Memoria Técnica — Flujo “Reserva de cita personal” (n8n)

Autor: Celena Perea, Yorleisy Ramos y Juan Esteban Ortiz.

Fecha: 2025-10-24

---

1. Resumen ejecutivo

Este documento describe paso a paso la implementación, ejecución y mejoras del workflow de **reserva de citas** desarrollado en **n8n**. Está pensado para que cualquier persona con acceso a n8n y a las credenciales de Cal.com, correo y Telegram pueda replicarlo.

Contenido:

- Requisitos y preparación
- Descripción de la arquitectura del flujo
- Documentación por nodo: qué hace, cómo se configura, entradas/salidas y ejemplos
- Manejo de errores y validaciones
- Mejora propuesta e instrucciones para desplegar y monitorizar
- Anexos: snippets JSON y comandos útiles

---

2. Requisitos previos

1. Cuenta y credenciales en **n8n** con permisos para crear flujos.
2. API key de **Cal.com** con permisos para crear reservas.
3. Cuenta de correo (Gmail) con OAuth o credenciales para lectura (si se usa nodo Gmail).
4. Token/API para enviar mensajes (Telegram/Bot) o servicio SMS.

5. Conocimientos básicos de expresiones de n8n (`{{$json[...]}}`, `{{ ... }}`) y formato ISO 8601 para fechas.

6. Opcional: ambiente de pruebas (sandbox) en Cal.com para verificar sin afectar producciones.

---

3. Arquitectura del flujo (visión general)

El flujo implementa la siguiente secuencia lógica:

1. **Webhook** — recibe la solicitud inicial (formulario/entrada externa).

2. **Verificación** — (If / Function) valida que la solicitud sea para espacio/fecha válidos.

3. **Preparar datos de reserva** — construye el JSON que requiere Cal.com (fechas, timezone, respuestas).

4. **Consultar disponibilidad** — (HTTP Request) consulta slots disponibles en Cal.com.

5. **Reservar una cita** — (HTTP Request / nodo Cal.com) envía la reserva final.

6. **Responder al webhook** — devuelve al origen la confirmación o error.

7. **Notificaciones** — envía correo/telegram con los detalles al usuario o al equipo.

> En la implementación de ejemplo se han usado dos ramas: una que intenta reservar directamente y otra que primero consulta disponibilidad si la primera falla.

---

4. Flujo por nodo — implementación y ejecución

A continuación se documenta cada nodo: configuración, propósito, entradas y salidas, y ejemplos.

Nodo: Webhook

- **Tipo:** Webhook (Trigger)

- **Propósito:** Recibir la petición de reserva desde una fuente externa (formulario web, integración, otra API).
- **Configuración:**
    - Método: `POST`
    - Path: `/reservas/cita-personal` (o el que prefieras)
    - Response mode: `On Received` o `Last Node` según quieras respuesta inmediata o esperar procesamiento.
- **Entrada esperada (ejemplo):**

```JSON
{
  "firstName": "Yorle",
  "email": "yorle@example.co",
  "startTime": "2025-10-24T14:00:00-05:00",
  "endTime": "2025-10-24T14:30:00-05:00",
  "appointmentTypeId": "3697380"
}
```

- **Salida:** Los datos llegan tal cual a la siguiente etapa. Verificar `View Output` para depuración.

Consejo: activar `Response Mode: Last Node` si quieres devolver el resultado del proceso (confirmación). Para pruebas usa `On Received`.

Nodo: La verificación es una solicitud de espacio... (If node / Function)

- **Tipo:** If (o Function)
- **Propósito:** Validar que el mensaje recibido contiene los campos obligatorios y que el `appointmentTypeId` corresponde al servicio deseado.
- **Validaciones mínimas:**

    - `firstName` no vacío
    - `email` válido (regex simple)
    - `startTime` y `endTime` presentes y en formato ISO
    - `appointmentTypeId` == `3697380` (si aplica)
- **Salida:** bifurca el flujo a `verdadero` (continúa) o `falso` (responde con error)

Ejemplo de Function (JS) para validar campos:

```javascript
const item = $json;
if (!item.firstName || !item.email || !item.startTime) {
  throw new Error('Faltan campos obligatorios');
}
return item;
```

Nodo: Preparar datos de reserva (Set / Function)

- **Tipo:** Set o Function
- **Propósito:** Construir los campos que el API de Cal.com requiere (start, end, timezone, responses, status). Normalizar fechas a ISO UTC.
- **Configuración típica:** Añadir campos calculados:
    - `eventTypeId`: `3697380`
    - `startTime` y `endTime` en ISO con zona UTC (ej. `2025-10-24T19:00:00Z`)
    - `timeZone`: `America/Bogota`
    - `responses` (objeto) con `name` y `email` como strings
- **Ejemplo de expresión para construir ISO UTC (en n8n):**

```
JavaScript
{{ new Date($json.startTime).toISOString() }}
```

Nota: si se necesita lógica más compleja para evitar reservar en el pasado, usar una Function con JS para calcular `start` y `end` dinámicamente.

Nodo: Consultar espacio disponible en Cal.com (HTTP Request)

- **Tipo:** HTTP Request
- **Método:** `GET` (endpoint de disponibilidad de Cal.com)
- **URL ejemplo:** `https://api.cal.com/v1/event-types/3697380/availability?from=2025-10-24T00:00:00Z&to=2025-10-25T00:00:00Z`
- **Headers:**
  - `Authorization: Bearer <CAL_API_KEY>`
  - `Content-Type: application/json`
- **Propósito:** Obtener los slots disponibles para el `eventTypeId` y elegir uno válido.
- **Salida:** lista de franjas horarias. En la respuesta elegir el primer slot válido o aplicar lógica de priorización.

Consejo: Guardar y revisar la respuesta en `Execution → Output` para mapear correctamente los campos.

Nodo: Reservar una cita (Book an Appointment / HTTP Request)

- **Tipo:** Nodo oficial Cal.com o HTTP Request (POST)
- **URL (ejemplo POST):** `https://api.cal.com/v1/bookings`
- **Headers:**

    - `Authorization: Bearer <CAL_API_KEY>`
    - `Content-Type: application/json`
- **Body (JSON Body) — ejemplo final (válido para Cal.com):**

```JSON
{
  "eventTypeId": 3697380,
  "start": "{{$json.start}}",
  "end": "{{$json.end}}",
  "timeZone": "America/Bogota",
  "language": "es",
  "metadata": {},
  "responses": {
    "name": "Celena Perea",
    "email": "cele@example.co"
  },
  "status": "ACCEPTED"
}
```

- **Errores comunes y cómo resolverlos:**
    - `Attempting to book a meeting in the past` → ajustar `start` a futuro
    - `booking_time_out_of_bounds_error` → elegir slot dentro de la disponibilidad del `eventTypeId`
    - `invalid_type in 'responses'` → enviar `responses` como objeto con strings

Nodo: Responder al webhook (Respond to Webhook)

- **Tipo:** Respond to Webhook

- **Propósito:** Devolver al origen la confirmación o el error del proceso. Ideal si el Webhook usa `Response Mode: Last Node`.
- **Ejemplo de payload de respuesta:**

```
JSON
{"status":"success","bookingId":"12345","start":"2025-10-24T19:00:00Z"}
```

Consejo: En caso de error devolver un objeto con `status: error` y `message` legible.

Nodo: Recibir un mensaje (Gmail) — opcional

- **Tipo:** Gmail
- **Propósito:** Procesar emails entrantes relacionados con la reserva (confirmaciones de Cal.com, respuestas del usuario, etc.)
- **Configuración:** Conexión OAuth, permisos `https://mail.google.com/`.
- **Uso típico:** leer el correo, extraer información y actualizar registros o notificar al equipo.

Nodo: Enviar un mensaje de texto / Telegram

- **Tipo:** Telegram (o HTTP Request para SMS)
- **Propósito:** Notificar al cliente o al equipo sobre la reserva.
- **Ejemplo:** enviar mensaje con `bookingId`, `start` y `enlace`.

Consejo: almacenar la respuesta de Cal.com y usar sus campos (`bookingId`, `invitee_url`) en la plantilla del mensaje.

---

5. Manejo de errores y reintentos

1. **Validaciones tempranas (If/Function):** rechazar peticiones sin datos obligatorios.
2. **Transformación de fechas:** asegurar ISO 8601 y zona horaria.
3. **Reintentos en HTTP:** configurar `Retry` en HTTP Request para errores temporales (429, 5xx).
4. **Notificación de fallos:** enviar correo/telegram al admin si un nodo crítico falla.
5. **Registro (logging):** guardar en base de datos o Google Sheets cada intento (entrada, resultado, error) para auditoría.

---

6. Mejoras sugeridas

1. **Consulta de disponibilidad automática:** antes de intentar reservar, consultar `/availability` y seleccionar el primer slot libre.
2. **Pool de reservas:** en horas pico, implementar una lógica que escoja el próximo slot libre y notifique al usuario si la hora solicitada no está disponible.
3. **Interfaz de seguimiento:** registrar todas las reservas en Google Sheets o Firebase para buscarlas y mantener un dashboard.
4. **Tests automatizados:** usar ejecuciones de prueba en sandbox para validar cambios.
5. **Manejo avanzado de zonas horarias:** normalizar todas las fechas a UTC en el backend y mostrar en la zona del usuario.
6. **Control de concurrency:** bloquear múltiples requests simultáneos para el mismo slot (locking).

---

7. Despliegue y monitorización

1. **Entorno de pruebas:** usar API key de sandbox y flujo de prueba.
2. **Monitorización:** configurar alertas en n8n (o mediante webhooks) cuando el flujo falla repetidamente.

3. **Backups:** exportar el flujo (JSON) y guardar en control de versiones.
4. **Documentación:** mantener este documento actualizado con los cambios.

---

8. Checklist para replicar (rápido)

- Crear webhook en n8n (POST)
- Conectar credenciales de Cal.com (API Key)
- Crear nodo de verificación (If / Function)
- Normalizar fechas en Set/Function a ISO/UTC
- Consultar disponibilidad (HTTP GET)
- Reservar (HTTP POST / nodo Cal.com) con body válido
- Responder al origen
- Notificar por correo/Telegram

---

9. Anexos — snippets útiles

JSON Body final de ejemplo (copiar en nodo Book an Appointment - Send Body as JSON)

```json
{
  "eventTypeId": 3697380,
  "start": "2025-10-23T14:00:00Z",
  "end": "2025-10-23T14:30:00Z",
  "timeZone": "America/Bogota",
  "language": "es",
  "metadata": {},
  "responses": {
    "name": "Yorle Ramos",
    "email": "yorle@example.co"
```

```
  },
  "status": "ACCEPTED"
}
```

Expresión JS en n8n para start +1 hora (ISO UTC)

```JavaScript
{{ new Date(Date.now() + 60*60*1000).toISOString() }}
```

---

10. Conclusión

Con este documento tienes la guía completa para replicar el flujo de reserva en n8n, validar entradas, consultar disponibilidad en Cal.com y realizar reservas de forma segura.
Implementa las mejoras propuestas para robustecer el flujo (consulta de disponibilidad, logs y manejo de concurrencia).
