🚀 INSTRUCCIONES DE DEPLOYMENT - SPRINT 2

✅ ESTADO ACTUAL

Branch: `v2/production-ready-clean`
Commit: `1ee8e11` - Sprint 2: Integración completa Evolution API para WhatsApp
Fecha: 5 de Octubre de 2025

📋 RESUMEN DE LO COMPLETADO

Sprint 1 ✅ (Completado anteriormente)

- APIs de Contactos (CRUD, import/export CSV)
- APIs de Agentes RAG (CRUD, chat con contexto)
- Servicio de Embeddings
- Schema de Prisma con pgvector

Sprint 2 ✅ (RECIÉN COMPLETADO)

- **Servicio Evolution API** (`lib/services/evolution-api.ts`)
- **Webhook WhatsApp** (`app/api/webhooks/whatsapp/route.ts`)
- **API Gestión de Instancias** (`app/api/whatsapp/instance/route.ts`)
- **API Envío de Mensajes** (`app/api/whatsapp/send/route.ts`)
- **API Estado de Conexión** (`app/api/whatsapp/status/route.ts`)
- **Schema actualizado** con campos para Evolution API

🔧 PASOS PARA DEPLOYMENT EN PRODUCCIÓN

1. Conectar al Servidor ⚠️ IMPORTANTE

```
ssh root@89.116.73.62
cd /srv/comodin_ia/comodin_ia
```

2. Hacer Pull del Código

```
git fetch origin
git checkout v2/production-ready-clean
git pull origin v2/production-ready-clean
```

3. Configurar Variables de Entorno

```
nano /srv/comodin_ia/comodin_ia/app/.env
```

Agregar/verificar estas variables:

```
# Evolution API (🔴 CRÍTICO - OBTENER DEL DASHBOARD DE EVOLUTION API)
EVOLUTION_API_URL=http://89.116.73.62:8080
EVOLUTION_API_KEY=tu-api-key-aqui  # 🔴 OBTENER DE EVOLUTION API DASHBOARD

# Webhook URL
NEXTAUTH_URL=https://comodinia.com

# Otras variables ya configuradas
DATABASE_URL=postgresql://postgres:22N3m3s1@?123456@db.ovpcxvotqfmiqqrdmloi.supabase.co:5432/postgres
OPENAI_API_KEY=sk-proj-bHE2Dz3dd3CCmsbMXdywsgSIiUju-5sWub6k8pCNxJvICyl-Bd_cPuiUhgJKcE_EuCBleKCq7YVT3BlbkFJ2IKamEOhUXjBmxbZRJJPICrntijt1d3ZmN5t9GWWJBzD2I5IjfmXrw0aPTNg7RI42CdtQPgOgA
STRIPE_SECRET_KEY=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
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_live_51LkF9aHZmGOpXSGYhRwtSFpfyJfbQSBH9ADCV2CNrhVpMtdpEUdUcq32mZ122yqxOGzVNBbLetZv8jp7ubDs92iv00ERgJ5WKA
```

4. Instalar Dependencias Nuevas

```
cd /srv/comodin_ia/comodin_ia/app
yarn install
```

5. Aplicar Migración de Base de Datos 🔴 CRÍTICO

```
cd /srv/comodin_ia/comodin_ia/app

# Verificar conexión a base de datos
yarn prisma db push

# Generar cliente de Prisma
yarn prisma generate
```

⚠️ **IMPORTANTE:** Esta migración agrega los siguientes campos a `WhatsAppChannel`:
- `instanceId` (String? @unique)
- `webhookUrl` (String?)
- `errorMessage` (String? @db.Text)
- Cambio de `qrCode` a tipo `Text` para soportar base64

6. Obtener Evolution API Key 🔴 CRÍTICO

Opción A: Desde el Dashboard de Evolution API

1. Ir a: `http://89.116.73.62:8080`
2. Login con tus credenciales
3. Ir a Settings → API Keys
4. Generar nueva API Key
5. Copiar el key

Opción B: Crear API Key manualmente

```
# Conectar a Evolution API
curl -X POST http://89.116.73.62:8080/manager/generate-apikey \
  -H "Content-Type: application/json" \
  -d '{
    "name": "comodin-ia-production"
  }'
```

Agregar el key al `.env`:

```
EVOLUTION_API_KEY=tu-api-key-generada-aqui
```

7. Build de Producción

```
cd /srv/comodin_ia/comodin_ia/app
yarn build
```

Deberías ver:

```
✓ Compiled successfully
✓ Generating static pages (128/128)
```

8. Reiniciar PM2

```
# Reiniciar la aplicación
pm2 restart comodin-ia

# Ver logs en tiempo real
pm2 logs comodin-ia --lines 100
```

9. Verificar que Todo Funciona

```
# Verificar que la app está corriendo
curl http://localhost:3000 -I

# Verificar logs
pm2 logs comodin-ia --lines 50

# Verificar estado de PM2
pm2 status
```

10. Probar Evolution API desde la App

```
# Probar conexión con Evolution API
curl -X GET http://localhost:3000/api/whatsapp/status \
  -H "Content-Type: application/json" \
  -H "Cookie: tu-session-cookie-aqui"
```

🧪 TESTING POST-DEPLOYMENT

1. Probar Creación de Instancia

```
POST /api/whatsapp/instance
{
  "name": "Canal de Prueba",
  "phoneNumber": "+52 55 1234 5678"
}
```

Debería devolver:
- `success: true`
- `channel: {...}`
- `qrCode: "..."`

2. Escanear QR Code

- Copiar el base64 del QR code
- Convertir a imagen
- Escanear con WhatsApp

3. Probar Envío de Mensaje

```
POST /api/whatsapp/send
{
  "channelId": "clx...",
  "contactId": "clx...",
  "message": "Hola, este es un mensaje de prueba"
}
```

4. Verificar Webhook

- Enviar mensaje desde WhatsApp al número conectado
- Verificar que aparece en la base de datos
- Verificar logs: `pm2 logs comodin-ia`

🔍 TROUBLESHOOTING

Error: “Can’t reach database server”

Solución:

```
# Verificar que DATABASE_URL apunta a Supabase
echo $DATABASE_URL

# Si es localhost, cambiar a:
DATABASE_URL="postgresql://postgres:22N3m3s1@?
123456@db.ovpcxvotqfmiqqrdmloi.supabase.co:5432/postgres"
```

Error: “EVOLUTION_API_KEY no está configurado”

Solución:

1. Obtener API key de Evolution API (ver paso 6)
2. Agregar a `.env`: `EVOLUTION_API_KEY=tu-key`
3. Reiniciar PM2: `pm2 restart comodin-ia`

Error: “Failed to fetch instances”

Solución:

```
# Verificar que Evolution API está corriendo
curl http://89.116.73.62:8080/instance/fetchInstances \
  -H "apikey: tu-api-key-aqui"

# Si no responde, reiniciar Evolution API
docker restart evolution-api  # O el comando que uses
```

Webhook no recibe mensajes

Solución:

1. Verificar que NEXTAUTH_URL está configurado correctamente
2. Verificar que el webhook se configuró en Evolution API
3. Ver logs: `pm2 logs comodin-ia | grep "WhatsApp Webhook"`

📊 ENDPOINTS NUEVOS DISPONIBLES

1. Gestión de Instancias

- `GET /api/whatsapp/instance` - Listar instancias
- `POST /api/whatsapp/instance` - Crear instancia
- `DELETE /api/whatsapp/instance?id={id}` - Eliminar instancia

2. Estado

- `GET /api/whatsapp/status` - Estado de Evolution API
- `GET /api/whatsapp/status?instanceId={id}` - Estado de instancia

3. Mensajes

- `POST /api/whatsapp/send` - Enviar mensaje

4. Webhook

- `POST /api/webhooks/whatsapp` - Recibir eventos de WhatsApp

📝 NOTAS IMPORTANTES

Seguridad

- ✅ Todos los endpoints requieren autenticación
- ✅ Aislamiento por organización (multi-tenant)
- ✅ API key de Evolution API segura en variables de entorno

Performance

- Los mensajes se guardan en base de datos automáticamente
- Las instancias se sincronizan con Evolution API
- Los webhooks procesan eventos en tiempo real

Límites

- Evolution API tiene límites de mensajes por día
- QR codes expiran en 60 segundos
- Las instancias deben mantenerse conectadas

🚀 PRÓXIMOS PASOS (Sprint 3)

Tareas Pendientes:

1. **Configurar Stripe Webhook** (webhook ya existe)
2. **Testing E2E de WhatsApp**
3. **Configurar MercadoPago** (opcional)
4. **Optimizaciones de UI/UX**
5. **Deploy final a producción**

📞 SOPORTE

Si encuentras problemas:

1. **Revisar logs:**

```
pm2 logs comodin-ia --lines 200
```

1. **Revisar estado:**

```
pm2 status
```

1. **Reiniciar aplicación:**

```
pm2 restart comodin-ia
```

1. **Revisar documentación completa:**
   - `SPRINT2_EVOLUTION_API_COMPLETADO.md`
   - `SPRINT1_COMPLETADO.md`

✅ CHECKLIST DE DEPLOYMENT

- [ ] Código actualizado desde GitHub

- [ ] Variables de entorno configuradas
- [ ] Evolution API Key obtenida y configurada
- [ ] Dependencias instaladas (`yarn install`)
- [ ] Migración de base de datos aplicada (`yarn prisma db push`)
- [ ] Cliente de Prisma generado (`yarn prisma generate`)
- [ ] Build de producción exitoso (`yarn build`)
- [ ] PM2 reiniciado (`pm2 restart comodin-ia`)
- [ ] Logs sin errores (`pm2 logs comodin-ia`)
- [ ] Endpoint de status responde correctamente
- [ ] Instancia de WhatsApp creada exitosamente
- [ ] QR code generado y funcional
- [ ] Mensaje enviado y recibido correctamente
- [ ] Webhook recibe eventos de WhatsApp

¡Listo para producción! 🎉

Una vez completado el checklist, el sistema estará completamente funcional con integración de WhatsApp a través de Evolution API.

Desarrollado por: Agente de Diseño y Desarrollo
Fecha: 5 de Octubre de 2025
Versión: Sprint 2 - Evolution API Integration