

Sprint 2 - Integración WhatsApp Evolution API

Resumen Ejecutivo

El Sprint 2 implementa una integración completa con WhatsApp Evolution API, permitiendo el envío automático de notificaciones y recordatorios de citas a los clientes de forma totalmente automatizada y configurable.








Versión: v1.9.0

Fecha: Octubre 15, 2025







Estado:  Completado

Objetivos Cumplidos








Fase 1: Configuración Base

-  Modelo de datos para configuración de WhatsApp
-  Sistema de encriptación de API Keys
-  Panel de administración en `/dashboard/settings/whatsapp`
-  API endpoints para CRUD de configuración
-  Servicio de conexión con Evolution API
-  Sistema de logs detallado
-  Prueba de conexión y validación

Fase 2: Notificaciones de Citas

-  Notificación automática al crear cita
-  Notificación al modificar cita
-  Notificación al cancelar cita
-  Plantillas personalizables con variables dinámicas
-  Seed script con plantillas predeterminadas en español
-  Integración transparente en hooks de citas

Fase 3: Recordatorios Automáticos

-  Sistema de cron jobs para recordatorios programados
 -  Recordatorio 24 horas antes
 -  Recordatorio 1 hora antes
 -  Configuración de tiempos personalizables
 -  API endpoint `/api/cron/send-reminders`
 -  Tabla de tracking de recordatorios
 -  Prevención de duplicados
-

Estructura de Archivos Creados

Base de Datos (Prisma)

```

app/prisma/
├── schema.prisma                # Modelos actualizados
├── migrations/
│   └── 20251015_whatsapp_integration/
│       └── migration.sql        # Migración SQL completa
├── seeds/
│   └── whatsapp-templates.ts    # Seed de plantillas

```

Servicios

```

app/lib/services/
├── whatsappService.ts           # Servicio principal de WhatsApp
├── reminderService.ts           # Servicio de recordatorios
└── whatsappNotificationHelper.ts # Helper no bloqueante

```

API Endpoints

```

app/api/
├── whatsapp/
│   ├── config/route.ts          # CRUD configuración
│   ├── test-connection/route.ts # Prueba de conexión
│   ├── logs/route.ts           # Historial de mensajes
│   ├── send/route.ts           # Envío manual
│   └── templates/route.ts       # CRUD plantillas
├── cron/
│   └── send-reminders/route.ts  # Cron job de recordatorios

```

UI (Dashboard)





```

app/dashboard/settings/whatsapp/
├── page.tsx                    # Página principal
├── components/
│   ├── WhatsAppConfigPanel.tsx # Panel de configuración
│   ├── MessageTemplatesPanel.tsx # Gestión de plantillas
│   ├── MessageLogsPanel.tsx     # Logs de mensajes
│   └── ReminderStatsPanel.tsx   # Estadísticas

```

Modificaciones en Endpoints Existentes

```

app/api/calendar/appointments/
├──  route.ts                 + Notificación al crear
└──  [id]/reschedule/route.ts  + Notificación al modificar

```

Modelos de Base de Datos

1. WhatsAppConfig

Configuración de Evolution API por sucursal o general.

```
model WhatsAppConfig {
  id                String @id @default(cuid())
  apiUrl            String
  apiKey            String @db.Text (encrypted)
  instanceName      String
  phoneNumber       String
  isActive          Boolean @default(true)
  isDefault         Boolean @default(false)

  // Opciones
  sendOnCreate      Boolean @default(true)
  sendOnUpdate      Boolean @default(true)
  sendOnCancel      Boolean @default(true)
  sendReminder24h   Boolean @default(true)
  sendReminder1h    Boolean @default(true)

  tenantId          String
  branchId          String? // null = configuración general

  // Relations
  whatsappLogs      WhatsAppLog[]
  reminderLogs      ReminderLog[]
}
```

2. WhatsAppLog

Registro de todos los mensajes enviados.

```
model WhatsAppLog {
  id                String @id @default(cuid())
  configId          String
  appointmentId     String?
  messageType       String
  recipient          String
  message            String @db.Text
  status            WhatsAppLogStatus @default(PENDING)
  response           String? @db.Text
  error              String? @db.Text
  sentAt            DateTime?
}
```

3. MessageTemplate

Plantillas personalizables de mensajes.

```

model MessageTemplate {
  id          String @id @default(cuid())
  name        String
  type        MessageType
  content     String @db.Text
  variables   Json?
  isActive    Boolean @default(true)
  isDefault   Boolean @default(false)
  tenantId    String
  branchId    String?
}

```

4. ReminderLog

Tracking de recordatorios enviados.

```

model ReminderLog {
  id          String @id @default(cuid())
  configId    String
  appointmentId String
  reminderType ReminderType
  sentAt      DateTime
  status      WhatsAppLogStatus
  response    String? @db.Text
  error       String? @db.Text
}

```



API Endpoints

Configuración

POST /api/whatsapp/config

Crear nueva configuración de WhatsApp.

Request:

```

{
  "apiUrl": "https://api.example.com",
  "apiKey": "your-api-key",
  "instanceName": "my-instance",
  "phoneNumber": "521234567890",
  "sendOnCreate": true,
  "sendOnUpdate": true,
  "sendOnCancel": true,
  "sendReminder24h": true,
  "sendReminder1h": true,
  "branchId": "optional-branch-id"
}

```

Response:

```
{
  "success": true,
  "message": "Configuración creada exitosamente",
  "data": { .. }
}
```

GET /api/whatsapp/config

Obtener configuraciones del tenant.

Query Params:

- `branchId` (opcional): Filtrar por sucursal

PUT /api/whatsapp/config

Actualizar configuración existente.

DELETE /api/whatsapp/config?id={configId}

Eliminar configuración.

Prueba de Conexión

POST /api/whatsapp/test-connection

Validar conexión con Evolution API.

Request:

```
{
  "configId": "optional-config-id",
  "branchId": "optional-branch-id"
}
```

Logs

GET /api/whatsapp/logs

Obtener historial de mensajes enviados.

Query Params:

- `configId` (opcional)
- `appointmentId` (opcional)
- `status` (opcional): PENDING, SENT, DELIVERED, FAILED, READ
- `limit` (opcional): Default 50
- `offset` (opcional): Default 0

Envío Manual

POST /api/whatsapp/send

Enviar mensaje manual.

Request:

```
{
  "recipient": "521234567890",
  "message": "Tu mensaje aquí",
  "configId": "optional-config-id",
  "appointmentId": "optional-appointment-id"
}
```

Plantillas

GET /api/whatsapp/templates

Obtener plantillas del tenant.

POST /api/whatsapp/templates

Crear nueva plantilla.

PUT /api/whatsapp/templates

Actualizar plantilla.

DELETE /api/whatsapp/templates?id={templateId}

Eliminar plantilla.

Cron Job

GET /api/cron/send-reminders

Enviar todos los recordatorios programados (24h y 1h).

Headers:

Authorization: Bearer {CRON_SECRET}

Response:

```
{
  "success": true,
  "data": {
    "reminders24h": {
      "total": 10,
      "sent": 9,
      "failed": 0,
      "skipped": 1
    },
    "reminders1h": {
      "total": 5,
      "sent": 5,
      "failed": 0,
      "skipped": 0
    },
    "duration": 1234
  }
}
```



Plantillas Predeterminadas


Variables Disponibles


- {cliente} - Nombre completo del cliente
- {servicio} - Nombre del servicio
- {fecha} - Fecha completa en español
- {hora} - Hora de la cita
- {profesional} - Nombre del profesional
- {sucursal} - Nombre de la sucursal
- {direccion} - Dirección de la sucursal
- {telefono} - Teléfono de la sucursal
- {precio} - Precio del servicio
- {duracion} - Duración del servicio


1. Confirmación de Cita


¡Hola {cliente}! ✨


Tu cita ha sido confirmada exitosamente:


 Fecha: {fecha}

 Hora: {hora}

 Servicio: {servicio}

 Profesional: {profesional}

 Sucursal: {sucursal}

 Precio: {precio}

Dirección: {direccion}

Teléfono: {telefono}


¡Te esperamos!


2. Modificación de Cita


Hola {cliente} ✎


Tu cita ha sido modificada:

 Nueva Fecha: {fecha}

 Nueva Hora: {hora}

 Servicio: {servicio}

 Profesional: {profesional}

 Sucursal: {sucursal}

Dirección: {direccion}

3. Cancelación de Cita

Hola {cliente} ❌

Tu cita del {fecha} a las {hora} ha sido cancelada.

Si deseas reagendar, por favor contáctanos:

- 📍 Sucursal: {sucursal}
- ☎️ Teléfono: {telefono}

4. Recordatorio 24 Horas

¡Hola {cliente}! 🔔

Te recordamos que mañana tienes una cita:

- 📅 Fecha: {fecha}
- 🕒 Hora: {hora}
- 👤 Servicio: {servicio}
- 👤 Profesional: {profesional}
- 📍 Sucursal: {sucursal}

¡Te esperamos! 😊

5. Recordatorio 1 Hora

¡Hola {cliente}! ⌚

Tu cita con {profesional} es en 1 hora:

- 🕒 Hora: {hora}
- 📍 Sucursal: {sucursal}
- 🏠 Dirección: {direccion}

¡Nos vemos pronto! 🙌

🔒 Seguridad

Encriptación de API Keys

Las API Keys se encriptan antes de guardar en la base de datos usando AES-256-CBC.

Variables de entorno requeridas:

```
WHATSAPP_ENCRYPTION_KEY=your-32-character-encryption-key
CRON_SECRET=your-cron-secret-key
```

Autenticación de Cron Jobs

El endpoint de cron requiere un token secreto en el header Authorization:

```
Authorization: Bearer {CRON_SECRET}
```

Configuración de Cron Jobs en Easypanel

1. Crear Cron Job

En Easypanel, configura un cron job que ejecute cada 15 minutos:

```
*/15 * * * * curl -X GET \  
-H "Authorization: Bearer ${CRON_SECRET}" \  
https://citaplanner.com/api/cron/send-reminders
```

2. Variables de Entorno

Asegúrate de configurar en Easypanel:

- `CRON_SECRET` - Token secreto para autenticar cron jobs
- `WHATSAPP_ENCRYPTION_KEY` - Clave de encriptación (32 caracteres)



Uso

1. Configurar WhatsApp

1. Accede a `/dashboard/settings/whatsapp`
2. Completa el formulario con tus credenciales de Evolution API
3. Configura las opciones de notificación
4. Prueba la conexión
5. Guarda la configuración

2. Personalizar Plantillas

1. Ve a la pestaña "Plantillas"
2. Edita las plantillas predeterminadas
3. Agrega variables personalizadas
4. Activa/desactiva plantillas según necesites

3. Monitorear Mensajes

1. Pestaña "Logs" - Ver historial completo
 2. Pestaña "Estadísticas" - Métricas de recordatorios
 3. Filtrar por estado, fecha, sucursal, etc.
-

Testing

Probar Envío Manual

```
curl -X POST https://citaplanner.com/api/whatsapp/send \
-H "Content-Type: application/json" \
-d '{
  "recipient": "521234567890",
  "message": "Mensaje de prueba"
}'
```

Ejecutar Recordatorios Manualmente

```
curl -X GET https://citaplanner.com/api/cron/send-reminders \
-H "Authorization: Bearer ${CRON_SECRET}"
```

Sembrar Plantillas

```
cd app
npm run ts-node -r tsconfig-paths/register prisma/seeds/whatsapp-templates.ts
```

Métricas y Logs

Logs del Sistema

Todos los intentos de envío se registran en `WhatsAppLog` :

- Estado del mensaje (PENDING, SENT, DELIVERED, FAILED, READ)
- Respuesta de la API
- Mensajes de error
- Timestamp de envío

Estadísticas de Recordatorios

Los recordatorios se rastrean en `ReminderLog` :

- Tipo de recordatorio (24h, 1h)
- Estado de envío
- Prevención de duplicados

Troubleshooting

Error: “WhatsApp not configured”

- Verifica que existe una configuración activa para el tenant/sucursal
- Revisa que la configuración tenga `isActive: true`

Error: “Connection validation failed”

- Verifica que la URL de Evolution API sea correcta
- Confirma que la API Key sea válida

- Asegúrate de que la instancia esté activa en Evolution API

Mensajes no se envían

- Revisa los logs en `/api/whatsapp/logs`
- Verifica que las opciones de notificación estén activadas
- Confirma que el teléfono del cliente tenga formato correcto (con código de país)



Flujo de Notificaciones

Creación de Cita

1. Usuario crea cita `POST /api/calendar/appointments`
2. Cita guardada en BD
3. Helper envía notificación (async) `sendAppointmentNotification()`
4. WhatsappService obtiene config y plantilla
5. Procesa variables dinámicas
6. Envía a Evolution API
7. Registra en WhatsAppLog
8. Usuario recibe confirmación instantánea

Recordatorios Automáticos

1. Cron ejecuta cada 15min `GET /api/cron/send-reminders`
2. `ReminderService` busca citas próximas
3. Filtra citas sin recordatorio enviado
4. Para cada cita:
 - a. Obtiene config de WhatsApp
 - b. Obtiene plantilla de recordatorio
 - c. Procesa variables
 - d. Envía mensaje
 - e. Registra en `ReminderLog`
5. Retorna estadísticas del batch



Notas Técnicas

Arquitectura

- **Envío asíncrono:** Las notificaciones no bloquean las respuestas de la API
- **Configuración por sucursal:** Cada sucursal puede tener su propia configuración
- **Fallback a configuración general:** Si no hay config de sucursal, usa la del tenant
- **Prevención de duplicados:** `ReminderLog` previene envíos múltiples del mismo recordatorio

Performance

- Delay de 1 segundo entre mensajes en batch (evitar rate limiting)
- Logs optimizados con índices en campos críticos
- Encriptación eficiente con AES-256

Escalabilidad

- Soporte multi-tenant completo

- Configuración flexible por sucursal
 - Plantillas personalizables por tenant
 - Sistema modular y extensible
-

Checklist de Deployment

- ☐ Configurar variables de entorno en Easypanel
 - ☐ Ejecutar migración de base de datos
 - ☐ Sembrar plantillas predeterminadas
 - ☐ Configurar cron job en Easypanel
 - ☐ Probar conexión con Evolution API
 - ☐ Enviar mensaje de prueba
 - ☐ Verificar logs y estadísticas
 - ☐ Documentar credenciales de Evolution API
-

Soporte

Para preguntas o issues relacionados con la integración de WhatsApp:

1. Revisa esta documentación
 2. Consulta los logs en el dashboard
 3. Verifica la configuración de Evolution API
 4. Contacta al equipo de desarrollo
-

Desarrollado por: Equipo CitaPlanner

Versión: v1.9.0

Última actualización: Octubre 15, 2025