Actividad Personalizada SMS para Journey Builder

Presentación de Arquitectura y Flujo

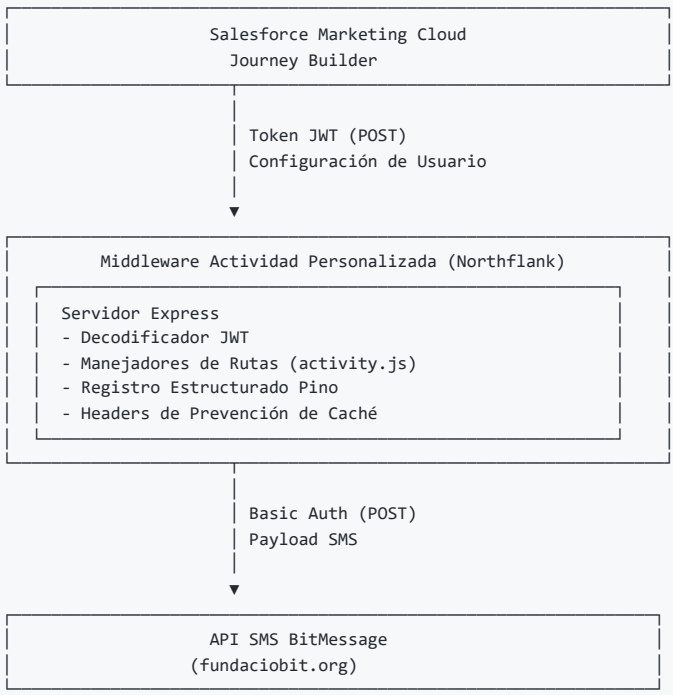
📋 Resumen General

Middleware de actividad personalizada que integra Salesforce Marketing Cloud Journey Builder con la API de SMS de BitMessage.

Stack Tecnológico:

- Node.js + Express
- Pino Logger
- Autenticación JWT
- Cliente HTTP Axios
- Desplegado en Northflank

📐 Diagrama de Arquitectura



🔄 Flujo Completo del Journey

Fase 1: Configuración (Marketer)

```
1. Marketer arrastra actividad → Canvas de Journey Builder

2. Marketer hace clic en Configurar
   │
   ├─→ Carga index.html (UI)
   │
   └─→ Usuario completa formulario:
       - Texto del mensaje
       - Campos de vinculación de datos

3. Usuario hace clic en "Listo"
   │
   ├─→ POST /validate (validación opcional)
   │
   └─→ POST /save (guarda configuración)
```

Fase 2: Activación del Journey

```
Marketer hace clic en "Activar Journey"
   │
   └─→ POST /publish (configuración única)
```

Fase 3: Ejecución en Tiempo Real (Por Contacto)

```
Contacto entra en la actividad
   │
   └─→ POST /execute
       │
       ├─→ Decodificar token JWT
       ├─→ Extraer datos del contacto (teléfono, mensaje)
       ├─→ Llamar API BitMessage
       │   │
       │   ├─→ Éxito (ENVIADO/CONFIRMADO)
       │   │   └─→ Retorna: {"branchResult": "sent"}
       │   │       └─→ Contacto sigue ruta "Enviado" ✅
       │   │
       │   └─→ Error (ERROR/timeout)
       │       └─→ Retorna: {"branchResult": "notsent"}
       │           └─→ Contacto sigue ruta "No Enviado" ❌
```

Fase 4: Desactivación del Journey

```
Marketer detiene el Journey
   │
   └─→ POST /stop (limpieza)
```

📡 Endpoints de la API

Endpoints de Configuración (UI de Journey Builder)

Endpoint	Cuándo se Llama	Propósito
`/save`	Usuario hace clic en “Listo”	Guardar configuración de actividad
`/validate`	Antes de guardar	Validar configuración
`/publish`	Journey activado	Configuración/inicialización
`/edit`	Usuario edita actividad	Cargar configuración guardada
`/stop`	Journey detenido	Limpieza de recursos

Endpoint de Ejecución (Motor de Ejecución)

Endpoint	Cuándo se Llama	Propósito
`/execute`	Contacto entra en actividad	Enviar SMS vía BitMessage

🔐 Seguridad y Autenticación

Journey Builder → Middleware

- **Método:** JWT (JSON Web Token)
- **Content-Type:** `application/jwt`
- **Body:** Token JWT sin procesar
- **Secreto:** Almacenado en variables de entorno

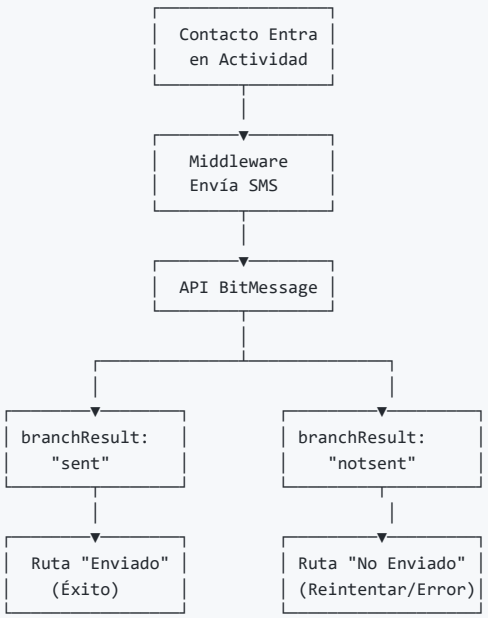
Middleware → API BitMessage

- **Método:** Autenticación Básica
- **Usuario/Contraseña:** Desde variables de entorno
- **Content-Type:** `application/json`

📊 Flujo del Endpoint Execute (Detallado)

```
POST /execute
   │
   ├─→ [1] Verificar Token JWT
   │   ├─→ Válido: Continuar
   │   └─→ Inválido: Retornar 401 No Autorizado
   │
   ├─→ [2] Extraer Datos del Contacto
   │   - telefono (número de teléfono)
   │   - texto (texto del mensaje)
   │   - campanyaReferencia (referencia de campaña)
   │
   ├─→ [3] Llamar API BitMessage envio instántaneo
   │   │
   │   └─→ POST https://bitmessage.fundaciobit.org/api/v1/envios/mensaje/send
   │       {
   │         "telefono": "654162594",
   │         "texto": "Tu mensaje aquí",
   │         "campanyaReferencia": "SOIB"
   │       }
   │
   ├─→ [4] Manejar Respuesta
   │   │
   │   ├─→ estado: "ENVIADO" o "CONFIRMADO"
   │   │   └─→ Retorna: {"branchResult": "sent"}
   │   │
   │   └─→ estado: "ERROR" o fallo de API
   │       └─→ Retorna: {"branchResult": "notsent"}
   │
   └─→ [5] Journey Builder enruta el contacto según el resultado
```

🎯 Lógica de Ramificación del Journey



📈 Escalabilidad y Rendimiento

Configuración Actual:

- **Hosting:** Northflank (auto-escalado)
- **Logging:** Pino (logs estructurados de alto rendimiento)
- **Caché:** Deshabilitado (siempre datos frescos)
- **Peticiones Concurrentes:** Configurable en config.json

Maneja:

- Múltiples journeys simultáneamente
- Miles de contactos por journey
- Entrega de SMS en tiempo real
- Seguimiento detallado de errores

🔍 Monitoreo y Depuración

Niveles de Logging:

```
[INFO]  - Endpoint llamado, operaciones exitosas
[WARN]  - Datos inválidos, errores de API
[ERROR] - Fallos JWT, errores críticos
[DEBUG] - Tokens JWT (solo desarrollo)
```

Logs Clave a Monitorear:

1. “Execute endpoint called” - Contacto entró
2. “Calling BitMessage API” - Envío de SMS iniciado
3. “SMS sent successfully” - Entrega confirmada
4. “BitMessage API call failed” - Ocurrió un error

🚀 Checklist de Despliegue

- ✅ Código desplegado en Northflank
- ✅ Variables de entorno configuradas:
  - `jwtSecret`
  - `BITMESSAGE_INSTANT_SMS_API`
  - `BITMESSAGE_USERNAME`
  - `BITMESSAGE_PASSWORD`
  - `BITMESSAGE_CAMPANYA`
- ✅ Actividad personalizada registrada en Journey Builder
- ✅ config.json apunta a URL de producción
- ✅ Monitoreo habilitado (logs de Northflank)

📝 Ejemplo de Flujo de Datos

Escenario: Enviar SMS de bienvenida a nuevo cliente

```
1. Cliente se registra → Journey disparado

2. Journey Builder envía a /execute:
   {
     "inArguments": [{
       "telefono": "654162594",
       "texto": "¡Bienvenido a nuestro servicio!"
     }]
   }

3. Middleware llama a BitMessage:
   POST https://bitmessage.fundaciobit.org/.../send
   {
     "telefono": "654162594",
     "texto": "¡Bienvenido a nuestro servicio!",
     "campanyaReferencia": "SOIB"
   }

4. BitMessage responde:
   {
     "id": 16312,
     "estado": "ENVIADO",
     "telefono": "654162594",
     "texto": "¡Bienvenido a nuestro servicio!",
     ...
   }

5. Middleware retorna a Journey Builder:
   {"branchResult": "sent"}

6. Journey Builder enruta cliente a ruta de éxito ✅
```

✨ Beneficios Clave

1. **Separación de Responsabilidades**
    - Lógica de UI aislada (index.html)
    - Lógica de negocio separada (activity.js)
    - Integración API abstraída (sendBitMessageSMS)
2. **Manejo de Errores**
    - Validación JWT
    - Manejo de fallos de API
    - Logging comprensivo
3. **Mantenibilidad**
    - Estructura de código limpia
    - Configuración basada en entorno
    - Fácil de probar y depurar
4. **Seguridad**
    - Autenticación JWT
    - Credenciales en variables de entorno
    - Nunca expuestas en código

🎓 Aspectos Técnicos Destacados

- **JavaScript Moderno:** ES6+ con async/await
- **Logging:** Pino para logs estructurados de grado producción
- **Diseño API:** RESTful con códigos de estado apropiados
- **Resiliencia ante Errores:** Degradación elegante en fallos
- **Prevención de Caché:** Siempre configuraciones frescas
- **Arquitectura Limpia:** Responsabilidades separadas, código testeable

---

📞 Soporte y Mantenimiento

Puntos de Monitoreo:

- Estado de despliegue en Northflank
- Logs de aplicación (salida de Pino)
- Tiempos de respuesta API BitMessage
- Uso de actividad en Journey Builder

Problemas Comunes:

- Discrepancia de secreto JWT → Verificar variables de entorno
- Timeouts de API → Verificar estado de API BitMessage
- Números de teléfono inválidos → Validar datos de entrada
- Problemas de caché → Verificar headers de caché
