# **Flujo de Automatización en n8n para WhatsApp - Gimnasio de Escalada**

## **Arquitectura del Flujo**

Este flujo utiliza WhatsApp Business Cloud API integrado con n8n para automatizar todo el proceso de atención.​

## **Nodos del Workflow**

## **1. WhatsApp Trigger - "On Message Received"**

Tipo: WhatsApp Business Cloud Trigger  
Función: Escucha y recibe mensajes entrantes de WhatsApp en tiempo real.​

Configuración:

* Webhook URL: URL generada por n8n
* Autenticación: WhatsApp Business API Token
* Eventos: Escuchar mensajes entrantes

Salida del nodo:

json

{

"from": "+56912345678",

"timestamp": "2025-10-29T16:15:00Z",

"message": {

"type": "text",

"text": {

"body": "Hola, quiero información sobre clases"

}

},

"contact": {

"profile": {

"name": "Juan Pérez"

}

}

}

## **2. Set Node - "Extraer Datos Usuario"**

Función: Normaliza los datos del mensaje para facilitar el procesamiento.​

Configuración:

javascript

{

"telefono": "{{$json.from}}",

"nombre\_usuario": "{{$json.contact.profile.name}}",

"mensaje": "{{$json.message.text.body}}",

"timestamp": "{{$json.timestamp}}"

}

## **3. Switch Node - "Clasificador de Intención"**

Función: Enruta el flujo según el tipo de consulta detectada.​

Reglas (Routes):

Route 1 - Consulta de Precios:

* Condición: {{$json.mensaje}} contiene "precio", "costo", "tarifa", "cuánto"
* Salida: → Nodo "Respuesta Precios"

Route 2 - Consulta de Horarios:

* Condición: {{$json.mensaje}} contiene "horario", "hora", "cuándo", "disponible"
* Salida: → Nodo "Consultar Horarios"

Route 3 - Reservar Clase:

* Condición: {{$json.mensaje}} contiene "reservar", "agendar", "clase", "inscribir"
* Salida: → Subflow "Proceso de Reserva"

Route 4 - Información General:

* Condición: {{$json.mensaje}} contiene "ubicación", "dónde", "dirección", "info"
* Salida: → Nodo "Info General"

Route 5 - Default (Consulta Compleja):

* Condición: No coincide con anteriores
* Salida: → Nodo "AI Agent RAG"

## **4A. Function Node - "Respuesta Precios"**

Función: Genera respuesta con información de precios.​

Código:

javascript

const precios = {

"individual": "15.000",

"bono\_5": "65.000",

"bono\_10": "120.000",

"mensual": "85.000"

};

const respuesta = `🧗 \*PRECIOS CLIMBZONE\*

• Clase individual: $${precios.individual}

• Bono 5 clases: $${precios.bono\_5}

• Bono 10 clases: $${precios.bono\_10}

• Mensualidad ilimitada: $${precios.mensual}

¿Te gustaría reservar una clase?

Responde \*"reservar"\* para agendar 📅`;

return {

json: {

respuesta: respuesta,

telefono: $input.item.json.telefono

}

};

## **4B. HTTP Request - "Consultar Horarios Google Calendar"**

Función: Consulta disponibilidad en Google Calendar.​​

Configuración:

* Method: GET
* URL: https://www.googleapis.com/calendar/v3/calendars/PRIMARY/events
* Authentication: OAuth2 Google
* Query Parameters:
  + timeMin: Fecha inicio (hoy)
  + timeMax: Fecha fin (+7 días)
  + orderBy: startTime

Salida procesada:

javascript

*// Filtrar eventos disponibles*

const eventos = $json.items;

const disponibles = eventos.filter(e =>

e.attendees && e.attendees.length < e.attendeeLimit

);

*// Formatear respuesta*

let mensaje = "📅 \*HORARIOS DISPONIBLES\*\n\n";

disponibles.forEach(evento => {

const fecha = new Date(evento.start.dateTime);

mensaje += `• ${fecha.toLocaleDateString('es-CL')}\n`;

mensaje += ` ${fecha.toLocaleTimeString('es-CL', {hour: '2-digit', minute:'2-digit'})}\n`;

mensaje += ` ${evento.summary} (${evento.attendeeLimit - evento.attendees.length} cupos)\n\n`;

});

mensaje += "Para reservar, responde con la fecha y hora que prefieres 🎯";

return { json: { respuesta: mensaje } };

## **4C. Subflow - "Proceso de Reserva Completo"**

## **4C.1. IF Node - "¿Usuario Registrado?"**

Función: Verifica si el usuario ya tiene datos guardados.​

Condición:

javascript

*// Consultar base de datos de usuarios*

const telefono = $json.telefono;

const usuario = $('Base\_Usuarios').item.json.usuarios.find(u => u.telefono === telefono);

if (usuario) {

return [true, null]; *// Usuario existe*

} else {

return [null, true]; *// Usuario nuevo*

}

## **4C.2. WhatsApp Send Message - "Solicitar Datos"**

Función: Pide datos al usuario si es nuevo.​

Mensaje:

text

¡Hola! Para reservar necesito algunos datos:

1️⃣ Nombre completo

2️⃣ Email (opcional)

Por favor envíalos en un solo mensaje 😊

## **4C.3. WhatsApp Wait for Response**

Función: Espera respuesta del usuario con sus datos.​

Configuración:

* Timeout: 300 segundos
* Resume on: Incoming message

## **4C.4. Set Node - "Guardar Datos Usuario"**

Función: Extrae y normaliza los datos proporcionados.

javascript

const mensaje = $json.message.text.body;

const lineas = mensaje.split('\n');

return {

json: {

nombre: lineas[0],

email: lineas[1] || null,

telefono: $json.from

}

};

## **4C.5. Postgres/Google Sheets - "Registrar Usuario"**

Función: Guarda el nuevo usuario en la base de datos.​

## **4C.6. WhatsApp Send Message - "Mostrar Horarios"**

Función: Muestra opciones de horarios disponibles.​​

Mensaje con botones interactivos:

json

{

"messaging\_product": "whatsapp",

"to": "{{$json.telefono}}",

"type": "interactive",

"interactive": {

"type": "list",

"header": {

"type": "text",

"text": "📅 Selecciona un horario"

},

"body": {

"text": "Elige la clase que deseas reservar:"

},

"action": {

"button": "Ver horarios",

"sections": [

{

"title": "Mañana 30/Oct",

"rows": [

{"id": "2025-10-30\_10:00\_principiante", "title": "10:00 AM - Principiante", "description": "5 cupos"},

{"id": "2025-10-30\_12:00\_intermedio", "title": "12:00 PM - Intermedio", "description": "3 cupos"}

]

},

{

"title": "Pasado Mañana 31/Oct",

"rows": [

{"id": "2025-10-31\_14:00\_avanzado", "title": "14:00 PM - Avanzado", "description": "2 cupos"},

{"id": "2025-10-31\_18:00\_boulder", "title": "18:00 PM - Boulder", "description": "4 cupos"}

]

}

]

}

}

}

## **4C.7. WhatsApp Wait for Response**

Función: Espera selección del usuario.​

## **4C.8. Function Node - "Parsear Selección"**

Función: Extrae fecha, hora y tipo de clase.

javascript

const seleccion = $json.message.interactive.list\_reply.id;

const [fecha, hora, tipo] = seleccion.split('\_');

return {

json: {

fecha: fecha,

hora: hora,

tipo\_clase: tipo,

usuario: $('Guardar\_Datos\_Usuario').item.json

}

};

## **4C.9. Google Calendar - "Crear Evento"**

Función: Crea la reserva en Google Calendar.​​

Configuración:

* Operation: Create Event
* Calendar: ClimbZone Calendar
* Event Data:

json

{

"summary": "Clase {{$json.tipo\_clase}} - {{$json.usuario.nombre}}",

"start": {

"dateTime": "{{$json.fecha}}T{{$json.hora}}:00-03:00",

"timeZone": "America/Santiago"

},

"end": {

"dateTime": "{{$json.fecha}}T{{$json.hora + 2}}:00-03:00",

"timeZone": "America/Santiago"

},

"attendees": [

{"email": "{{$json.usuario.email}}"}

],

"description": "Reserva vía WhatsApp\nTeléfono: {{$json.usuario.telefono}}"

}

## **4C.10. WhatsApp Send Message - "Confirmación Reserva"**

Función: Envía confirmación al usuario.​

Mensaje:

javascript

const datos = $json;

return {

json: {

message: `✅ \*RESERVA CONFIRMADA\*

👤 ${datos.usuario.nombre}

📅 ${new Date(datos.fecha).toLocaleDateString('es-CL')}

🕐 ${datos.hora}

🧗 Clase: ${datos.tipo\_clase.toUpperCase()}

📍 ClimbZone - Av. Principal 123

Te enviaremos un recordatorio 24 hrs antes.

¡Nos vemos! 💪

\_Para cancelar, responde "cancelar" al menos 2 hrs antes.\_`

}

};

## **4C.11. Postgres/Google Sheets - "Registrar Reserva"**

Función: Guarda la reserva en base de datos para seguimiento.​

## **4D. OpenAI/Ollama - "AI Agent con RAG"**

Función: Procesa consultas complejas usando IA.​

Configuración:

* Model: GPT-4 o Llama local
* System Prompt:

text

Eres el asistente virtual de ClimbZone, un gimnasio de escalada.

INFORMACIÓN DEL GIMNASIO:

- Horarios: Lun-Vie 6:00-22:00, Sáb 8:00-20:00, Dom 10:00-18:00

- Clases: Principiante, Intermedio, Avanzado, Boulder

- Política cancelación: 2 horas antes sin cargo

- Dirección: Av. Principal 123, Santiago

INSTRUCCIONES:

- Responde en español de forma amigable y concisa

- Usa emojis apropiados

- Si no sabes algo, ofrece contacto con personal

- Máximo 150 palabras por respuesta

- Si mencionan reserva/precio/horario, sugiere usar comandos específicos

User Message: {{$json.mensaje}}

## **5. Pinecone/Vector Database - "Base de Conocimientos"**

Función: Búsqueda semántica en documentación del gimnasio.​

Datos indexados:

* FAQs sobre equipamiento, seguridad, instructores
* Políticas detalladas
* Tipos de membresías
* Promociones vigentes

## **6. IF Node - "¿Requiere Escalamiento?"**

Función: Detecta si necesita intervención humana.​

Condiciones:

* Mensaje contiene "reclamo", "mal servicio", "problema"
* Usuario envía más de 5 mensajes sin resolver consulta
* IA responde "necesito ayuda humana"

## **7. Telegram/Email - "Notificar Personal"**

Función: Alerta al equipo cuando se requiere atención.​

Mensaje:

text

🚨 Escalamiento requerido

Usuario: {{$json.nombre\_usuario}}

Teléfono: {{$json.telefono}}

Último mensaje: {{$json.mensaje}}

Responder por WhatsApp lo antes posible.

## **8. WhatsApp Send Message - "Enviar Respuesta Final"**

Función: Envía la respuesta generada al usuario.​

Configuración:

* To: {{$json.telefono}}
* Message: {{$json.respuesta}}

## **9. Google Sheets/Postgres - "Registrar Interacción"**

Función: Almacena todas las conversaciones para análisis.​

Datos guardados:

javascript

{

telefono: $json.telefono,

nombre: $json.nombre\_usuario,

timestamp: $json.timestamp,

mensaje\_usuario: $json.mensaje,

respuesta\_bot: $json.respuesta,

tipo\_consulta: $json.clasificacion,

escalado: $json.requiere\_humano || false

}

## **Flujo Paralelo: Recordatorios Automáticos**

## **10. Schedule Trigger - "Diario 9:00 AM"**

Función: Ejecuta workflow de recordatorios.​​

## **11. Google Calendar - "Obtener Eventos Mañana"**

Función: Consulta reservas para el día siguiente.

Query:

* timeMin: Mañana 00:00
* timeMax: Mañana 23:59

## **12. Loop Node - "Iterar Reservas"**

Función: Procesa cada reserva individualmente.

## **13. WhatsApp Send Message - "Enviar Recordatorio"**

Función: Envía recordatorio personalizado.​​

Mensaje:

javascript

const evento = $json;

return {

json: {

to: evento.attendees[0].phoneNumber,

message: `🔔 \*RECORDATORIO DE CLASE\*

Hola ${evento.attendees[0].displayName},

Te recordamos tu clase mañana:

📅 ${new Date(evento.start.dateTime).toLocaleDateString('es-CL')}

🕐 ${new Date(evento.start.dateTime).toLocaleTimeString('es-CL', {hour: '2-digit', minute: '2-digit'})}

🧗 ${evento.summary}

📍 ClimbZone - Av. Principal 123

Si necesitas cancelar, responde a este mensaje.

¡Nos vemos! 💪`

}

};