper respuestas

│

├── /config/

│ ├── database.php # Conexión BD

│ ├── app.php # Constantes globales

│ ├── mail.php # Config email

│ └── payments.php # Config pasarelas pago

│

├── /includes/

│ ├── functions.php # Funciones globales

│ ├── auth.php # Funciones autenticación

│ ├── empresa.php # Funciones multi-empresa

│ ├── email-validator.php # Validación emails

│ └── security.php # Headers y validaciones

│

├── /database/

│ ├── schema-saas.sql # Estructura completa BD

│ ├── seeds.sql # Datos iniciales

│ └── migrations/ # Cambios incrementales

│

├── /whatsapp-service/ # Servicio Node.js

│ ├── /src/

│ │ ├── index.js # Entry point

│ │ ├── sessionManager.js # Manejo multi-sesión

│ │ ├── messageHandler.js # Procesamiento mensajes

│ │ ├── botAgent.js # Lógica bot ventas

│ │ ├── catalogReader.js # Lector de PDFs

│ │ ├── database.js # Conexión MySQL

│ │ └── api.js # API REST

│ ├── /sessions/ # Carpeta sesiones WhatsApp

│ ├── .env.example

│ └── package.json

│

├── /cron/

│ ├── check-payments.php # Verificar pagos vencidos

│ ├── send-reminders.php # Enviar recordatorios

│ └── clean-sessions.php # Limpiar sesiones viejas

│

├── /logs/

│ ├── error.log

│ ├── access.log

│ └── whatsapp.log

│

└── /docs/

└── api-documentation.md

**🔧 Módulo 1: Landing Page y Registro**

**Archivos involucrados:**

* public/index.php - Landing page
* public/registro.php - Formulario registro
* public/login.php - Login unificado
* app/api/v1/auth/registro.php - API registro
* app/api/v1/auth/google-oauth.php - Google OAuth
* includes/email-validator.php - Validación emails

**Base de datos:**

sql

CREATE TABLE empresas (

id INT PRIMARY KEY AUTO\_INCREMENT,

nombre\_empresa VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password\_hash VARCHAR(255),

google\_id VARCHAR(100),

telefono VARCHAR(20),

metodo\_registro ENUM('email', 'google'),

token\_verificacion VARCHAR(100),

email\_verificado BOOLEAN DEFAULT 0,

plan\_id INT DEFAULT 1, *-- Trial*

fecha\_registro DATETIME DEFAULT CURRENT\_TIMESTAMP,

fecha\_expiracion\_trial DATETIME,

ultimo\_acceso DATETIME,

activo BOOLEAN DEFAULT 1,

ip\_registro VARCHAR(45),

INDEX idx\_email (email),

INDEX idx\_google\_id (google\_id)

);

**Flujo de registro con email:**

1. Usuario llena formulario en registro.php
2. JavaScript valida y envía POST a api/v1/auth/registro.php
3. API valida:
   * Email no temporal (usando email-validator.php)
   * Email no existe en BD
   * reCAPTCHA válido
4. Crea registro con email\_verificado = 0
5. Envía email con código de 6 dígitos
6. Usuario ingresa código en verificar-email.php
7. Sistema activa cuenta y asigna trial 2 días

**Flujo Google OAuth:**

1. Click en "Registrar con Google"
2. Redirige a Google OAuth
3. Callback a google-oauth.php
4. Si email no existe, crea empresa
5. Login automático y trial activado

**Validación de emails temporales:**

php

*// includes/email-validator.php*

class EmailValidator {

private $dominiosBloqueados = [

'10minutemail.com', 'tempmail.com',

'guerrillamail.com', 'mailinator.com',

'throwawaymail.com', 'yopmail.com'

];

public function validar($email) {

*// 1. Formato válido*

*// 2. Dominio no temporal*

*// 3. MX records existen*

}

}

**🔧 Módulo 2: Panel Cliente - Dashboard**

**Archivos involucrados:**

* app/cliente/dashboard.php - Vista principal
* app/cliente/layouts/\* - Plantillas comunes
* app/api/v1/cliente/dashboard/stats.php - Estadísticas
* includes/empresa.php - Helper multi-empresa

**Funcionalidad clave - Multi-empresa:**

php

*// includes/empresa.php*

function getEmpresaActual() {

return $\_SESSION['empresa\_id'];

}

function scopeQuery($query) {

$empresa\_id = getEmpresaActual();

return $query . " AND empresa\_id = $empresa\_id";

}

**Widgets del dashboard:**

1. **Estado WhatsApp**: Conectado/Desconectado + QR
2. **Mensajes hoy**: Contador
3. **Conversaciones activas**: Lista
4. **Saldo/Plan**: Días restantes trial o plan activo

**Comunicación con Node.js:**

javascript

*// Verificar estado WhatsApp*

async function checkWhatsAppStatus() {

const response = await fetch('/api/v1/cliente/whatsapp/status.php', {

method: 'POST',

body: JSON.stringify({empresa\_id: EMPRESA\_ID})

});

*// Actualizar UI*

}

**🔧 Módulo 3: Configuración del Negocio**

**Archivos involucrados:**

* app/cliente/modulos/negocio-config.php - Vista
* app/api/v1/cliente/negocio/actualizar.php - API
* app/api/v1/cliente/negocio/subir-catalogo.php - Upload PDF
* public/uploads/catalogos/ - Almacenamiento PDFs

**Base de datos:**

sql

CREATE TABLE configuracion\_negocio (

id INT PRIMARY KEY AUTO\_INCREMENT,

empresa\_id INT UNIQUE NOT NULL,

nombre\_negocio VARCHAR(100),

telefono VARCHAR(20),

direccion TEXT,

horario\_apertura TIME,

horario\_cierre TIME,

dias\_laborales JSON, *-- ["lunes", "martes", ...]*

catalogo\_pdf VARCHAR(255),

catalogo\_fecha\_subida DATETIME,

cuentas\_pago JSON,

*-- Ejemplo: {"yape": "999888777", "plin": "999888777", "bcp": "123-456789-0-12"}*

logo\_url VARCHAR(255),

FOREIGN KEY (empresa\_id) REFERENCES empresas(id)

);

**Upload de catálogo:**

php

*// api/v1/cliente/negocio/subir-catalogo.php*

$uploadDir = \_\_DIR\_\_ . '/../../../../../public/uploads/catalogos/';

$filename = $empresa\_id . '\_' . time() . '.pdf';

*// Validaciones:*

*// - Solo PDF*

*// - Máximo 5MB (plan básico) o 20MB (premium)*

*// - Eliminar catálogo anterior*

**Gestión de cuentas de pago:**

javascript

*// Frontend - agregar cuenta*

function agregarCuenta() {

const tipo = $('#tipo\_cuenta').val();

const numero = $('#numero\_cuenta').val();

cuentas[tipo] = numero;

actualizarVistaCuentas();

}

**🔧 Módulo 4: Bot IA - Agente de Ventas**

**Archivos involucrados:**

* app/cliente/modulos/bot-config.php - Configuración
* app/api/v1/cliente/bot/configurar.php - API config
* whatsapp-service/src/botAgent.js - Lógica principal
* whatsapp-service/src/catalogReader.js - Lector PDF

**Base de datos:**

sql

CREATE TABLE configuracion\_bot (

id INT PRIMARY KEY AUTO\_INCREMENT,

empresa\_id INT UNIQUE NOT NULL,

activo BOOLEAN DEFAULT 1,

saludo\_inicial TEXT,

personalidad TEXT DEFAULT 'amigable y profesional',

delay\_respuesta INT DEFAULT 3, *-- segundos*

horario\_atencion BOOLEAN DEFAULT 1,

mensaje\_fuera\_horario TEXT,

promociones JSON,

*-- Ejemplo: [{"titulo": "2x1 Pizzas", "descripcion": "Todos los martes", "activa": true}]*

palabras\_escalamiento JSON DEFAULT '["reclamo", "queja", "problema", "ya pagué", "deposité"]',

FOREIGN KEY (empresa\_id) REFERENCES empresas(id)

);

CREATE TABLE conversaciones\_bot (

id INT PRIMARY KEY AUTO\_INCREMENT,

empresa\_id INT NOT NULL,

numero\_cliente VARCHAR(20),

nombre\_cliente VARCHAR(100),

mensaje\_cliente TEXT,

respuesta\_bot TEXT,

tipo\_mensaje ENUM('texto', 'audio', 'imagen'),

audio\_transcripcion TEXT,

productos\_cotizados JSON,

total\_cotizado DECIMAL(10,2),

estado ENUM('normal', 'escalado', 'completado'),

escalado\_razon VARCHAR(100),

fecha\_hora DATETIME DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_empresa\_numero (empresa\_id, numero\_cliente)

);

**Flujo del Bot:**

javascript

*// whatsapp-service/src/botAgent.js*

class BotAgent {

async procesarMensaje(mensaje, empresaId) {

*// 1. Obtener configuración*

const config = await this.getConfiguracion(empresaId);

*// 2. Obtener historial*

const historial = await this.getHistorial(mensaje.from);

*// 3. Leer catálogo si es necesario*

const catalogo = await this.leerCatalogo(empresaId);

*// 4. Generar respuesta con OpenAI*

const respuesta = await this.generarRespuesta(

mensaje.body,

historial,

catalogo,

config

);

*// 5. Detectar si escalar*

if (this.debeEscalar(mensaje.body)) {

await this.escalarAHumano(mensaje.from, empresaId);

}

*// 6. Guardar conversación*

await this.guardarConversacion(*/\*...\*/*);

return respuesta;

}

}

**Integración OpenAI:**

javascript

async generarRespuesta(mensaje, historial, catalogo, config) {

const prompt = `

Eres un agente de ventas ${config.personalidad} para ${config.nombre\_negocio}.

CATÁLOGO:

${catalogo}

PROMOCIONES ACTIVAS:

${config.promociones}

CUENTAS PARA PAGO:

${config.cuentas\_pago}

HISTORIAL:

${historial}

Cliente dice: ${mensaje}

Instrucciones:

- Saluda amablemente si es primera vez

- Cotiza productos del catálogo

- Sugiere promociones cuando sea relevante

- Proporciona cuentas de pago

- Pregunta si desea algo más

- Si detectas confirmación de pago o reclamo, indica que un asesor lo contactará

`;

*// Llamada a OpenAI con API key global*

}

**🔧 Módulo 5: WhatsApp Service Multi-Sesión**

**Archivos involucrados:**

* whatsapp-service/src/sessionManager.js - Gestor de sesiones
* whatsapp-service/src/index.js - Entry point
* whatsapp-service/sessions/ - Carpeta sesiones

**Gestión de sesiones:**

javascript

*// sessionManager.js*

class SessionManager {

constructor() {

this.sessions = new Map();

}

async createSession(empresaId) {

const session = await wppconnect.create({

session: `empresa\_${empresaId}`,

folderNameToken: `./sessions/empresa\_${empresaId}`,

headless: true,

devtools: false,

useChrome: false,

debug: false,

logQR: false,

browserArgs: ['--no-sandbox']

});

this.sessions.set(empresaId, session);

this.setupListeners(session, empresaId);

return session;

}

async getSession(empresaId) {

if (!this.sessions.has(empresaId)) {

return await this.createSession(empresaId);

}

return this.sessions.get(empresaId);

}

}

**API REST del servicio:**

javascript

*// api.js*

app.post('/api/whatsapp/qr', async (req, res) => {

const { empresaId } = req.body;

const session = await sessionManager.getSession(empresaId);

const qr = await session.getQrCode();

res.json({ qr });

});

app.post('/api/whatsapp/send', async (req, res) => {

const { empresaId, numero, mensaje } = req.body;

const session = await sessionManager.getSession(empresaId);

await session.sendText(`${numero}@c.us`, mensaje);

res.json({ success: true });

});

**🔧 Módulo 6: SuperAdmin**

**Archivos involucrados:**

* app/superadmin/modulos/empresas.php - Gestión usuarios
* app/superadmin/modulos/configuracion.php - Config global
* app/api/v1/superadmin/\* - APIs administrativas

**Base de datos:**

sql

CREATE TABLE configuracion\_plataforma (

clave VARCHAR(50) PRIMARY KEY,

valor TEXT,

tipo ENUM('texto', 'numero', 'boolean', 'json'),

descripcion TEXT

);

*-- Configuraciones iniciales*

INSERT INTO configuracion\_plataforma VALUES

('openai\_api\_key', '', 'texto', 'API Key de OpenAI'),

('trial\_dias', '2', 'numero', 'Días de trial'),

('whatsapp\_soporte', '', 'texto', 'WhatsApp de soporte'),

('email\_remitente', 'noreply@mensajeropro.com', 'texto', 'Email remitente'),

('mercadopago\_access\_token', '', 'texto', 'Token MercadoPago'),

('paypal\_client\_id', '', 'texto', 'PayPal Client ID'),

('paypal\_secret', '', 'texto', 'PayPal Secret');

CREATE TABLE plantillas\_email (

id INT PRIMARY KEY AUTO\_INCREMENT,

codigo VARCHAR(50) UNIQUE,

asunto VARCHAR(200),

contenido\_html TEXT,

variables JSON, *-- ["nombre", "empresa", "dias\_restantes"]*

activa BOOLEAN DEFAULT 1

);

**Panel de control:**

php

*// superadmin/modulos/empresas.php*

*// - Lista de todas las empresas*

*// - Suspender/activar*

*// - Cambiar plan*

*// - Extender trial*

*// - Ver logs de actividad*

**🔧 Módulo 7: Sistema de Pagos**

**Archivos involucrados:**

* app/api/v1/webhooks/mercadopago.php - Webhook MP
* app/api/v1/webhooks/paypal.php - Webhook PayPal
* app/api/v1/cliente/pagos/crear.php - Crear pago
* cron/check-payments.php - Verificar vencimientos

**Base de datos:**

sql

CREATE TABLE planes (

id INT PRIMARY KEY AUTO\_INCREMENT,

nombre VARCHAR(50),

precio\_mensual DECIMAL(10,2),

precio\_anual DECIMAL(10,2),

limite\_catalogos INT DEFAULT 1,

limite\_mb\_catalogo INT DEFAULT 5,

bot\_24h BOOLEAN DEFAULT 0,

promociones\_ilimitadas BOOLEAN DEFAULT 0,

soporte\_prioritario BOOLEAN DEFAULT 0,

activo BOOLEAN DEFAULT 1

);

CREATE TABLE suscripciones (

id INT PRIMARY KEY AUTO\_INCREMENT,

empresa\_id INT NOT NULL,

plan\_id INT NOT NULL,

tipo ENUM('mensual', 'anual'),

fecha\_inicio DATE,

fecha\_fin DATE,

estado ENUM('activa', 'cancelada', 'vencida'),

auto\_renovar BOOLEAN DEFAULT 1,

FOREIGN KEY (empresa\_id) REFERENCES empresas(id),

FOREIGN KEY (plan\_id) REFERENCES planes(id)

);

CREATE TABLE pagos (

id INT PRIMARY KEY AUTO\_INCREMENT,

empresa\_id INT NOT NULL,

suscripcion\_id INT,

monto DECIMAL(10,2),

metodo ENUM('mercadopago', 'paypal'),

referencia\_externa VARCHAR(100),

estado ENUM('pendiente', 'aprobado', 'rechazado'),

fecha\_creacion DATETIME DEFAULT CURRENT\_TIMESTAMP,

fecha\_actualizacion DATETIME ON UPDATE CURRENT\_TIMESTAMP,

respuesta\_gateway JSON,

FOREIGN KEY (empresa\_id) REFERENCES empresas(id),

INDEX idx\_referencia (referencia\_externa)

);

**Flujo de pago MercadoPago:**

php

*// api/v1/cliente/pagos/crear.php*

$preference = new MercadoPago\Preference();

$item = new MercadoPago\Item();

$item->title = "Plan " . $plan->nombre;

$item->quantity = 1;

$item->unit\_price = $tipo === 'anual' ? $plan->precio\_anual : $plan->precio\_mensual;

$preference->items = [$item];

$preference->back\_urls = [

"success" => APP\_URL . "/cliente/pago-exitoso",

"failure" => APP\_URL . "/cliente/pago-fallido",

"pending" => APP\_URL . "/cliente/pago-pendiente"

];

$preference->auto\_return = "approved";

$preference->external\_reference = $empresa\_id . "\_" . time();

$preference->save();

**🔧 Módulo 8: Cron Jobs**

**Archivos:**

* cron/check-payments.php - Cada hora
* cron/send-reminders.php - Diario 9 AM
* cron/clean-sessions.php - Diario 3 AM

**Configuración crontab:**

bash

*# Verificar pagos vencidos - cada hora*

0 \* \* \* \* php /var/www/mensajeropro-saas/cron/check-payments.php

*# Enviar recordatorios - diario 9 AM*

0 9 \* \* \* php /var/www/mensajeropro-saas/cron/send-reminders.php

*# Limpiar sesiones viejas - diario 3 AM*

0 3 \* \* \* php /var/www/mensajeropro-saas/cron/clean-sessions.php

**🔒 Seguridad y Mejores Prácticas**

**Headers de seguridad (includes/security.php):**

php

header("X-Content-Type-Options: nosniff");

header("X-Frame-Options: DENY");

header("X-XSS-Protection: 1; mode=block");

header("Strict-Transport-Security: max-age=31536000");

header("Content-Security-Policy: default-src 'self'");

**Validaciones obligatorias:**

1. **SQL Injection**: Usar prepared statements SIEMPRE
2. **XSS**: htmlspecialchars() en todas las salidas
3. **CSRF**: Token en todos los formularios
4. **Sesiones**: Regenerar ID después de login
5. **Archivos**: Validar tipo MIME real, no solo extensión

**Multi-empresa en consultas:**

php

*// NUNCA hacer:*

$sql = "SELECT \* FROM contactos WHERE id = " . $\_GET['id'];

*// SIEMPRE hacer:*

$sql = "SELECT \* FROM contactos WHERE id = ? AND empresa\_id = ?";

$stmt->execute([$\_GET['id'], getEmpresaActual()]);

**🚀 Guía de Implementación por Fases**

**Fase 1 (Semana 1-2): MVP Base**

1. Estructura de carpetas
2. Base de datos completa
3. Landing + Registro (email + Google)
4. Panel cliente básico
5. WhatsApp multi-sesión
6. Bot básico con catálogo PDF

**Fase 2 (Semana 3): Pagos**

1. Integración MercadoPago
2. Integración PayPal
3. Gestión de suscripciones
4. Emails transaccionales

**Fase 3 (Semana 4): SuperAdmin**

1. Panel SuperAdmin completo
2. Configuración dinámica
3. Gestión de usuarios
4. Plantillas de email

**Fase 4 (Semana 5): Optimización**

1. Cron jobs
2. Limpieza de código
3. Testing completo
4. Documentación API

**📝 Checklist de Desarrollo**

**Para cada módulo nuevo:**

* Crear estructura de archivos según esta guía
* Implementar validación multi-empresa
* Agregar logs de auditoría
* Validar permisos y seguridad
* Crear APIs con response.php
* Documentar endpoints
* Probar en ambiente local
* Verificar responsive en móvil

**Antes de producción:**

* Cambiar credenciales de desarrollo
* Configurar SSL
* Optimizar base de datos (índices)
* Configurar backups automáticos
* Monitoreo de errores
* Rate limiting en APIs
* CDN para assets

Esta documentación debe actualizarse con cada cambio significativo. Versión: 2.0.0 SaaS