🚀 Guía de Inicio Rápido - Web Agent N8N

Esta guía te ayudará a poner en marcha tu agente web en aproximadamente **15 minutos**.

📋 Antes de empezar

Asegúrate de tener acceso a:

- [ ] Tu instancia de **n8n**
- [ ] Una base de datos **Redis** funcionando
- [ ] **API key de OpenAI** o **Google AI**
- [ ] (Opcional) Servicios **MCP** para calendario y correo
- [ ] (Opcional) **Google Sheet** para registro de leads

Paso 1: Importar el workflow

1. Abre tu instancia de n8n
2. Ve a **Workflows** en el menú lateral
3. Click en **Import from File**
4. Selecciona el archivo `workflow/AGENTE_WEB_CLEAN_2_.json`
5. El workflow se cargará con todos sus nodos

✅ **Resultado**: Deberías ver el workflow completo con varios módulos organizados (ENTRADA, AGENTE, VERIFICAR RESPUESTA, etc.).

Paso 2: Configurar credenciales básicas

2.1 Redis

1. Ve a **Credentials** (esquina superior derecha)
2. Busca las credenciales de tipo **Redis**
3. Crea una nueva credencial con:
   - **Host**: Nombre o IP de tu contenedor Redis
   - **Port**: `6379` (por defecto)
   - **Password**: Tu contraseña de Redis
   - **Database**: `0`
4. Click en **Test** para verificar la conexión
5. Guarda la credencial

2.2 OpenAI o Google AI

Opción A: OpenAI

1. Crea credencial de tipo **OpenAI**
2. Pega tu API key
3. Guarda

Opción B: Google AI (Gemini)

1. Crea credencial de tipo **Google PaLM API**
2. Pega tu API key
3. Guarda

💡 **Tip**: El workflow usa ambos como backup, pero funciona con solo uno configurado. Gemini es el motor principal, OpenAI es el fallback.

2.3 Google Sheets (Opcional - para captura de leads)

1. En **Credentials**, busca **Google Service Account**
2. Crea una nueva credencial
3. Pega el contenido del archivo JSON de tu service account
4. Guarda la credencial
5. Comparte tu Google Sheet con el email del service account

2.4 MCP Services (Opcional - para calendario y correo)

Para cada servicio MCP:

1. En **Credentials**, busca **HTTP Header Auth**
2. Crea una nueva credencial con:
   - **Name**: `Authorization`
   - **Value**: `Bearer tu_token_mcp`
3. Guarda

Paso 3: Asignar credenciales a los nodos

Ahora necesitas decirle a cada nodo qué credencial usar:

3.1 Nodos de Redis

1. Busca el nodo **"Redis Chat Memory"**
2. Ábrelo y selecciona tu credencial de Redis
3. Guarda

3.2 Nodos de IA

1. Abre el nodo **"Google Gemini Chat Model"**
   - Selecciona tu credencial de Google AI
2. Abre el nodo **"OpenAI Chat Model4"**
   - Selecciona tu credencial de OpenAI
3. Abre el nodo **"OpenAI Chat Model1"** (para formateo)
   - Selecciona tu credencial de OpenAI

3.3 Servicios MCP (si los configuraste)

1. Abre el nodo **"MCP Calendario1"**
   - Endpoint URL: Tu URL del servicio MCP de calendario
   - Selecciona tu credencial HTTP Header Auth
2. Abre el nodo **"MCP Correo1"**
   - Endpoint URL: Tu URL del servicio MCP de correo
   - Selecciona tu credencial HTTP Header Auth

3.4 Google Sheets (si lo configuraste)

1. Abre el nodo **"AddLEADS"**
   - Selecciona tu credencial de Google Service Account
   - **Document ID**: El ID de tu Google Sheet (de la URL)
   - **Sheet Name**: Nombre de la hoja (ej: "Leads")

Paso 4: Obtener URL del webhook

1. Activa el workflow (switch en la parte superior: **Inactive** → **Active**)
2. La URL del webhook se genera automáticamente
3. Será algo como: `https://tu-n8n.com/webhook/AGENTE_WEB_GITHUB`
4. **Copia esta URL** - la necesitarás para integrar en tu web

💡 **Importante**: El workflow debe estar **Active** para que el webhook funcione.

Paso 5: Personalizar el agente (opcional pero recomendado)

1. Busca el nodo llamado **"AI Agent1"**
2. Ábrelo y busca el **System Message** (System Prompt)
3. Personaliza el prompt con:
   - Nombre de tu empresa
   - Servicios que ofreces
   - Tono de comunicación deseado
   - Horarios de atención
   - Información de contacto

Ejemplo de prompt personalizado:

```
Eres [NOMBRE], asistente virtual de [TU EMPRESA].

Ayudas a los clientes con:
- Información sobre [SERVICIO 1]
- Información sobre [SERVICIO 2]
- Agendar citas
- Responder preguntas frecuentes

Tu tono es amable, profesional y cercano.

Horario de atención: Lunes a Viernes, 10:00 - 14:00 y 15:00 - 19:00h
Ubicación: [TU CIUDAD]

Cuando agendes una cita, SIEMPRE:
1. Verifica disponibilidad con MCP Calendario
2. Solicita: nombre, email, teléfono
3. Confirma los datos con el usuario
4. Crea la cita en el calendario
5. Envía email de confirmación
6. Registra el lead en Google Sheets
```

Paso 6: Integrar en tu sitio web

Opción A: Cliente JavaScript simple

Crea un archivo `chat.js` en tu web:

javascript

```javascript
class ChatWebAgent {
  constructor(webhookUrl) {
    this.webhookUrl = webhookUrl;
    this.sessionId = this.generateSessionId();
  }

  generateSessionId() {
    // Genera un ID único por usuario/navegador
    let sessionId = localStorage.getItem('chat_session_id');
    if (!sessionId) {
      sessionId = 'session_' + Date.now() + '_' +
                  Math.random().toString(36).substr(2, 9);
      localStorage.setItem('chat_session_id', sessionId);
    }
    return sessionId;
  }

  async sendMessage(message) {
    try {
      const response = await fetch(this.webhookUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          chatInput: message,
          sessionId: this.sessionId
        })
      });

      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }

      const data = await response.json();
      return data.message;
    } catch (error) {
      console.error('Error:', error);
      return 'Lo siento, ha ocurrido un error. Por favor, intenta de nuevo.';
    }
  }
}

// Uso
```

```
const agent = new ChatWebAgent('https://tu-n8n.com/webhook/AGENTE_WEB_GITHUB');

// Ejemplo: enviar mensaje
agent.sendMessage('Hola').then(response => {
  console.log('Agente:', response);
});
```

Opción B: Widget de chat completo (HTML + CSS + JS)

Crea un archivo `chat-widget.html`:

html

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Chat con Agente IA</title>
 <style>
  * {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
  }

  body {
   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
   background: #f5f5f5;
  }

  #chat-widget {
   position: fixed;
   bottom: 20px;
   right: 20px;
   width: 380px;
   height: 600px;
   background: white;
   border-radius: 15px;
   box-shadow: 0 5px 40px rgba(0,0,0,0.16);
   display: flex;
   flex-direction: column;
   overflow: hidden;
  }

  #chat-header {
   background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
   color: white;
   padding: 20px;
   font-weight: bold;
   font-size: 18px;
  }

  #chat-messages {
   flex: 1;
   overflow-y: auto;
   padding: 20px;
```

```
  background: #f9f9f9;
}

.message {
  margin-bottom: 15px;
  padding: 12px 16px;
  border-radius: 18px;
  max-width: 80%;
  word-wrap: break-word;
  animation: fadeIn 0.3s;
}

@keyframes fadeIn {
  from { opacity: 0; transform: translateY(10px); }
  to { opacity: 1; transform: translateY(0); }
}

.user-message {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  margin-left: auto;
  text-align: right;
}

.agent-message {
  background: white;
  color: #333;
  border: 1px solid #e0e0e0;
}

.agent-message.typing {
  opacity: 0.6;
}

#chat-input-container {
  padding: 15px;
  background: white;
  border-top: 1px solid #e0e0e0;
  display: flex;
  gap: 10px;
}

#message-input {
  flex: 1;
  padding: 12px 16px;
  border: 1px solid #ddd;
  border-radius: 25px;
```

```
  font-size: 14px;
  outline: none;
  transition: border-color 0.3s;
}

#message-input:focus {
  border-color: #667eea;
}

#send-button {
  width: 45px;
  height: 45px;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  border: none;
  border-radius: 50%;
  color: white;
  cursor: pointer;
  display: flex;
  align-items: center;
  justify-content: center;
  transition: transform 0.2s;
}

#send-button:hover {
  transform: scale(1.05);
}

#send-button:active {
  transform: scale(0.95);
}

.typing-indicator {
  display: flex;
  gap: 4px;
  padding: 12px 16px;
}

.typing-indicator span {
  width: 8px;
  height: 8px;
  background: #999;
  border-radius: 50%;
  animation: typing 1.4s infinite;
}

.typing-indicator span:nth-child(2) {
```

```
    animation-delay: 0.2s;
  }

  .typing-indicator span:nth-child(3) {
    animation-delay: 0.4s;
  }

  @keyframes typing {
    0%, 60%, 100% { transform: translateY(0); }
    30% { transform: translateY(-10px); }
  }
 </style>
</head>
<body>
 <div id="chat-widget">
  <div id="chat-header">
   💬 Asistente Virtual
  </div>
  <div id="chat-messages"></div>
  <div id="chat-input-container">
   <input
    type="text"
    id="message-input"
    placeholder="Escribe tu mensaje..."
    autocomplete="off"
   >
   <button id="send-button">
    <svg width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
     <line x1="22" y1="2" x2="11" y2="13"></line>
     <polygon points="22 2 15 22 11 13 2 9 22 2"></polygon>
    </svg>
   </button>
  </div>
 </div>

 <script>
  class ChatWebAgent {
   constructor(webhookUrl, messagesContainer) {
    this.webhookUrl = webhookUrl;
    this.messagesContainer = messagesContainer;
    this.sessionId = this.generateSessionId();
    this.isTyping = false;
   }

   generateSessionId() {
    let sessionId = localStorage.getItem('chat_session_id');
    if (!sessionId) {
```

```
    sessionId = 'session_' + Date.now() + '_' +
                Math.random().toString(36).substr(2, 9);
    localStorage.setItem('chat_session_id', sessionId);
  }
  return sessionId;
}

addMessage(text, isUser) {
  const messageDiv = document.createElement('div');
  messageDiv.className = 'message ' +
                         (isUser ? 'user-message' : 'agent-message');
  messageDiv.textContent = text;
  this.messagesContainer.appendChild(messageDiv);
  this.scrollToBottom();
}

showTypingIndicator() {
  const typingDiv = document.createElement('div');
  typingDiv.className = 'message agent-message typing-indicator';
  typingDiv.id = 'typing-indicator';
  typingDiv.innerHTML = '<span></span><span></span><span></span>';
  this.messagesContainer.appendChild(typingDiv);
  this.scrollToBottom();
}

hideTypingIndicator() {
  const indicator = document.getElementById('typing-indicator');
  if (indicator) {
    indicator.remove();
  }
}

scrollToBottom() {
  this.messagesContainer.scrollTop =
    this.messagesContainer.scrollHeight;
}

async sendMessage(message) {
  if (this.isTyping) return;

  try {
    this.isTyping = true;
    this.addMessage(message, true);
    this.showTypingIndicator();

    const response = await fetch(this.webhookUrl, {
```

```
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          chatInput: message,
          sessionId: this.sessionId
        })
      });

      this.hideTypingIndicator();

      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }

      const data = await response.json();
      this.addMessage(data.message, false);
    } catch (error) {
      this.hideTypingIndicator();
      console.error('Error:', error);
      this.addMessage(
        'Lo siento, ha ocurrido un error. Por favor, intenta de nuevo.',
        false
      );
    } finally {
      this.isTyping = false;
    }
  }
}

// Inicialización
const messagesContainer = document.getElementById('chat-messages');
const agent = new ChatWebAgent(
  'https://tu-n8n.com/webhook/AGENTE_WEB_GITHUB', // ⚠️ CAMBIA ESTO
  messagesContainer
);

const messageInput = document.getElementById('message-input');
const sendButton = document.getElementById('send-button');

// Event listeners
sendButton.addEventListener('click', () => {
  const message = messageInput.value.trim();
  if (message && !agent.isTyping) {
    agent.sendMessage(message);
```

```
    messageInput.value = '';
  }
});

messageInput.addEventListener('keypress', (e) => {
  if (e.key === 'Enter' && !agent.isTyping) {
    sendButton.click();
  }
});

// Mensaje de bienvenida
setTimeout(() => {
  agent.sendMessage('Hola');
}, 500);
</script>
</body>
</html>
```

⚠️ **Importante**: Cambia la URL del webhook en el código JavaScript por tu URL real.

Paso 7: Probar el agente

1. Abre el archivo `chat-widget.html` en tu navegador
2. Deberías ver el widget de chat
3. El agente te saludará automáticamente
4. Envía un mensaje de prueba

Mensajes de prueba sugeridos:

- `"Hola"`
- `"¿Qué servicios ofrecen?"`
- `"Quiero agendar una cita"`
- `"¿Cuál es el horario de atención?"`

🔍 Verificar que todo funciona

✅ Checklist de funcionamiento

- [ ] El workflow está en estado **"Active"**
- [ ] Los mensajes llegan al webhook (ver en **Executions** de n8n)
- [ ] El agente responde a los mensajes
- [ ] Las conversaciones mantienen contexto (prueba enviando varios mensajes)
- [ ] El formato de las respuestas es legible (sin markdown)
- [ ] El `sessionId` se mantiene entre mensajes

🚨 Si algo no funciona

El agente no responde:

1. Revisa que el workflow esté **activo**
2. Verifica las credenciales de **Redis**
3. Comprueba que **OpenAI** o **Gemini** tengan créditos
4. Mira los logs de ejecución en n8n (ícono de reloj → **Executions**)
5. Verifica la URL del webhook en el código

Error de credenciales:

1. Ve a cada nodo con error (aparece un triángulo rojo)
2. Abre el nodo y asigna la credencial correcta
3. Guarda y reactiva el workflow

Error de CORS en el navegador:

Si ves errores de CORS en la consola:

```javascript
// Solución temporal: Usar durante desarrollo
const proxyUrl = 'https://cors-anywhere.herokuapp.com/';
const webhookUrl = proxyUrl + 'https://tu-n8n.com/webhook/...';
```

Nota: En producción, configura CORS en tu servidor n8n o usa un proxy reverso (Nginx, Cloudflare, etc.).

Redis no conecta:

```bash
# Verifica que Redis esté corriendo
docker ps | grep redis

# Si no está, inicialo
docker start redis

# Prueba la conexión
docker exec -it redis redis-cli -a tu_password ping
```

🎯 Próximos pasos

Ahora que tu agente funciona, puedes:

1. **Personalizar el diseño** del chat para tu marca
2. **Añadir herramientas** - Consulta TECHNICAL.md para agregar más funcionalidades
3. **Configurar servicios MCP** - Para calendario y correo automático
4. **Integrar Google Sheets** - Para capturar leads automáticamente
5. **Optimizar el prompt** - Ajusta el comportamiento del agente
6. **Añadir analíticas** - Monitorea conversaciones y métricas

---

📚 ¿Necesitas más ayuda?

- **Instalación detallada**: INSTALLATION.md
- **Documentación técnica**: TECHNICAL.md
- **Problemas comunes**: Revisa la sección de troubleshooting en TECHNICAL.md

---

💡 Tips adicionales

Mejorar el tiempo de respuesta:

1. Usa modelos más rápidos (GPT-4.1-mini en lugar de GPT-4.1)
2. Reduce la temperatura a 0.1 para respuestas más consistentes
3. Optimiza el system prompt (menos texto, más directo)

Personalizar el comportamiento:

```javascript
// En el System Message del AI Agent1
- Cambia el nombre del agente
- Define tu tono (formal, casual, técnico)
- Especifica qué puede y no puede hacer
- Añade información de tu negocio
```

Mantener la memoria limpia:

```bash
# Limpia sesiones antiguas de Redis periódicamente
redis-cli -a tu_password KEYS "chat_memory:*" | xargs redis-cli -a tu_password DEL
```

¿Todo funcionando? ¡Genial! Ahora tienes un agente web completamente automatizado. 🎉

¿Problemas? Revisa TECHNICAL.md o contacta para soporte profesional.