Informe comercio conversacional con twilio

Backend Intern Developers:

Daniel Holguín Domínguez

Santiago Marín Ocampo

25-05-2023

KEYRUS S.A.S

Contenido

¿Qué es Twilio?

Twilio es una plataforma y librería de comunicaciones para desarrolladores la cual permite incorporar capacidades de voz, video, mensajería y autenticación en sus aplicaciones y servicios mediante el uso de su API. Esto significa que los desarrolladores pueden utilizar las herramientas y funciones de Twilio para agregar y gestionar la comunicación en tiempo real en sus propias aplicaciones, sitios web y servicios en la nube.

Algunas de las funciones que provee twilio dentro de su plataforma son la creación de líneas telefónicas virtuales, el envío y recepción de mensajes de texto, la realización de llamadas de voz y video, la autenticación de usuarios mediante mensajes SMS o llamadas de voz, y mucho más. La plataforma de Twilio es altamente escalable y ha sido adoptada por numerosas empresas de todo el mundo para mejorar la comunicación con sus clientes y usuarios.

El modelo de cobro de twilio es pago por consumo, es decir, que los clientes pagan por los recursos que utilizan en función de factores como el volumen de mensajes enviados o recibidos, la duración de las llamadas, la cantidad de números telefónicos virtuales utilizados, entre otros.

Desarrollo en Twilio

Durante la exploración de la plataforma nos centramos en la creación de un flujo conversacional utilizando como medio de mensajeria un numero de whatsapp y un canal de facebook messenger. En los siguientes puntos explicaremos cada uno de los modulos usados para crear un flujo y posteriormente veremos un pequeño tutorial de como se construye.

Modulos

Studio:

Studio es el módulo de twilio para crear los flows mediante un canvas, donde se construyen arrastrando unos WIDGET. Dado que twilio brinda un sistema de mensajería por voz, brinda múltiples widgets para la funcionalidad de este, pero en este acercamiento con la plataforma, solo se exploraron los widgets relacionando con los mensajes de texto que se describirán a continuación:

- **Trigger**: Este es el comienzo del flujo, que puede ser llamado mediante una REST API URL o WEBHOOK URL, mediante estas URL es donde se hace la integración con los múltiples canales de mensajería, que serán explicados más adelante.

- **Send message**: Solo envía un mensaje y continua con el flujo desde de enviarlo.

- **Send and wait for reply:** Envia el mensaje y espera a la respuesta del usuario para continuar el flujo. Tiene la opción de configurar un tiempo de espera límite para continuar con el flujo en caso de que el usuario no responda.

- **Split Based On…:** Es la manera de dividir el flujo, de acuerdo con una variable. Se puede validar la condición de la variable mediante:

    - Equal To
    - Not Equal To
    - Matches Any Of
    - Does Not Match Any Of
    - Is Blank
    - Is Not Blank
    - Regex
    - Contains
    - Does Not Contain
    - Starts With
    - Does Not Start With
    - Less Than
    - Greater Than
    - Is Before Time
    - Is After Time
    - Is Before Date
    - Is After Date

- **Set variables:** Se utiliza para guardar variables, como por ejemplo la respuesta del usuario a una pregunta. Más adelante se explicará el manejo de variables en flujo

- **Run subflow:** Se utiliza para llamar otro flujo; cuando el flujo entra por el run subflow continua con este nuevo flujo y una vez este flujo termine volverá al flujo inicial continuando desde este widget.

- Nota: Un flujo termina cuando llega a un widget y no encuentra siguientes conexiones a otros widgets.

- **Run function:** Se utiliza para ejecutar una función desde el módulo de Functions and Assets, se puede añadir los respectivos parámetros desde variables del flujo y su respuesta también es guardada en una variable del flujo.
- Nota: este widget tiene dos transacciones cuando la función falla y cuando es exitosa; se entiende que es exitosa la función cuando se retorna un código de respuesta de 200 y que falla cuando se retorna un 500.

- **Make HTTP Request:** Se utiliza para hacer peticiones HTTP desde el flujo. La manera cómo funciona sus transacciones es similar a la de Run function.

- **Send to flex:** Se utiliza para crear una tarea que recibe la plataforma flex, que será explicada más adelante y trasfiere el flujo hacia un asesor humano.

Variables en el flujo

Para obtener el valor de una variable se sigue el siguiente formato:

{{<<**referencia de la variable**>>}}

Ejemplos

- Si se quiere obtener la respuesta del usuario desde un Send and wait for reply. Se obtiene con la siguiente estructura:

  {{widgets**.<<nombre del widget>>**.inbound.Body }}

- Si se quiere obtener el valor de una variable guardada con el widget set variable:

  {{flow.variables**.<<nombre de la variable puesta en el set varaible>>** }}

- Si se quiere obtener la respuesta del widget Run function y Make HTTP Request:

  {{widgets**.<<nombre del widget Run function/ Make HTTP Request** >>.parsed}}

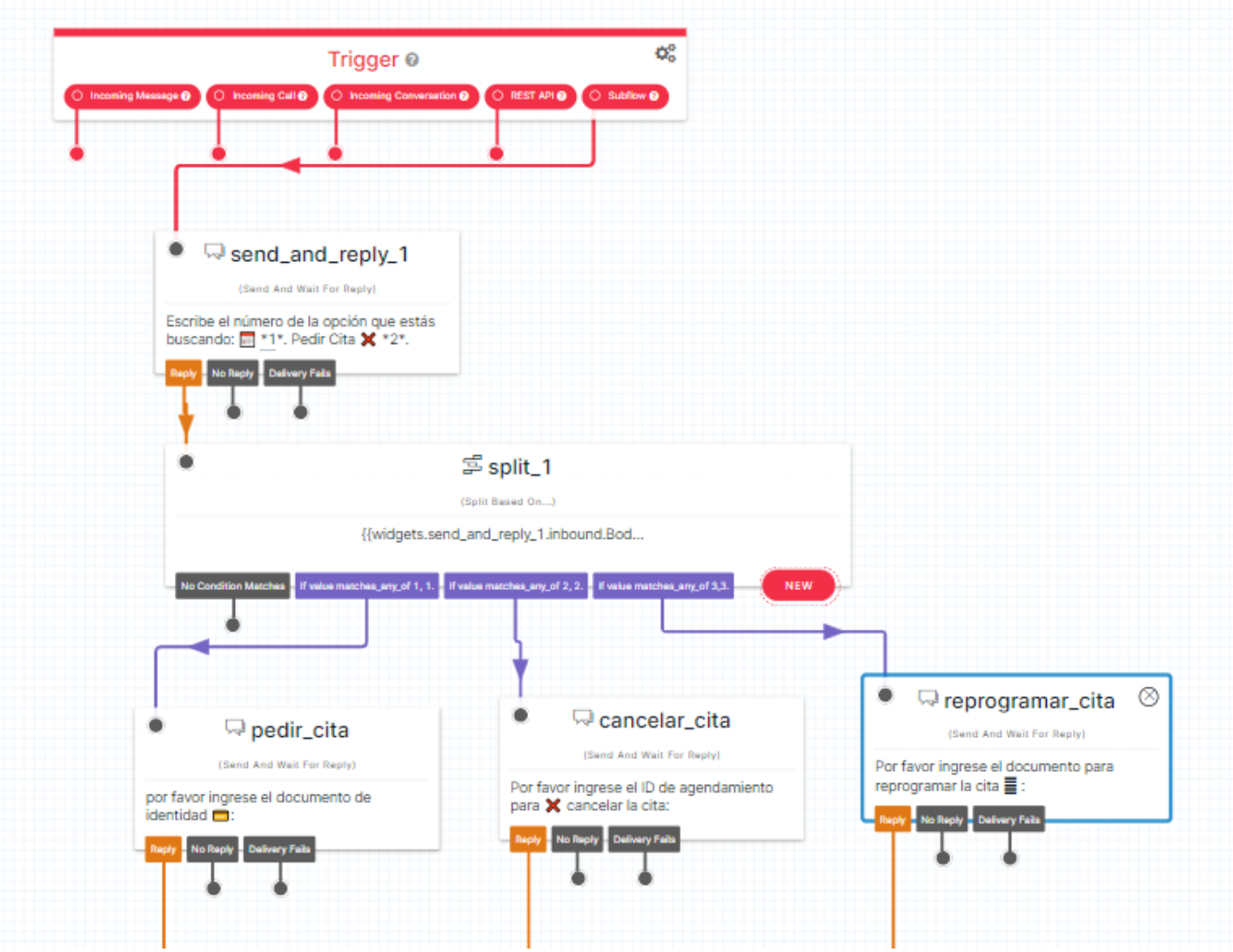
  Nota:

  De esta manera se accede a la raíz de la respuesta, para acceder a una variable especifica en esta raíz basta con poner .<<nombre de la variable>>

  {{widgets**.<<nombre del widget Run function**>>. parsed **.<<nombre de la variable**>>}}

Nota:

Twilio no brinda la opción de poner botos de respuesta rápida en sus mensajes mediante el módulo de studio, pero existe una manera de implementarlo mediante las platillas de mensajes de WhatsApp; Se crea una plantilla con el mismo mensaje que será mando en el flujo y se le agregan los botones de respuesta rápida a la plantilla, una vez que esta esté aprobada, le aparecerán los botones al usuario cuando se le mande el mensaje desde el flujo.



Functions and Assets:

Twilio está orientado a ser implementado como librería / dependencia en un backend independiente, sin embargo, ofrece un módulo para implementar este backend en su propia plataforma.
El módulo permite crear aplicaciones de comunicación mediante código *Javascript* usando *Node JS* y la librería de Twilio. Este código y ejecución de este es totalmente alojado en la infraestructura de nube que ofrece Twilio. Adicionalmente, permite almacenar archivos de distintos tipos como .mp3, jpg y más para su uso.

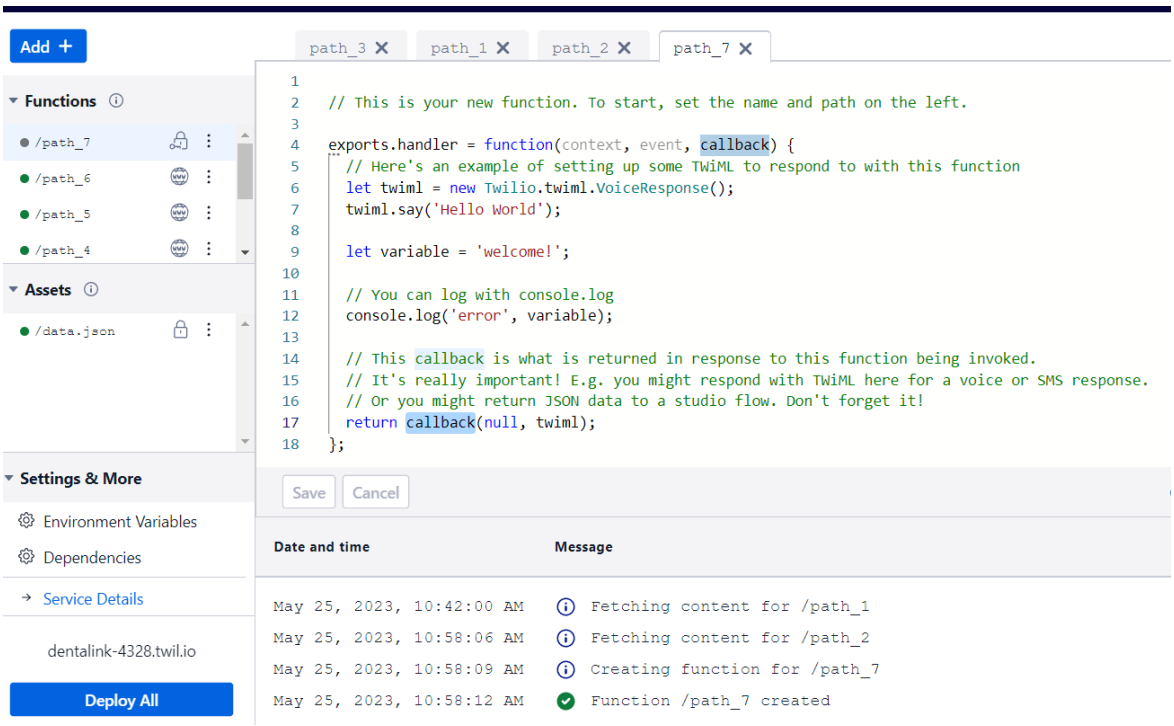
La creación de estas funciones se hace mediante lo que twilio llama servicios, un servicio es un contenedor de aplicaciones para almacenar todas sus Funciones y Activos, y se utiliza para administrar implementaciones y entornos separados. Probablemente creará un nuevo Servicio para cada nuevo proyecto en el que trabaje.

Notas:

- Las funciones de twilio se ejecutan una única vez, y si dentro de su código existe un ciclo sin condición de parada, la infraestructura de twilio detendrá el código en determinado tiempo.

- El uso del *callback* es indispensable para que la función se ejecute correctamente como se ve en el siguiente ejemplo: al final se hace el *return callback* y si NO se pasa con el parámetro

null y el código funciona bien, retorna un 500. Por lo que es necesario pasar el *null* al callback para que retorne un 200 si la aplicación funciona bien.

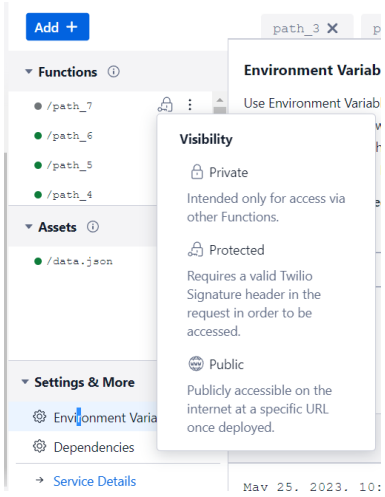


- Para usar las variables que se pasan como parámetros de función, se utiliza el event. Ejemplo, si a la función se le pasa un dato llamado nombre, en el código para acceder a este valor y guardarlo en una variable local tendrá que hacer lo siguiente: nombre = event.nombre

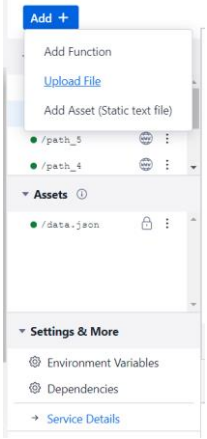
- A parte de Node JS y la librería de twilio, la plataforma permite añadir más dependencias/librerías externas como por ejemplo *axios*, la cual facilita el manejo del consumo y envío de datos con APIs. Para agregar una dependencia se dirige al apartado de «*dependencies*»

- Para no usar las credenciales de la cuenta de twilio dentro del código debe dirigirse al aparatado de «Enviroment Variables», ahí existe una opción de usar las credenciales de twilio en las variables de ambiente y usarlas con context.getTwilioClient(). Adicionalmente, ese apartado permite añadir más variables con datos sensibles para almacenarlos de manera segura, como por ejemplo tokens de autenticación, entre otros.

- La visibilidad de la función es importante por si se quiere acceder/ejecutar desde un sistema externo o desde otro apartado de twilio, como lo es el studio. Para cambiarla en el apartado de Functions hay un icono de candado, se le da clic ahí y se despliegan las opciones que ofrece de privacidad



- Para cargar un archivo debe oprimir el botón «Add +» y se desplegará un menú contextual con tres opciones, añadir una función, subir un archivo y agregar un texto plano estático.

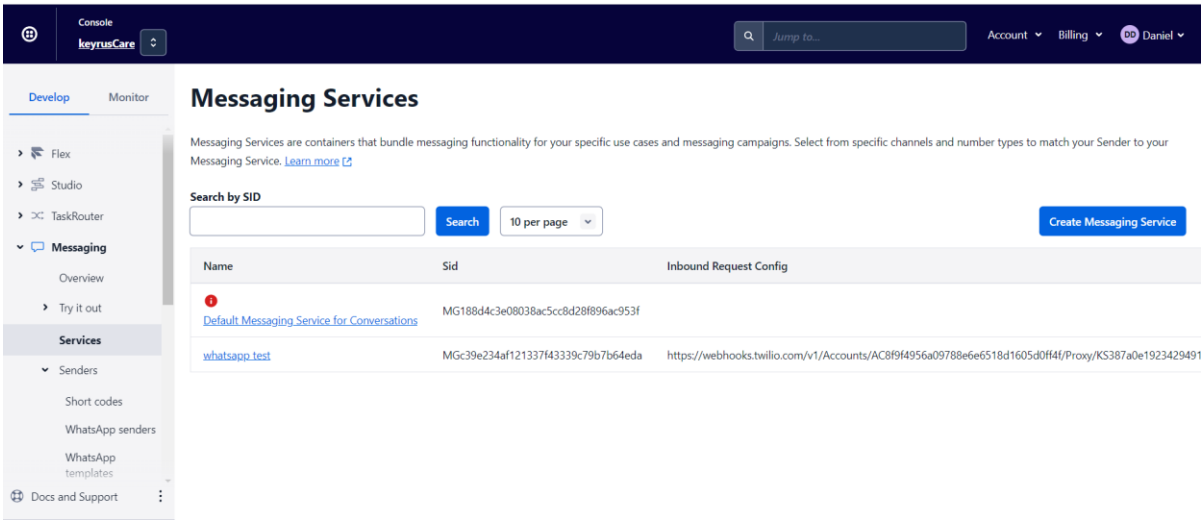


- Para acceder a un archivo desde el código es necesaria la siguiente línea: `const Jsondata = Runtime.getAssets()['/data.json'].open()` como se ve en la imagen anterior, existe un archivo en formato JSON ya subido llamado data.json al cual se accede mediante una url y la función Runtime.getAssets()[urldelassets].open().

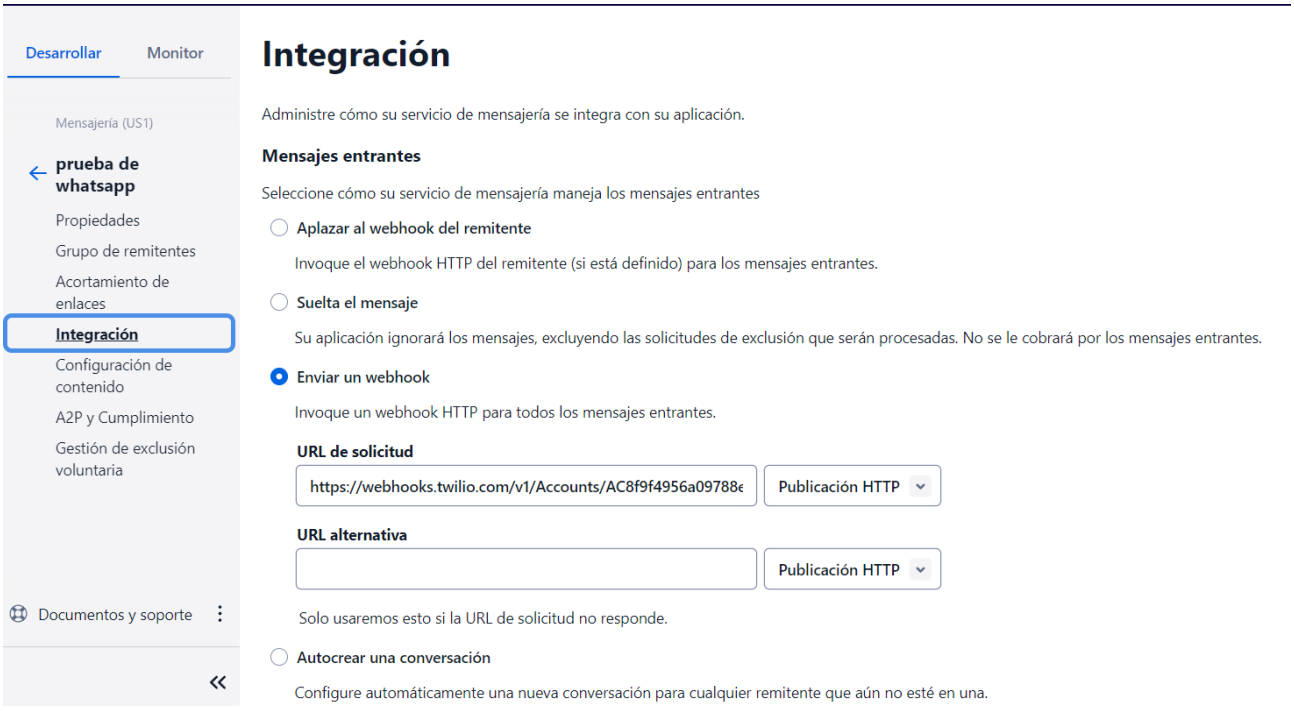
Messaging:

Este módulo permite configurar el envío de mensajes asociando un número de teléfono tradicional o uno de WhatsApp según sea el caso de uso.

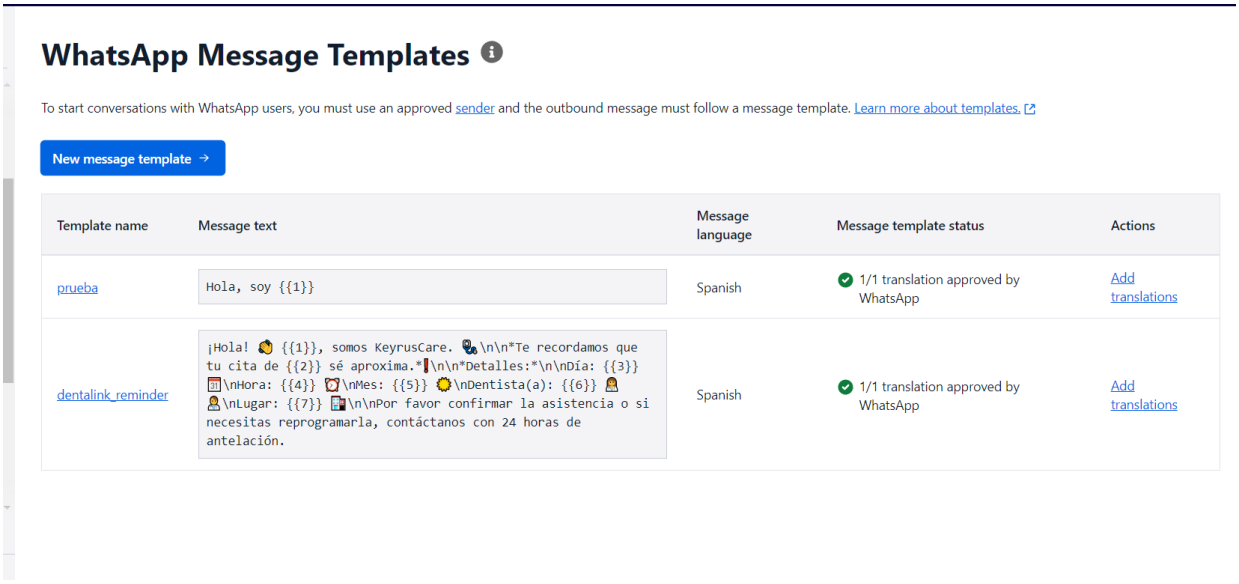
El apartado «*Services*» en *Messaging* son contenedores que agrupan la funcionalidad de mensajería para sus casos de uso y campañas de mensajería específicos. Seleccione entre canales específicos y tipos de números para hacer coincidir su remitente con su servicio de mensajería.



Es necesario crear un servicio ya que este genera un ID para asociarlo a servicios de notificación de WhatsApp, por ejemplo, adicionalmente en el apartado integraciones podrá conectarlo a flujo del studio mediante webhook. Esta URL que se pone en el webhook es extraída del disparador (*Trigger*) flujo del studio.



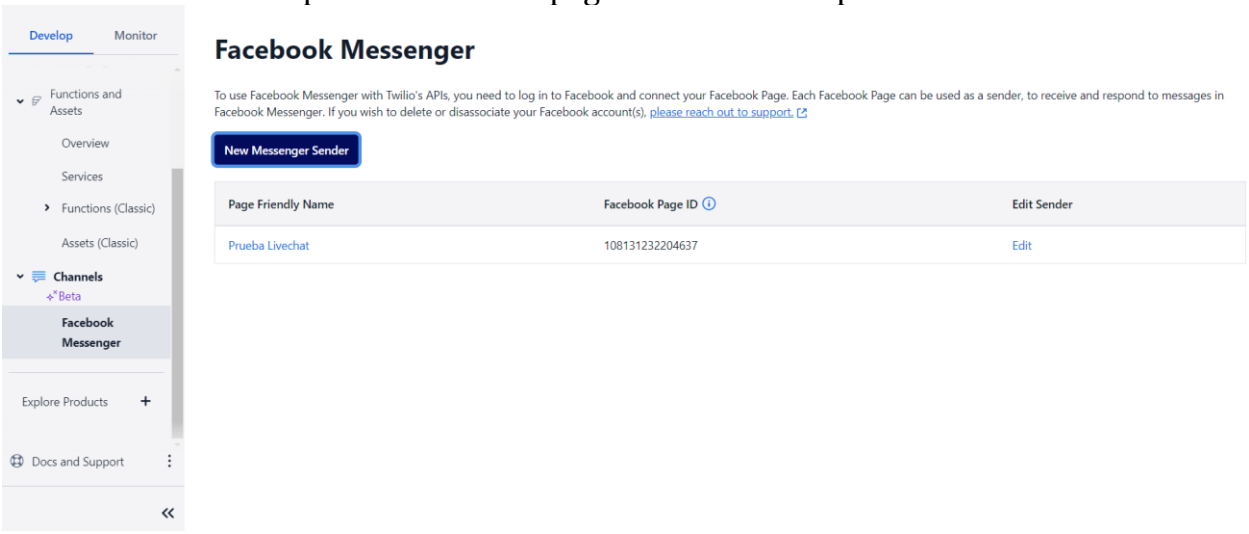
Para utilizar WhatsApp como canal de mensajería es necesario registrarlo en el apartado de *Senders* enlazando la cuenta de negocios de meta / WhatsApp. Adicionalmente, si desea usar botones en los mensajes de WhatsApp debe mandar a meta una plantilla de mensaje genérica para que ellos la aprueben según sus lineamientos.



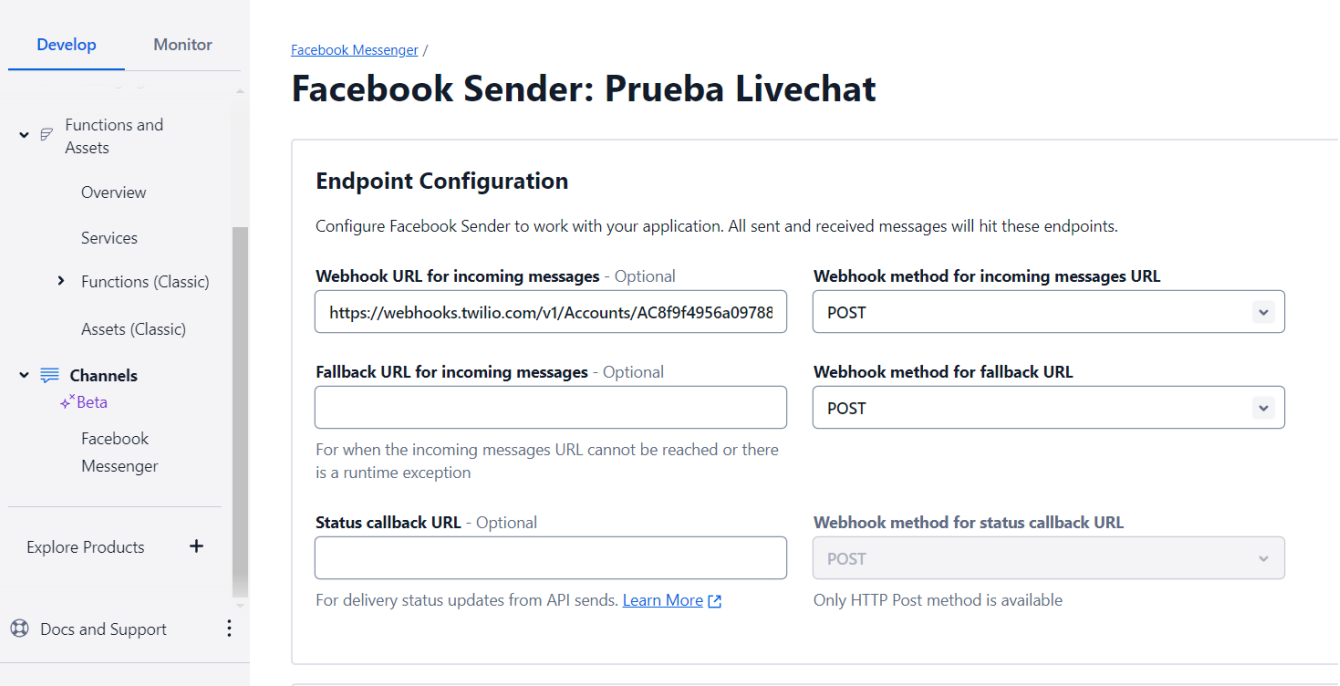
Channels:

Los canales son una colección de integraciones de terceros que te permiten enviar y recibir mensajes en múltiples canales con las API de Twilio. En la actualidad, Facebook Messenger (Beta Publica) y la API de WhatsApp Business son compatibles. Twilio permite utilizar Facebook Messenger a través de *Twilio Programmable Messaging* y WhatsApp a través de *Twilio Programmable Messaging y, Conversations*

Para usar el canal de Facebook es necesario hacer *login* con la cuenta de negocio y esta cuenta de negocio debe tener creada previamente una página de Facebook para la conexión

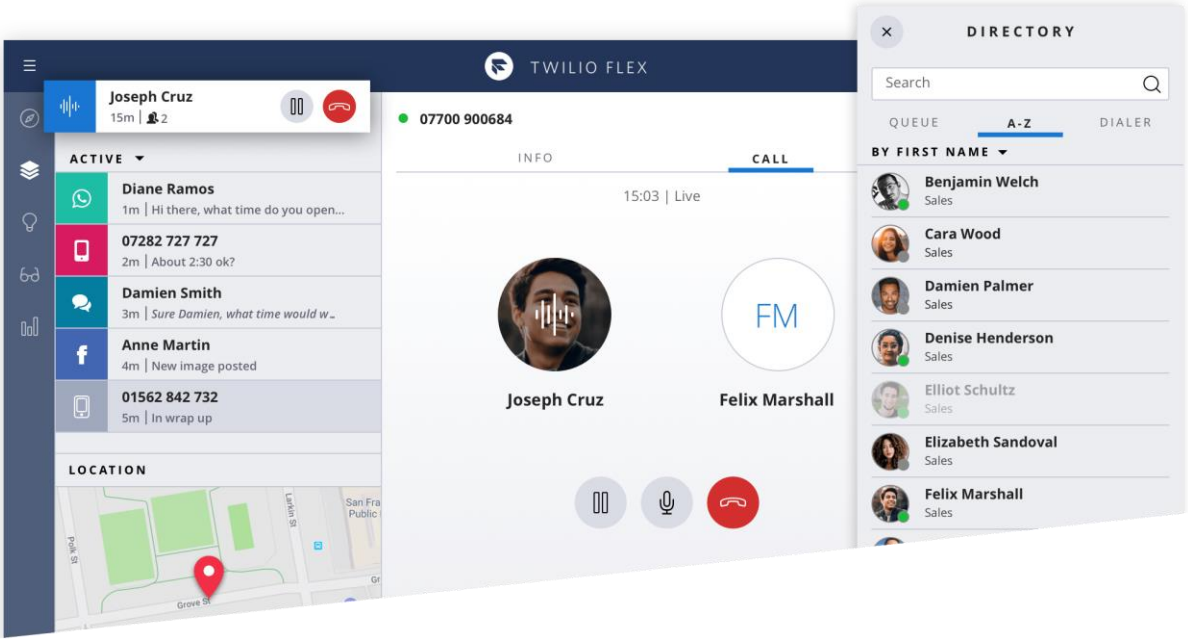


Una vez realizada la conexión en la parte de editar se puede hacer la conexión al disparador (Trigger) del flujo del studio mediante la URL del webhook al igual que con el canal de WhatsApp



Flex:

Flex es la plataforma de twilio para manejar todo el tema de los agentes humanos. Permite conectarse con el flujo, haciendo posible la transferencia del bot a un agente si el usuario lo requiere, también brinda toda la interfaz donde los agentes pueden atender estas solicitudes y conversar con los usuarios.



Flex se ayuda con el módulo de TaskRouter de twilio que brinda toda la configuración para el ruteo de los usuarios mediante colas de trabajo programables. La creación de agentes; que son llamados workers en TaskRouter, asignación de prioridad y tiempos de espera todo esto se configura en TaskRouter.

Notas:

- La plataforma Flex brinda inicialmente una base muy sencilla; sin muchas funcionalidades, pero brinda un sistema flexible donde el programador puede modificar la plataforma y agregarle funcionalidades personalizadas según sus necesidades o integraciones con sistemas externo a twilio.

- Flex cobra 1 dólar por hora de asesor en la plataforma al día de elaboración de este informe, lo cual supone un gran costo en para cliente.

- Para poner conectar los flujos a Flex se deben cambiar algunas configuraciones:

  1. Agregar el canal al apartado de address que se encuentra en el módulo de Flex, sub menú manage, messaging. Estas direcciones se tienen agregar que mediante una petición HTTP a la API de twilio. Si se agregan mediante la interfaz no funcionaran.

     Ejemplos:

     Url de petición: https://webhooks.twilio.com/v1/Accounts/<< **Account SID** >>/Proxy/<< **Flex Proxy Service SID**>>/Webhooks/Message

     Para conectar whatsapp:

     curl -X POST "https://flex-api.twilio.com/v1/FlexFlows" \
     --data-urlencode "ChannelType=whatsapp" \
     --data-urlencode "Enabled=true" \
     --data-urlencode "IntegrationType=studio" \
     --data-urlencode "ContactIdentity=whatsapp**:+<<número de whatsapp**>> " \
     --data-urlencode "FriendlyName=Flex WhatsApp FlexFlow" \
     --data-urlencode "IntegrationFlowSid=<<**SID del flow a conectar con el canal**>> " \
     --data-urlencode 'ChatServiceSid=<<**Chat Service Sid** >>' \
     -u << **Account SID** >>**: <<Auth Token**>>

     Para conectar messenger:

     curl -X POST 'https://flex-api.twilio.com/v1/FlexFlows' \
     --data-urlencode 'ChannelType=facebook' \
     --data-urlencode "IntegrationType=studio" \
     --data-urlencode 'Enabled=true' \
     --data-urlencode 'ContactIdentity=messenger**:<< Facebook Page ID** >> ' \
     --data-urlencode 'FriendlyName=Facebook Messenger Flow' \
     --data-urlencode 'JanitorEnabled=true' \
     --data-urlencode "IntegrationFlowSid=<<**SID del flow a conectar con el canal**>> " \
     --data-urlencode 'ChatServiceSid=<<**Chat Service Sid** >>' \
     -u << **Account SID** >>: <<**Auth Token**>>

  2. Cambiar la URL de los webhooks para los mensajes entrantes tanto para WhatsApp como para messeger; en vez de poner la url del trigger del flujo se cambia por la siguiente url: https://webhooks.twilio.com/v1/Accounts/<< **Account SID** >>/Proxy/<< **Flex Proxy Service SID**>>/Webhooks/Message
     que es la misma donde se hacen la peticiones para crear las direcciones de los canales.

Librería Twilio:

Para enviar un mensaje usando la librería de twilio se requiere el siguiente código en Node JS:

```
const accountSid = process.env.TWILIO_ACCOUNT_SID;

const authToken = process.env.TWILIO_AUTH_TOKEN;

const client = require('twilio')(accountSid, authToken);

client.messages
    .create({
      from: 'whatsapp:+14155238886',
      body: 'Hello there!',
      to: 'whatsapp:+15005550006'
    })
    .then(message => console.log(message.sid));
```

Aparte de Node JS, la librería de twilio es compatible con otros lenguajes de programación como Python, Java, C#, PHP, GO y RUBY. Para conocer como se ejecuta en dichos lenguajes consulte: https://www.twilio.com/es-mx/docs/whatsapp/quickstart

Como se puede observar el *body* es el mensaje que se le va a enviar al destinatario, en esa *string* se puede incluir emojis, en Windows con la tecla Windows + . pueden insertar dichos emojis.

La librería de twilio permite programar mensajes en el futuro en un intervalo de 15 minutos y 7 días. Para hacerlo es necesario el siguiente código en Node JS:

```
const accountSid = process.env.TWILIO_ACCOUNT_SID;

const authToken = process.env.TWILIO_AUTH_TOKEN;

const client = require('twilio')(accountSid, authToken);

client.messages
    .create({
      messagingServiceSid: 'MGXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
      body: 'This is a scheduled message',
      sendAt: new Date(Date.UTC(2021, 10, 30, 20, 36, 27)),
      scheduleType: 'fixed',
      statusCallback: 'https://webhook.site/xxxxx',
      to: '+15558675310'
    })
    .then(message => console.log(message.sid));
```

Como se puede observar las diferencias con respecto al código de enviar un mensaje inmediato es que existe dos parámetros adicionales necesarios llamados messagingServiceSid, sendAt y scheduleType (statusCallback es opcional) el messagingServiceSid reemplaza el parámetro from y es el que indica desde donde se va a enviar el mensaje programado, el sendAt es donde se ingresa la fecha en que se enviará el mensaje, este va en formato UTC completo y por último el parámetro scheduleType indica si el recordatorio se va a programar en una hora fija.

A los mensajes programados se les puede añadir imágenes, con el siguiente código:

```
client.messages
  .create({
    messagingServiceSid: 'MGXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
    body: 'This is a scheduled message with an image url',
    mediaUrl: ['https://c1.staticflickr.com/3/2899/14341091933_1e92e62d12_b.jpg'],
    sendAt: new Date(Date.UTC(2021, 10, 30, 20, 36, 27)),
    scheduleType: 'fixed',
    statusCallback: 'https://webhook.site/xxxxx',
    to: '+15558675310'
  })
  .then(message => console.log(message.sid));
```

El único parámetro adicional es mediaUrl, donde se pasa el link de la imagen que será enviada. Estos ejemplos están hechos en JavaScript, pero es compatible con los demás lenguajes mencionados anteriormente, para más información consultar: https://www.twilio.com/es-mx/docs/sms/api/message-resource#schedule-a-message-resource

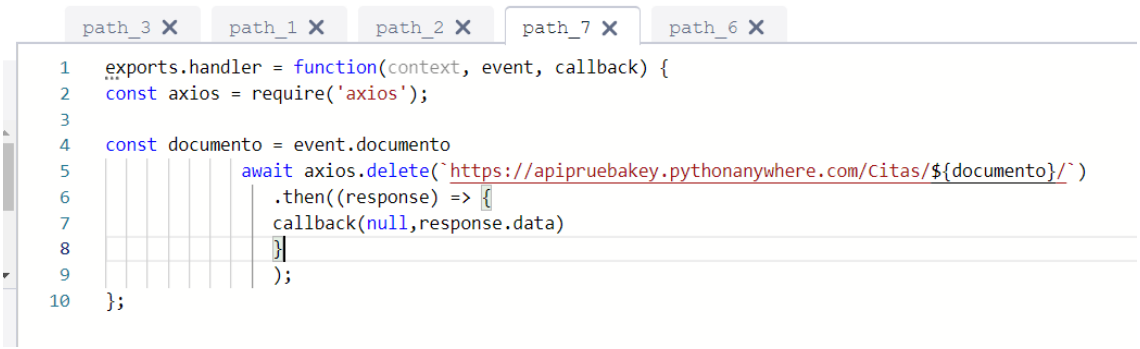
Tutorial

En este pequeño tutorial se mostrará como construir un flujo desde cero conectado con la plataforma Flex y los canales de comunicación de WhatsApp y Facebook Messenger.

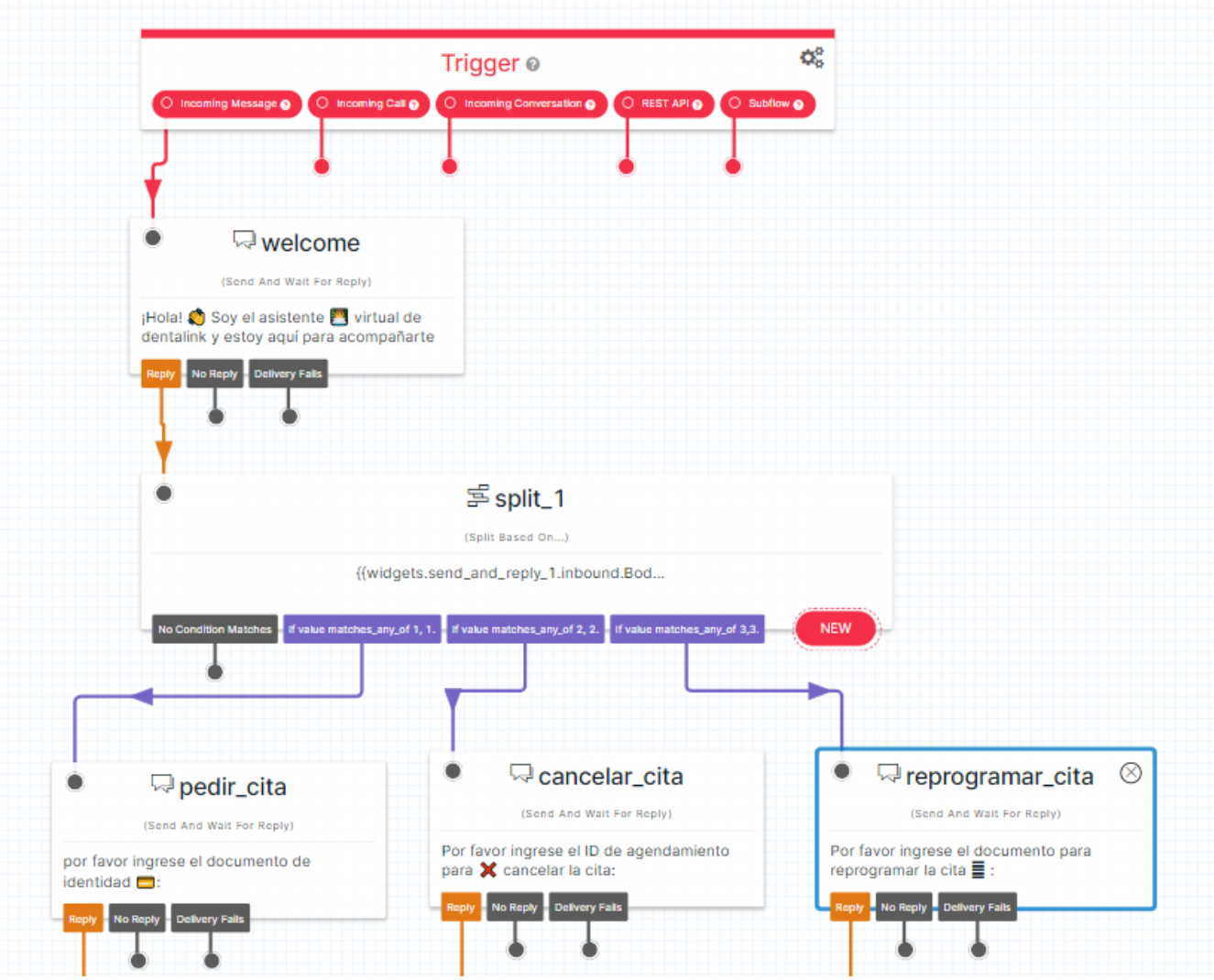
El propósito del flujo será permitir a un usuario, cancelar una cita médica mediante WhatsApp y Facebook Messenger y comunicarse con un agente.

1. Conectamos nuestro propio número de whatsApp a WhatsApp sender .
2. Conectamos nuestra página de Facebook a Messenger sender.
3. Crear servicio para cancelar cita en la base de datos.

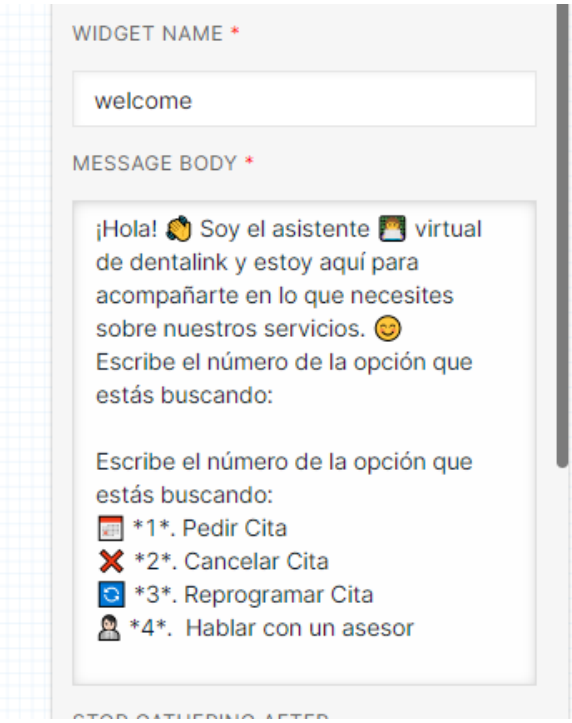
   Como se ve en la siguiente imagen, usamos la librería de axios para modificar la base de datos mediante una API



4. Crear nuestro flujo en studio:



Empezamos con el *trigger*, añadiendo la transición de *incoming message* a nuestros mensajes de bienvenida que será un *send and wait for reply*, donde le brindaremos al usuario las múltiples opciones del flujo.



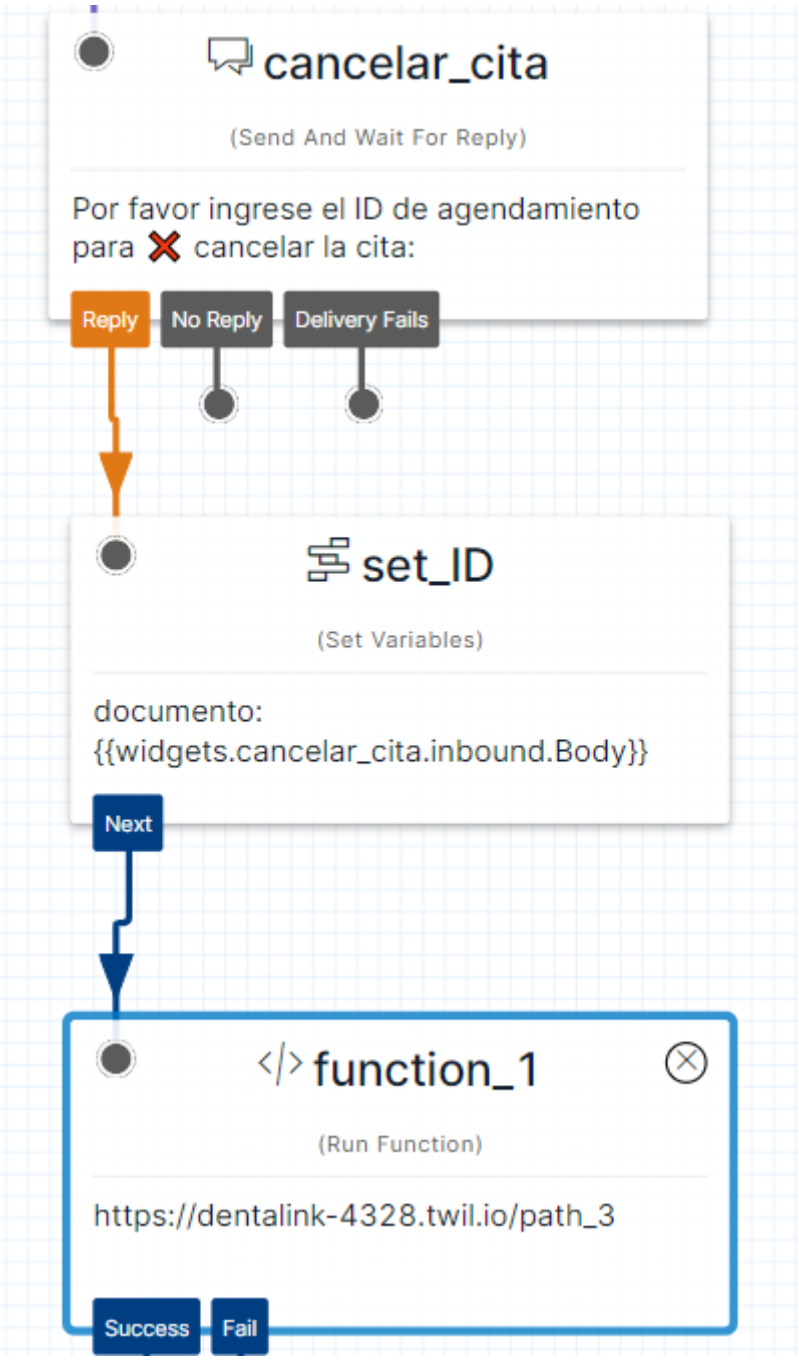
Una vez que el usuario responda, seguiremos en el flujo con un Split evaluando la respuesta del usuario.

Que se guardara en la siguiente variable del flujo:

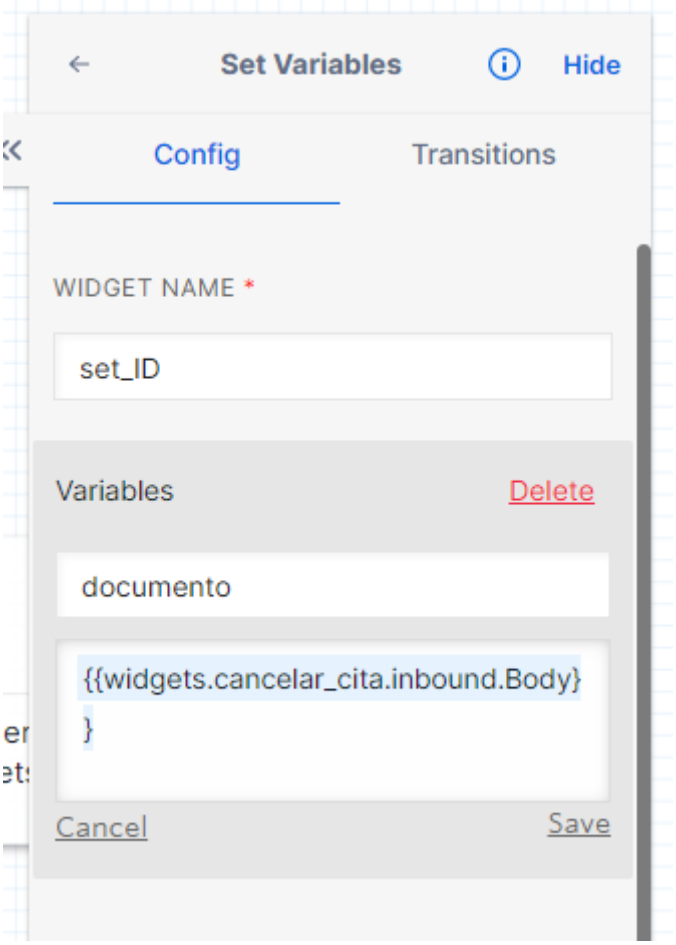
{{widgets.send_and_reply_1.inbound.Body}}

según la respuesta del usuario el flujo continuara por las diferentes transiciones del Split.

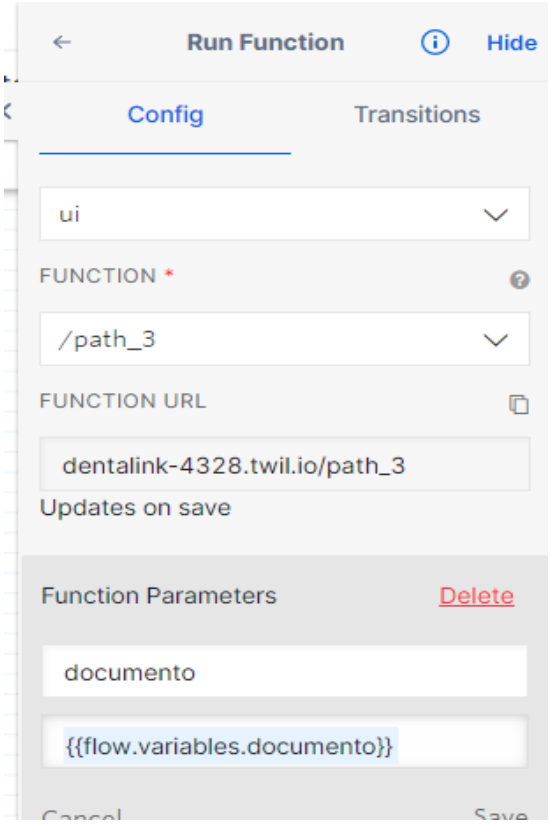
Continuaremos por la rama de cancelar cita:



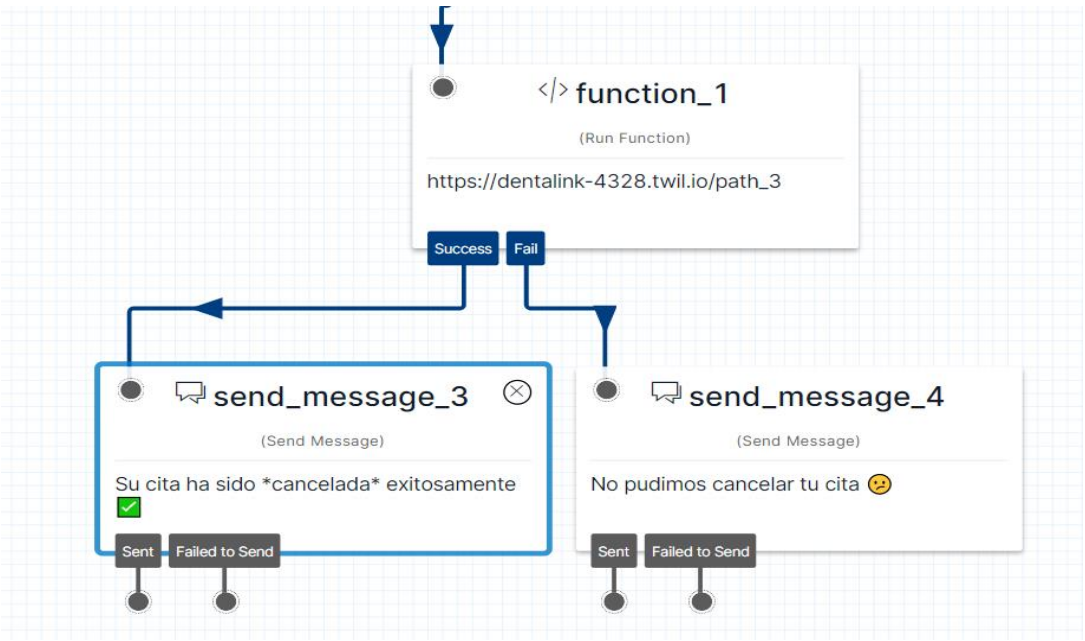
Cuando el usuario ya entro en la rama de cancelar cita, se le pedirá un identificador para cancelar la cita. El cual será guardado en una variable para su posterío uso.



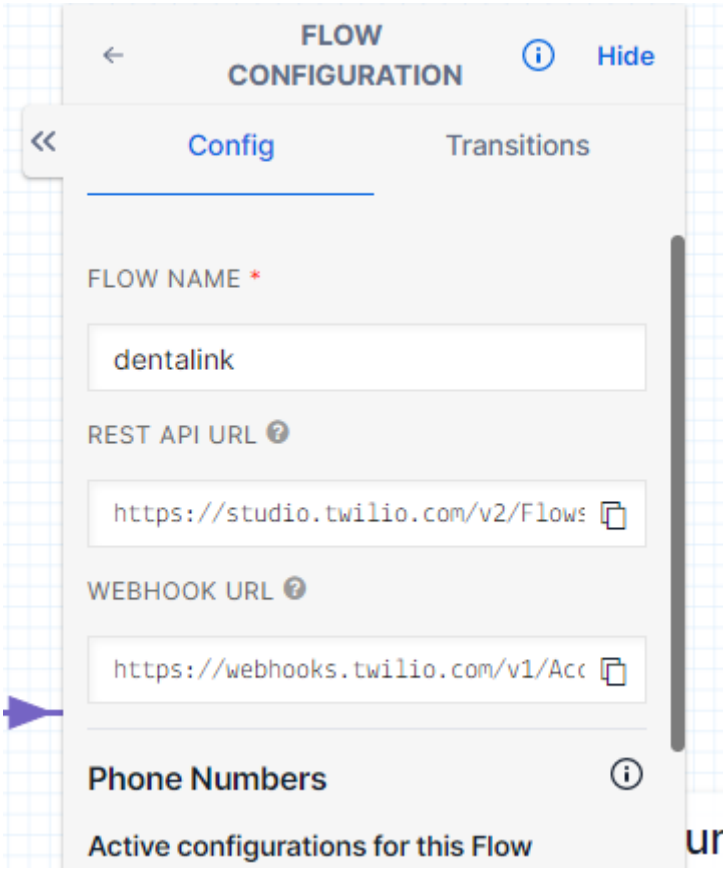
Una vez guardo el identificador de la cita a cancelar se pasa a llamar a la función que se encargara de la lógica de cancelar la cita, pasándole como parámetro la variable guardada con el identificador.



Por el ultimo, manejamos las posibles respuestas de la función como *success* avisándole al usuario que su cita fue cancelada exitosamente y *fail* informado que algo sucedió mal y no se pudo cancelar la cita.



5. Buscar la url del webhook del trigger del flujo:



6. Poner la url del trigger en whatsapp sender y en Facebook messeger en webhook URL for incoming messages:

WhatsApp Sender: +573245634749

Status

Connected to WhatsApp

Endpoint configuration

Configure WhatsApp to work with your application. All sent and received messages will hit these endpoints.

How would you like to configure this sender?

- ○ Use a Messaging Service (recommended) Learn more
- ◉ Use webhooks

Webhook URL for incoming messages - Optional
https://webhooks.twilio.com/v1/Accounts/AC8f9f4956a097

Webhook method for incoming messages URL
HTTP Post

Fallback URL for incoming messages - Optional
https://example.com/fallback

Webhook method for fallback URL
HTTP Post

For when the incoming messages URL cannot be reached or there is a runtime exception

Facebook Messenger /

Facebook Sender: Prueba Livechat

Endpoint Configuration

Configure Facebook Sender to work with your application. All sent and received messages will hit these endpoints.

Webhook URL for incoming messages - Optional
https://webhooks.twilio.com/v1/Accounts/AC8f9f4956a09788

Webhook method for incoming messages URL
POST

Fallback URL for incoming messages - Optional

Webhook method for fallback URL
POST

For when the incoming messages URL cannot be reached or there is a runtime exception

Status callback URL - Optional

Webhook method for status callback URL
POST

For delivery status updates from API sends. Learn More

Only HTTP Post method is available

Conclusiones

Con la exploración de Twilio podemos concluir que es una herramienta útil para los siguientes casos de uso:

1. Notificar a clientes: ya sea mediante mensajes de texto (SMS) o mensajes de WhatsApp cosas como recordatorios de citas, notificaciones de entrega, actualizaciones de estado de pedidos, confirmaciones de transacciones, entre otros.

2. Líneas telefónicas virtuales: proporcionando números telefónicos virtuales que permiten a las empresas establecer líneas de atención al cliente, líneas de soporte técnico o líneas de ventas sin la necesidad de configurar una infraestructura telefónica física. Estas líneas telefónicas pueden reenviar llamadas a números existentes o utilizarse con IVR (respuesta de voz interactiva) para enrutar y gestionar las llamadas de manera eficiente

3. Autenticación y verificación de usuarios: Twilio ofrece servicios de autenticación de dos factores (2FA) y verificación de números telefónicos. Esto permite a las empresas agregar una capa adicional de seguridad a sus aplicaciones y servicios, asegurando que los usuarios sean verificados correctamente antes de acceder a ciertas funcionalidades o información confidencial.

4. Integración en un backend independiente: La librería de Twilio permite que se implemente en diversos lenguajes y en plataformas diferentes a twilio por medio de código, brindando flexibilidad y escalabilidad en la implementación en un sistema diferente.

5. Twilio Flex: Una plataforma de centro de contacto basada en la nube de Twilio que permite personalizarse al 100%, adaptándose a las necesidades específicas del negocio ofreciendo escalabilidad e integraciones con CRMs y permitiendo que los agentes atiendan canales múltiples como lo son SMS, WhatsApp, Facebook y línea telefónica.

Si bien existen mas casos de uso que ofrece twilio, estos cinco son los que mas destaca en el mercado entre opciones que no ofrecen dichas funcionalidades. Aun así, hay que tener en cuenta el precio sobre todo de el ultimo caso de uso que puede llegar a ser muy costoso para empresas pequeñas o con poca demanda de usuarios al mes. Si no es necesario el uso de Flex puede ser una gran opción para empresas pequeñas ya que se cobra por mensaje enviado un precio muy mínimo según sea el canal.

Lista de precios hoy 25/05/2023

WhatsApp: A partir de USD 0,0042 para enviar un mensaje de plantilla de WhatsApp y USD 0,005 para mensajes de sesión de WhatsApp.

SMS: A partir de USD 0,0079 para enviar o recibir un mensaje.

Flex: USD 1 por hora de agente activo o USD 150 por agente nombrado al mes.

Voz programable: A partir de USD 0,0085 por minuto para recibir llamadas y USD 0,013 por minuto para realizar llamadas.

SMS Verificación/autenticación: A partir de USD 0,05 por verificación.