E.T.S. de Ingeniería Industrial, Informática
y de Telecomunicación

Creación de un chatbot para atender peticiones sanitarias

Grado en Ingeniería Informática

Trabajo Fin de Grado

Óscar Del Barrio Farran

Daniel Paternain Dallo

Pamplona, 06/09/2021

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen

Un chatbot es un asistente que se comunica con los usuarios a través de mensajes de texto. De cada vez más, las empresas apuestan por este canal digital de comunicación para dar servicio a sus clientes.

En este Trabajo Fin de Grado se ha realizado un chatbot con múltiples funcionalidades que permita satisfacer las peticiones sanitarias de un cliente. La primera parte ha consistido en la documentación sobre las funcionalidades de un chatbot y, lo más importante, qué herramientas utilizar para poder implementarlo con éxito. Una vez asentados los conceptos básicos, la siguiente tarea ha consistido en realizar un prototipo utilizando las herramientas escogidas y hacer pruebas con él de cara a la implementación final. Por último, se ha trabajado en el chatbot final, realizando las modificaciones pertinentes con el fin de integrarlo en un servicio público para poder evolucionar el prototipo a un chatbot más completo.

Palabras clave

- Chatbot

- NLP: Procesamiento del Lenguaje Natural

- Telegram

- DialogFlow

- AWS: Amazon Web Services

- SoapUI

Índice

Capítulo 1

Introducción y objetivos

1.1 Contextualización

Pensar en una conversación entre un robot y una persona, hace años era cosa de ciencia ficción. Sin embargo, gracias al crecimiento y evolución de los chatbots y, sobre todo, al desarrollo de la inteligencia artificial (IA), se ha conseguido que se convierta en una acción cotidiana para todas las personas. La importancia de la inteligencia artificial es tal que, gracias a ella, los chatbots son capaces de participar cada vez más en conversaciones naturales y entablar relaciones con los usuarios como si de un humano se tratara [1].

La inteligencia artificial la conocemos como una amplia rama de la informática que se ocupa de la construcción de máquinas inteligentes capaces de realizar tareas que normalmente requieren inteligencia humana. Su funcionamiento es sencillo, de manera resumida, la máquina recibe datos (preparados previamente), los procesa y responde a ellos [2].

Gracias a los avances en la potencia informática, la disponibilidad de enormes cantidades de datos y la aparición de nuevos algoritmos, han permitido que se den grandes avances en la transformación digital de la sociedad. Muchos son los sectores los que aplican la IA en su trabajo cotidiano [3]:

- *Publicidad*: en este sector, se busca que el anuncio publicitado llegue al usuario en el momento adecuado y con un contenido apropiado que lo posicione en un lugar destacado. La inteligencia artificial es una técnica muy eficaz y poderosa para lograrlo. Gracias a algoritmos predictivos que anticipen cuál va a ser el comportamiento y las demandas del usuario, se van a conseguir campañas publicitarias totalmente personalizadas a las expectativas, necesidades y gustos individuales consiguiendo de esta manera un mayor éxito y un considerable aumento de las ventas [4].

- *Ciberseguridad*: los sistemas de inteligencia artificial son clave para ayudar a reconocer y luchar contra ciberataques y posibles amenazas que atenten contra nuestra privacidad. Dichos sistemas, se basan en los datos que reciben continuamente, reconociendo patrones y de esta forma, impidiendo ataques. Gracias a ello, eliminamos las tareas de investigación que por lo general son muy costosas y también se facilita la tarea de los analistas reduciendo el tiempo que se necesita para tomar decisiones críticas y neutralizar amenazas. En definitiva, se consigue una mayor velocidad de respuesta evitando así posibles catástrofes informáticas [5].

- *Automoción*: cuando pensamos en la inteligencia artificial aplicada a la automoción, la conducción autónoma es lo primero que se nos viene a la mente. Dicha tecnología, es a la que más importancia se le está dando en la actualidad, ya que para muchas industrias automovilísticas es el futuro. Los vehículos se están transformando rápidamente en dispositivos conectados gracias a las conexiones vía wifi, bluetooth…

La tecnología IA más avanzada, que se está incorporando de manera habitual al sistema de los vehículos es la de los asistentes de voz, utilizada para ayudar al conductor en cualquier situación al volante. Como se puede observar, año tras año se buscan diferentes maneras de mejorar la experiencia del conductor utilizando como pilar la inteligencia artificial [6].

- *Asistente personal*: es uno de los usos cada vez más comunes de la inteligencia artificial. Su función es clara, responder a instrucciones y preguntas impuestas por un usuario para ayudar a completar tareas. Los ejemplos más reconocibles son los asistentes de voz de los smartphones (Siri), los altavoces inteligentes (Alexa) y los chatbots. Para estos asistentes, la utilización del procesamiento del lenguaje natural es de vital importancia para entender el contexto de las preguntas, interpretar las voces humanas y en definitiva, para que se puedan dar conversaciones más naturales entre las personas y sus dispositivos [7].

Tras haber visto varios ejemplos sobre la aplicación de la inteligencia artificial en nuestra sociedad, nos quedamos con este último sobre los asistentes personales. Vemos que se introduce uno de los campos más conocidos de la inteligencia artificial como es el Procesamiento del Lenguaje Natural (NLP), que se ocupa de investigar la manera de comunicar las máquinas con las personas a través de lenguas naturales, como pueden ser el español, el inglés o el chino.

La información que recibe el asistente para que sea procesada puede expresarse, o bien por escrito (texto), o bien de forma oral (voz). El procesador del lenguaje natural por lo general está más desarrollado en el tratamiento de textos, donde hay más facilidades para tratar los datos al no tener que realizar ningún proceso extra. En cambio, para procesar los audios el proceso es más costoso, ya que es necesario transcribirlos en letras o caracteres y, a partir de ahí, entender la petición del usuario para darle respuesta [8].

Podemos diferenciar dos fases principales para el procesamiento del lenguaje natural, el preprocesamiento de datos y el desarrollo de algoritmos.

El preprocesamiento de datos implica preparar y “limpiar” la información recibida para que la máquina pueda analizarla sin coste alguno. Existen una serie de técnicas para realizar dicho proceso: la tokenización, la eliminación de palabras frecuentes que no aportan semántica al texto (stop-words) y lemmatization o stemming.

Una vez los datos han sido preprocesados con la ayuda de las técnicas nombradas anteriormente, es necesario el desarrollo de algoritmos para procesarlos. Finalmente, se da respuesta al usuario que previamente ha realizado la consulta y se concluye así con el proceso de procesamiento de la información [9].

1.2 Creación chatbot

Se ha podido comprobar tras la contextualización realizada en el anterior apartado que, tanto la inteligencia artificial como el procesamiento del lenguaje natural son clave para el desarrollo de cualquier asistente personal, más concretamente de un chatbot. En este apartado, se va a exponer una explicación más detallada de la implementación de un chatbot indicando las diferentes fases de desarrollo y lo necesario para su implementación.

Un chatbot está compuesto por los siguientes elementos: una app de mensajería para recibir la información a procesar, un procesador del lenguaje natural que dé respuesta a la información recibida y, por último, un servicio web que permita realizar consultas más complejas (Ver Figura 1).

Figura 1. Arquitectura de un chatbot

En cuanto a las aplicaciones de mensajería, prácticamente todo el mundo conoce alguna, como pueden ser WhatsApp, Telegram o Signal. A la hora de elegir la aplicación, es interesante que cualquier usuario pueda acceder y que esta proporcione permisos para trabajar con la API con la mayor libertad posible, ya que de esta forma se podrá realizar un chatbot con una funcionalidad plena.

El procesador del lenguaje natural es el encargado de recibir los mensajes escritos por el usuario, procesarlos y responder como si de un humano se tratase. Para ello, se deberá conectar la app de mensajería, donde se ha creado el chatbot, con el procesador del lenguaje natural elegido para que reciba dicha información y la procese.

El programador de forma manual deberá indicar el árbol de decisión que debe seguir el procesador para dar respuesta al usuario de la manera más lógica posible. La capacidad de un procesador es muy limitada, ya que no es posible realizar a través de él una programación profunda que permita realizar consultas a bases de datos o servicios web. Simplemente se permiten respuestas simples sin ningún tipo de información trascendente para el usuario.

Para dotar a un chatbot de una funcionalidad plena, es imprescindible la implementación de un servicio web que sustente las respuestas lógicas del chatbot al usuario. Para realizar la conexión entre el servicio web y el procesador del lenguaje natural, es esencial la utilización de webhooks, canales utilizados para enviar información en tiempo real a otras aplicaciones. A través de ellos, el procesador enviará el mensaje recibido por el usuario para que el servicio web lo reciba y lo procese dándole respuesta y, por último, enviando la información de vuelta para que se la entregue al usuario.

Tras haber explicado todo el proceso de creación del chatbot, queda claro que se necesita una buena base informática para conseguir que un simple chatbot pueda convertirse en una máquina inteligente capaz de seguir una conversación con un ser humano [10].

1.3 Objetivos

El objetivo de este Trabajo Fin de Grado (TFG) es el estudio y desarrollo de un chatbot para una app de mensajería mediante un procesador del lenguaje natural para atender peticiones sanitarias a través de respuestas automatizadas.

Para conseguir este objetivo, se han planteado los siguientes subobjetivos:

- Realizar un análisis de mercado de apps de mensajería y procesadores del lenguaje natural.

- Desarrollar un chatbot utilizando Telegram como app de mensajería, DialogFlow como plataforma de desarrollo NLP y Amazon Web Services como servicio web.

Capítulo 2

Análisis de mercado de aplicaciones de mensajería y de procesadores del lenguaje natural

Antes de proceder al desarrollo del proyecto de creación del chatbot, he creído conveniente realizar un análisis de mercado sobre qué aplicación de mensajería y qué plataforma de desarrollo NLP utilizar para obtener un chatbot con las condiciones que demanda el proyecto.

2.1 Aplicaciones de mensajería

En cuanto a las apps de mensajería la idea es clara, llegar al mayor número de usuarios posible. Por ello, el análisis lo he centrado en dos aplicaciones que ahora mismo son las más utilizadas del mundo como son WhatsApp [11] y Telegram [12]. Ambas proporcionan lo necesario para la creación de un chatbot, pero es necesario analizarlas bien para encontrar aquellas diferencias que permitan decantar la balanza.

2.1.1 WhatsApp

Es la aplicación de mensajería más utilizada en el mundo con más de 5.000 millones de descargas, es gratuita y tiene infinidad de posibilidades como enviar mensajes de texto, fotografías, audios, vídeos, etc. Además, también tiene la opción de realizar llamadas o incluso videollamadas tanto individuales como grupales sin coste alguno. En la actualidad, WhatsApp se considera el mejor canal de comunicación y, por el momento, nadie es capaz de alcanzar su enorme progresión.

A principios de 2018, WhatsApp presentó "*WhatsApp Business*" como una aplicación paralela a la original diseñada para atender las necesidades de las pequeñas empresas (ver Figura 2). Gracias a esta aplicación, se consigue facilitar la comunicación con los clientes, enseñar los productos y servicios al cliente de manera instantánea, responder a las preguntas que tengan durante la experiencia de compra... En definitiva, es la aplicación ideal para dar a conocer un negocio y evolucionar en el ámbito internacional aprovechando su gran cobertura mundial.

Figura 2. Ejemplo WhatsApp Business

Pero lo realmente interesante de esta nueva aplicación, y sobre lo que realmente interesa trabajar, es la posibilidad de crear una herramienta para automatizar, ordenar y responder mensajes rápidamente, es decir, la creación de un chatbot. Para ello, se utiliza la API de WhatsApp Business, diseñada y habilitada solamente para medianas y grandes empresas, es decir, si eres un simple aficionado a la programación o no formas parte de una empresa reconocida, no obtendrás permisos de desarrollo ni podrás acceder a la API de WhatsApp.

Para obtener los permisos necesarios existen dos posibilidades, o bien inscribirse directamente, o bien usando un proveedor de soluciones. Dichos proveedores usan la integración técnica de la API de WhatsApp Business para brindar asistencia a cada negocio, y también se encargan de crear y administrar cada cuenta de WhatsApp Business.

En el caso de realizar el trámite por cuenta propia, el responsable de la empresa deberá rellenar un formulario (ver Figura 3) que será enviado a Facebook, donde se encargarán de tramitar la solicitud y verificar si realmente la empresa que demanda sus servicios es o no apta. Pero previamente se debe conocer que, por lo general, intentarlo directamente tiene una baja probabilidad de éxito, ya que a menos que la marca de la empresa tenga una buena relación con Facebook o sea reconocida internacionalmente, esta vía no es particularmente efectiva. En cambio, trabajar con un proveedor de soluciones de WhatsApp Business ayudará a que la empresa tenga una mayor posibilidad de aprobación y acelerará el proceso.

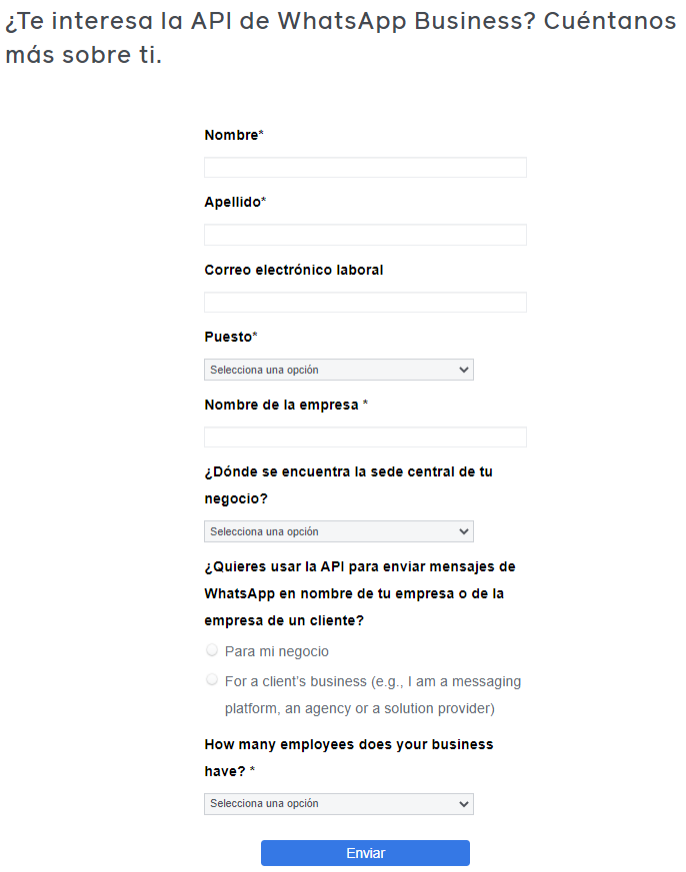

Figura 3. Formulario de solicitud

Una vez recibida la aprobación, se deberá facilitar más información para activar el número de teléfono asociado a la empresa en la aplicación de WhatsApp Business. Este proceso puede llegar a tardar hasta 2 o 3 semanas. Se demanda la siguiente información:

- La identificación del Business Manager de clientes en Facebook.
- El número de teléfono de tu cuenta de WhatsApp Business.
- El nombre asociado al número de WhatsApp.
- La región de tu aplicación (EEUU o UE).

Tras la verificación de todos los datos y la activación del número de teléfono por parte de WhatsApp, todas las comunicaciones pasarán a alojarse en un servidor virtual. Por lo que ya no se tendrá acceso a los mensajes de los usuarios desde el móvil, sino que se gestionarán en el panel de control o en el sistema del proveedor de soluciones elegido. En el caso de haber solicitado la API de WhatsApp Business por cuenta propia, la propia empresa deberá crear su propio servidor virtual. Puede suponer un gran desafío si no se disponen de los conocimientos necesarios para ello.

Por último, faltaría por crear las plantillas de mensajes (ver Figura 4), es decir, formatos de mensajes específicos que se usan para dar respuesta a través de notificaciones o mensajes de atención al cliente a las personas que hayan realizado la consulta. Para ello, se debe tener en cuenta las reglas establecidas por la propia compañía de Facebook en lo que a la forma y contenido se refiere. Tanto es así, que debemos recibir la aprobación de WhatsApp para poder poner en uso nuestros mensajes. Una vez que se haya aprobado el mensaje, se permite comenzar a trabajar con el desarrollador para agregar las plantillas de mensaje a la API y finalizar de este modo la implementación de un chatbot en WhatsApp.

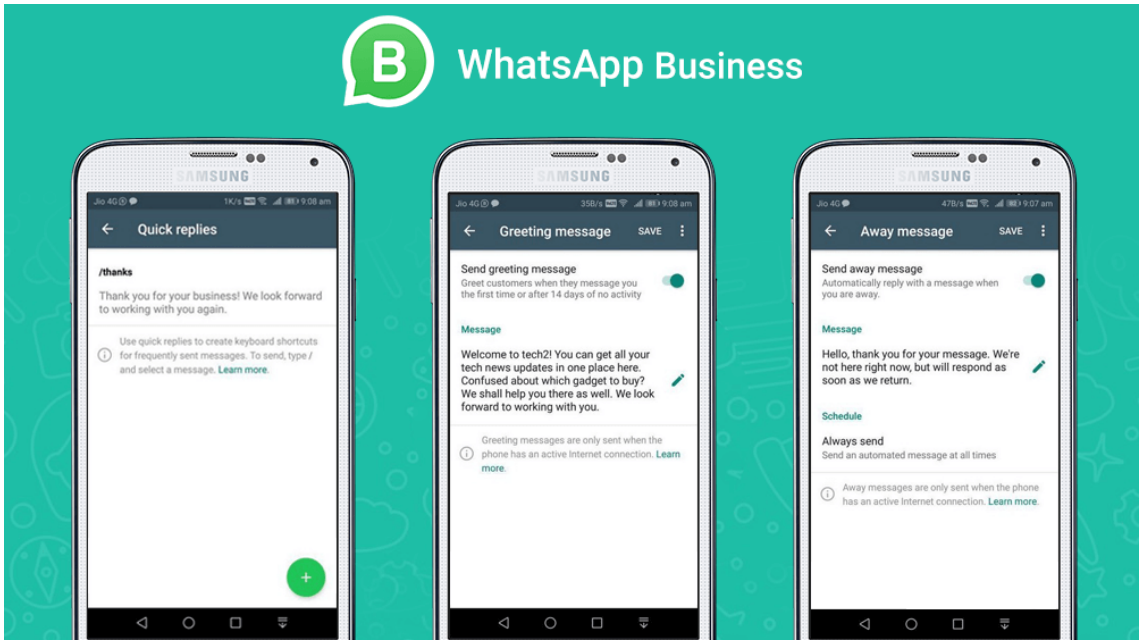

Figura 4. Ejemplos plantillas WhatsApp Business

2.1.2 Telegram

Es una de las aplicaciones de mensajería que mayor evolución ha tenido a lo largo de estos años, llegando a más de 500 millones de descargas. Al igual que en WhatsApp, su uso es gratuito y se tiene la posibilidad de enviar cualquier tipo de mensaje de texto, fotografías, audios, vídeos, etc. En 2020, Telegram buscó incorporar las videollamadas a su sistema de llamadas, aunque solamente de persona a persona. En la actualidad, Telegram es una de las apps de mensajería más populares y, aunque no tenga la misma cantidad de usuarios que otras aplicaciones, sigue siendo una excelente opción.

Algo que caracteriza esta plataforma, y que la diferencia de otros, es la variedad de trucos en cuanto a temas, chats secretos, soporte multiplataforma y la capacidad de unirte a canales y supergrupos de diferentes categorías, permitiendo así poder conectarse con miles de personas a nivel mundial. Con Telegram se pueden hacer muchas más cosas que sólo chatear o enviar mensajes de voz, gracias a su plataforma de bots que permite tener infinidad de funcionalidades como reproducir música, simular una calculadora, un traductor, etc.

Usar los bots en Telegram es sencillo, aunque antes de interactuar con alguno de ellos se deben iniciar entrando en sus perfiles de usuario y pulsando el botón de Iniciar. Existen dos posibilidades para acceder a estos bots, o bien a través de un enlace directo, o bien usando el buscador incorporado en la aplicación.

En Telegram hay básicamente dos tipos: aquellos que funcionan directamente mientras escribes, llamados inline bots, y aquellos que debes usar interactuando en un chat privado o a través de comandos, considerados los normal bots.

- Inline bots: en este caso, en el mensaje que escriba el usuario se deberá mencionar al bot, añadiendo a continuación la consulta. Entonces se recibe por parte del bot un resultado o sugerencia que no se enviará al chat hasta que el usuario la acepte (Ver Figura 5).

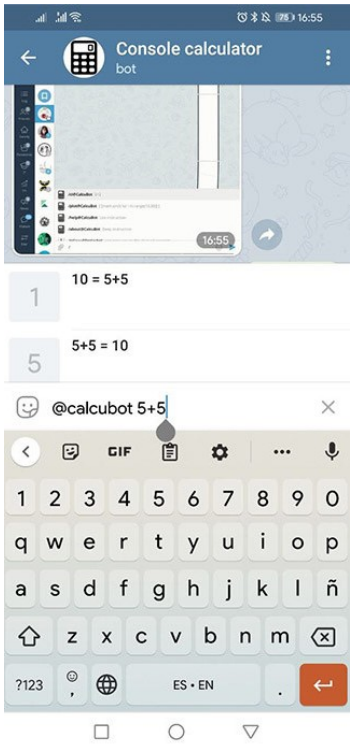

Figura 5. Ejemplo de Inline Bot

- Normal bots: se debe interactuar con ellos mediante mensajes privados o, si se encuentra en un grupo, mediante comandos en el chat. En este caso, se envía la consulta como si fuese un mensaje normal (Ver Figura 6).

Figura 6. Ejemplo de Normal Bot

Tras haber analizado y comprendido la utilidad de los bots en Telegram, llegamos a la parte más importante, la creación de chatbots propios. Es una funcionalidad incorporada en la aplicación que permite al usuario realizar sus propias creaciones con una libertad de decisión absoluta y sin necesidad de ningún tipo de permiso por parte de la compañía. Dicho proceso se realiza a través de un "*BotFather*" programado por Telegram para dar servicio a los usuarios y controlar todos los bots.

El proceso es muy sencillo, lo primero que se tiene que hacer es iniciar una conversación con el "*BotFather*". Tras iniciarla, se desplegará una lista con todos los comandos que entiende el bot, y con ellos, se podrá crear un bot propio haciendo todo lo necesario para configurarlo y ponerlo en marcha (Ver Figura 7).

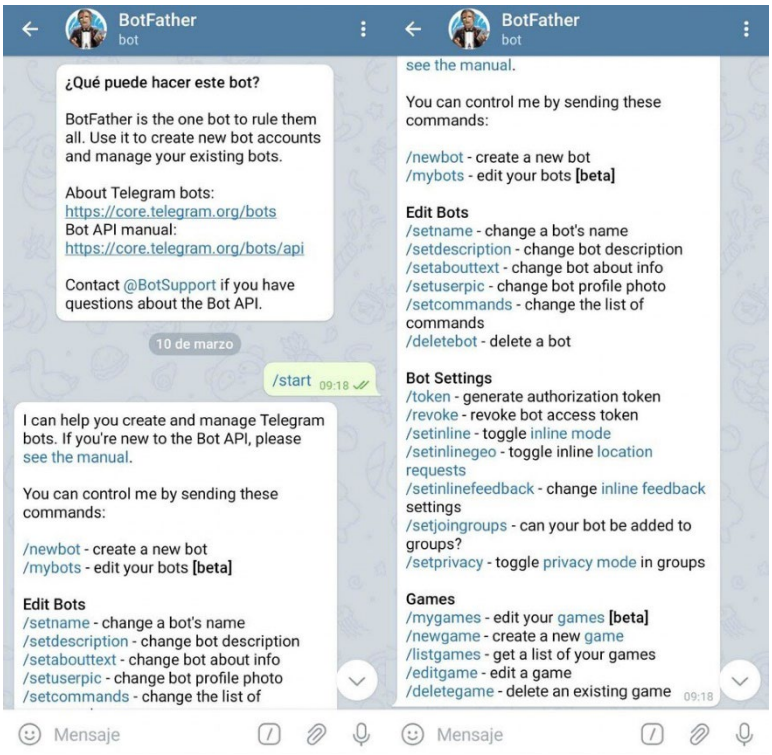

Figura 7. Ilustración comandos BotFather

Para crear uno nuevo se utilizará el comando *"/newbot*" dándole un nombre y un nombre de usuario. Acto seguido, el "*BotFather*" proporcionará un token de autorización, un código alfanumérico, que será necesario para que el bot emplee la API de Telegram (Ver Figura 8).

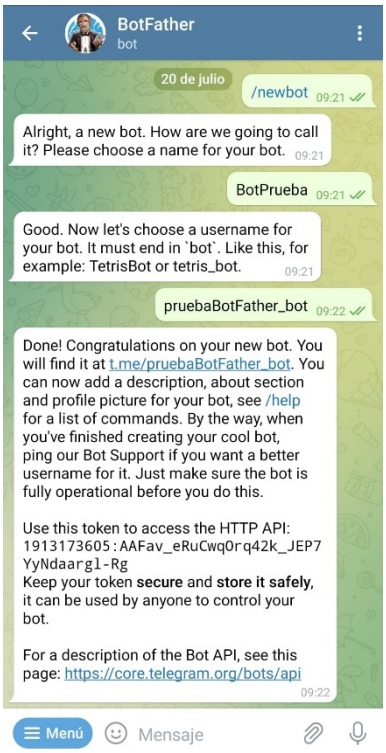

Figura 8. Proceso de creación chatbot

En cuanto a su comportamiento, el usuario tendrá la libertad de definir ante qué comandos responderá y qué flujo de conversación debe seguir. Una vez se ha completado el proceso de creación, se da por finalizada la implementación de un chatbot propio en Telegram.

2.1.3 Conclusiones

Tras haber analizado ambas plataformas de mensajería, tengo bastante claro que la aplicación idónea para implementar un chatbot es Telegram. A pesar de que WhatsApp es la app de mensajería por excelencia, su servicio deja mucho que desear. Básicamente, todo son pegas a la hora de pedir permisos para desarrollar y como he comentado en su correspondiente apartado, a no ser que seas una empresa mediana o grande, no tienes ninguna posibilidad de obtener los permisos para implementar un chatbot propio en la aplicación. Además, en el caso de obtener los permisos de desarrollo, cada plantilla de mensajes que se programe debe pasar por el filtro de Facebook para poder ser implementada, algo que resulta muy molesto e irritante, ya que la libertad del programador se ve ofuscada.

En cambio, Telegram ofrece una libertad de programación envidiable donde cada usuario tiene la potestad de decidir qué tipo de chatbot crear, sin necesidad de aprobaciones de altos cargos y también el poder decidir cuándo hacerlo sin tiempos de espera interminables. En definitiva, si lo que se busca es un desarrollo de un asistente virtual que dé respuesta a cualquier consulta con total libertad, Telegram es sin duda la aplicación escogida.

2.2 Plataformas de desarrollo NLP

Las plataformas de desarrollo NLP nos van a permitir construir chatbots más avanzados, capaces de incorporar funcionalidades más completas [13]. Hay bastantes plataformas que realizarían la función que deseamos sin ningún problema, no obstante, voy a realizar un estudio sobre alguna de las más usadas con el fin de decidir qué plataforma sería la idónea para implementar nuestro proyecto. Cada una de ellas, tiene sus pros y sus contras, y su mayor o menor idoneidad va a depender tanto del objetivo y funcionalidad del chatbot que se quiere construir como de los recursos disponibles.

2.2.1 DialogFlow

Adquirida por Google en septiembre de 2016, DialogFlow es una de las plataformas de desarrollo de chatbots más utilizada en el momento. Destaca el uso de intents (intenciones) y entities (entidades), además de la gestión de subcontextos, en base a los intents detectados. La utilización de subcontextos es clave para la detección de otros intents. En el siguiente capítulo abordaremos con más profundidad el uso de intents y entities para entender mejor su funcionamiento.

La plataforma permite el despliegue de sus chatbots en muchas aplicaciones de mensajería, en páginas web, asistentes virtuales y aplicaciones propias. Algunas de ellas son: WhatsApp, Telegram, Facebook Manager, Alexa, Cortana, etc.

Por último, añadir tres aspectos muy importantes que hacen de la plataforma una de las mejores del mercado como son: la incorporación de herramientas de análisis y monitorización, la posibilidad de integrar otros sistemas como bases de datos, APIs o servicios web, gracias al uso de webhooks, y la permisividad de configuración en algunos aspectos del algoritmo de Machine Learning que se utiliza para el procesamiento de los mensajes.

La excelente documentación aportada por Google y el buen número de ejemplos en forma de videos encontrados en YouTube por parte de otros programadores, permite facilitar la utilización de la plataforma y el desarrollo de chatbots.

En la Figura 9 podemos apreciar visualmente la arquitectura que sigue DialogFlow, explicada anteriormente.

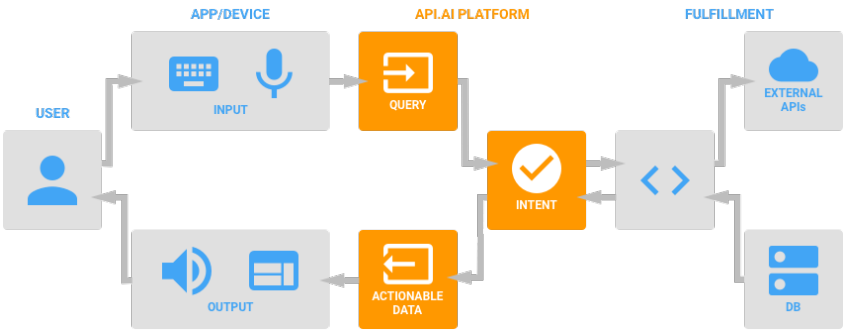

Figura 9. Descripción general de la arquitectura de DialogFlow

2.2.2 Facebook Bot Engine

Es una plataforma creada en 2016 por Facebook, que sirvió para abrir paso a la empresa en el mundo del procesamiento del lenguaje natural. Solo permite el despliegue de chatbots en su propia aplicación de mensajería, "*Facebook Messenger*". Su desarrollo se basa en el uso de intents y entities, e incorpora las llamadas stories (historias), elementos clave para definir el comportamiento y funcionamiento del chatbot.

Una story es una plantilla de una conversación o un conjunto de intenciones relacionadas (grafo de intents). La plataforma permite añadir ramas al grafo que se activan dependiendo de la información contenida por las frases propuestas por el usuario. Gracias a este sistema, se puede definir un flujo de conversación.

Facebook Bot Engine, tiene habilitado un mecanismo de marcadores que permite saltar entre intenciones e historias sin seguir estrictamente el flujo marcado. Existe a su vez una funcionalidad llamada "*Facebook Analytics*", que permite al programador hacer un seguimiento y monitorización de la actividad del chatbot en cualquier momento.

Facebook aporta una buena documentación y muchos ejemplos sobre cómo utilizar la plataforma consiguiendo que el programador no tenga ningún problema a la hora de implementar su propio chatbot.

En la Figura 10 podemos apreciar visualmente la arquitectura que sigue Facebook Bot Engine, explicada anteriormente.

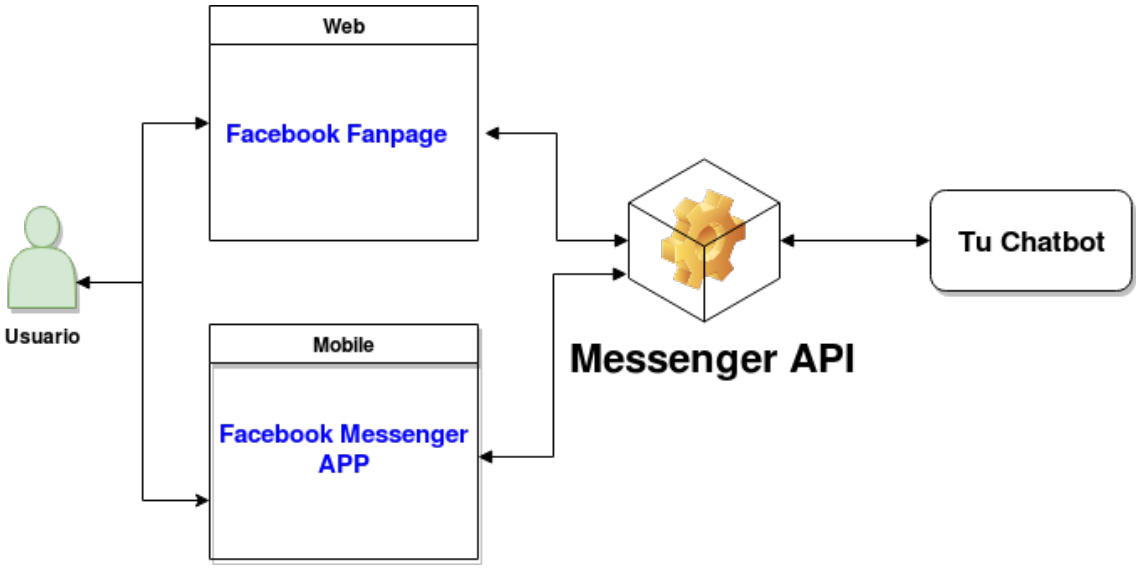

Figura 10. Descripción general de la arquitectura de Facebook Bot Engine

2.2.3 Microsoft Bot Framework

La plataforma de desarrollo de Microsoft divide la implementación de un chatbot en tres partes muy diferenciadas:

- *Portal de desarrollo*: permite realizar desarrollos en Node.js y .Net, frameworks de programación maduros. En la actualidad, son de amplio uso y gracias a ellos, se han conseguido desarrollar muchas funcionalidades aplicadas al desarrollo de los chatbot.

- *Bot Connector*: es una clase que permite la multicanalidad de los chatbots. Así a través de dicha clase, el programador puede desplegar su chatbot en aplicaciones de mensajería como WhatsApp o Telegram, en páginas web, asistentes virtuales como Cortana o aplicaciones propias.

- *Bot Directory*: es un repositorio de chatbots desarrollados con Microsoft Bot Framework al que cualquier usuario puede acceder para testearlos.

Dicha plataforma permite monitorizar, hacer analíticas e integrar servicios como bases de datos, APIs o servicios web. Para el procesamiento del lenguaje natural, se utiliza un servicio basado en aprendizaje automático desarrollado por Microsoft, llamado LUIS (Language Understanding Intelligent Service).

LUIS se basa en intents, entities y features (características), para analizar el contenido de la información proporcionada por el usuario a través de los mensajes entrantes, y definir el comportamiento del chatbot. Las features son diccionarios de palabras o expresiones que permiten que el modelo aprenda más rápido, de tal forma que el chatbot pueda reconocer una entidad en el menor número de ejemplos posibles.

Microsoft ofrece una documentación muy buena y proporciona un buen número de ejemplos que facilitan el desarrollo e implementación de chatbots.

En la Figura 11 podemos apreciar visualmente la arquitectura que sigue Microsoft Bot Framework, explicada anteriormente.

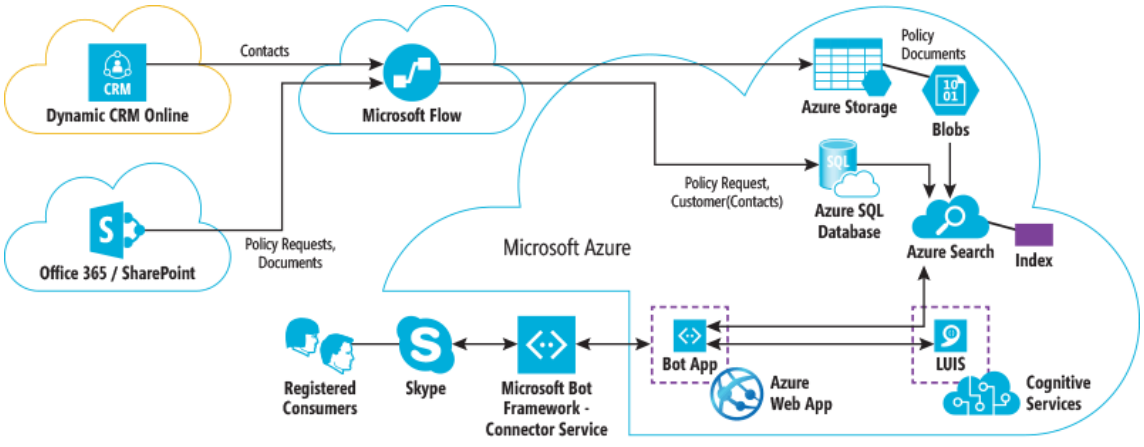

Figura 11. Descripción general de la arquitectura de Microsoft Bot Framework

2.2.4 IBM Watson

En el cloud de IBM, la construcción de chatbots se realiza mediante los "*Conversation Services*" de Watson. Los chatbots se denominan "*workshops*" y se pueden construir utilizando frameworks de desarrollo como pueden ser Node.js, Python, .Net, Android, iOS, etc.

El procesamiento del lenguaje natural está basado en el uso de intents, entities y synonyms (sinónimos). Los synonyms son conjuntos de palabras reconocidas dentro de una misma entity como, por ejemplo: camiseta, falda y vestido, estarían dentro de la entity "*ropa*".

IBM Watson dispone de una función que permite reconocer las palabras que el usuario escribe mal, llamada "*Fuzzy Matching*". Dicha plataforma, también permite la creación de flujos de diálogos de forma gráfica, la integración de otras APIs de Watson o bases de datos y el despliegue de chatbots en páginas web y aplicaciones propias como Slack y Twilio. Por último, comentar que se incluye una herramienta de monitorización y analítica utilizada sobre los resultados obtenidos de las conversaciones entre usuarios y chatbots.

IBM dispone de una buena documentación y se facilitan bastantes ejemplos que aseguran el buen desarrollo de los chatbots.

En la Figura 12 podemos apreciar visualmente la arquitectura que sigue IBM Watson, explicada anteriormente.

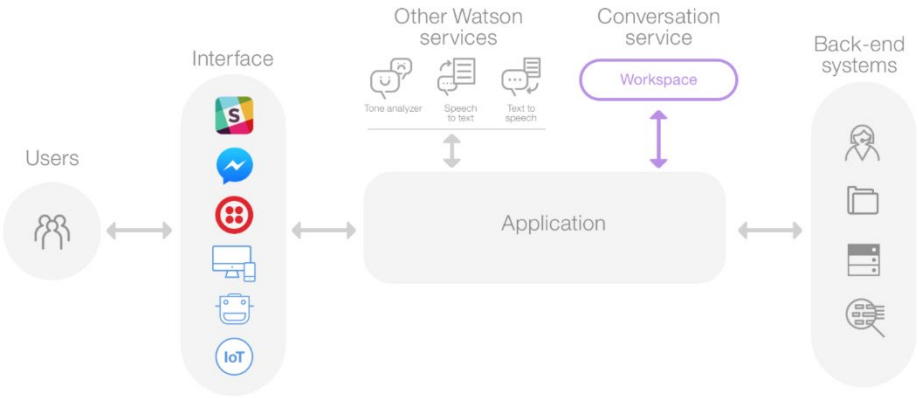

Figura 12. Descripción general de la arquitectura de IBM Watson

2.2.5 Amazon Lex

Es un servicio de Amazon Web Services para crear interfaces conversacionales o chatbots. El sistema de procesamiento del lenguaje natural está basado en el uso de intents y slots. Los slots son los parámetros que puede requerir un intent, aunque no son obligatorios.

Por ejemplo, el intent "*ReservarVuelo*" podría requerir de slots como origen, destino y clase, teniendo cada uno de ellos un tipo como:

- Origen y destino: tendrán un listado de aeropuertos.
- Clase: estará compuesto por las distintas posibilidades como pueden ser "*Turista*", "*Business*" o "*Primera*".

El usuario podrá responder con un valor de slot que incluya palabras adicionales, como *"Quiero viajar desde Pamplona a Madrid"* o *"Me gustaría la clase Primera"* y el sistema de Amazon Lex seguirá entendiendo el valor del slot integrado.

Amazon Lex cuenta con los siguientes entornos de programación: Node.js, Python, Java, JavaScript, PHP, .Net, Android y iOS. Incluye herramientas para la monitorización y analítica, y también permite integraciones con otros servicios como APIs, se hace a través de los AWS Lambda. Además, los chatbots desarrollados con Amazon Lex pueden desplegarse en aplicaciones como Facebook Manager, Slack y Twilio.

Por último, añadir que Amazon aporta buena documentación y varios ejemplos sobre los que trabajar el desarrollo de chatbots.

En la Figura 13 podemos apreciar visualmente la arquitectura que sigue Amazon Lex, explicada anteriormente.

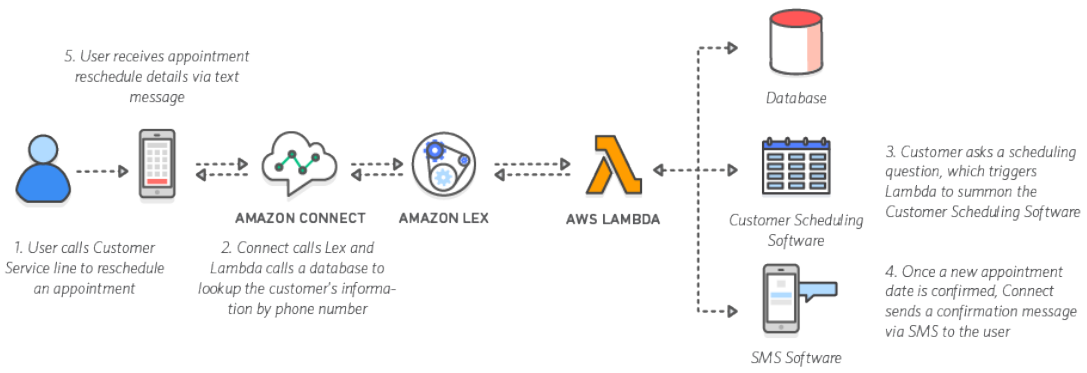

Figura 13. Descripción general de la arquitectura de Amazon Lex

2.2.6 Aspect Customer Experience Platform (CXP)

Es una plataforma enfocada al diseño, implementación y despliegue de aplicaciones dedicadas al servicio al cliente a través de la utilización de chatbots. Sus canales de comunicación más utilizados son páginas web y redes sociales como Facebook y Twitter.

Aspect CXP no permite el uso de lenguajes de programación, el diseño y desarrollo de chatbots se hace a través de una herramienta propietaria, llamada "*Aspect CXP Designer*". Es una aplicación con muchas funcionalidades que se podría considerar como una mezcla entre una aplicación gráfica y un lenguaje de programación. Para obtener el mayor partido a la aplicación mencionada, se deben tener altos conocimientos de lógica de programación. También permite la integración de otros servicios como pueden ser las APIs, bases de datos, servicios cognitivos, etc.

Para el procesamiento de la información, Aspect usa una herramienta propia llamada Aspect NLU (Natrural Language Understanding) que permite recorrer los mensajes entrantes e ir identificando las partes relevantes y su significado. Las implementaciones de NLP hechas con Aspect NLU son más robustas, pero menos flexibles y más complejas de construir.

Aspect aporta una documentación de una calidad aceptable, los ejemplos de desarrollo de chatbots son realmente escasos y su código es inaccesible.

En la Figura 14 podemos apreciar visualmente la arquitectura que sigue Aspect CXP, explicada anteriormente.

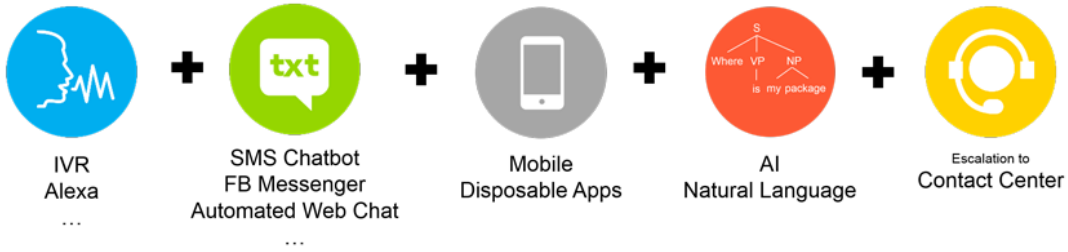

Figura 14. Descripción general de la arquitectura de Aspect CXP

2.2.7 Gupshup

Es una plataforma de desarrollo apta tanto para desarrolladores como para aquellos que no lo son, ya que ofrece dos herramientas:

- *Flow Bot Builder*: es un editor visual diseñado para usuarios sin conocimiento de programación, que permite la construcción e implementación de chatbots sin necesidad de programar. Claramente el resultado son chatbots sencillos a los que no se les puede dar una gran lógica de programación. A pesar de no ser una herramienta muy potente, sigue siendo muy útil para aficionados a la informática.

- *Bot Builder IDE*: es una herramienta de desarrollo que permite el uso de varios entornos de programación como Node.js, Python, Java, C#, Python, Android y iOS. El NLP que utiliza por defecto llamado "*NLP on the fly*", es muy sencillo de implementar debido a que solo se manejan intents y entities. A su vez, utiliza herramientas de analítica y monitorización permitiendo recibir un feedback sobre el funcionamiento del chatbot.

Tanto una herramienta como la otra permiten el despliegue en numerosos canales como pueden ser aplicaciones de mensajería, aplicaciones propias, páginas web, etc. También permiten la integración de servicios como APIs o bases de datos.

La calidad de la documentación de Gupshup es aceptable, aunque sobre los ejemplos de desarrollo de chatbots, el contenido es muy pobre.

En la Figura 15 podemos apreciar visualmente la arquitectura que sigue Gupshup, explicada anteriormente.

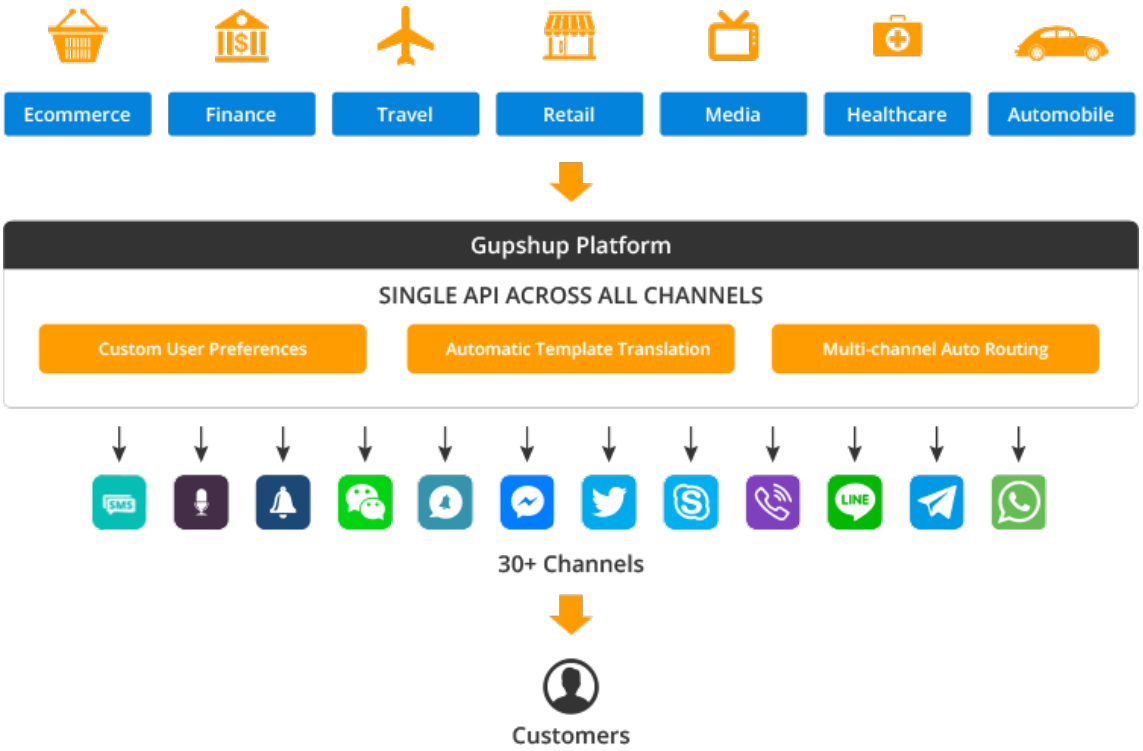

Figura 15. Descripción general de la arquitectura de Gupshup

2.2.8 Conclusiones

Tras haber analizado las distintas plataformas de desarrollo NLP, voy a realizar una valoración de cada plataforma puntuando del uno al tres, siendo uno la nota más alta y tres la más baja, cada uno de los aspectos fundamentales para el desarrollo de un chatbot. En la Tabla 1 se reflejan los aspectos a valorar elegidos:

Tabla 1. Aspectos por valorar

FD	Frameworks de desarrollo	A/M	Analítica / Monitorización
NLP	Procesamiento del lenguaje natural	D	Documentación
IS	Integración con otros servicios	EC	Ejemplos de chatbots
CC	Canales de comunicación		

Tras la valoración de cada plataforma (Tabla 2), se puede concluir que DialogFlow es la plataforma de desarrollo NLP más completa de todas obteniendo la máxima puntuación. Por lo tanto, para la realización de un chatbot propio, la aplicación por excelencia y la que se debe escoger es DialogFlow sin ninguna duda.

Tabla 2. Valoración de todos los aspectos

	FD	NLP	IS	CC	A/M	D	EC
DialogFlow	1	1	1	1	1	1	1
Facebook Bot Engine	1	2	2	3	2	2	2
Microsoft Bot Framework	1	2	1	2	2	1	1
IBM Watson	2	2	1	3	2	2	2
Amazon Lex	1	2	1	3	1	2	3
Aspect CXP	3	1	2	3	1	3	3
Gupshup	1	3	1	2	2	2	3

Capítulo 3

Herramientas clave para la creación de un chatbot

Antes de comenzar con el desarrollo de mi chatbot, conviene explicar de una manera más detallada el funcionamiento de las herramientas que se han ido utilizando a lo largo del proceso de creación del chatbot de cara a facilitar la comprensión del desarrollo.

3.1 Funcionamiento de DialogFlow

En este punto, se explicará el funcionamiento de la plataforma de desarrollo NLP anteriormente escogida, DialogFlow [14] [15].

3.1.1 Intents

Los intents son una relación entre la entrada del usuario y la acción que se debe tomar. Para cada acción, se debe crear un intent que será activado con entradas de usuario. Para intents más complejos se da mucha importancia al uso de eventos, pero sobre todo al uso de contextos.

Un contexto representa el estado real de la solicitud del usuario. Por ejemplo, un usuario puede estar pidiendo un pantalón y durante el proceso de compra puede decir "Quiero uno gris". En este caso, la herramienta puede entender que quiere un pantalón gris porque el intent anterior ha agregado el contexto del pantalón. Cada intent puede tener contextos de entrada y salida. Los contextos de entrada son los que se necesitan para desencadenar el intent deseado. Los contextos de salida son los nuevos que aparecen después de activar un intent. Entonces, para comenzar un intent, debe coincidir en el contexto de entrada y en una frase o evento.

Cuando un usuario comienza a usar el chatbot, se le puede solicitar un contexto y si este contexto se asigna a un intent, se iniciará la acción asignada. Lo mismo ocurre cuando el usuario envía un texto a la herramienta. En este caso, DialogFlow analizará si el texto coincide o es similar a una frase de entrenamiento y luego ejecutará la acción asignada.

En la Figura 16 mostramos un ejemplo. Supongamos que disponemos de un chatbot que atiende consultas meteorológicas y un usuario dice "*¿Cuál es el pronóstico?*", la plataforma detecta el intent "*Pronóstico*". Una vez el usuario indique la localización, acciones asociadas a ese intent, se genera el subcontexto "*LocalizaciónEscogida*". En el caso de que el usuario siguiera consultando y, por ejemplo, dijera *"¿Qué temperatura hará mañana en Pamplona?"*, el chatbot solamente podría responder debido a la existencia del subcontexto mencionado anteriormente, "*LocalizaciónEscogida*", ya que daría acceso al otro intent "ObtenerInformaciónLocalización", que guarda dicha información. Si no existiera ningún subcontexto asociado, no podría haber acceso al intent de la información del pedido generando problemas a la hora de dar respuesta al usuario.

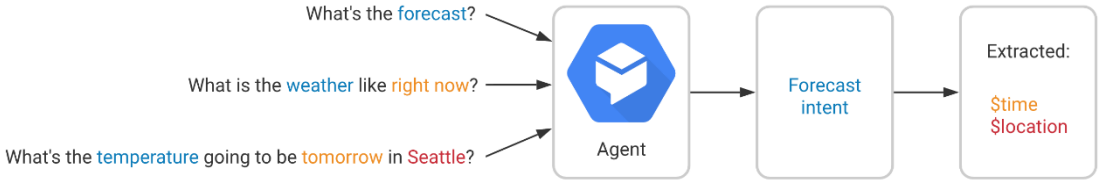

Figura 16. Ejemplo explicación Intents

Un intent básico está compuesto principalmente por frases de entrenamiento, parámetros y respuestas. En la Figura 17, se muestra el flujo básico para la coincidencia de intents y respuestas al usuario final.

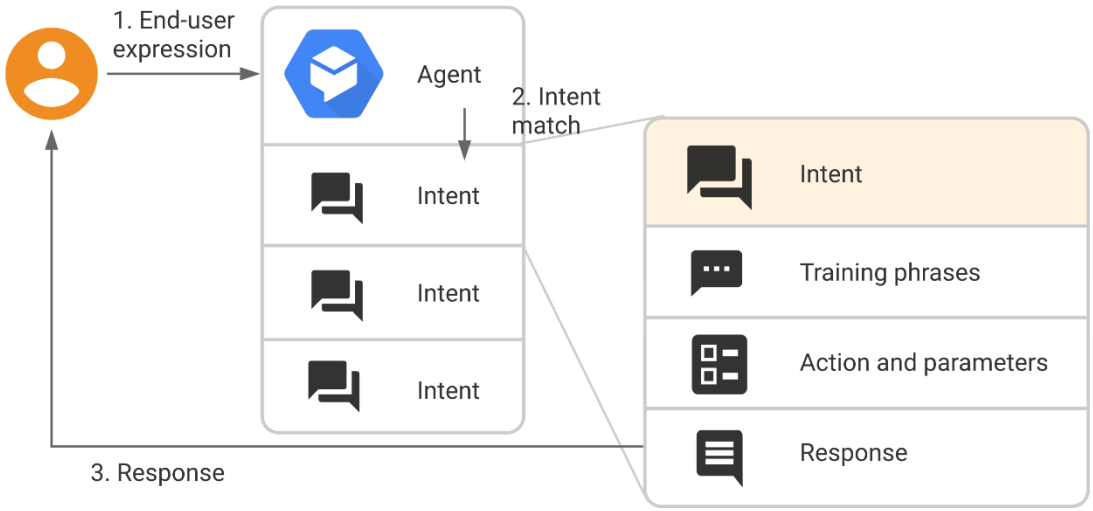

Figura 17. Flujo básico intents

Las frases de entrenamiento son frases que se insertan manualmente por parte del desarrollador con la intención de que DialogFlow aprenda cuándo debe activar el intent deseado. Por ejemplo, tal como mostramos en la Figura 18, algunas frases pueden ser "Quiero una camiseta" o "Me gustaría comprar una camiseta". Entonces, si un usuario escribe "Me gustaría una camiseta" o "Quiero comprar una camiseta" se reconocerá el intent, ya que estas frases son similares a las introducidas por el desarrollador. En el caso de que la frase no coincida con ninguna intención, se activará el intent alternativo predeterminado. Este intent tiene el objetivo de notificar al usuario que no se ha entendido su frase.

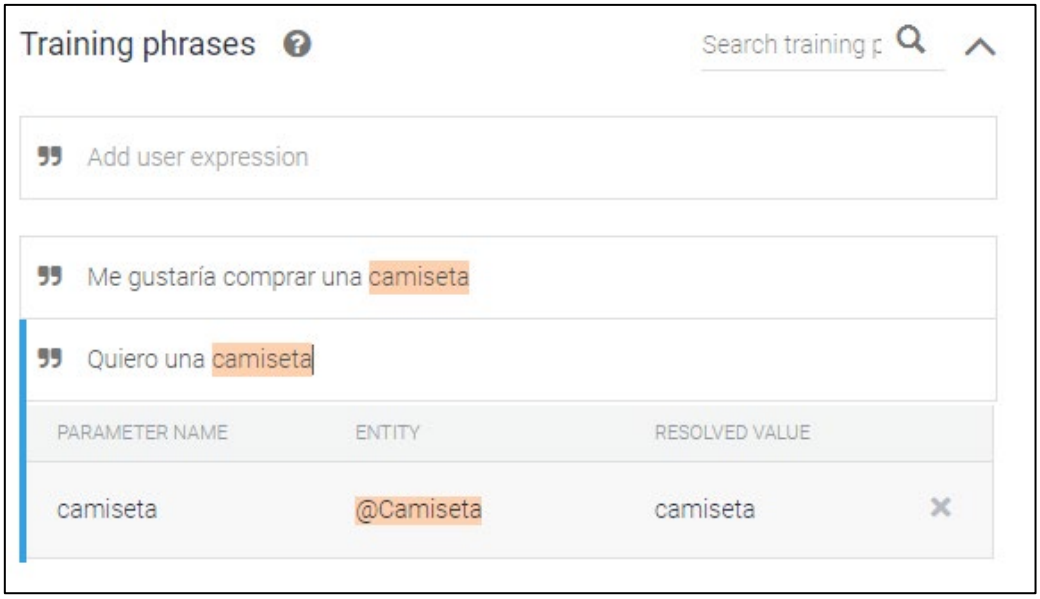

Figura 18. Ejemplo frases de entrenamiento

Un intent puede tener un "*fulfillment*" para proporcionar una respuesta más dinámica. Esto significa que cuando se activa un intent, se puede ejecutar algún código externo para realizar una tarea. Por ejemplo, si un usuario le pregunta a un chatbot cómo estará el clima en una semana, se debe ejecutar un código para consultar esta información. Los "*fulfillment*" funcionan mediante webhooks, herramientas que entregan información a medida que sucede algo. En este caso, cada intent puede tener un webhook asignado que se activará cuando se active el intent (Ver Figura 19).

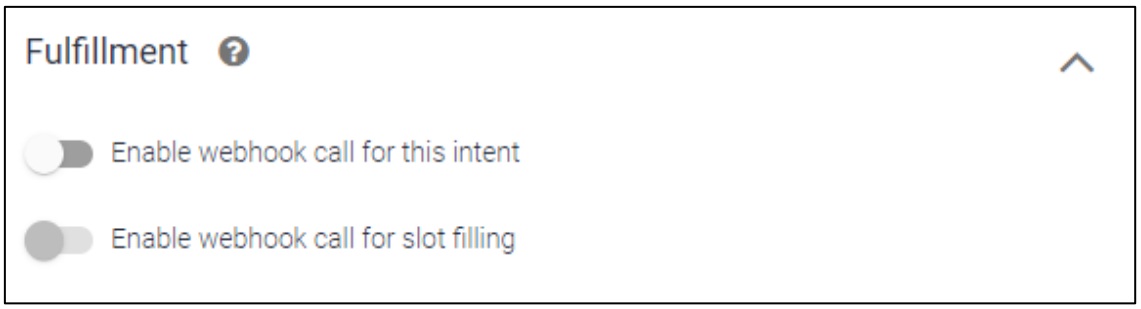

Figura 19. Opciones de la herramienta Fulfillment

Pero para enviar la información, se deben recopilar algunos parámetros, ya que los webhooks pueden necesitar información extra del usuario que recopilará el chatbot. Estos parámetros deben especificarse en el intent para activar el webhook solo cuando se consiga toda la información (Ver Figura 20).

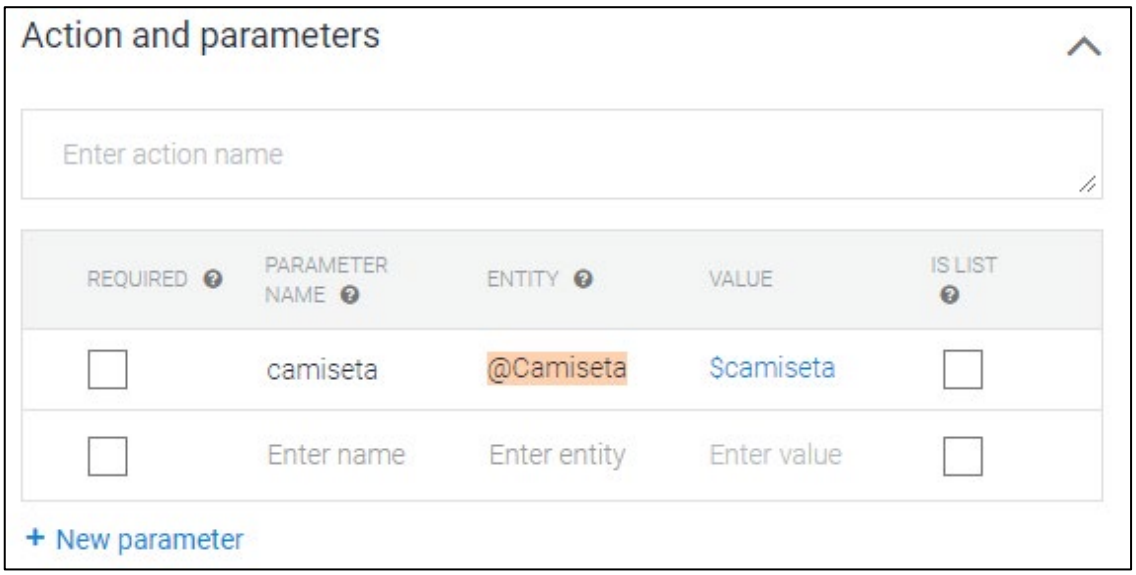

Figura 20. Ejemplo parámetro

Finalmente, los intents tienen la opción de definir respuestas predeterminadas. Estas respuestas son las que se envían al usuario después de que se active un intent. Estas se pueden adaptar a cada plataforma donde se va a consumir el servicio y aprovechar al máximo las herramientas que ofrece cada aplicación de mensajería. Por ejemplo, en Telegram podemos agregar respuestas normales, imágenes, formularios, respuestas rápidas y respuestas más complejas (Ver Figura 21).

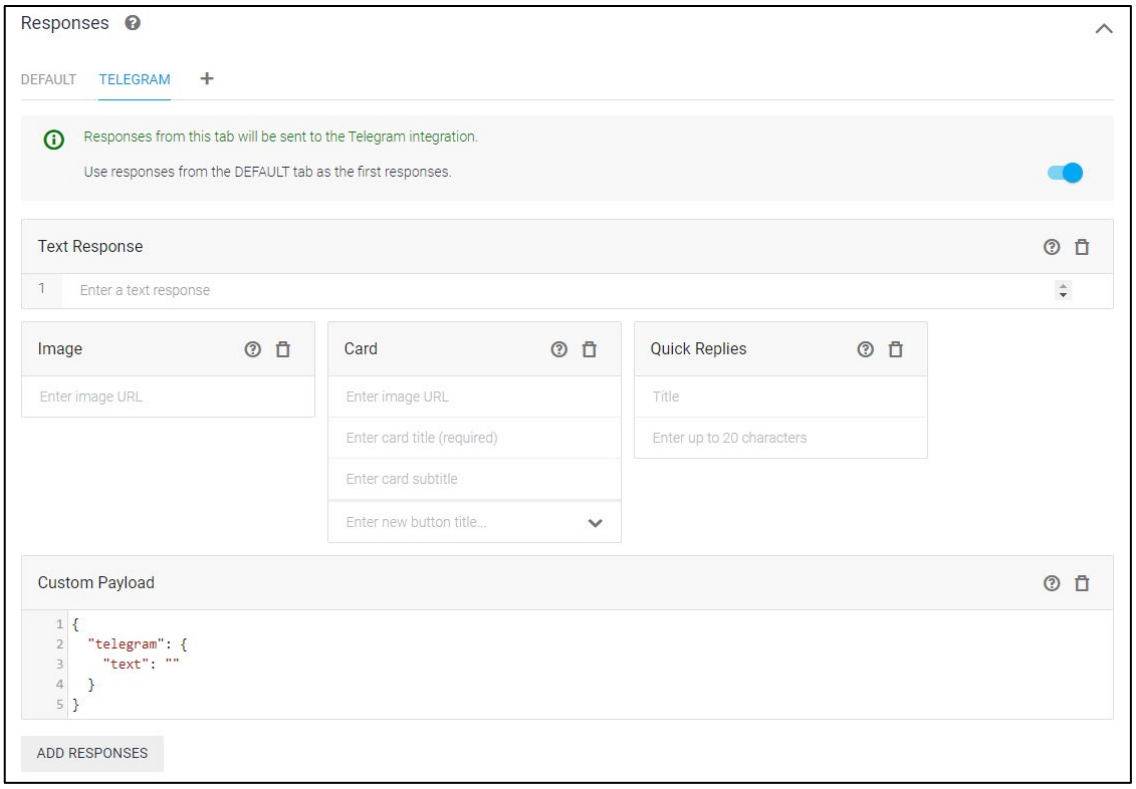

Figura 21. Tipos de respuestas

3.1.2 Entities

Las entities son un mecanismo utilizado para extraer parámetros del texto enviado por los usuarios. Cada parámetro tiene un tipo, denominado tipo de entidad, que determina de forma exacta cómo se extraen los datos de una expresión del usuario. Por ejemplo, un usuario puede querer comprar un producto y decir "*pantalón corto*" en lugar de "*pantaloneta*". Esto es totalmente válido, ya que ambas palabras han sido introducidas anteriormente en la entity "*Ropa*" como sinónimos. Gracias al haber definido dicha entity, cada vez que un usuario envíe una consulta que incluya las palabras mencionadas, DialogFlow directamente deducirá que el usuario está realizando una consulta sobre ropa (Ver Figura 22).

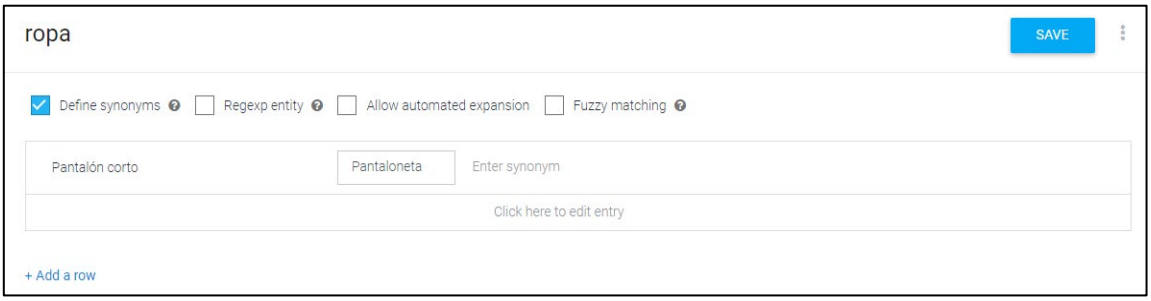

Figura 22. Ejemplo entity

El chatbot puede tener varias entities, y cada una de ellas puede tener varias entradas. La ventaja de tener entities con diferentes entradas es que los sinónimos se pueden agrupar por el tema que elijamos, lo que nos ayudará a crear frases de entrenamiento.

Además, las entities sirven para verificar la información del usuario. Si el chatbot le pide al usuario su correo electrónico personal, si este parámetro usa la entity *"@sys.email*", el chatbot solo aceptará el correo electrónico del usuario si sigue la expresión regular *"*@*.*"* (* = cualquier cadena). Estas entities son las que ofrece la herramienta de NLP.

El usuario no puede crear entities que verifiquen expresiones regulares, pero puede usar las que ofrece DialogFlow como en el caso del correo electrónico.

3.1.3 Training

Esta sección se utiliza para entrenar al chatbot. Aquí el programador puede seleccionar los cuadros de diálogo entre el usuario y el chatbot, y verificar si todas las frases fueron con la intención correcta o si algunas de ellas fallaron y activaron el intent por defecto. Además, la herramienta "*Training*" da la posibilidad de corregir las palabras que fallaron indicando a que intent podrían asociarse en vez de activar el intent por defecto, de tal forma que la plataforma aprenda del error cometido y no se vuelva a cometer en adelante. Gracias a ello, se consigue que el chatbot esté en continuo aprendizaje y por tanto tenga la capacidad de responder a más tipos de mensajes. Las funciones de esta característica son limitadas, ya que está en fase beta.

En la Figura 23, se puede observar un ejemplo en el que un usuario se ha despedido con la palabra "adios" y DialogFlow lo ha procesado a la perfección deduciendo que es un mensaje de despedida asociando la palabra al intent "*Despedida*".

Figura 23. Ejemplo Intent "Despedida"

3.1.4 Integrations

Esta herramienta permite la integración del chatbot en una (o más) aplicaciones de la lista. Normalmente estas integraciones son bastante sencillas, en ellas el desarrollador solo tiene que configurar algunos parámetros.

Hay muchas plataformas disponibles. Las más conocidas son Facebook Messenger, Twitter, Skype o Telegram, pero hay otros (Ver Figura 24).

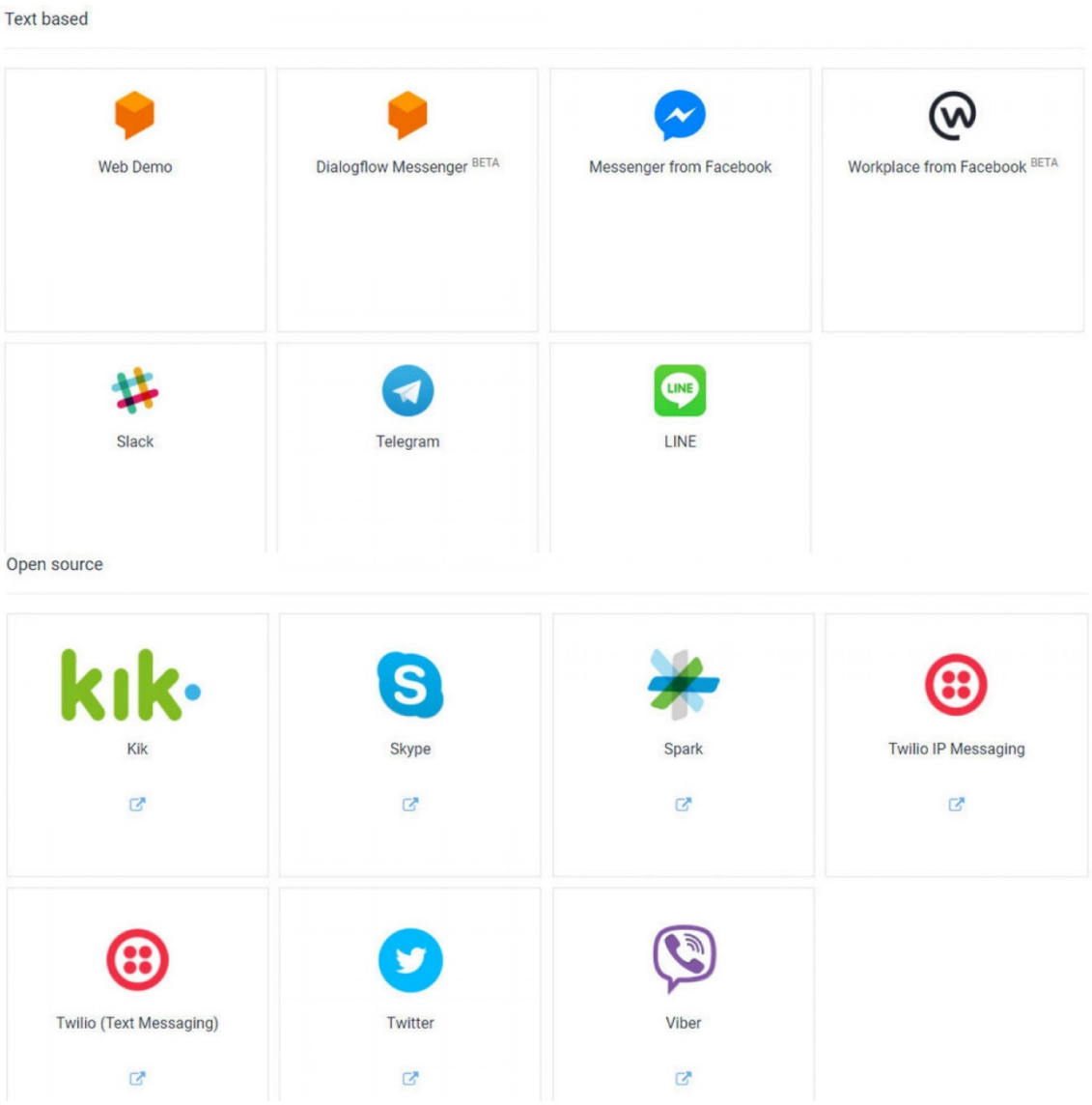

Figura 24. Plataformas disponibles DialogFlow

3.1.5 Analytics

Esta herramienta permite hacer un seguimiento del funcionamiento del chatbot por parte del desarrollador (ver Figura 25). Se pueden ver gráficos donde se indica el número de usuarios activos. Además, se puede ver cuántas veces se ha utilizado un intent y cuánto tiempo ha tardado en responder al usuario. Finalmente, existe una herramienta llamada "*Session Flow*" que indica diferentes flujos y el porcentaje de uso de cada uno.

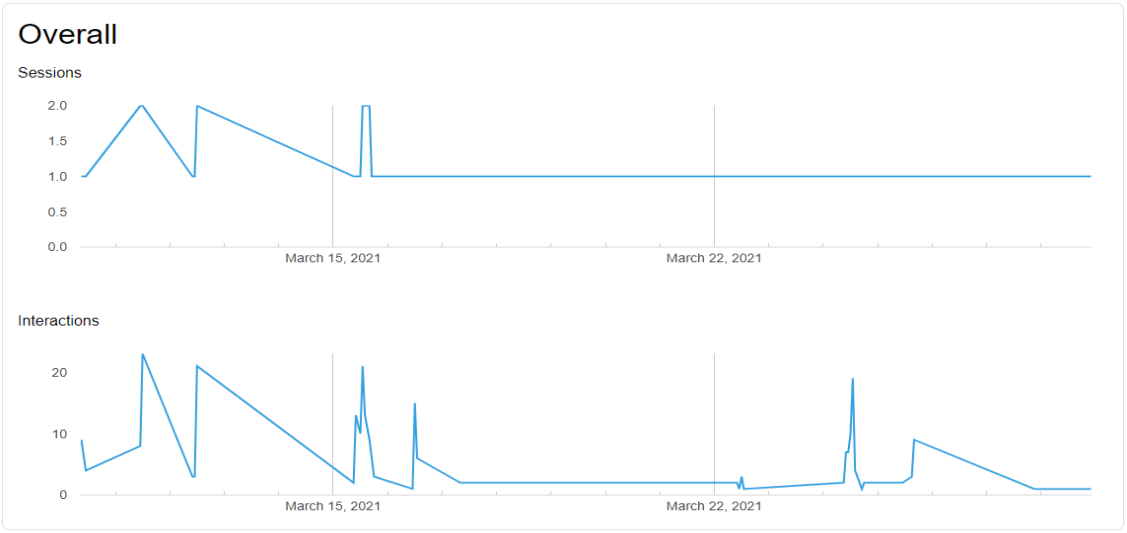

Figura 25. Gráficos sobre la interacción con mi chatbot

3.1.6 Fulfillment

En este último apartado se puede configurar toda la información del webhook. Aquí, el desarrollador puede introducir la URL del webhook y los encabezados necesarios para usarlo (ver Figura 26). Además, Google ofrece la posibilidad de utilizar sus propios servidores webhook (Firebase). En este caso, se pueden codificar las funciones usando su editor en línea. En este desarrollo no se ha utilizado esta característica, ya que los servidores de AWS fueron los elegidos.

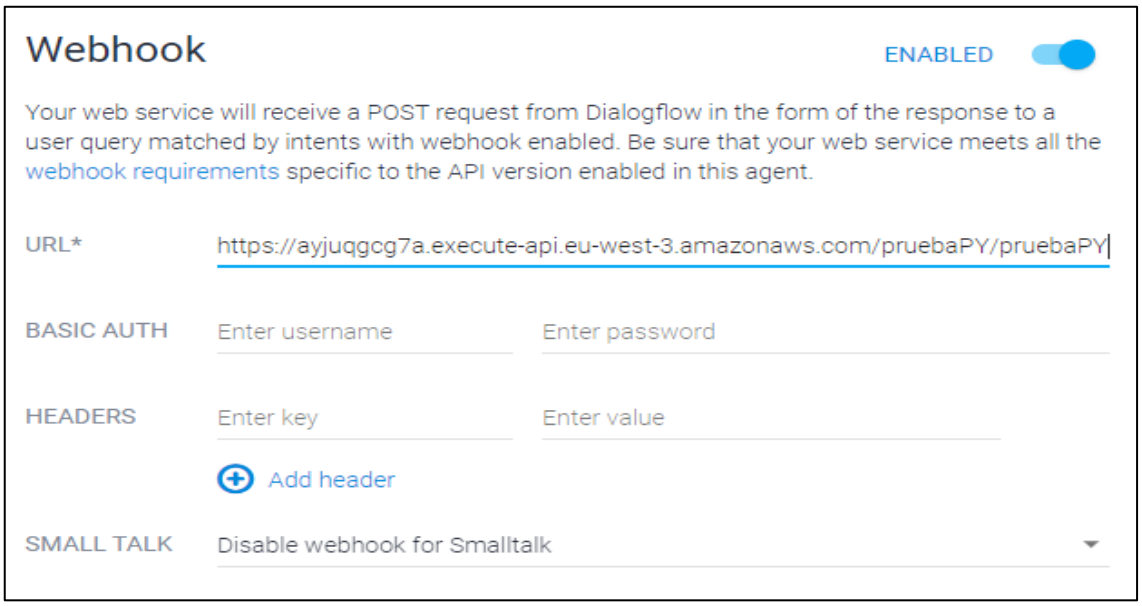

Figura 26. Conexión DialogFlow con AWS

3.2 Funcionamiento Servicios Web

Para conseguir un chatbot dinámico, es decir, un chatbot que pueda emular conversaciones que se podrían tener con una persona real, el uso de servicios web es de vital importancia. Es necesario complementar a la plataforma de desarrollo NLP con un servicio web que permita realizar consultas para responder a las peticiones más complejas de los usuarios.

En este apartado, se explicarán brevemente los servicios utilizados para la creación de un servicio web como son Amazon Web Services (AWS) y SoapUI. Además, se tratará la importancia de la proactividad en un chatbot dinámico.

3.2.1 Amazon Web Services

Amazon Web Services es un proveedor de servicios en la nube que nos permite disponer de almacenamiento, recursos de computación, aplicaciones móviles, bases de datos y un largo etcétera en la modalidad de "*cloud computing*" o más conocida para nosotros como "*informática en la nube*" [16].

En este proyecto se ha utilizado AWS para cargar el código fuente que permita responder a las peticiones más complejas del usuario llegadas desde Telegram a DialogFlow. La conexión entre DialogFlow y AWS se realizará a través de webhooks, lo que permitirá obtener en todo momento la información proporcionada por parte del usuario en Telegram.

La lógica del programa se verá reflejada en una función. La función puede ser programada en el lenguaje que el desarrollador elija y ha de ser almacenada en Amazon Web Services (AWS), concretamente en su producto AWS Lambda. Para este proyecto, el lenguaje elegido fue Python.

Para lograr esta tarea, se debe crear una cuenta en AWS y, acto seguido, la función comentada anteriormente en la herramienta AWS Lambda. La plataforma ofrece la opción de codificar directamente en un editor web, pero si el desarrollo requiere de algunos módulos extra, es necesario empaquetar todo el código en un zip y posteriormente cargarlo para su buen funcionamiento (Ver Figura 27).

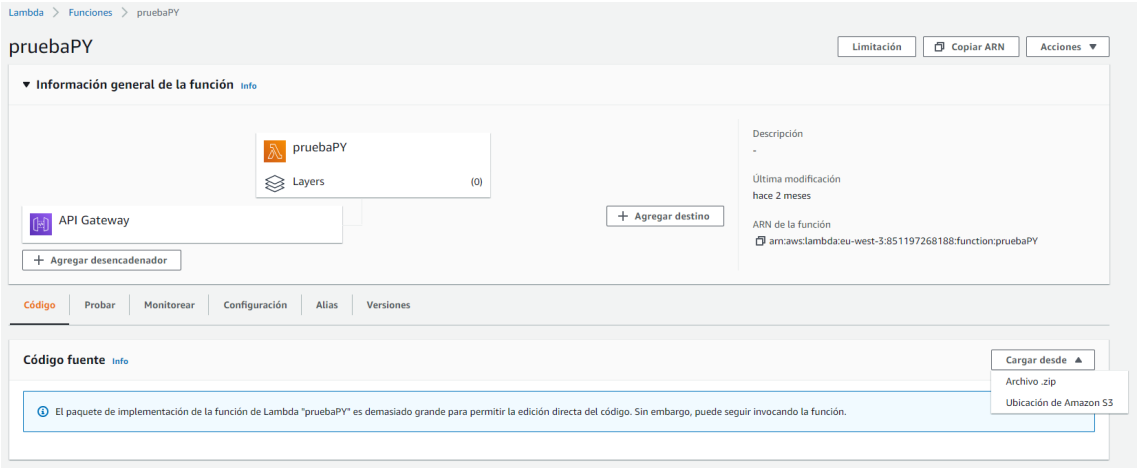

Figura 27. Ejemplo AWS Lambda

En definitiva, Amazon Web Services es la mejor plataforma para conseguir la funcionalidad extra necesaria que requiere cualquier chatbot.

3.2.2 SoapUI

SoapUI es la herramienta de mayor difusión para probar servicios web en arquitecturas orientadas a servicios (SOA) y transferencia de estado representacional (REST). Es una aplicación muy versátil que nos permite probar, simular y generar código de servicios web de forma ágil [17]. Es por ello por lo que es la herramienta escogida para dar soporte y respuesta a las consultas más complejas del usuario.

Para su utilización, es necesario que haya detrás un servicio que permita responder a las llamadas SOAP que se realizarán en cada petición web. En mi caso, la empresa en la que estaba realizando las prácticas, me permitió acceso a la VPN de la Mancomunidad de la Comarca de Pamplona (MCP) para poder maquetar mi servicio web para poder dar respuesta a las peticiones pertinentes.

El desarrollo de una llamada SOAP es realmente sencillo, simplemente se crea un proyecto nuevo con su respectiva conexión al servicio web maquetado y se realiza la programación de la llamada en el lenguaje de programación XML (Ver Figura 28).

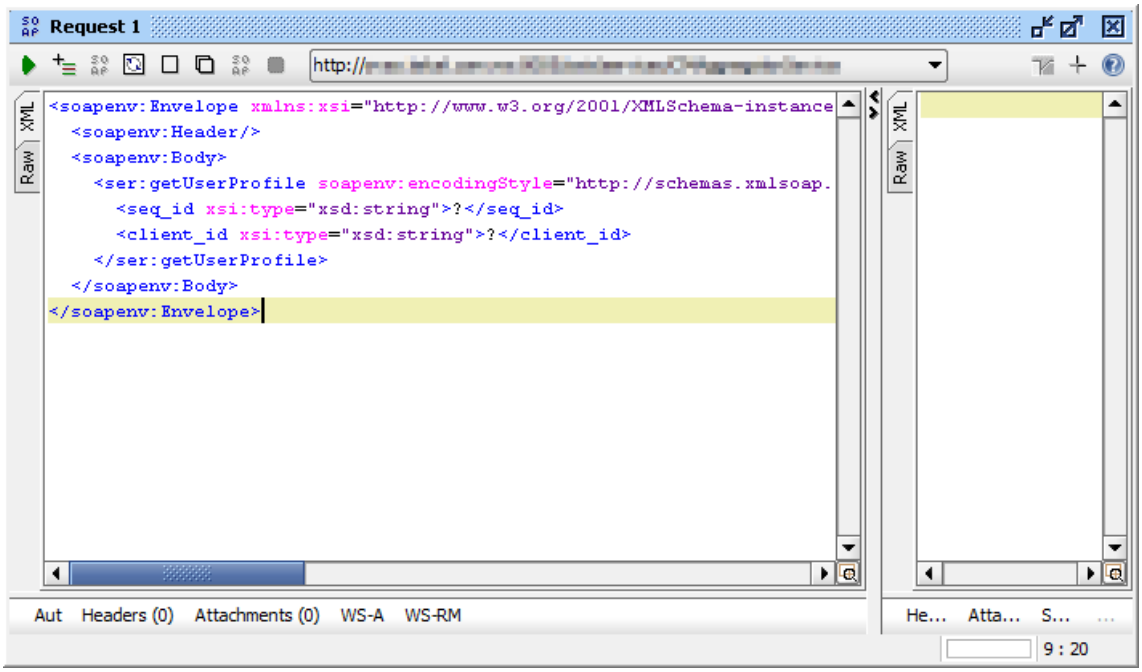

Figura 28. Ejemplo llamada SOAP

Finalmente, una vez realizada la llamada, se procede a su implementación en la función AWS Lambda, comentada anteriormente, para que el chatbot pueda utilizar sus servicios y de esta forma conseguir un chatbot plenamente dinámico.

3.2.3 Proactividad

Pocos usuarios iniciarán una conversación por su cuenta con un chatbot. Normalmente, un asistente virtual se dedica única y exclusivamente a responder mensajes recibidos por parte de un usuario. Pero en ocasiones, es posible que para ser más efectivos y motivar el compromiso, deba ser proactivo, es decir, tener la capacidad de enviar mensajes automáticamente para llamar la atención del usuario y devolver la actividad al servicio [18].

Los mensajes proactivos pueden ser útiles en diversos escenarios, pero el más utilizado es el servicio de alertas. Por ejemplo, si un usuario ha solicitado con anterioridad a un chatbot que supervise el precio de un coche, el bot se encargará de alertar proactivamente al usuario si el precio ha disminuido respecto a la consulta anterior.

Ante estas situaciones, el chatbot activará una notificación de inserción y contactará al usuario, ofreciendo un servicio que corresponda a la consulta. Si el momento es el correcto, el usuario se inclinará a responder y mantendrá una conversación con el asistente virtual, quien aclarará las necesidades del usuario y guiará al usuario en la conversión. Por el contrario, una notificación enviada en el momento incorrecto o con un mensaje incorrecto podría dar al usuario la impresión de ser estafado o incluso amenazado por la intrusión. Es por ello por lo que se precisa de la supervisión de un profesional en la temática para evitar problemas de este tipo y conseguir la mejor versión de un chatbot proactivo.

En definitiva, un chatbot proactivo asume el papel de un verdadero vendedor, capaz de adaptarse a las expectativas del usuario con una enorme eficiencia. Para ello, es necesario basar la configuración del chatbot en escenarios relevantes que permitan adaptarse a las acciones realizadas por los usuarios con la mayor eficacia posible. Como resultado, se podrá iniciar un diálogo de forma inteligente replicando una conversación humana con un éxito absoluto.

Capítulo 4

Desarrollo de un chatbot

En este capítulo, se detalla el proceso de creación de mi chatbot para atender peticiones sanitarias, desde la creación del bot en Telegram, pasando por el procesamiento de la información en DialogFlow, hasta el uso de servicios web para atender las peticiones más complejas de los usuarios. El chatbot tendrá la capacidad de responder a peticiones sobre citas médicas, recetas electrónicas e información COVID.

4.1 Creación en Telegram

Para la creación del chatbot en Telegram, se contactó con el "*BotFather*" iniciando una conversación con él. Como ya se ha descrito en el apartado 2.1.2, sobre el funcionamiento de Telegram, para la creación de un chatbot es necesario utilizar el comando "/newbot".

Tras iniciar la conversación, el bot solicitó un nombre para el chatbot. Al ser un bot enfocado a la salud decidí llamarle "*BotSalud*", y un nombre de usuario que identificara el bot, decidí el nombre de "PrototipoSaludBot" (Ver Figura 29).

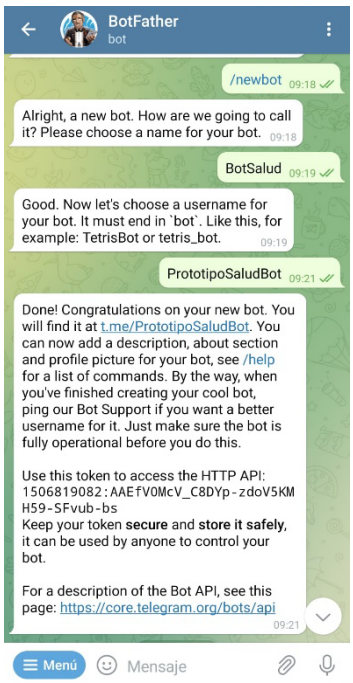

Figura 29. Creación BotSalud

Finalmente, conseguí crear mi chatbot en dos pasos y de una manera muy sencilla. Para acceder a él desde cualquier plataforma, simplemente se necesita el link de acceso que aparece en el último párrafo de la Figura 29, "*t.me/PrototipoSaludBot*". Gracias a dicho link, cualquier persona puede ponerse en contacto con el chatbot y entablar una conversación con él.

Si se accede al bot creado a través del link propuesto, se puede comprobar que las credenciales que aparecen en su perfil se ajustan a las credenciales que se utilizaron para su creación (Ver Figura 30).

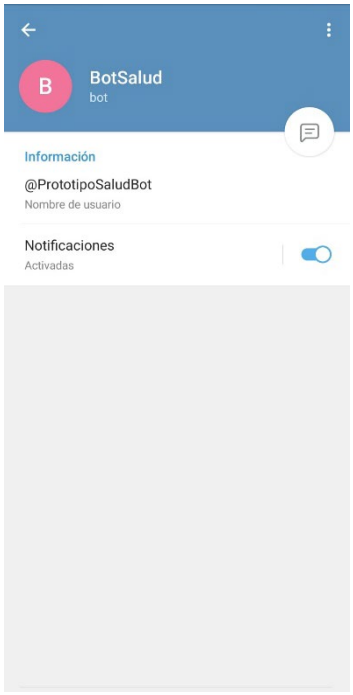

Figura 30. Credenciales BotSalud

Una vez se ha comprobado que el chatbot está operativo, se procedió a su integración en la plataforma de desarrollo NLP elegida, DialogFlow. Para que DialogFlow ofrezca servicio al chatbot, es necesario el token que aparece en el último párrafo de la Figura 29, "*1506819082:AAEfV0McV_C8DYp-zdoV5KMH59-SFvub-bs*". Dicho token se introducirá en la parte de "*Integrations*", concretamente en la parte de "*Telegram*" de DialogFlow para completar la integración y finalizar así el proceso de creación (Ver Figura 31).

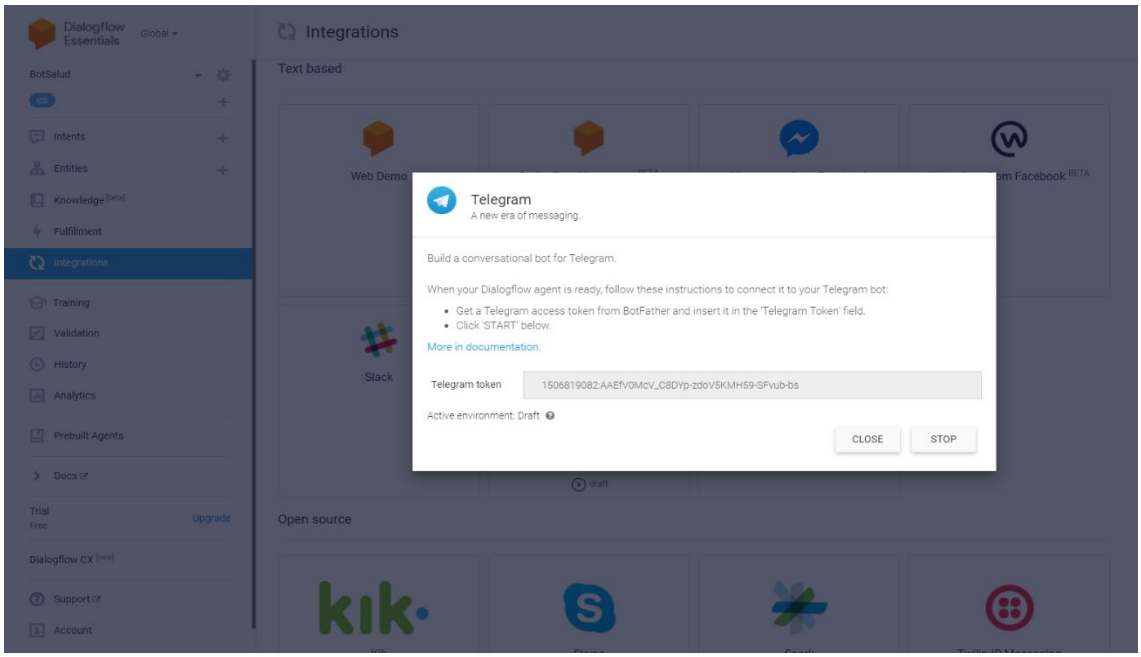

Figura 31. Integración chatbot en DialogFlow

4.2 Implementación en DialogFlow

Tras su creación en Telegram, el chatbot se encuentra disponible para que cualquier usuario contacte con él. Pero no en su totalidad, ya que todavía no ha sido programado para procesar ningún tipo de información por lo que será incapaz de dar una respuesta lógica a las peticiones de los usuarios que utilicen el servicio.

DialogFlow va a ser la herramienta encargada de dar el soporte necesario para procesar cualquier tipo de información que reciba el chatbot y ser capaz de buscar una respuesta con sentido que satisfaga las necesidades del usuario.

Como ya he comentado anteriormente, este chatbot se va a centrar en 3 cuestiones sanitarias: citas médicas, recetas electrónicas e información COVID. Por lo tanto, es muy importante que el usuario sea informado de ello antes de que comience a usar nuestro servicio, ya que, en caso contrario, podría comenzar a realizar peticiones al chatbot que éste no sería capaz de responder, lo que provocaría una disconformidad no deseada en el usuario. Es por ello por lo que el chatbot deberá informar al usuario sobre este aspecto nada más iniciar la consulta para evitar malentendidos.

Una vez se ha comprendido el objetivo de nuestro servicio, es importante saber modelarlo en nuestra plataforma de desarrollo NLP, de tal forma que consigamos un flujo de procesamiento del lenguaje sin error alguno. Para ello en DialogFlow, se crearon 6 intents que ayudasen a gestionar lo anteriormente comentado, dichos intents son los siguientes: "*Bienvenida*", "*Cita*", "*Receta*", "*InformacionCOVID*", "*Despedida*" y "*Error*".

En cada uno de los intents, es necesaria una fase de entrenamiento previa para que de esta manera DialogFlow sepa en todo momento a qué intent debe acudir y qué flujo debe seguir. Pero para abordar dicha fase, es primordial la creación de un entity que nos ayude a procesar cada frase de entrenamiento, ya que de esta forma conseguiremos sustraer de cada frase las palabras que nos interesan para posteriormente deducir que tipo de consulta se está realizando.

Para ello, se creó una entity "peticiones" que abarcara todas las peticiones que el chatbot está dispuesto a procesar, como son las citas médicas, las recetas electrónicas y la información COVID. En dicha entity, definí las palabras clave que consideré, con sus respectivos sinónimos, para que DialogFlow fuese capaz de comprender cada frase del usuario. De esta forma, si el chatbot recibe una frase con alguna de estas palabras, automáticamente se deduce que se trata de una petición del usuario (Ver Figura 32).

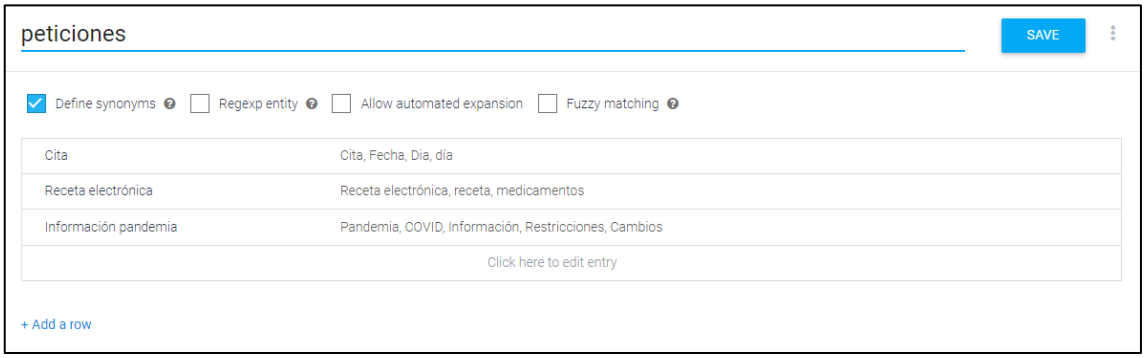

Figura 32. Entity "peticiones"

Tras haber definido las palabras que nos permiten comprender la intención de cada frase del usuario, es el momento de abarcar la explicación sobre la creación y preparación de cada intent.

- *Intent "Bienvenida"*: este intent será el encargado de procesar y dar respuesta a los primeros mensajes de bienvenida que realicen los usuarios al contactar por primera vez con el chatbot. Para que dicho intent fuese capaz de procesar un mensaje de bienvenida, fue necesario realizar un entrenamiento previo a través de frases de entrenamiento. (ver Figura 33)

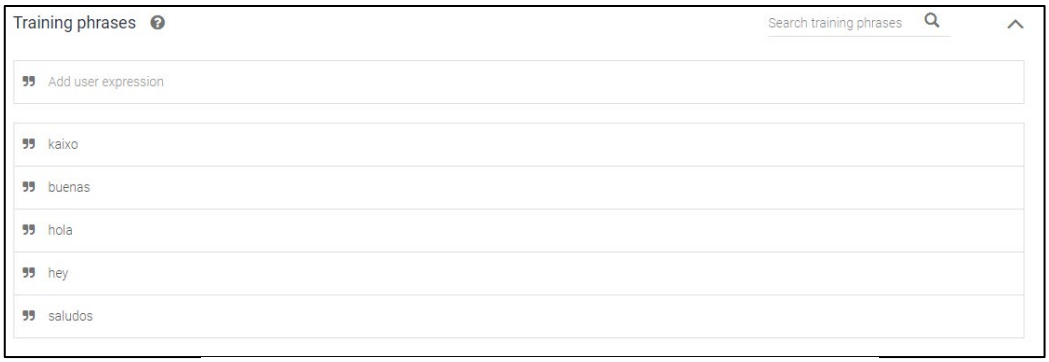

Figura 33. Frases de entrenamiento Intent "Bienvenida"

Una vez el chatbot ha comprendido que la intención del usuario es la de comenzar una conversación, es necesario automatizar una respuesta al usuario. Dicha respuesta contendrá también un mensaje de bienvenida y lo más importante, una breve explicación de aquello a lo que el chatbot es capaz de responder para evitar disconformidades como anteriormente he comentado. En la Figura 34, se puede apreciar la respuesta que se realizará cada vez que un usuario contacte con nuestro servicio de atención.

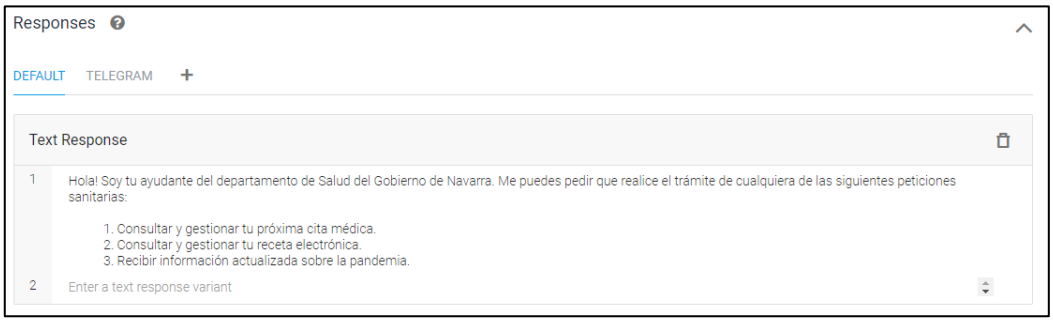

Figura 34. Respuesta de bienvenida

Tras haber respondido al usuario, el chatbot se queda a la espera de cualquier petición que el usuario le pueda realizar.

- *Intent "Cita"*: este intent se encargará de procesar las peticiones de los usuarios que quieran conocer sus futuras citas médicas. Para ello, se debe realizar un entrenamiento con frases que contengan las palabras clave que hemos definido en nuestra entity relacionadas con citas para que automáticamente aprenda a asociarlas a nuestro intent "*Cita*" (ver Figura 35).

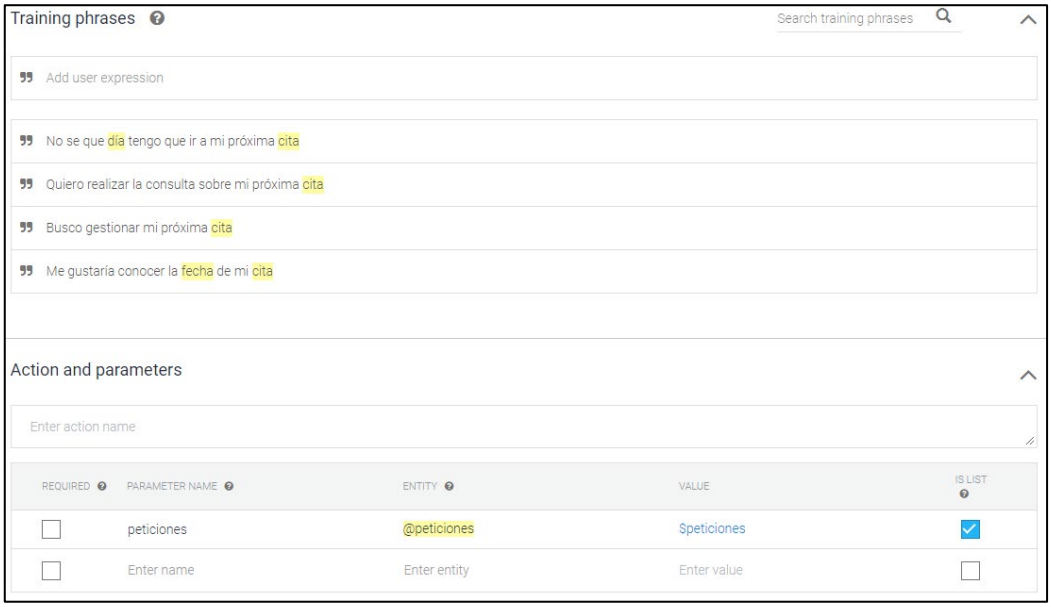

Figura 35. Frases de entrenamiento Intent "Cita"

Una vez se ha detectado que el usuario está realizando una petición sobre citas médicas, el chatbot deberá responderle dando los detalles exactos de su cita como son: la fecha, la hora, el lugar y el nombre de la planta a la que debe acudir. Para realizar dicha consulta se deberá recurrir a un servicio web, dicho proceso será comentado y explicado en profundidad en el apartado 4.3.

- *Intent "Receta"*: al igual que en el intent "*Cita*", se procedió a realizar un entrenamiento con frases, pero en este caso con palabras clave relacionadas con recetas para que se asocien al propio intent (ver Figura 36).

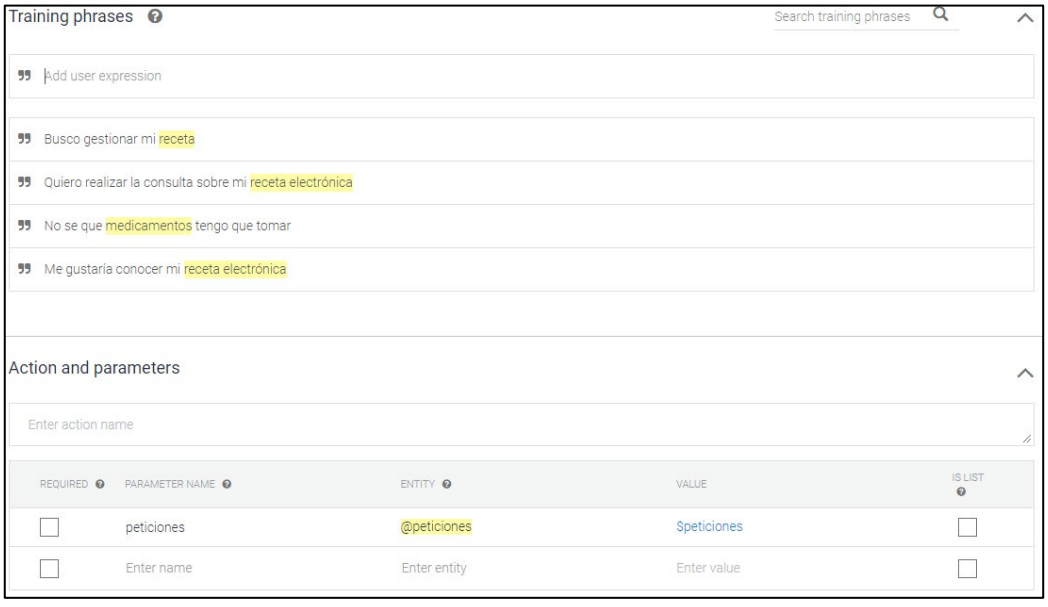

Figura 36. Frases de entrenamiento Intent "Receta"

Igualmente, el chatbot deberá responder al usuario, pero en este caso dando los detalles exactos sobre su receta como son: el medicamento, la farmacia y la fecha desde su disponibilidad en dicha farmacia.

- *Intent "InformacionCOVID"*: siguiendo el patrón de los dos últimos intents, se realizó un entrenamiento, pero en este caso con palabras clave relacionadas con la información COVID para que se asocien a este intent (ver Figura 37).

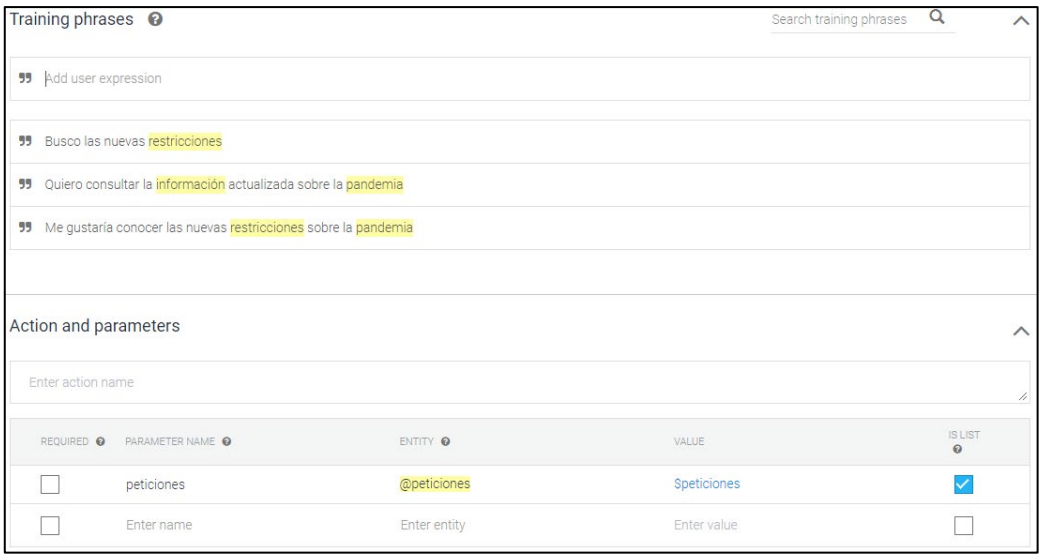

Figura 37. Frases de entrenamiento Intent "InformacionCOVID"

Al igual que en los anteriores intents, el chatbot responderá al usuario, pero en este caso dando información actualizada sobre la situación de la pandemia a través de links a las noticias de los periódicos más relevantes de este país.

- *Intent "Despedida"*: este intent se encargará de procesar y dar respuesta al último mensaje que realicen los usuarios, es decir, el de despedida. Para que dicho intent fuese capaz de procesar un mensaje de despedida, fue necesario realizar un entrenamiento previo a través de frases de entrenamiento (ver Figura 38).

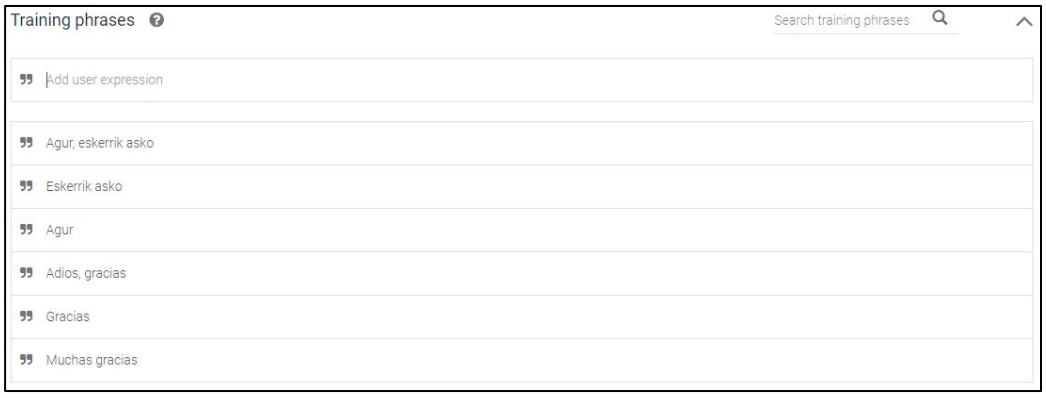

Figura 38. Frases de entrenamiento Intent "Despedida"

Una vez se conoce que la intención del usuario es la de despedirse y finalizar la conversación, el chatbot responderá al usuario con una respuesta de despedida automatizada finalizando así el servicio prestado.

- *Intent "Error"*: en el caso de que el usuario escriba algo que el chatbot procese y no sepa darle respuesta, se activará este intent. De esta forma, el chatbot dará a entender al usuario que lo que demanda, o bien no es capaz de responderlo, o bien se ha cometido algún error en la escritura de la palabra que impide procesar a la perfección su petición. Para ello, se darán una serie de respuestas automatizadas al usuario (ver Figura 39).

Figura 39. Frases de entrenamiento Intent "Error"

Tras dar respuesta al error, el chatbot se mantendrá a la espera de la corrección del usuario para poder atender de manera satisfactoria su petición.

Una vez definidos todos los intents, el chatbot se encuentra totalmente capacitado para procesar cualquier tipo de información que le llegue, pero se ha podido comprobar cómo es necesario un servicio web para poder dar respuesta a algunas de las consultas del usuario. Es por ello por lo que en el siguiente apartado abordaremos la maquetación de un servicio web que atienda las peticiones comentadas.

4.3 Creación y utilización de un Servicio Web

Se ha podido comprobar a lo largo del proceso de desarrollo de un chatbot que es primordial el uso de un servicio web para poder convertir un servicio estático en uno dinámico. El hecho de poder realizar consultas lógicas más allá de una simple respuesta aporta una gran mejoría en nuestro chatbot y una mayor credibilidad de cara al usuario que contacta con él.

Para la creación del servicio web me apoyé en los servicios de la Mancomunidad de la Comarca de Pamplona, ya que la empresa trabaja asiduamente con ellos y me dieron el permiso para maquetar el servicio de consultas. Al ser el chatbot un simple prototipo, realice un servicio para atender peticiones sobre citas médicas y recetas de un solo usuario.

Para atender ambas peticiones, fueron necesarios dos datos: el id de Telegram y el CIPNA. El id de Telegram, es el id que se le asigna a cada usuario al crearse una cuenta en la aplicación, por lo tanto, es un dato fundamental para conocer que usuario está contactando con el chatbot para saber así a quien exactamente devolver la respuesta. Para conocer nuestro propio id es tan fácil como utilizar uno de los bots proporcionados por Telegram llamado "*userinfobot*" e iniciar una conversación con él. En la Figura 40, se puede observar cómo se obtiene toda la información sobre mi usuario consiguiendo de esta forma el id que voy a utilizar para cada consulta, es decir, el "*1268433834*".

Figura 40. Información usuario Telegram

Al estar realizando peticiones sanitarias, el CIPNA del usuario (código de identificación personal de la tarjeta sanitaria) es un dato obligatorio para cualquier tipo de consulta, ya que cualquier información sanitaria se encuentra asociada a ese número. En la Figura 41, aparece un ejemplo de una tarjeta sanitaria donde se señala el CIPNA que buscamos obtener para realizar cada consulta.

Figura 41. Ejemplo CIPNA

Una vez conseguí que el servicio web se encontrara operativo, me centré en obtener el id de mi cuenta de Telegram y generé un código CIPNA para realizar las consultas. A dicho CIPNA le asigné un par de citas médicas y una receta electrónica como ejemplos para poder realizar todas las comprobaciones. Tras estos breves pasos, ya pude comenzar a maquetar el servicio de consultas utilizando las conocidas y comentadas llamadas SOAP.

Tanto para la primera llamada sobre las citas médicas como para la segunda llamada sobre las recetas fue necesaria la utilización de mi id y del CIPNA generado. En las Figuras 42 y 43, se pueden observar ambas llamadas.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://ws.t
   <soapenv:Header/>
   <soapenv:Body>
      <ws:consultaCitas>
         <!--Optional:-->
         <ws:idUsuarioTelegram>1268433834.0</ws:idUsuarioTelegram>
         <!--Optional:-->
         <ws:cipna>00576198</ws:cipna>
      </ws:consultaCitas>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 42. Llamada SOAP citas

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://ws.t
   <soapenv:Header/>
   <soapenv:Body>
      <ws:consultaRecetas>
         <!--Optional:-->
         <ws:idUsuarioTelegram>1268433834.0</ws:idUsuarioTelegram>
         <!--Optional:-->
         <ws:cipna>00559731</ws:cipna>
      </ws:consultaRecetas>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 43. Llamada SOAP recetas

En las Figuras 44 y 45, se muestran los enlaces utilizados para realizar la conexión al servicio web creado en los servicios de la MCP para responder a las llamadas.

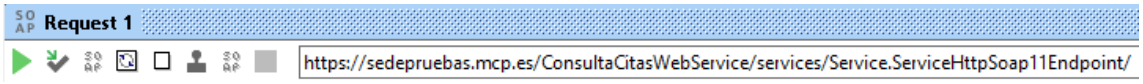

Figura 44. Enlace consulta citas

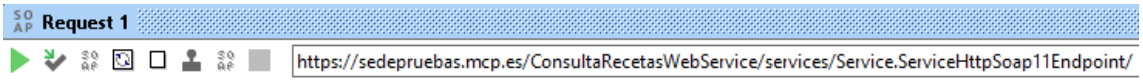

Figura 45. Enlace consulta recetas

Tras la configuración de la conexión y la realización de las llamadas, solamente quedaba por obtener la respuesta a las peticiones para comprobar que el servicio funcionaba a la perfección. En las Figuras 46 y 47, observamos la receta y las dos citas médicas pendientes por el usuario con CIPNA "*00559731*".

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header/>
   <soapenv:Body>
      <ns:consultaCitasResponse xmlns:ns="http://ws.tecdoc.ieci.com">
         <ns:return xsi:type="ax21:ConsultaCitasResponse" xmlns:ax21="http://responses.tecdoc.ieci.com/xsd" xmlns:
            <ax21:citas xsi:type="ax22:Cita">
               <ax22:fecha>2021-08-23</ax22:fecha>
               <ax22:fecha>10:30</ax22:hora>
               <ax22:lugar>Pabellón A consulta 203</ax22:lugar>
               <ax22:nombre>Oftalmología</ax22:nombre>
            </ax21:citas>
            <ax21:citas xsi:type="ax22:Cita">
               <ax22:fecha>2021-08-26</ax22:fecha>
               <ax22:fecha>11:30</ax22:hora>
               <ax22:lugar>Edificio Principal consulta 102</ax22:lugar>
               <ax22:nombre>Traumatología</ax22:nombre>
            </ax21:citas>
         </ns:return>
      </ns:consultaCitasResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 46. Respuesta a llamada SOAP citas

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header/>
   <soapenv:Body>
      <ns:consultaRecetasResponse xmlns:ns="http://ws.tecdoc.ieci.com">
         <ns:return xsi:type="ax21:ConsultaRecetasResponse" xmlns:ax21="http://responses.tecdoc.ieci.com/xsd" xml
            <ax21:recetas xsi:type="ax22:Receta">
               <ax22:fechaDisponibilidad>2021-07-27</ax22:fechaDisponibilidad>
               <ax22:nombreMedicamento>Focusín</ax22:nombreMedicamento>
               <ax22:nombreFarmacia>Farmacia Lorca Díaz</ax22:nombreFarmacia>
            </ax21:recetas>
         </ns:return>
      </ns:consultaRecetasResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 47. Respuesta a llamada SOAP recetas

Esta información es la que se debe enviar como respuesta al usuario que realice la consulta. Para ello, es necesario implementar ambas llamadas en nuestra función AWS Lambda, ya que cuando se reciba la información sobre la petición del usuario a través de los webhooks, es necesario que todo este implementado en dicha función. De esta forma, el chatbot tendrá la capacidad de responder a ambas consultas y de devolver al usuario lo que demanda consiguiendo un chatbot plenamente funcional y dinámico.

Capítulo 5

Conclusiones y líneas futuras

Tras haber finalizado el proyecto, se puede concluir que la parte diferencial a la hora de desarrollar un chatbot no se encuentra plenamente en la programación, sino que una parte muy importante, y lo que seguramente marque el éxito del producto, se encuentra en el estudio previo de mercado sobre las aplicaciones y plataformas que se vayan a usar.

El saber elegir de manera adecuada qué aplicación de mensajería y qué plataforma de desarrollo NLP utilizar, va a ser clave para el devenir del producto. Si se falla en la elección, por muy bien diseñado y programado que esté el chatbot, seguramente no tenga el éxito que hubiese tenido de haber realizado una buena elección. Por tanto, es fundamental antes de comenzar con la implementación de un chatbot considerar pausada y detenidamente todas las opciones posibles y una vez se tenga claro el objetivo, lanzarse a ello.

Por otro lado, he podido comprobar que un chatbot puede tener muchos niveles de complejidad, desde un servicio estático que simplemente responde con respuestas automatizadas, hasta un servicio dinámico que tenga la capacidad de realizar consultas para así obtener la capacidad de realizar respuestas más complejas asemejándose a las de un ser humano.

En cuanto a la arquitectura de un chatbot, tras su desarrollo he podido diferenciar tres bloques claros como son el del usuario, el del procesador del lenguaje natural y el del servicio web. Tanto el primer como el segundo bloque son primordiales a la hora de desarrollar un simple chatbot estático. En el caso de querer desarrollar un chatbot dinámico, el tercer bloque se considera fundamental, ya que el procesador del lenguaje NLP se encuentra muy limitado a la hora de conseguir respuestas lógicas y es por ello por lo que se utiliza un servicio web consiguiendo un mayor nivel en las respuestas. En la Figura 48, se puede apreciar la arquitectura gráficamente visualizando de una forma muy clara la distribución de los tres bloques.

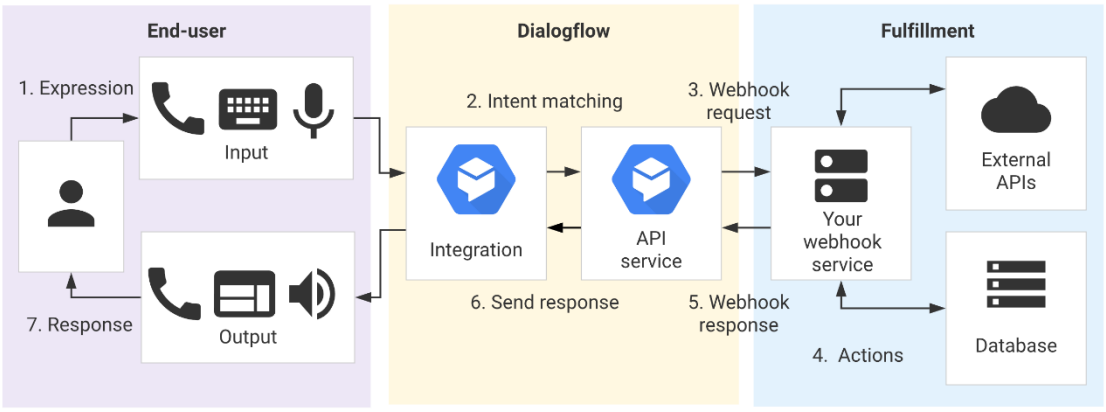

Figura 48. Arquitectura desarrollo chatbot

En resumen, se concluye que para que un chatbot tenga éxito es necesario un buen análisis previo de mercado y tener las mejores herramientas para construir un servicio dinámico que pueda ayudar a responder cualquier tipo de consulta que un usuario pueda demandar de la manera más lógica posible.

En cuanto a las líneas futuras, tengo claro que me gustaría evolucionar mi prototipo de chatbot a un chatbot que tuviera la capacidad de atender a más de un usuario y pudiera ser aplicado, por tanto, en el Servicio Navarro de Salud. Para ello, necesitaríamos tener acceso a la información médica de cada usuario a través de sus CIPNAs y poder de esta manera atender sus peticiones sin ningún tipo de impedimento.

Bibliografía

[1] Hub, R., 2021. "*La evolución de los chatbots: cómo han llegado para ser esenciales en nuestra vida" - Contact Center Hub*. [En línea] Contact Center Hub. Disponible en: https://contactcenterhub.es/la-evolucion-de-los-chatbots-como-han-llegado-para-ser-esenciales-en-nuestra-vida-2021-25-30406

[2] Builtin.com. 2021. "*What is Artificial Intelligence? How Does AI Work?"*. [En línea] Disponible en: https://builtin.com/artificial-intelligence

[3] Europarl.europa.eu. 2021. "*¿Qué es la inteligencia artificial y cómo se usa?"*. [En línea] Disponible en: https://www.europarl.europa.eu/news/es/headlines/society/20200827STO85804/que-es-la-inteligencia-artificial-y-como-se-usa

[4] Factoria, L., 2021. "*Inteligencia artificial: ¿qué impacto produce en la publicidad y el marketing?"*. [En línea] La Factoría Creativa. Disponible en: https://www.lafactoriacreativa.com/blog/marketing-online/inteligencia-artificial-publicidad-marketing/

[5] Ibm.com. 2021. "*Inteligencia artificial para una ciberseguridad más inteligente"*. [En línea] Disponible en: https://www.ibm.com/es-es/security/artificial-intelligence

[6] DirectorTIC. 2021. "*La inteligencia artificial en la industria automovilística"*. [En línea] Disponible en: https://directortic.es/noticias/la-inteligencia-artificial-en-la-industria-automovilistica-2019120123223.htm

[7] Brita Inteligencia Artificial. 2021. "*¿Qué es un Asistente virtual?"*. [En línea] Disponible en: https://brita.mx/que-es-un-asistente-virtual

[8] Instituto de Ingeniería del Conocimiento. 2021. "*Procesamiento del lenguaje natural ¿qué es?"*. [En línea] Disponible en: https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural

[9] SearchEnterpriseAI. 2021. "*What is Natural Language Processing? An Introduction to NLP".* [En línea] Disponible en: https://searchenterpriseai.techtarget.com/definition/natural-language-processing-NLP

[10] Landbot.io. 2021. "*Cómo Solicitar la API de WhatsApp para Empresas: Todo lo que necesitas saber"*. [En línea] Disponible en: https://landbot.io/es/blog/activar-api-whatsapp-para-empresas

[11] WhatsApp.com. 2021. "*Business API"*. [En línea] Disponible en: https://www.whatsapp.com/business/api

[12] Ramírez, I., 2021. "*25 bots de Telegram recomendados para sacar todo el partido a la app"*. [En línea] Xatakandroid.com. Disponible en: https://www.xatakandroid.com/comunicacion-y-mensajeria/25-bots-telegram-recomendados-para-sacar-todo-partido-a-app

[13] Datahack. 2021. "*DESARROLLO DE CHATBOTS"*. [En línea] Disponible en: https://www.datahack.es/blog/big-data/desarrollo-chatbots

[14] Chatimize. 2021. "*Dialogflow Review 2021: Should you use this chatbot builder?"*. [En línea] Disponible en: https://chatimize.com/reviews/dialogflow

[15] Google Cloud. 2021. "*Concepts | Dialogflow ES | Google Cloud*". [En línea] Disponible en: https://cloud.google.com/dialogflow/es/docs/concepts

[16] Gimenez, M., 2021. "*Amazon Web Services (AWS): ¿qué es y qué ofrece?*". [En línea] Blog de Hiberus Tecnología. Disponible en: https://www.hiberus.com/crecemos-contigo/amazon-web-services-aws-que-es-y-que-ofrece

[17] Ejemplo, T., 2021. "*Tutorial de SoapUI en español*". [En línea] Pmoinformatica.com. Disponible en: http://www.pmoinformatica.com/p/soapui-tutorial-en-espanol.html

[18] Smart Business Solutions for the Information Technology World | serem. 2021. "*¡Elija un asistente virtual proactivo para atraer a sus usuarios!*". [En línea] Disponible en: https://www.serem.com/blog/2018/07/26/elija-asistente-virtual-proactivo-atraer-usuarios