📝 Instrucciones para Pull Request - Integración de Chatwoot

🔗 Crear Pull Request

Link directo para crear el PR:
https://github.com/qhosting/citaplanner/pull/new/feature/chatwoot-integration

Branch: - **From:** `feature/chatwoot-integration` - **To:** `develop` (o `main` según tu flujo)

📋 Título Sugerido del PR

```
feat: Integrate Chatwoot live chat support (v1.11.1) 💬
```

📄 Descripción Completa del PR

Copia y pega esta descripción en el PR:

```
## 🎯 Resumen

Integración completa de **Chatwoot**, una plataforma de soporte y
        chat en vivo de código abierto, con soporte **multi-
        tenant**, identificación automática de usuarios y atributos
        personalizados.

Esta integración permite que cada tenant (y opcionalmente cada
        sucursal) tenga su propio canal de comunicación con clientes
        directamente desde CitaPlanner.


## ✨ Características Implementadas

### Componentes Frontend

- ✅ **ChatwootWidget** - Widget de chat interactivo
  - Carga dinámica del SDK de Chatwoot
  - Identificación automática de usuarios autenticados
  - Envío de atributos personalizados (tenantId, role, branchId)
  - Configuración personalizable (posición, idioma, colores)
  - Control programático del widget
  - Soporte para usuarios anónimos
```

- ✅ **ChatwootProvider** - Provider para el layout principal
  - Carga configuración desde API
  - Inicialización automática del widget
  - Gestión de sesión y re-identificación

API REST

- ✅ **`/api/chatwoot/config`** - Endpoints CRUD completos
  - `GET` - Obtener configuración del tenant actual
  - `POST` - Crear nueva configuración (solo ADMINs)
  - `PUT` - Actualizar configuración existente
  - `DELETE` - Eliminar configuración
  - Validación de permisos (ADMIN/SUPERADMIN)
  - Sanitización de URLs
  - Validación de datos

Backend y Database

- ✅ **Modelo `ChatwootConfig`** en Prisma
  - `websiteToken` - Token del inbox de Chatwoot
  - `baseUrl` - URL de la instancia de Chatwoot
  - `isActive` - Estado habilitado/deshabilitado
  - `isDefault` - Configuración por defecto del tenant
  - `position` - Posición del widget (left/right)
  - `locale` - Idioma del widget (es, en, fr, etc.)
  - `widgetColor` - Color personalizado
  - `tenantId` - Relación con tenant (CASCADE)
  - `branchId` - Relación opcional con sucursal (CASCADE)

- ✅ **Migración Prisma** `add_chatwoot_integration`
  - Tabla `chatwoot_configs`
  - Índices en `tenantId`, `branchId`, `isActive`
  - Foreign keys con CASCADE delete

Utilidades y Tipos

- ✅ **`lib/chatwoot/types.ts`** - Tipos TypeScript completos
- ✅ **`lib/chatwoot/config.ts`** - Configuración y validaciones
- ✅ **`lib/chatwoot/server.ts`** - Funciones server-side
- ✅ **`lib/chatwoot/index.ts`** - Exportaciones centralizadas

Características Avanzadas

- 🏢 **Multi-tenant**: Cada tenant puede tener su propia instancia/inbox
- 🏪 **Por sucursal**: Configuración opcional específica por branch
- 👤 **Identificación automática**: Usuarios autenticados identificados con email, nombre, avatar

- 📊 **Atributos personalizados**: TenantId, role, branchId enviados automáticamente
- 🎨 **Personalización**: Widget configurable (posición, idioma, colores)
- 🔒 **Seguro**: Validación de permisos y sanitización de datos
- ☁️ **Flexible**: Compatible con Chatwoot Cloud o self-hosted

📁 Archivos Creados

Componentes (3 archivos):

app/components/chatwoot/ ├── ChatwootWidget.tsx (144 líneas) ├── ChatwootProvider.tsx (42 líneas) └── index.ts (2 líneas)

Backend (4 archivos):

app/lib/chatwoot/ ├── types.ts (71 líneas) ├── config.ts (55 líneas) ├── server.ts (61 líneas) └── index.ts (3 líneas)

API (1 archivo):

app/api/chatwoot/ └── config/ └── route.ts (258 líneas)

Database (1 archivo):

app/prisma/migrations/ └── 20251112064144_add_chatwoot_integration/ └── migration.sql

Scripts (1 archivo):

app/scripts/ └── generate-version.sh (actualizado para v1.11.1)

Documentación (2 archivos):

CHATWOOT_INTEGRATION.md (1800+ líneas, 60+ páginas)
CHANGELOG.md (actualizado con v1.11.1)

📝 Archivos Modificados

- ✏️ `app/prisma/schema.prisma` - Modelo `ChatwootConfig` agregado
- ✏️ `app/components/providers.tsx` - `ChatwootProvider` integrado en el layout
- ✏️ `app/.env.example` - Variables de Chatwoot agregadas
- ✏️ `CHANGELOG.md` - Versión 1.11.1 documentada
- ✏️ `DEVELOPMENT_ROADMAP.pdf` - Actualizado con integración de Chatwoot

```
## 📖 Documentación

### `CHATWOOT_INTEGRATION.md` (60+ páginas)

Documentación completa y exhaustiva que incluye:

- 📋 Resumen ejecutivo
- 🎯 Qué es Chatwoot y por qué se integró
- 🏗️ Arquitectura de la integración con diagramas
- ⚙️ Configuración paso a paso:
  - Crear cuenta en Chatwoot (Cloud o self-hosted)
  - Obtener website token
  - Configurar variables de entorno
  - Configurar por tenant en base de datos
- 🔌 Uso completo de la API de configuración
- 🎨 Personalización del widget
- 📊 Atributos personalizados y su uso
- 👤 Identificación de usuarios
- 🔍 Troubleshooting detallado (10+ escenarios)
- 💡 Ejemplos de código (15+ snippets)
- 📸 Diagramas de arquitectura y flujo
- ❓ FAQ completa (15+ preguntas)
- 📚 Referencias y recursos

## 🚀 Instrucciones de Deployment

### 1. Variables de Entorno (Opcionales)

Estas variables son **opcionales** y sirven como configuración
global por defecto:

```env
NEXT_PUBLIC_CHATWOOT_WEBSITE_TOKEN=tu_token_aqui
NEXT_PUBLIC_CHATWOOT_BASE_URL=https://app.chatwoot.com
```

⚠ **Importante**: Para multi-tenant, es mejor usar la base de datos en lugar de variables de entorno.

2. Migración de Base de Datos

La migración se ejecuta automáticamente en el `docker-entrypoint.sh`:

```
npx prisma migrate deploy
```

Verificación manual:

```
-- Conecta a PostgreSQL y ejecuta:
SELECT table_name
FROM information_schema.tables
```
```

```sql
WHERE table_name = 'chatwoot_configs';

-- Debe retornar: chatwoot_configs
```

3. Configurar Primer Tenant

Opción A: Via API (Recomendado)

```
POST /api/chatwoot/config
Authorization: Bearer <token_admin>
Content-Type: application/json

{
  "websiteToken": "TU_WEBSITE_TOKEN",
  "baseUrl": "https://app.chatwoot.com",
  "isActive": true,
  "position": "right",
  "locale": "es"
}
```

Opción B: Directamente en PostgreSQL

```sql
INSERT INTO chatwoot_configs (
  id,
  "websiteToken",
  "baseUrl",
  "isActive",
  "isDefault",
  position,
  locale,
  "tenantId",
  "createdAt",
  "updatedAt"
) VALUES (
  gen_random_uuid(),
  'TU_WEBSITE_TOKEN',
  'https://app.chatwoot.com',
  true,
  true,
  'right',
  'es',
  (SELECT id FROM tenants LIMIT 1),
  NOW(),
  NOW()
);
```

4. Obtener Website Token de Chatwoot

Pasos:

1. Crea cuenta en Chatwoot Cloud o instala self-hosted
2. Crea un Inbox de tipo "Website"
3. Ve a Settings → Inboxes → [Tu Inbox] → Configuration
4. Copia el **Website Token** (similar a `ABC123xyz456`)
5. Copia también la **Base URL** (ej: `https://app.chatwoot.com`)

5. Verificar en Producción

1. Deploy la aplicación con los cambios
2. Inicia sesión con un usuario del tenant configurado
3. Verifica que aparece el widget de chat (esquina inferior derecha)
4. Envía un mensaje de prueba
5. Revisa en Chatwoot que llegó el mensaje con los atributos personalizados

✅ Checklist de Revisión

Antes de aprobar el PR, verificar:

Código

- ☐ Todos los archivos TypeScript compilan sin errores
- ☐ No hay console.logs innecesarios
- ☐ Imports están organizados y son correctos
- ☐ Nombres de variables/funciones son descriptivos
- ☐ Código sigue convenciones del proyecto

Funcionalidad

- ☐ Widget de Chatwoot se carga correctamente
- ☐ Usuarios autenticados son identificados
- ☐ Atributos personalizados se envían correctamente
- ☐ API endpoints responden correctamente (GET/POST/PUT/DELETE)
- ☐ Validación de permisos funciona (solo ADMINs pueden crear/editar)
- ☐ Configuración multi-tenant funciona
- ☐ Widget puede ser personalizado (posición, idioma, colores)

Base de Datos

- ☐ Migración ejecuta correctamente
- ☐ Tabla `chatwoot_configs` se crea con todos los campos
- ☐ Índices están presentes
- ☐ Foreign keys funcionan (tenantId, branchId)

- [ ] CASCADE delete funciona correctamente

Seguridad

- [ ] No hay secrets expuestos en el código
- [ ] Validación de entrada en API endpoints
- [ ] Sanitización de URLs funciona
- [ ] Permisos verificados en todas las rutas
- [ ] HTTPS requerido en producción

Documentación

- [ ] `CHATWOOT_INTEGRATION.md` está completo
- [ ] `CHANGELOG.md` actualizado con v1.11.1
- [ ] Comentarios en código son claros
- [ ] `.env.example` tiene las nuevas variables
- [ ] README actualizado (si aplica)

Testing

- [ ] Probar con usuario ADMIN
- [ ] Probar con usuario normal (USER)
- [ ] Probar sin configuración de Chatwoot
- [ ] Probar con configuración desactivada (`isActive: false`)
- [ ] Probar en diferentes browsers (Chrome, Firefox, Safari)
- [ ] Probar en mobile

🔍 Puntos de Atención

⚠ Cosas a Verificar

1. **Sparse Checkout**: Algunos archivos pueden estar fuera del sparse checkout. Asegurar que todos los archivos necesarios están incluidos.

2. **Variables de Entorno**: Recordar que las variables `NEXT_PUBLIC_CHATWOOT_*` son opcionales y globales. Para multi-tenant, usar la base de datos.

3. **HTTPS Requerido**: Chatwoot requiere HTTPS en producción. No funcionará con HTTP.

4. **CORS**: Asegurar que el dominio de CitaPlanner está permitido en la configuración de Chatwoot.

5. **SDK Externo**: El SDK de Chatwoot se carga desde la URL configurada. Verificar que la URL es accesible.

🎓 Recursos Adicionales

Documentación

- **Completa**: `CHATWOOT_INTEGRATION.md` (en la raíz del proyecto)
- **Changelog**: `CHANGELOG.md` versión 1.11.1
- **Chatwoot oficial**: https://www.chatwoot.com/docs
- **API Reference**: https://www.chatwoot.com/developers/api

Testing

```
# Verificar que la migración está lista
cd app
npx prisma migrate status

# Ver el schema actualizado
npx prisma db pull

# Generar cliente de Prisma
npx prisma generate
```

Ejemplos de Uso

```
// Uso básico en layout
import { ChatwootProvider } from '@/components/chatwoot';

export default function RootLayout({ children }) {
  return (
    <ChatwootProvider>
      {children}
    </ChatwootProvider>
  );
}

// Obtener configuración en API route
import { getChatwootConfig } from '@/lib/chatwoot/server';

const config = await getChatwootConfig(session.user.tenantId);

// Control programático del widget
window.$chatwoot?.toggle('open'); // Abrir
window.$chatwoot?.toggle('close'); // Cerrar
```

📊 Métricas del PR

- **Archivos creados**: 9
- **Archivos modificados**: 5

- **Líneas agregadas**: ~4,400
- **Líneas de documentación**: ~1,800
- **Componentes React**: 2
- **API Endpoints**: 4 (GET, POST, PUT, DELETE)
- **Funciones server-side**: 3
- **Tipos TypeScript**: 5 interfaces
- **Migración Prisma**: 1

🎉 Beneficios

- 📞 **Soporte en tiempo real** para clientes
- 🏢 **Aislamiento por tenant** - cada negocio su canal
- 👤 **Contexto rico** - agentes ven info completa
- 💰 **Open Source** - sin costos de licencia
- 🌐 **Multi-canal** - integra WhatsApp, Facebook, Email
- 📊 **Analytics** - reportes por tenant/sucursal
- 🤖 **Automatización** - bots y respuestas automáticas

🚧 Próximos Pasos (Futuras PRs)

- [ ] Panel de administración en UI para gestionar configuraciones
- [ ] Webhooks para escuchar eventos de Chatwoot
- [ ] Integración con WhatsApp desde CitaPlanner
- [ ] Dashboard de analytics de conversaciones
- [ ] Notificaciones en CitaPlanner cuando hay mensaje nuevo
- [ ] Chat interno entre sucursales usando Chatwoot

👥 Reviewers Sugeridos

- @qhosting (Owner)
- Backend team (API y Prisma)
- Frontend team (React components)
- DevOps team (Deployment y migrations)

✅ Este PR está listo para revisión y merge

Documentación completa en: `CHATWOOT_INTEGRATION.md`

Versión: 1.11.1
Fecha: Noviembre 2024
Branch: `feature/chatwoot-integration`

```
## 🔧 Comandos Útiles

### Para el Reviewer

```bash
# Clonar el branch
git fetch origin
git checkout feature/chatwoot-integration

# Ver cambios
git diff develop...feature/chatwoot-integration

# Ver archivos modificados
git diff --name-status develop...feature/chatwoot-integration

# Contar líneas agregadas/eliminadas
git diff --stat develop...feature/chatwoot-integration
```

Para Testing Local

```bash
# Instalar dependencias
cd app
npm install

# Ejecutar migración
npx prisma migrate dev

# Generar cliente Prisma
npx prisma generate

# Iniciar desarrollo
npm run dev

# Verificar tipos
npx tsc --noEmit
```

Para Deployment

```bash
# Build production
npm run build

# Verificar que todo compila
npm run build 2>&1 | grep -i error

# Ejecutar migraciones en producción
npx prisma migrate deploy
```

```

📞 Contacto

Si hay preguntas sobre esta integración:

1. Revisar `CHATWOOT_INTEGRATION.md` (documentación completa)
2. Revisar el código y comentarios
3. Contactar al autor del PR
4. Consultar documentación oficial de Chatwoot

¡Gracias por revisar este PR! 💬🚀