# Análisis Completo: Workflow WhatsApp AI Chatbot en n8n

## El Problema que Tuvimos (5+ horas)

El chatbot recibía mensajes de WhatsApp pero el AI Agent respondía con texto genérico sobre "nodos" en vez del mensaje real del usuario.

---

## 1. CÓMO FUNCIONA EL FLUJO DE DATOS EN n8n

**Concepto Fundamental: Cada nodo recibe datos del nodo anterior**

```
Nodo A → Nodo B → Nodo C
         ↑
         Nodo B recibe el OUTPUT de Nodo A
         Nodo B NO sabe qué hay en Nodo C
```

**En tu workflow:**

```
Webhook → Edit Fields → AI Agent → Send Message
  │          │           │           │
  │          │           │           └─ Recibe output del AI Agent
  │          │           │
  │          │           └─ Recibe output de Edit Fields
  │          │
  │          └─ Recibe output del Webhook (datos de Meta)
  │
  └─ Recibe datos directamente de Meta (JSON crudo)
```

---

## 2. LOS DOS ERRORES PRINCIPALES

**ERROR 1: Texto literal en vez de expresión**

**Lo que tenías:**

```
json

"text": "=Take from previous node automatically"
```

**El problema:**

- En n8n, `=` al inicio indica "quiero usar una expresión"
- PERO el texto que sigue NO era una expresión válida

- n8n lo interpretó como texto literal
- El AI Agent recibía literalmente "Take from previous node automatically" como prompt
- Por eso respondía explicando qué son los nodos

**La solución:**

```
json

"text": "={{ $('Edit Fields').item.json.chatInput }}"
```

**ERROR 2: Sintaxis incorrecta de expresión**

**Intentamos:**

```
{{ $json.chatInput }}
```

**Por qué no funcionó:**

- `$json` solo funciona para acceder a datos del nodo INMEDIATAMENTE anterior
- Pero el AI Agent es un nodo especial (langchain) que maneja los datos diferente
- Necesita referencia EXPLÍCITA al nodo origen

**La solución que funcionó:**

```
{{ $('Edit Fields').item.json.chatInput }}
```

---

## 3. EXPRESIONES EN n8n: GUÍA COMPLETA

**Tipos de referencias:**

| Sintaxis | Significado | Cuándo usar |
|---|---|---|
| `{{ $json.campo }}` | Campo del nodo anterior inmediato | Nodos simples en secuencia |
| `{{ $('Nombre Nodo').item.json.campo }}` | Campo de un nodo específico | Siempre que necesites certeza |
| `{{ $input.first().json.campo }}` | Primer item del input | Cuando hay múltiples items |
| `{{ $node['Nombre'].json.campo }}` | Otra forma de referenciar nodo | Alternativa válida |

**REGLA DE ORO:**

> Cuando tengas dudas, usa SIEMPRE la sintaxis completa: `{{ $('Nombre del Nodo').item.json.campo }}`

---

## 4. POR QUÉ EL WEBHOOK EJECUTA 2 VECES

**Meta envía DOS tipos de eventos al mismo endpoint:**

**Tipo 1: Mensaje entrante** (lo que queremos)

```json
{
  "body": {
    "entry": [{
      "changes": [{
        "value": {
          "messages": [{
            "from": "34619794604",
            "text": { "body": "Hola" }
          }]
        }
      }]
    }]
  }
}
```

**Tipo 2: Status update** (delivered, read, etc.)

```json
{
  "body": {
    "entry": [{
      "changes": [{
        "value": {
          "statuses": [{
            "status": "delivered",
            "recipient_id": "34619794604"
          }]
        }
      }]
    }]
  }
}
```

**El problema:**

- Tu webhook recibe AMBOS
- El workflow intenta procesar ambos

- En status updates, `messages[0]` no existe → da `undefined`
- Por eso veías `null` en los campos

**Solución (opcional):**

Añadir nodo IF que filtre solo mensajes reales:

```
Condición: {{ $json.body.entry[0].changes[0].value.messages }} exists
```

---

## 5. WEBHOOK GENÉRICO vs WHATSAPP TRIGGER

**Tu enfoque: Webhook genérico**

```
Webhook (POST) → Edit Fields → AI Agent → WhatsApp Send
```

**Ventajas:**

- Más control sobre el flujo
- Funciona con cualquier versión de n8n
- No requiere OAuth de WhatsApp

**Desventajas:**

- Debes extraer campos manualmente (Edit Fields)
- Recibes TODOS los eventos (mensajes + status)
- Expresiones más complejas: `$json.body.entry[0].changes[0].value.messages[0].from`

**Alternativa: WhatsApp Trigger (nodo nativo)**

```
WhatsApp Trigger → AI Agent → WhatsApp Send
```

**Ventajas:**

- Filtra automáticamente solo mensajes
- Datos ya parseados: `$json.messages[0].from`
- Más simple

**Desventajas:**

- Requiere OAuth configurado con Meta
- Solo un webhook por app de Meta

- Puede dar problemas entre test y producción

---

## 6. ESTRUCTURA DE DATOS DE META WHATSAPP

**JSON completo que llega al Webhook:**

```json
```

```
{
 "headers": { ... },
 "params": {},
 "query": {},
 "body": {
  "object": "whatsapp_business_account",
  "entry": [
   {
    "id": "TU_BUSINESS_ACCOUNT_ID",
    "changes": [
     {
      "value": {
       "messaging_product": "whatsapp",
       "metadata": {
        "display_phone_number": "15551658897",
        "phone_number_id": "846885438510763"
       },
       "messages": [
        {
         "from": "34619794604",
         "id": "wamid.xxx",
         "timestamp": "1234567890",
         "type": "text",
         "text": {
          "body": "Hola, necesito ayuda"
         }
        }
       ],
       "contacts": [
        {
         "profile": {
          "name": "José"
         },
         "wa_id": "34619794604"
        }
       ]
      },
      "field": "messages"
     }
    ]
   }
  ]
```

```
  }
}
```

**Cómo acceder a cada campo:**

| Dato | Expresión |
|---|---|
| Número del remitente | `$json.body.entry[0].changes[0].value.messages[0].from` |
| Texto del mensaje | `$json.body.entry[0].changes[0].value.messages[0].text.body` |
| Nombre del contacto | `$json.body.entry[0].changes[0].value.contacts[0].profile.name` |
| Phone Number ID | `$json.body.entry[0].changes[0].value.metadata.phone_number_id` |

---

## 7. TU WORKFLOW FINAL (FUNCIONANDO)

**Estructura:**

```
Webhook (POST) ──→ Edit Fields ──→ AI Agent ──→ Send Message
                                      ↑
                                OpenAI Chat Model
                                Simple Memory

Webhook1 (GET) ──→ Respond to Webhook (verificación Meta)
```

**Configuración de cada nodo:**

**Webhook (POST):**

- Path: `whatsapp-webhook`
- Method: POST

**Edit Fields:**

- sender: `{{ $json.body.entry[0].changes[0].value.messages[0].from }}`
- message: `{{ $json.body.entry[0].changes[0].value.messages[0].text.body }}`
- chatInput: `{{ $json.body.entry[0].changes[0].value.messages[0].text.body }}`

**AI Agent:**

- Prompt Type: Define
- Text: `{{ $('Edit Fields').item.json.chatInput }}`

**Simple Memory:**

- Session Key: `{{ $json.sender }}`

**Send Message:**

- Phone Number ID: `846885438510763`
- Recipient: `{{ $('Edit Fields').item.json.sender }}`
- Text Body: `{{ $json.output }}`

---

## 8. LECCIONES APRENDIDAS

**1. Siempre verificar el OUTPUT de cada nodo**

- En n8n, haz clic en cada nodo después de ejecutar
- Revisa la pestaña "OUTPUT" para ver qué datos salen
- Si ves `null` o `undefined`, el problema está en la expresión

**2. Usar expresiones explícitas**

- NO confiar en "Take from previous node automatically"
- SIEMPRE usar `{{ $('Nombre Nodo').item.json.campo }}`

**3. Entender la estructura de datos de Meta**

- Meta envuelve todo en `body.entry[0].changes[0].value`
- Los mensajes están en `.messages[0]`
- Los status están en `.statuses[0]`

**4. Probar con datos reales**

- El modo "Test" de n8n puede dar resultados diferentes a producción
- Meta envía datos ligeramente diferentes en cada caso

---

## 9. CREDENCIALES DE REFERENCIA

**App de Meta (Izumi Hotel):**

- App ID: `820864327516745`
- Phone Number ID: `846885438510763`
- Business Account ID: `681354961503657`

**Webhook:**

- URL: `https://n8n-production-bb2d.up.railway.app/webhook/whatsapp-webhook`
- Verify Token: `myhostbizmate`

**n8n:**

- URL: `https://n8n-production-bb2d.up.railway.app`
- Workflow ID: `mEth5Jcuswp2cNXt`

---

## 10. PRÓXIMOS PASOS RECOMENDADOS

1. **Añadir filtro IF** para evitar doble ejecución
2. **Personalizar el System Prompt** del AI Agent para Izumi Hotel
3. **Añadir manejo de errores** por si falla la API de WhatsApp
4. **Considerar migrar a WhatsApp Trigger** nativo cuando estabilices

---

*Documento creado el 2 de Diciembre 2025 después de 5+ horas de troubleshooting*