

Integraciones Externas - Parte 2B

Índice

1. [Visión General](#)
 2. [Integraciones Disponibles](#)
 3. [Arquitectura de Integración](#)
 4. [Flujos de Datos](#)
 5. [Configuración General](#)
 6. [Seguridad y Mejores Prácticas](#)
-

Visión General

La Parte 2B de la Plataforma de Incentivos implementa integraciones con servicios externos clave que extienden las funcionalidades de la plataforma:

- **WhatsApp Business API:** Notificaciones y comunicación directa con deudores
- **CRM (Salesforce, HubSpot, Zoho):** Sincronización bidireccional de datos de deudores y deudas
- **Mercado Pago:** Procesamiento de pagos en línea y gestión de transacciones

Estas integraciones transforman la plataforma en un ecosistema completo que conecta múltiples sistemas y servicios para maximizar la eficiencia operativa.

Integraciones Disponibles

1. WhatsApp Business API

Propósito: Enviar notificaciones proactivas y automáticas a los deudores por WhatsApp.

Casos de Uso:

- Recordatorios de pago próximos a vencer
- Confirmaciones de acuerdos aceptados
- Notificaciones de pagos recibidos
- Alertas de incentivos disponibles
- Nuevas ofertas de negociación
- Logros y subidas de nivel (gamificación)

Características:

- ☒ Mensajes de texto personalizados
- ☒ Templates predefinidos para cada tipo de notificación
- ☒ Envío masivo de mensajes
- ☒ Integración con sistema de notificaciones existente

Documentación: Ver [CONFIGURACION_WHATSAPP.md](#) (./CONFIGURACION_WHATSAPP.md)

2. Integración con CRM

Propósito: Mantener sincronizados los datos entre la plataforma y sistemas CRM empresariales.






CRMs Soportados:

- Salesforce
- HubSpot
- Zoho CRM

Casos de Uso:

- Importación automática de deudores desde el CRM
- Sincronización bidireccional de información de contacto
- Importación masiva de deudas pendientes
- Actualización de estados de pago en el CRM
- Registro de actividades y comunicaciones
- Creación y seguimiento de acuerdos de pago

Características:

-  Interfaz unificada para múltiples CRMs
-  Sincronización completa e incremental
-  Mapeo automático de campos
-  Operaciones masivas (bulk operations)
-  Detección automática del CRM configurado

Documentación: Ver [CONFIGURACION_CRM.md](#) (./CONFIGURACION_CRM.md)







3. Mercado Pago

Propósito: Procesar pagos de deudas y cuotas de forma segura y automática.

Casos de Uso:

- Generación de links de pago para deudas completas
- Pagos de cuotas de acuerdos de pago
- Procesamiento de webhooks de confirmación
- Gestión de reembolsos
- Reportes de transacciones

Características:

-  Checkout redirect de Mercado Pago
-  Soporte para múltiples métodos de pago
-  Webhooks para confirmaciones automáticas
-  Cálculo y otorgamiento automático de incentivos
-  Modo sandbox para testing
-  Integración con wallet de incentivos

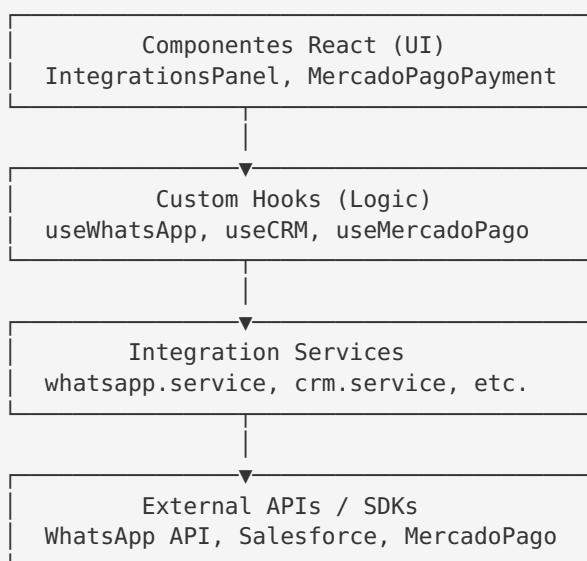
Documentación: Ver [CONFIGURACION_MERCADOPAGO.md](#) (./CONFIGURACION_MERCADOPAGO.md)

Arquitectura de Integración

Principios de Diseño

1. **Modularidad:** Cada integración está aislada en su propio servicio
2. **Abstracción:** Hooks personalizados simplifican el uso desde componentes React
3. **Resiliencia:** Manejo robusto de errores y reintentos
4. **Seguridad:** Credenciales en variables de entorno, nunca en el código
5. **Observabilidad:** Logging completo de operaciones

Capas de la Arquitectura



Estructura de Archivos

```

src/
├── services/
│   └── integrations/
│       ├── whatsapp.service.js
│       ├── mercadopago.service.js
│       └── crm/
│           ├── crm.service.js (genérico)
│           ├── salesforce.service.js
│           ├── hubspot.service.js
│           └── zoho.service.js
├── hooks/
│   └── integrations/
│       ├── useWhatsApp.js
│       ├── useCRM.js
│       ├── useMercadoPago.js
│       └── index.js
└── components/
    └── integrations/
        ├── IntegrationsPanel.jsx
        ├── MercadoPagoPayment.jsx
        ├── CRMSyncStatus.jsx
        ├── WhatsAppNotificationSettings.jsx
        └── index.js
  
```

Flujos de Datos

Flujo 1: Notificación WhatsApp al Pagar una Deuda

```
[Usuario paga deuda]
  → [Mercado Pago webhook]
  → [Actualiza deuda en BD]
  → [Otorga incentivo]
  → [Envía confirmación WhatsApp]
  → [Registra actividad en CRM]
```

Flujo 2: Sincronización con CRM

```
[Admin inicia sync]
  → [Obtiene deudores del CRM]
  → [Mapea campos al formato de plataforma]
  → [Guarda en Supabase]
  → [Obtiene deudas del CRM]
  → [Actualiza estados locales]
```

Flujo 3: Procesamiento de Pago

```
[Deudor selecciona pagar]
  → [Genera preferencia en Mercado Pago]
  → [Redirige a checkout]
  → [Usuario completa pago]
  → [Webhook de confirmación]
  → [Actualiza deuda + wallet]
  → [Notifica al usuario]
```

Configuración General

1. Variables de Entorno

Todas las integraciones se configuran mediante variables de entorno en el archivo `.env` :

```
# WhatsApp
VITE_WHATSAPP_ACCESS_TOKEN=tu-token
VITE_WHATSAPP_PHONE_NUMBER_ID=tu-phone-id
VITE_WHATSAPP_BUSINESS_ACCOUNT_ID=tu-account-id

# Salesforce
VITE_SALESFORCE_ACCESS_TOKEN=tu-token
VITE_SALESFORCE_INSTANCE_URL=https://yourinstance.salesforce.com

# HubSpot
VITE_HUBSPOT_ACCESS_TOKEN=tu-token

# Zoho
VITE_ZOHO_ACCESS_TOKEN=tu-token
VITE_ZOHO_API_DOMAIN=https://www.zohoapis.com

# Mercado Pago
VITE_MERCADOPAGO_ACCESS_TOKEN=tu-token
VITE_MERCADOPAGO_PUBLIC_KEY=tu-public-key
VITE_MERCADOPAGO_SANDBOX=true
```

2. Verificación de Configuración

Cada servicio proporciona un método `isConfigured()` que verifica si las credenciales están presentes:

```
import whatsappService from './services/integrations/whatsapp.service';

const status = whatsappService.isConfigured();
console.log(status.message); // "WhatsApp configurado" o "Faltan credenciales"
```

3. Panel de Administración

El componente `IntegrationsPanel` muestra el estado de todas las integraciones:

```
import { IntegrationsPanel } from './components/integrations';

function AdminPage() {
  return (
    <div>
      <h1>Configuración del Sistema</h1>
      <IntegrationsPanel />
    </div>
  );
}
```

Seguridad y Mejores Prácticas

1. Gestión de Credenciales

✗ NUNCA hagas esto:

```
const token = "mi-token-secreto-123"; // ¡NO!
```

✓ SIEMPRE usa variables de entorno:

```
const token = import.meta.env.VITE_WHATSAPP_ACCESS_TOKEN;
```

2. Validación de Datos

Siempre valida los datos antes de enviarlos a APIs externas:

```
// Validar número de teléfono
const phoneNumber = input.replace(/\D/g, ''); // Solo dígitos
if (phoneNumber.length < 10) {
  throw new Error('Número de teléfono inválido');
}
```

3. Manejo de Errores

Implementa manejo robusto de errores:

```
try {
  const result = await whatsappService.sendMessage(phone, message);
  if (!result.success) {
    console.error('Error:', result.error);
    // Notificar al usuario
  }
} catch (error) {
  console.error('Error crítico:', error);
  // Registrar en sistema de monitoreo
}
```

4. Rate Limiting

Implementa límites de tasa para evitar abusos:

```
// Ejemplo: Limitar envíos de WhatsApp
const sendWithRateLimit = async (phone, message) => {
  const lastSent = localStorage.getItem(`last_whatsapp_${phone}`);
  const now = Date.now();

  if (lastSent && now - lastSent < 60000) { // 1 minuto
    throw new Error('Espera 1 minuto antes de enviar otro mensaje');
  }

  await whatsappService.sendMessage(phone, message);
  localStorage.setItem(`last_whatsapp_${phone}`, now);
};
```

5. Logging y Auditoría

Registra todas las operaciones importantes:

```
console.log(✅ WhatsApp enviado a ${phone}:`, response.data);
console.log(📦 Sincronización CRM: ${result.summary.debtors} deudores`);
console.log(💰 Pago procesado: ${amount}`);
```

6. Testing

Usa los modos sandbox/test antes de producción:

```
// Mercado Pago  
VITE_MERCADOPAGO_SANDBOX=true // Usar credenciales de test  
  
// WhatsApp  
// Usa el número de prueba de Meta para testing
```

Próximos Pasos

1. **Configura tus credenciales:** Lee las guías de configuración específicas
2. **Prueba cada integración:** Usa los modos de prueba/sandbox
3. **Revisa logs:** Asegúrate de que todo funcione correctamente
4. **Pasa a producción:** Cambia a credenciales de producción

Recursos Adicionales

- [Configuración WhatsApp](#) (./CONFIGURACION_WHATSAPP.md)
- [Configuración CRM](#) (./CONFIGURACION_CRM.md)
- [Configuración Mercado Pago](#) (./CONFIGURACION_MERCADOPAGO.md)
- [Manual Técnico](#) (./MANUAL_TECNICO_INTEGRACIONES.md)
- [Guía de Despliegue](#) (./GUIA_DESPLIEGUE.md)

Soporte

Si encuentras problemas:

1. Revisa los logs en la consola del navegador
2. Verifica que las credenciales estén correctamente configuradas
3. Consulta la documentación oficial de cada servicio
4. Revisa los ejemplos de código en este repositorio

Versión: 1.0.0

Última actualización: Octubre 2025