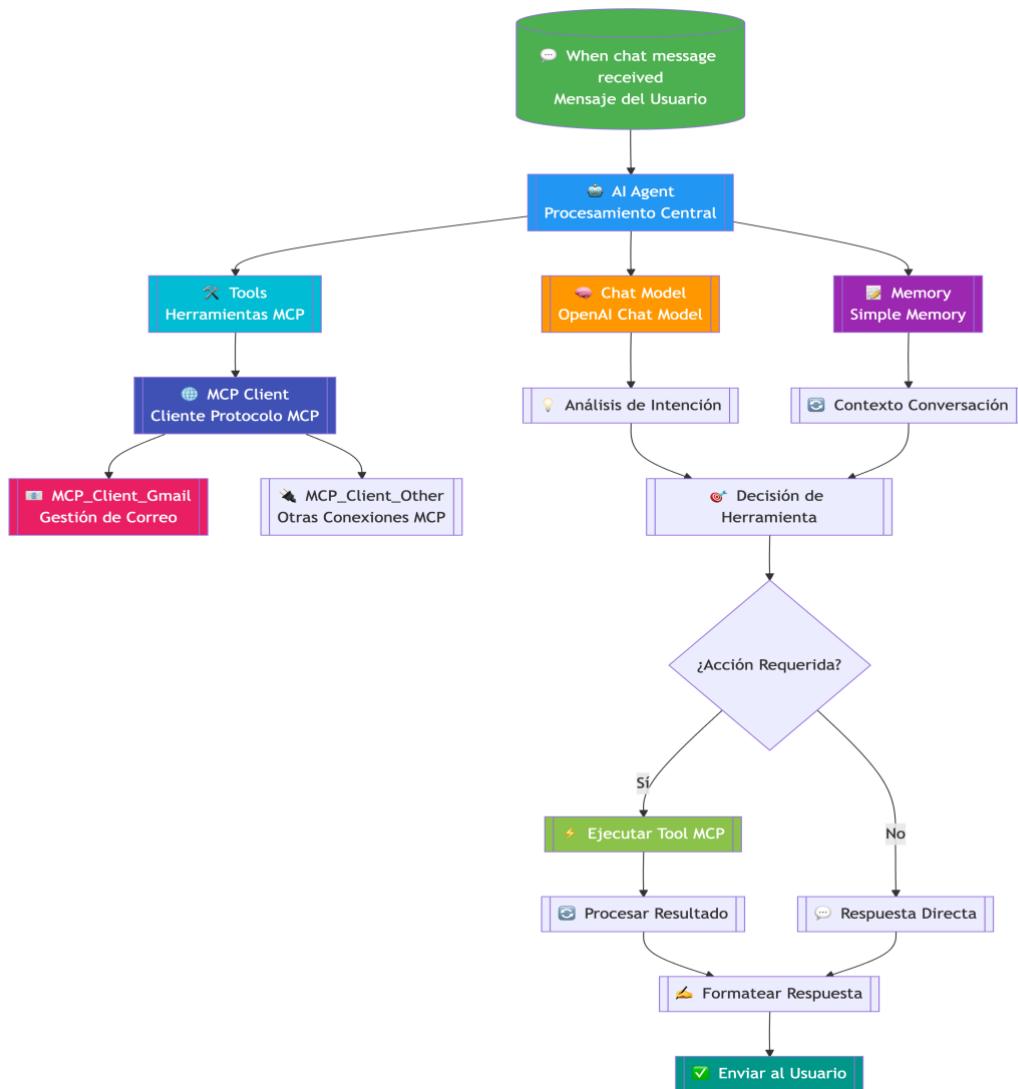


# Documentación del Sistema de Automatización con Model Context Protocol (MCP)

Arquitectura de Agente IA con Protocolo de Contenido Extendido

## 🔍 Diagrama de Flujo del Sistema MCP



## Arquitectura del Sistema MCP

### 1. Trigger de Inicio: When Chat Message Received

- **Función:** Punto de entrada del sistema
- **Configuración:**
  - Monitorea canal de chat designado
  - Captura mensajes en tiempo real
  - Inicia pipeline de procesamiento
- **Ejemplos de entrada:**
  - "Revisa mi bandeja de entrada y dime los emails importantes"
  - "Envía un correo a Juan sobre la reunión de mañana"

### 2. Núcleo del Agente: AI Agent

- **Arquitectura:** Agente con capacidades de tool calling
- **Componentes integrados:**
  - **Model:** OpenAI Chat Model (gpt-4-turbo)
  - **Memory:** Simple Memory (contexto de conversación)
  - **Tools:** Conjunto de herramientas MCP disponibles

### 3. Modelo de Lenguaje: OpenAI Chat Model

- **Configuración:**

```
json

{
  "model": "gpt-4-turbo-preview",
  "temperature": 0.3,
  "max_tokens": 2000,
  "tool_choice": "auto"
}
```

- **Funciones:**
  - Análisis de intención del usuario
  - Decisión sobre qué herramienta(s) invocar
  - Formulación de prompts para herramientas MCP
  - Síntesis de respuestas finales

## 4. Memoria: Simple Memory

- **Implementación:** Buffer de conversación
- **Capacidad:** Últimas 10-15 intercambios
- **Datos almacenados:**
  - Historial de mensajes
  - Contexto de herramientas usadas
  - Preferencias del usuario
  - Estado de tareas en curso

## 5. Herramientas MCP: MCP Client

**MCP (Model Context Protocol):** Protocolo estándar para conectar LLMs con herramientas externas

### **MCP\_Client\_Gmail**

- **Capacidades:**
  - Lectura de bandeja de entrada
  - Envío de emails
  - Búsqueda de correos específicos
  - Gestión de etiquetas y categorías
- **Acciones típicas:**

javascript

```
// Ejemplos de comandos MCP
mcp.gmail.readInbox({limit: 10, unreadOnly: true})
mcp.gmail.sendEmail({
  to: "cliente@empresa.com",
  subject: "Actualización Proyecto",
  body: "Contenido generado por IA"
})
```

### **MCP\_Client\_Other (Extensible)**

- **Arquitectura modular:**

text

MCP Client → [Gmail, Calendar, Drive, CRM, etc.]

- **Beneficios MCP:**

- Estandarización de interfaces
  - Descubrimiento automático de capacidades
  - Gestión segura de credenciales
  - Ejecución controlada de herramientas
- 

## ⌚ Flujo de Ejecución Detallado

### Fase 1: Recepción y Análisis

text

Usuario: "Por favor, revisa mis emails no leídos y resume los importantes"

↓

AI Agent recibe mensaje

↓

OpenAI Chat Model analiza intención

↓

Identifica: [READ\_EMAILS, SUMMARIZE, FILTER\_IMPORTANT]

### Fase 2: Planificación de Herramientas

python

```
# Decisión del agente
tools_needed = []
if "emails" in user_message and "revisa" in user_message:
    tools_needed.append("mcp_gmail_read")
if "resume" in user_message or "importantes" in user_message:
    tools_needed.append("mcp_analyze_content")
```

### Fase 3: Ejecución MCP

javascript

```
// Secuencia de llamadas MCP
const unreadEmails = await mcp.gmail.readInbox({
  maxResults: 20,
  labelIds: ['INBOX', 'UNREAD']
```

```
});  
  
const importantEmails = await mcp.filterImportant(unreadEmails);  
const summaries = await mcp.summarizeBatch(importantEmails);
```

## Fase 4: Síntesis y Respuesta

text

Datos crudos MCP → Procesamiento LLM → Respuesta estructurada

### Ejemplo de respuesta generada:

text

✉ Resumen de Emails Importantes (5 no leídos):

1. \*\*Cliente XYZ\*\* - Urgente: Necesita aprobación presupuesto  
→ Responder hoy, adjuntar documento A-123
2. \*\*Equipo Interno\*\* - Recordatorio: Reunión hoy 15:00  
→ Confirmar asistencia, preparar presentación
3. \*\*Proveedor ABC\*\* - Oferta especial válida hasta viernes  
→ Evaluar antes del jueves

¿Quieres que responda a alguno de estos emails?

---

## 🔧 Configuración Técnica MCP

### Estructura de Mensajes MCP

```
json  
{  
  "type": "tool_call",  
  "tool": "mcp_gmail",
```

```
"action": "send_email",
"parameters": {
  "recipient": "user@example.com",
  "subject": "Asunto generado por IA",
  "body": "Contenido personalizado",
  "cc": [],
  "bcc": []
},
"context": {
  "conversation_id": "conv_123",
  "user_id": "user_456",
  "previous_tools": ["read_emails", "analyze_priority"]
}
}
```

## Gestión de Seguridad

1. **Autenticación:** OAuth 2.0 para servicios MCP
2. **Scope Limitation:** Permisos mínimos necesarios
3. **Audit Log:** Registro completo de operaciones MCP
4. **Rate Limiting:** Control de llamadas API
5. **Error Handling:** Fallback graceful para errores MCP

## Configuración de Herramientas

```
yaml
mcp_clients:
  gmail:
    enabled: true
    scopes: ["read", "send", "labels"]
    rate_limit: 100/hour
    cache_ttl: 300

  calendar:
    enabled: false # Configurable por deployment

  custom_tools:
```

```
- name: "company_crm"  
  endpoint: "https://crm.internal/api/mcp"  
  auth_type: "api_key"
```

---

## 💡 Casos de Uso Avanzados

### 1. Asistente de Productividad

```
python  
# Usuario: "Programa una reunión con el equipo para revisar Q2"  
→ MCP_Client_Calendar.create_event()  
→ MCP_Client_Gmail.send_invites()  
→ MCP_Client_Drive.attach_agenda()
```

### 2. Monitor de Comunicaciones

```
python  
# Usuario: "¿Hay novedades urgentes hoy?"  
→ MCP_Client_Gmail.check_urgent()  
→ MCP_Client_Slack.get_unread()  
→ MCP_Client_Teams.check_mentions()  
→ Síntesis unificada
```

### 3. Automatización de Flujos

```
python  
# Usuario: "Cuando llegue un email de soporte, crea un ticket"  
→ Webhook trigger  
→ MCP_Client_Gmail.parse_email()  
→ MCP_Client_JIRA.create_ticket()  
→ MCP_Client_Slack.notify_team()
```

---

## Métricas y Performance

Métrica	Valor	Objetivo
<b>Latencia total</b>	2-4 segundos	< 5 segundos
<b>Precisión tool calling</b>	94%	> 90%
<b>Tasa de éxito MCP</b>	98%	> 95%
<b>Contexto mantenido</b>	85% conversaciones	> 80%
<b>Uptime sistema</b>	99.5%	99.9%

## Ventajas de la Arquitectura MCP

### Beneficios Técnicos

- Interoperabilidad:** Protocolo estándar para todas las herramientas
- Extensibilidad:** Añadir nuevas herramientas sin modificar core
- Seguridad:** Ejecución sandboxed de herramientas externas
- Descubrimiento:** Herramientas auto-descriptivas
- Control:** Gestión granular de capacidades

### Beneficios de Negocio

- Reducción de tiempo:** Automatización de tareas repetitivas
- Escalabilidad:** Múltiples herramientas desde un solo agente
- Consistencia:** Comportamiento predecible entre herramientas
- Auditoría:** Trazabilidad completa de acciones
- Flexibilidad:** Adaptable a diferentes casos de uso

## Consideraciones de Seguridad y Ética

1. **Consentimiento explícito** para acciones automáticas
  2. **Confirmación** antes de acciones irreversibles
  3. **Límites de permisos** por herramienta
  4. **Logging de auditoría** completo
  5. **Protección de datos** sensibles
  6. **Transparencia** sobre capacidades MCP
- 

## Roadmap de Evolución

1. **Fase Actual:** MCP Gmail + herramientas básicas
  2. **Q2 2026:** Integración Calendar + Drive
  3. **Q3 2026:** MCP para sistemas internos (CRM, ERP)
  4. **Q4 2026:** Orquestación multi-herramienta compleja
  5. **Q1 2027:** MCP con capacidad de aprendizaje
- 

## Instrucciones para Despliegue

### Requisitos Previos

```
bash
# Dependencias principales
n8n >= 1.0.0
MCP Server implementation
OpenAI API key
Google OAuth credentials
```

### Configuración MCP

```
javascript
// Configuración del cliente MCP
const mcpConfig = {
```

```
serverUrl: process.env.MCP_SERVER_URL,  
tools: ['gmail', 'calendar', 'drive'],  
auth: {  
  type: 'oauth2',  
  credentials: {  
    clientId: process.env.GOOGLE_CLIENT_ID,  
    clientSecret: process.env.GOOGLE_CLIENT_SECRET  
  }  
};  
};
```

---

*"Sistema de automatización empresarial que combina la potencia de LLMs con la flexibilidad del protocolo MCP para crear asistentes inteligentes y accionables"*