





Integración de Chatwoot con Sistema de Notificaciones



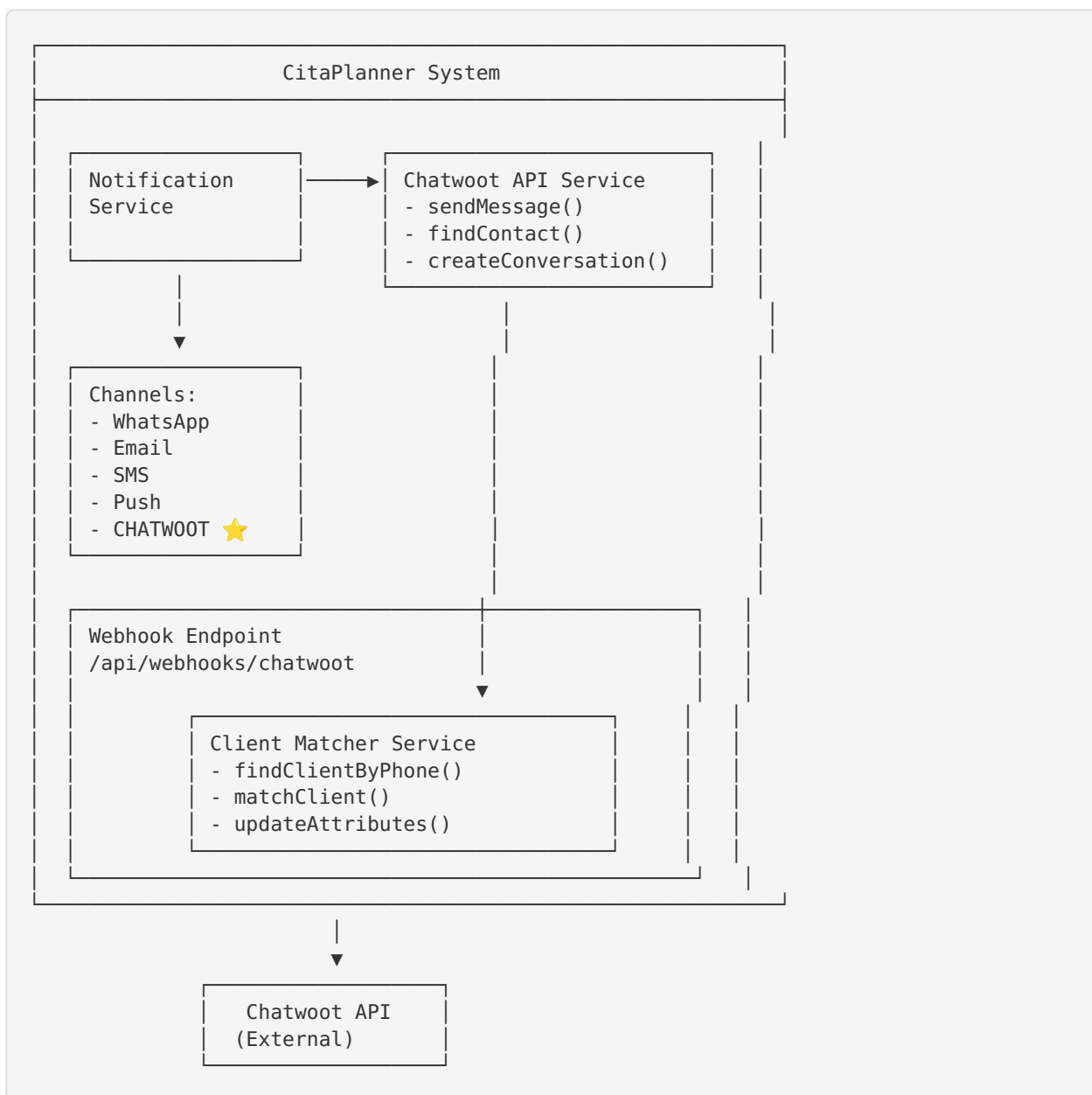
Resumen

Esta implementación integra **Chatwoot** de forma bidireccional con el sistema de notificaciones de CitaPlanner, permitiendo:

1.  **Envío de notificaciones** (citas, recordatorios, marketing) vía Chatwoot
 2.  **Detección automática de clientes** cuando interactúan por Chatwoot
 3.  **Vinculación automática** basada en número de teléfono
 4.  **Sincronización de datos** entre CitaPlanner y Chatwoot
-

Arquitectura

Componentes Implementados



Archivos Creados/Modificados

Nuevos Archivos

1. **`app/lib/chatwoot/api.ts`**
 - Servicio para interactuar con la API de Chatwoot
 - Métodos para enviar mensajes, crear contactos, gestionar conversaciones
2. **`app/lib/chatwoot/client-matcher.ts`**
 - Lógica de detección automática de clientes
 - Normalización de números telefónicos
 - Actualización de atributos en Chatwoot

3. `app/api/webhooks/chatwoot/route.ts`
 - Endpoint webhook para recibir eventos de Chatwoot
 - Procesamiento de mensajes entrantes
 - Detección y vinculación automática de clientes
4. `prisma/migrations/XXXXXX_add_chatwoot_integration/migration.sql`
 - Migración de base de datos con los nuevos campos

Archivos Modificados

1. `prisma/schema.prisma`
 - ☒ Agregado campo `chatwootContactId` en modelo `Client`
 - ☒ Agregados campos de API en modelo `ChatwootConfig`
 - ☒ Agregado canal `CHATWOOT` en enum `NotificationChannel`
 - ☒ Agregados campos de configuración en `NotificationSettings`
2. `app/lib/services/notificationService.ts`
 - ☒ Agregado soporte para canal `CHATWOOT`
 - ☒ Implementado método `sendChatwoot()`
3. `app/.env.example`
 - ☒ Agregadas variables de entorno para API de Chatwoot

Configuración

1. Variables de Entorno

Agregar al archivo `.env` :

```
# Chatwoot Configuration
# Widget Configuration (Frontend)
NEXT_PUBLIC_CHATWOOT_WEBSITE_TOKEN="your-chatwoot-website-token"
NEXT_PUBLIC_CHATWOOT_BASE_URL="https://app.chatwoot.com"

# API Configuration (Backend - for sending messages)
CHATWOOT_API_URL="https://app.chatwoot.com"
CHATWOOT_API_ACCESS_TOKEN="your-api-access-token"
CHATWOOT_ACCOUNT_ID="your-account-id"
CHATWOOT_INBOX_ID="your-inbox-id"
```

2. Obtener Credenciales de Chatwoot

API Access Token

1. Ir a Chatwoot → **Settings** → **Integrations** → **API**
2. Crear un nuevo **Access Token**
3. Copiar el token generado

Account ID e Inbox ID

1. En tu instancia de Chatwoot, la URL tendrá este formato:
`https://app.chatwoot.com/app/accounts/{ACCOUNT_ID}/inbox/{INBOX_ID}`
2. Extraer los IDs de la URL

3. Configurar Webhook en Chatwoot

1. Ir a **Settings** → **Integrations** → **Webhooks**
2. Agregar nuevo webhook:
 - **URL:** `https://tu-dominio.com/api/webhooks/chatwoot`
 - **Events:** Seleccionar:
 - ☒ `message_created`
 - ☒ `conversation_created`
 - ☒ `conversation_updated`

4. Migración de Base de Datos

Ejecutar la migración:

```
cd app
npx prisma migrate deploy
```

O si estás en desarrollo:

```
npx prisma migrate dev
```

5. Configuración por Tenant

Opción A: Usar `NotificationSettings` (Recomendado)

```
UPDATE notification_settings
SET
  chatwoot_enabled = true,
  chatwoot_api_url = 'https://app.chatwoot.com',
  chatwoot_api_token = 'tu-token-aqui',
  chatwoot_account_id = 'tu-account-id',
  chatwoot_inbox_id = 'tu-inbox-id'
WHERE tenant_id = 'tu-tenant-id';
```

Opción B: Usar `ChatwootConfig`

```
UPDATE chatwoot_configs
SET
  enable_notifications = true,
  api_access_token = 'tu-token-aqui',
  account_id = 'tu-account-id',
  inbox_id = 'tu-inbox-id'
WHERE tenant_id = 'tu-tenant-id';
```



Uso

Enviar Notificaciones por Chatwoot

```
import { notificationService } from '@lib/services/notificationService';
import { NotificationChannel, NotificationType } from '@prisma/client';

// Enviar notificación de cita por Chatwoot
await notificationService.sendNotification({
  type: NotificationType.APPOINTMENT_REMINDER,
  channel: NotificationChannel.CHATWOOT,
  recipientId: 'client-id',
  message: 'Hola! Te recordamos tu cita mañana a las 10:00 AM',
  variables: {
    clientName: 'Juan Pérez',
    appointmentDate: '2024-11-13',
    appointmentTime: '10:00',
  },
});
```

Verificar Estado del Webhook

```
curl https://tu-dominio.com/api/webhooks/chatwoot
```

Respuesta esperada:

```
{
  "success": true,
  "message": "Webhook de Chatwoot activo",
  "timestamp": "2024-11-12T16:30:00.000Z"
}
```

Sincronizar Clientes Existentes con Chatwoot

```
import { chatwootClientMatcher } from '@lib/chatwoot/client-matcher';

// Sincronizar todos los clientes de un tenant
const result = await chatwootClientMatcher.syncAllClientsToChatwoot('tenant-id');

console.log(`Sincronizados: ${result.synced} de ${result.total}`);
```



Flujo de Detección Automática

Cuando un cliente escribe por Chatwoot:

1. **Chatwoot envía webhook** → `/api/webhooks/chatwoot`
2. **Sistema extrae** número de teléfono del contacto
3. **Busca cliente** en BD por teléfono (con normalización)

4. Si encuentra cliente:

- ☒ Vincula `chatwootContactId` al cliente
- ☒ Actualiza atributos en Chatwoot:
 - `client_id`
 - `tenant_id`
 - `total_appointments`
 - `last_appointment`
 - etc.

5. Si NO encuentra cliente:



-  Registra que no se encontró (no crea automáticamente por seguridad)

Normalización de Números

El sistema normaliza números telefónicos para matching:

- Elimina espacios, guiones, paréntesis
- Agrega código de país si falta (+52 para México)
- Compara con variaciones del número

Ejemplos:

55 1234 5678		+525512345678
(555) 123-4567		+525551234567
+52 55 1234 5678		+525512345678



Testing

Probar Envío de Mensaje

```
import { chatwootApiService } from '@lib/chatwoot/api';

// Configurar servicio
await chatwootApiService.loadConfigForTenant('tenant-id');

// Enviar mensaje de prueba
const result = await chatwootApiService.sendMessageToContact({
  to: '+525512345678',
  message: 'Mensaje de prueba desde CitaPlanner',
  tenantId: 'tenant-id',
});

console.log(result);
```

Probar Detección de Cliente

```
import { chatwootClientMatcher } from '@lib/chatwoot/client-matcher';

// Buscar cliente por teléfono
const client = await chatwootClientMatcher.findClientByPhone(
  '+525512345678',
  'tenant-id'
);

console.log('Cliente encontrado:', client);
```

Simular Webhook de Chatwoot

```
curl -X POST https://tu-dominio.com/api/webhooks/chatwoot \
-H "Content-Type: application/json" \
-d '{
  "event": "message_created",
  "id": 123,
  "message_type": "incoming",
  "content": "Hola, quiero agendar una cita",
  "conversation": {
    "id": 456,
    "inbox_id": 789,
    "contact": {
      "id": 111,
      "name": "Juan Pérez",
      "phone_number": "+525512345678"
    }
  }
}'
```



Modelo de Datos

Client (actualizado)

```
model Client {
  id String @id @default(cuid())
  firstName String
  lastName String
  phone String
  email String?

  // NUEVO 🌟
  chatwootContactId String? // ID del contacto en Chatwoot

  @@index([chatwootContactId])
  @@unique([phone, tenantId])
}
```

ChatwootConfig (actualizado)

```
model ChatwootConfig {
  id String @id @default(cuid())
  websiteToken String @db.Text
  baseUrl String

  // NUEVOS ⭐
  apiAccessToken String? @db.Text
  accountId String?
  inboxId String?
  enableNotifications Boolean @default(false)
}
```

NotificationSettings (actualizado)

```
model NotificationSettings {
  id String @id @default(cuid())

  // Canales existentes
  whatsappEnabled Boolean @default(false)
  emailEnabled Boolean @default(true)
  smsEnabled Boolean @default(false)
  pushEnabled Boolean @default(false)

  // NUEVOS ⭐
  chatwootEnabled Boolean @default(false)
  chatwootApiUrl String?
  chatwootApiToken String? @db.Text
  chatwootAccountId String?
  chatwootInboxId String?
}
```

Seguridad

Creación Automática de Clientes

Por defecto, **la creación automática de clientes está deshabilitada** por seguridad. Solo se vinculan clientes existentes.

Para habilitar creación automática, descomentar código en:

app/lib/chatwoot/client-matcher.ts → método matchClientFromChatwootContact()

Validación de Webhooks

Para producción, se recomienda agregar validación de firma de webhooks:

```
// En app/api/webhooks/chatwoot/route.ts
function validateWebhookSignature(request: NextRequest): boolean {
  const signature = request.headers.get('x-chatwoot-signature');
  // Implementar validación de firma
  return true;
}
```

Troubleshooting

Problema: No se envían mensajes

Solución:

1. Verificar que `chatwootEnabled = true` en configuración
2. Verificar credenciales de API
3. Revisar logs en consola:

```
bash
```

```
# Ver logs de Chatwoot API
```

```
grep "ChatwootAPI" logs/app.log
```

Problema: No se detectan clientes

Solución:

1. Verificar que el webhook esté configurado correctamente
2. Verificar formato del número de teléfono en BD
3. Revisar logs del webhook:

```
bash
```

```
grep "ChatwootWebhook" logs/app.log
```

Problema: Migración falla

Solución:

```
# Resetear migración problemática  
npx prisma migrate reset
```

```
# Aplicar nuevamente  
npx prisma migrate dev
```

Próximos Pasos

Mejoras Sugeridas

1. Tabla de Interacciones

- Crear modelo `ChatwootInteraction` para tracking detallado
- Almacenar historial completo de mensajes

2. Automatizaciones

- Auto-respuestas basadas en horarios
- Asignación automática de conversaciones

3. Analytics

- Dashboard de métricas de Chatwoot
- Tiempo de respuesta promedio
- Tasa de conversión






4. Plantillas

- Sistema de plantillas para respuestas rápidas
 - Variables dinámicas
-

Changelog

v1.12.0 - 2024-11-12

🌟 Nuevas Funcionalidades

-  Integración de Chatwoot como canal de notificaciones
-  Detección automática de clientes por número de teléfono
-  Webhook para recibir mensajes de Chatwoot
-  Servicio de API de Chatwoot completo
-  Sincronización bidireccional de datos

Cambios en Base de Datos

- Agregado campo `chatwootContactId` en tabla `clients`
- Agregados campos de API en `chatwoot_configs`
- Agregado canal `CHATWOOT` en enum `NotificationChannel`
- Agregada configuración de Chatwoot en `notification_settings`

Documentación

- Creada guía completa de integración
- Ejemplos de uso y configuración
- Guía de troubleshooting

Contribuidores

- **Equipo CitaPlanner** - Implementación inicial
- **DeepAgent** - Desarrollo y documentación

Licencia

Este código es parte del proyecto CitaPlanner y está sujeto a su licencia.

Soporte

Para soporte o preguntas:

-  Email: soporte@citaplanner.com
 -  Docs: <https://docs.citaplanner.com>
 -  Chat: <https://citaplanner.com/chat>
-

Última actualización: 12 de Noviembre, 2024

Versión: 1.12.0