UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

TRABAJO DE GRADUACIÓN:

ANÁLISIS DE CHATBOTS PARA LA CREACIÓN DE CONVERSACIONES AUTOMATIZADAS EN TIEMPO REAL CON TECNOLOGÍAS EXISTENTES EN EL 2021 EN EL CONTEXTO DE SERVICIO AL CLIENTE EN EL SALVADOR.

MODALIDAD PROYECTO DE INVESTIGACIÓN

PARA OPTAR AL GRADO DE:

MAESTRO EN ARQUITECTURA DE SOFTWARE

AUTORES:

KAREN CECILIA CORNEJO DE VÁSQUEZ

JOSÉ RODRIGO MEJÍA GÁMEZ

ANA MARÍA RODRÍGUEZ HERNÁNDEZ

ASESOR:

MANUEL NAPOLEÓN CARDONA GUTIÉRREZ

ANTIGUO CUSCATLÁN, LA LIBERTAD, EL SALVADOR CENTROAMÉRICA

22 DE JULIO DEL 2022

AGRADECIMIENTO

Queremos agradecer primeramente a Dios, porque ha sido gracias a Él que hemos logrado culminar nuestra maestría y nos ha brindado la fortaleza para alcanzar esta meta.

A nuestras familias por todo el apoyo incondicional en cada sueño que perseguimos. Este trabajo es una superación no solo personal sino también familiar, sin nuestras familias jamás lo hubiéramos logrado.

A nuestro director de maestría y asesor de tesis por el apoyo que nos dio a lo largo de todo este proceso, por la paciencia y dedicación que dio, a nuestros maestros por compartir sus conocimientos con nosotros a lo largo de todo este tiempo.

Por último, agradecemos a todas las personas, compañeros y amigos que de una forma u otra han estado implicados en el desarrollo de este trabajo y en toda nuestra maestría, por su paciencia y su apoyo.

DEDICATORIA

Karen Cecilia Cornejo de Vásquez

Me gustaría dedicar esta tesis primeramente a Dios quien ha sido mi guía en la vida y la fortaleza para comenzar y finalizar este proyecto de maestría durante una pandemia que no esperábamos vivir.

Agradecer a mi esposo Mario Vásquez quien es y será siempre la inspiración para seguir adelante y querer ser mejor cada día, por su paciencia y apoyo durante el desarrollo de esta maestría.

A mis compañeros de tesis, Ana por ser siempre la persona encargada de motivarnos a trabajar duro hasta alcanzar nuestros objetivos, a Rodrigo por ser la persona con ideas frescas y siempre apoyar en todo, gracias a ellos este trabajo de graduación ha sido una aventura que ha valido la pena.

También agradecer a TELUS por su apoyo económico y por creer siempre en nuestro desarrollo profesional y personal.

Y por último agradecer a nuestro director de maestría que siempre busco brindarnos todos los recursos que necesitábamos para el desarrollo de las materias y a nuestro asesor por guiarnos en el desarrollo de este trabajo de graduación y siempre brindarnos retroalimentación en cómo mejorar en cada fase de esta.

José Rodrigo Mejía Gámez

Atravesar todo el camino de la maestría no ha sido algo sencillo, ha sido un camino con muchos retos, los cuales se han podido superar y llegar hasta el desarrollo y culminación de la presente tesis. Durante todo el trayecto siempre sentí el apoyo no sólo de Dios, sino que también el de mis compañeras en cada proyecto que se iba realizando, a las cuales agradezco mucho por su motivación y sus experiencias laborales que han servido de mucho tanto en los módulos como en la tesis.

Dedicada también a mi esposa, Andrea Cárcamo de Mejía, que durante todo el desarrollo de la tesis fue de gran ayuda y soporte emocional, por sus atenciones y apoyo para seguir adelante no sólo en los estudios, sino que en muchos ámbitos de la vida.

Agradeciéndoles a los docentes de la universidad, que compartieron su conocimiento y experiencia en cada uno de los módulos, cuyo trabajo no ha sido en vano, sino que ha fomentado a hacer crecer nuestro saber y poder aplicarlo en la vida profesional.

Ana María Rodríguez Hernández

Ante todo, agradecer a Dios por ayudarme a concluir esta maestría, brindándome la fuerza y la inteligencia para superar las adversidades.

Un agradecimiento especial a mi futuro esposo Jorge Burgos por su apoyo, comprensión y motivación durante todo este proceso, a mi papa y a mi familia Rodríguez por ser un ejemplo para buscar la superación profesional.

También un agradecimiento a mis compañeros de maestría y de tesis Karen y Rodrigo por su trabajo, apoyo, esfuerzo, entrega y dedicación para finalizar con éxito este trabajo y todo este proceso, ya que ha sido importante para finalizar con éxito esta etapa. También un agradecimiento a Telus International por el apoyo financiero para cursar esta maestría.

Un agradecimiento a los catedráticos, quienes se encargaron de guiarme y apoyarme durante cada módulo de la maestría, a nuestro asesor de tesis por el apoyo y soporte brindado en todo el proceso de tesis y a nuestro director de maestría por todo el apoyo y soporte brindado durante todo este proceso de aprendizaje.

RESUMEN

Las compañías de servicio al cliente son empresas BPO encargadas de proveer a los clientes la administración de diversos servicios generalmente orientados al trato directo con el cliente o usuario final. Su oferta de servicios puede variar con base en el caso de negocio que se presente, como atención directa, atención b2b (business-to-business), entre otras. Debido a la amplia diferencia entre cada proceso de negocio a tercerizar, la necesidad fundamental de toda empresa BPO es definir y validar los requerimientos específicos con el cliente, a fin de buscar la mejor estrategia para poder brindar el servicio que mejor se adapte a sus necesidades.

El objetivo principal de las compañías BPO es reducir los costos de las empresas clientes y ayudarles a crecer económicamente, a través de la optimización del tiempo y recursos de sus procesos de negocio. Inmerso en estos procesos de negocio se tienen los procesos de servicio al cliente, cuyo objetivo fundamental es proveer asesoría a los consumidores durante todo el proceso de compra y posterior, a fin de poder establecer una relación entre el negocio y el cliente, lo cual se traduce en fidelización, divulgación de marca y aumento en el volumen de las ventas.

En la actualidad y con los avances tecnológicos existentes, la tecnología forma parte de la vida diaria de las personas y se utiliza para realizar el trabajo de forma más eficiente y eficaz. Actualmente existen máquinas que hacen actividades que antes eran realizadas por humanos, con esto se ha logrado mejorar la productividad y los procesos existentes dentro de las empresas.

Los chatbots son la última tecnología que está cambiando y revolucionando al mundo en relación a la forma de cómo una máquina puede interactuar y simular una conversación entre dos entidades (humano y máquina). Esto se lleva a cabo a través de la interpretación del lenguaje natural de los humanos por parte de las máquinas, esta interpretación del lenguaje natural hace que esta tecnología pueda ser incorporada en las empresas de servicio al cliente para mejorar procesos de comunicación y consulta que contribuyen en la agilización del negocio y por consecuencia en la mejora de la atención prestada.

Los chatbots son programas de software que utilizan algoritmos, reglas y procesamiento de lenguaje natural con el objetivo de simular una conversación con un humano, están diseñados de forma que puedan permitir la comunicación automática entre humanos y máquinas a través de una serie de configuraciones a nivel de infraestructura y arquitectura de software. Estos poseen funcionalidades técnicas y administrativas como; contar con su propia base de datos, integración con otros sistemas nativos, módulo de supervisión y estadísticas de agente, integración analítica, operación en la nube y personalización con la base de conocimiento.

Hoy en día la interacción que el ser humano tiene con los distintos dispositivos electrónicos a través de sus distintas interfaces representa la principal forma de interacción y de recopilación de datos, es por ello que fue necesario crear distintas formas de interpretar estos datos para poder utilizar esta información para que los chatbot fueran capaces de "entender" al ser humano y es aquí donde la lingüística computacional ayuda utilizando herramientas de la informática como el procesamiento del lenguaje natural, la inteligencia artificial y el aprendizaje automático.

Estas tecnologías se utilizan para proporcionar resultados a las preguntas hechas en las interacciones con el ser humano, los chatbots deben anticiparse en cuanto a las posibles respuestas utilizando algoritmos con capacidad de lectura rápida, clasificación, interpretación, aprendizaje entre otros. Así las respuestas serán cada vez más acertadas o correctas dependiendo del entrenamiento previo y personalizado que se le dé.

En la actualidad existen una gran cantidad de proveedores para la creación de chatbots, pero para efectos de esta tesis sólo se investigaron los proveedores Google y Microsoft debido al posicionamiento de estas dos empresas en el mundo de la tecnología actual.

El proveedor de Google ofrece Dialogflow que es una plataforma de comprensión del lenguaje natural que facilita el diseño y la integración de una interfaz de usuario conversacional en diferentes formatos que puede analizar múltiples tipos de entradas, incluidas entradas de texto o audio. También puede responder de varias maneras, ya sea a través de texto o con voz sintética y además es parte de la oferta de IA conversacional dentro de GCP.

Dialogflow proporciona dos servicios de agente virtual (ES y CX), cada uno con su propio tipo de agente, interfaz de usuario, API, bibliotecas de clientes y documentación. También ofrece tres ediciones, de las cuales dos de ellas (ES y CX) tienen la modalidad de pago por uso (pay-as-you-go) lo que significa que solo se paga el tiempo y los elementos que se utilizan. Y la edición gratuita que proporciona la mayoría de las funciones del tipo de agente ES estándar, pero con cupo limitado y soporte para la comunidad y correo electrónico. Esta edición se recomienda para experimentar con Dialogflow.

Por otro lado, el proveedor de Microsoft ofrece Bot Framework Composer que es un canvas de código abierto para que permite diseñar y crear experiencias conversacionales con Language Understanding, QnA Maker y una composición sofisticada de respuestas de bot (Language Generation). Con la cual se puede crear un bot con la capacidad de hablar, escuchar, comprender y aprender de los usuarios con la integración nativa de Azure Cognitive Services. Azure Bot Service permite crear bots inteligentes de nivel empresarial con propiedad y control de los datos. Da la capacidad de crear un simple bot de preguntas y respuestas o crear un asistente virtual sofisticado.

Este está basado en Bot Framework SDK y es un IDE de código abierto para que los desarrolladores creen, prueben y administren experiencias conversacionales. Proporciona un poderoso lienzo de creación visual que permite la creación de diálogos, modelos de comprensión del lenguaje, bases de conocimiento de QnAMaker y respuestas de generación de lenguaje desde un solo lienzo y que permite que estas experiencias se amplíen con código para tareas más complejas, como la integración del sistema. Las experiencias resultantes se pueden probar en Composer y aprovisionar en Azure junto con los recursos dependientes. Composer está disponible como aplicación de escritorio para Windows, macOS y Linux. Si la aplicación de escritorio no se adapta también se puede compilar Composer desde el origen o alojar Composer en la nube.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS, SIMBOLOGÍA Y SIGLAS

AI: Se refiere a sistemas o máquinas que imitan la inteligencia humana para realizar tareas y pueden mejorar iterativamente a partir de la información que recopilan. La IA se manifiesta de varias formas. Algunos ejemplos son: Los chatbots utilizan la IA para comprender más rápido los problemas de los clientes y proporcionar respuestas más eficientes

Agentes inteligentes: Es un sistema perceptivo capaz de interpretar y procesar la información que recibe de su entorno, actuando en consecuencia de acuerdo a los datos que recoge y procesa. La forma de actuar de esta entidad es lógica y racional basándose en las reacciones del comportamiento normal de un sistema en concreto.

Algoritmo: Es cualquier procedimiento computacional bien definido que parte de un estado inicial y un valor o un conjunto de valores de entrada, a los cuales se les aplica una secuencia de pasos computacionales finitos, produciendo una salida o solución.

API: La Application Programming Interface (API) es un conjunto de patrones que forman parte de una interfaz que permite la creación de plataformas de una forma más sencilla y práctica para desarrolladores.

Bot: Los bots (aféresis de robot) se tratan de un software o programa informático que se sirve de la inteligencia artificial (IA) para realizar tareas automatizadas a través de internet como si se tratase de un ser humano.

Entidad: Son objetos concretos o abstractos que presentan interés para el sistema y sobre los que se recoge información que será representada en un sistema de bases de datos. Por ejemplo, clientes, proveedores y facturas serían entidades en el entorno de una empresa.

FaaS: La función como servicio (FaaS) es un tipo de servicio de cloud computing que permite que los desarrolladores diseñen, ejecuten y gestionen paquetes de aplicaciones como funciones sin tener que ocuparse del mantenimiento de su propia infraestructura.
Fulfillment.

GCP: Google Cloud Platform ofrece una infraestructura de cloud escalable, rentable y de gran rendimiento. Informática ofrece el medio más ágil para entregar, transformar, gestionar y sincronizar datos fiables e integrados de fuentes de multi-cloud, del entorno local y del Big Data en Google Cloud.

Instancia de bases de datos: Una instancia de Motor de base de datos es una copia del ejecutable de sqlservr.exe que se ejecuta como un servicio de sistema operativo. Cada instancia administra varias bases de datos del sistema y una o varias bases de datos de usuario.

Lenguaje natural: Un lenguaje de programación en que el programador escribe especificaciones sin tomar en consideración el formato o la sintaxis de la instrucción de computación; esencialmente, usando un lenguaje cotidiano para programar.

Robots: Es una máquina automática programable capaz de realizar determinadas operaciones de manera autónoma y sustituir a los seres humanos en algunas tareas.

Redes neuronales artificiales: Están basadas en el funcionamiento de las redes de neuronas biológicas. Las neuronas que todos tenemos en nuestro cerebro están compuestas de dendritas, el soma y el axón: Las dendritas se encargan de captar los impulsos nerviosos que emiten otras neuronas. Estos impulsos, se procesan en el soma y se transmiten a través del axón que emite un impulso nervioso hacia las neuronas contiguas.

Sistemas expertos: Son aplicaciones tecnológicas que pertenecen al gran campo de la inteligencia artificial y pueden identificarse como programas informáticos que reproducen la actuación de uno o varios expertos en un campo de actividad determinado (dominio). Estos sistemas están ya

bastante maduros y, por lo tanto, son ampliamente aplicables en muchos contextos; un ejemplo es la venta de productos complejos para la que es imprescindible contar con un configurador de productos.

Tecnología: Quiere decir arte, oficio o destreza. Por lo tanto, la tecnología no es una cosa sino un proceso, una capacidad de transformar o combinar algo ya existente para construir algo nuevo o bien darle otra función. Y esa capacidad de transformación puede ser intuitiva o bien (como sucede en nuestras sociedades actuales) se trata de un saber que proviene directamente del campo de las ciencias.

Webhook: Son eventos que desencadenan acciones. Su nombre se debe a que funcionan como «ganchos» de los programas en Internet y casi siempre se utilizan para la comunicación entre sistemas. Son la manera más sencilla de obtener un aviso cuando algo ocurre en otro sistema.

INTRODUCCIÓN

En la actualidad el uso de la tecnología en las empresas de servicio al cliente ha ido aumento cada vez, esto con el objetivo de aprovechar los beneficios que trae el uso de herramientas que utilizan la Inteligencia Artificial y con ello reducir gastos en recursos humanos en lo que se refiere a la interacción de clientes y soporte técnico.

Las nuevas tecnologías basadas en la administración y almacenamiento en la nube se encuentran incursionando en esta área de la IA a través de la disponibilidad de recursos como el diseño personalizado de "Chatbots", algo que permite a los desarrolladores crearlos y administrarlos según las necesidades de cada negocio.

Reducir la interacción entre el ser humano en procesos y procedimientos que pueden realizarse de forma remota se ha convertido en una de las prioridades para este tipo de empresas no solo en el mundo sino también en El Salvador debido a la pandemia por covid19, esto con el fin de reducir el contacto humano, sino también con el de reducir costos administrativos.

La implementación de estas tecnologías debe realizarse luego de evaluar los costos y los beneficios de esta nueva forma de atención al cliente, es por ello que en el siguiente trabajo de graduación se realizará un análisis desde el origen de esta nueva opción, así como el beneficio económico que puede representar para las empresas del país.

CAPÍTULO 1
1. PLANTEAMIENTO DEL PROBLEMA

1.1 FORMULACIÓN DEL PROBLEMA

¿Cómo los servicios ofrecidos por Google en conjunto con el lenguaje de programación Nodejs y la gestión de datos a través de bases de datos ayudan gestionar y automatizar el servicio al cliente a través de la implementación de chatbots para la creación de conversaciones automatizadas en tiempo real capaces de interpretar el lenguaje natural?

1.2 OBJETIVO GENERAL

Desarrollar una investigación descriptiva y exploratoria de cómo los servicios ofrecidos por Google en conjunto con el lenguaje de programación Nodejs y las bases de datos, ayudan a la implementación de chatbots para la creación de conversaciones automatizadas en tiempo real capaces de interpretar el lenguaje natural, los cuales pueden ser utilizados en el proceso de gestión y automatización del servicio al cliente.

1.3 OBJETIVOS ESPECÍFICOS

- Sentar los fundamentos teóricos de los chatbots y su capacidad para interpretar el lenguaje natural.
- Indagar las necesidades existentes en las compañías dedicadas al servicio al cliente en El Salvador.
- Investigar cómo el lenguaje de programación Nodejs, las bases de datos y las tecnologías ofrecidas por Google ayudan a la implementación de chatbots.
- Determinar la factibilidad económica de la implementación de un chatbot en las empresas de servicio al cliente.
- Describir cómo llevar a cabo la implementación de un chatbot para crear conversaciones automatizadas en tiempo real.

1.4 ALCANCES

- La investigación abarca conceptualmente algoritmos de machine learning e inteligencia artificial utilizados para la interpretación del lenguaje natural.
- La investigación incluye una guía para la creación de un chatbot como prueba de concepto utilizando las tecnologías ofrecidas por Google en conjunto con nodejs y una base relacional.
- La investigación incluirá el perfil técnico e infraestructura necesaria para el diseño y creación del chatbot basado en las tecnologías ofrecidas por el proveedor seleccionado Google.
- La investigación incluirá un análisis costo-beneficio en empresas localizadas en El Salvador y, por tanto, en las cantidades de pago promediadas brindadas por las mismas.

1.5 LIMITACIONES

- Las tecnologías utilizadas para la creación del chatbot son versiones gratuitas de los servicios ofrecidos por Google y GCP por lo tanto la prueba de concepto estará limitada a los servicios gratuitos ofrecidos por el proveedor.
- Debido a la limitación de utilizar versiones gratuitas en la prueba de concepto, no se podrán mostrar todas las bondades que proporciona un chatbot 100% funcional en una empresa de servicio al cliente en la vida real.
- Por motivos de confidencialidad, no se detalla de qué empresas es la información utilizada para realizar los análisis de costos, sin embargo, todas las empresas de las cuales se ha tomado la misma tienen operaciones en El Salvador.

1.6 JUSTIFICACIÓN

En la actualidad y con los avances tecnológicos existentes, la tecnología forma parte de la vida diaria de las personas y es utilizada para realizar el trabajo de forma más eficiente y eficaz. Actualmente existen máquinas que hacen actividades que antes eran realizadas por humanos, con esto se ha logrado mejorar la productividad y los procesos existentes dentro de las empresas.

Los chatbots son la última tecnología que está cambiando y revolucionando al mundo en relación a la forma de cómo una máquina puede interactuar y simular una conversación entre dos entidades (humano y máquina). Esto se lleva a cabo a través de la interpretación del lenguaje natural de los

humanos por parte de las máquinas, esta interpretación del lenguaje natural hace que esta tecnología pueda incorporarse en las empresas de servicio al cliente para mejorar procesos de comunicación y consulta que contribuyen en la agilización del negocio y por consecuencia en la mejora de la atención al cliente.

La finalidad de esta investigación es presentar y analizar los fundamentos teóricos que giran en torno a esta nueva tecnología de chatbots y su capacidad de interpretar el lenguaje natural que puede ser utilizado para crear conversaciones automatizadas en tiempo real.

Para esto se tomará como objeto de estudio tecnologías que ayuden a la creación e implementación de chatbots existentes en el 2021, como por ejemplo los servicios ofrecidos por Google, el lenguaje de programación Nodejs, la gestión de datos a través de bases de datos relacionales y la inteligencia artificial.

Esto es importante ya que para poder implementar una nueva tecnología primero se debe conocer las bases teóricas y las herramientas existentes que nos ayudarán a lograr una implementación exitosa que no solo ayude a solventar las necesidades del negocio sino a identificar áreas de mejora.

Esta investigación se realiza en el contexto del servicio al cliente por tanto el enfoque está orientado a este rubro, pero el análisis tecnológico podrá ayudar al lector a identificar en qué otras áreas de su empresa podrían utilizarse esta tecnología.

1.7 HIPÓTESIS

¿Cómo los servicios ofrecidos por Google en conjunto con el lenguaje de programación Nodejs y las bases de datos, pueden ser utilizados para la implementación de chatbots que ayuden a gestionar el servicio al cliente generando conversaciones automatizadas capaces de interpretar el lenguaje natural en tiempo real?

1.8 TIPO DE INVESTIGACION Y METODOLOGIA

Para definir la metodología de investigación es muy importante el seleccionar el tipo de investigación a realizar, en el caso de este proyecto se ha seleccionada la investigación tecnológica aplicada (Sánchez, 2011).

La investigación tecnológica tiene por finalidad la investigación de artefactos o procesos con el objeto de ofrecerlos al mercado para obtener beneficios. Esta suele apoyarse en la investigación aplicada que en muchos casos requiere del aporte de varias ramas especializadas, esta comprende el conjunto de actividades que tiene por finalidad el descubrir o aplicar conocimientos que puedan realizarse en productos o en procesos nuevos utilizables (Sánchez, 2011).

Este tipo de investigación se enfoca en encontrar estrategias para lograr un objetivo específico y ponerlo en práctica al ser tecnología nos da la oportunidad de aplicar el nuevo conocimiento con el propósito de favorecer la vida de las personas, en este caso se busca una forma de automatizar los procesos de servicio al cliente a través del análisis de chatbots para crear conversaciones automatizadas en tiempo real (Sánchez, 2011).

El desarrollo de esta investigación será basado en la aplicación de técnicas, procedimientos, herramientas y métodos. Para poder seleccionar las adecuadas es muy importante conocer el objetivo del trabajo, en nuestro caso se enfoca en desarrollar una investigación descriptiva y exploratoria sobre el análisis de chatbots para la creación de conversaciones automatizadas en tiempo real utilizando tecnologías existentes en el 2021 en el contexto de servicio al cliente en El Salvador.

Para poder definir la metodología de investigación se debe entender las técnicas a ser aplicadas. Las técnicas son los métodos prácticos de investigación que se utilizaran para obtener la evidencia necesaria que fundamente las opiniones y conclusiones. Para esta investigación se ha seleccionado la técnica documental de investigación, la cual consiste en la recopilación de información acudiendo a fuentes previas como investigaciones ajenas, libros, documentos electrónicos expuestos en la web, documentos oficiales, etc. (Moreno et al., 2020).

Adicionalmente la obtención de datos e información por medio de documentos secundarios como una recopilación por medio de una encuesta a personas dentro del rubro de servicio al cliente. Debido a que para poder tener un mejor conocimiento de las necesidades que presentan las empresas del rubro especificado, se necesita conocer de primera mano de las personas responsables de ofrecer el servicio.

Por otro lado, es muy importante definir el método a ser utilizado para realizar este proyecto de investigación. Se ha seleccionado el método operativo el cual es la combinación del método cualitativo y cuantitativo. Ya que Según Sampieri & Torres (2018), al utilizar el enfoque mixto en la investigación se obtienen diversas bondades como:

- Lograr una perspectiva más amplia y profunda del fenómeno.
- La percepción de este resulta más integral y completa, dándole más confianza a la investigación.
- Producir datos más ricos y variables mediante la multiplicidad de observaciones ya que se consideran diversas fuentes y tipos de datos que rompen con la investigación uniforme.

Adicionalmente esta investigación se llevará a cabo de forma remota, esto se basa en tener todas las interacciones utilizando la web o medios digitales. Esta forma de investigar es muy poco común pero debido a la situación actual de la pandemia por COVID-19 se presenta como la mejor opción.

Además, es muy importante el seleccionar las herramientas adecuadas ya que estas son el conjunto de elementos que permiten llevar a cabo las acciones definidas en las técnicas, es por ello que las herramientas principales a ser utilizadas son:

- Encuestas
- Libros
- Documentos oficiales
- Documentos digitales

Todas las herramientas seleccionadas se pueden aplicar de forma digital y remota dado que por la situación actual del COVID-19, es recomendable no tener interacciones físicas.

CAPITULO 2.
2. ESTADO DEL ARTE

2.1 CONTEXTO SOBRE LAS COMPAÑÍAS DE SERVICIO AL CLIENTE BPO

¿Qué son las compañías de servicio al cliente?

BPO son las siglas en inglés para Business Process Outsourcing (Externalización de Procesos de Negocio), el cual consiste en proveer a los clientes la administración de diversos servicios o funcionalidades, tales como manufacturación de productos, transporte, desarrollo de software, contabilidad, atención al cliente, etc. Estos servicios son brindados por compañías que se especializan en ciertas ramas del BPO, por ejemplo, existen empresas que proveen el servicio de manufactura y almacenamiento de productos, o que se enfocan en el área de desarrollo de software (Rantakari, 2010).

Generalmente las compañías de servicio al cliente BPO brindan servicios orientados al trato directo con el cliente o usuario final. Su oferta de servicios puede variar con base en el caso de negocio que se presente, como atención directa, atención b2b (business-to-business), entre otras.

¿Cuáles son las necesidades de las compañías de servicio al cliente?

Debido a la amplia diferencia entre cada proceso de negocio a tercerizar, la necesidad fundamental de toda empresa BPO es definir y validar los requerimientos específicos con el cliente, a fin de buscar la mejor estrategia para poder brindar el servicio que mejor se adapte a lo que se requiera. Este debe ser acompañado de un análisis costo-beneficio para determinar qué tan rentable resulta la transacción económica para la empresa, ayudando a establecer cuánto se le cobrará a la compañía solicitante de los servicios y cuáles de ellos se pueden ofrecer.

Otra necesidad fundamental que se tiene en este tipo de empresas y que está más orientada dentro de los procesos de negocio como tal, es la recepción de la retroalimentación y evaluación de los servicios brindados. Esta evaluación es muy necesaria para conocer qué tan óptimas han sido las estrategias implementadas para proveer las soluciones, a través de la asignación de valor monetario

a la calidad del servicio, para poder definir ciertas métricas que ayuden a la toma de decisiones ejecutivas.

2.2 OBJETIVO DE LAS COMPAÑÍAS DE SERVICIO AL CLIENTE

El objetivo principal de las compañías BPO es reducir los costos de las empresas clientes y ayudarles a crecer económicamente, a través de la optimización del tiempo y recursos de sus procesos de negocio. Inmerso en estos procesos de negocio se tienen los procesos de servicio al cliente, cuyo objetivo fundamental es proveer asesoría a los consumidores durante todo el proceso de compra y posterior a ello, a fin de poder establecer una relación entre el negocio y el cliente, lo cual se traduce en fidelización, divulgación de marca y aumento en el volumen de las ventas.

2.3 DESCRIPCIÓN DE PROCESOS COMUNES DE SERVICIO AL CLIENTE

En la industria de BPO se tienen dos procesos de negocio que se le ofrecen a las empresas clientes: el “Front-office” y el “Back-office”.

Front-Office

Los procesos front-office se encargan de las actividades externas de la compañía que ha solicitado los servicios BPO, las cuales implican contacto con clientes directos de la misma, como por ejemplo ventas, marketing, soporte técnico, etc. Las actividades de front-office están directamente relacionadas con la generación de ingresos de la empresa cliente y, por tanto, todo lo que se lleva a cabo en dichas actividades tiende a ser muy delicado.

El primer punto en todo el espectro de procesos front-office es la interacción con el cliente, el cual es atendido por un agente especializado en el área requerida para poder solventar las necesidades que se presenten. Esta atención se brinda a través de una llamada o chat, según se haya solicitado por el cliente. La transacción finaliza cuando el cliente haya determinado la solución a su problema, cuando el caso se haya escalado porque no ha podido ser solucionado o porque haya existido una interrupción en la interacción.

Al finalizar el contacto con el cliente, se realiza una encuesta de satisfacción del servicio brindado por el agente y se almacena el resultado, junto con las siguientes métricas adicionales:

- Tiempo de interacción
- Existió o no una escalación
- Categoría
- Línea de negocio
- Cuenta

Una vez al día, generalmente en el cierre, los departamentos de aseguramiento de calidad y de administración de personal revisan las métricas de todas las interacciones recibidas en el día para poder presentarlas a las empresas clientes y tener la visibilidad para generar planes de acción con respecto a sus estrategias de manejo de clientes. Adicionalmente esta información también es utilizada por el departamento de cobros para poder generar la facturación de servicios a la empresa solicitante.

Las empresas de servicio al cliente se enfocan principalmente en los procesos de front-office, por la misma naturaleza de las actividades, las cuales implican el contacto directo con el cliente para brindar soporte antes, durante o después de la adquisición de un producto o servicio, siendo estos procesos el punto central de toda la empresa.

Back-Office

Los procesos back office se encargan de los aspectos internos de un negocio, no directamente relacionados con la generación de ingresos del mismo, sino que a los procesos de soporte, como por ejemplo contabilidad, gestión de talento humano, gestión de planillas, inventarios, etc.

A diferencia del front-office, estos procesos nunca tienen interacción directa con los clientes de la empresa solicitante y por tanto solo se limitan a cumplir las directrices brindadas por la misma. Sin embargo, siempre se realizan encuestas de satisfacción para conocer áreas de mejora en cuanto a servicio proporcionado y generar métricas de calidad acorde a ellas. La frecuencia de estas encuestas es mucho menor que las de front-office, generalmente el período en el cual se realiza es

entre uno a tres meses calendario, dependiendo del arreglo contractual que se tengan con las empresas solicitantes.

2.4 DESCRIPCIÓN DE PRINCIPALES PROBLEMAS EN LOS PROCESOS DE SERVICIO AL CLIENTE

Las métricas más relevantes para las empresas BPO son las de los tiempos de interacción, si existió o no una escalación y la encuesta de satisfacción del cliente.

Los tiempos de interacción varían según la línea de negocio sobre la cual se esté tratando, sin embargo, existen diversos temas de ayuda y soporte que, por muy sencillos que sean, pueden incrementar estos tiempos de interacción debido a la brecha tecnológica que los usuarios poseen a la hora de realizar estos tipos de gestión, lo cual también aumenta la posibilidad de una escalación.

Este incremento afecta de manera negativa las métricas de las cuentas BPO, las cuales deben resolver de la manera más rápida y efectiva posible cualquier caso que se presente, y estos tiempos de interacción más elevados de lo normal pueden dar malas impresiones a los clientes y causar pérdidas monetarias a la empresa.

Adicionalmente a este problema, cuando un agente tiene tiempo de interacción alto en algún caso, se pierde la capacidad de poder atender otros casos en las 6 a 8 horas que un agente labora normalmente en estas industrias, causando que la empresa pueda verse afectada en el pago de horas extras a dichos empleados (GAO, 2008).

2.5 ANÁLISIS DE COSTOS DE PROCESOS DE SERVICIO AL CLIENTE

Como se ha mencionado en el punto 2.2, se tienen dos tipos de procesos de servicio al cliente: front-office y back-office. Los procesos más variables en todo el espectro son los front-office, dado que son los que implican contacto directo con los clientes y que tienen tiempos de interacción muy diferentes dependiendo del caso que se necesite resolver.

Para este análisis, se han tomado los costos operativos de cuatro empresas en el área de front-office, los cuales brindan el costo promedio por hora por agente en cada una de las mismas presentado a continuación:

Tabla 1. Tabla de costos promedios por agente por tipo de servicio por hora de empresas muestra.

	Support	Sales	Customer Service
Empresa 1	$5.65	$3.75	$4.25
Empresa 2	$4.64	$3.63	$3.75
Empresa 3	$4.72	$3.65	$3.80
Empresa 4	$5.43	$3.75	$4.08
Promedio	**$5.11**	**$3.70**	**$3.97**

2.6 CONTEXTO DE CHATBOTS Y LENGUAJE NATURAL

En la actualidad y con los avances tecnológicos existentes, la tecnología forma parte de la vida diaria de las personas y es utilizada para realizar el trabajo de forma más eficiente y eficaz. Actualmente existen máquinas que hacen actividades que antes eran realizadas por humanos, con esto se ha logrado mejorar la productividad y los procesos existentes dentro de las empresas.

Los chatbots son la última tecnología que está cambiando y revolucionando al mundo en relación a la forma de cómo una máquina puede interactuar y simular una conversación entre dos entidades (humano y máquina). Esto se lleva a cabo a través de la interpretación del lenguaje natural de los humanos por parte de las máquinas, esta interpretación del lenguaje natural hace que esta tecnología pueda incorporarse en las empresas de servicio al cliente para mejorar procesos de comunicación y consulta que contribuyen en la agilización del negocio y por consecuencia en la mejora de la atención prestada.

¿Qué es un chatbot?

Son programas de software que utilizan algoritmos, reglas y procesamiento de lenguaje natural (NLP: Natural Language Processing) que tienen como objetivo simular una conversación con un humano.

Los chatbots se encuentran diseñados de forma que puedan permitir la comunicación automática entre humanos y máquinas a través de una serie de configuraciones a nivel de infraestructura y arquitectura de software. Estos poseen funcionalidades técnicas y administrativas como; contar con su propia base de datos, integración con otros sistemas nativos, módulo de supervisión y estadísticas de agente, integración analítica, operación en la nube y personalización con la base de conocimiento (Mathieson, 2018).

Los C (Mathieson, 2018)hatbot nacieron de la I.A. En su artículo el autor manifiesta que las organizaciones comerciales en la actualidad independientemente de su campo ya sea científicos, comerciales, financieros, académicos, industriales, gubernamentales, entre otros, están utilizando esta tecnología para brindar soluciones correctas a sus clientes, utilizándolos como asistentes virtuales para el análisis profundo de las incidencias que presentan los usuarios de sus servicios, la I.A. les permite un aprendizaje automático para encontrar soluciones adecuadas.

Para proporcionar resultados a las preguntas en la interacción con el ser humano, los chatbots deben anticiparse en cuanto a las posibles respuestas utilizando algoritmos con capacidad de lectura rápida y clasificación, esta respuesta será cada vez más acertada o considerada correcta dependiendo del entrenamiento previo y personalizado a través de los desarrolladores o de los propios usuarios que utilicen la herramienta.

Los chatbots se caracterizan por establecer comunicación de forma multiusuario y omnicanal, esto quiere decir que en esta tecnología se encuentra preparada para satisfacer la interacción con ella en los diferentes dispositivos tecnológicos como: computadores, teléfonos inteligentes, tablets, etc. Es decir, conversar con el usuario por diferentes medios electrónicos y canales digitales y poder

analizar la información y generación de estadísticas de atención a usuarios en tiempo real mediante tableros de control (Microsoft Chattigo, 2018).

Sus orígenes

Todo comenzó con el Test de Turing el cual fue diseñado por el matemático británico Alan Turing en 1950, la implementación de este Test se basa en un juego, al que llamó "The Imitation Game", en este ejercicio participan dos personas y una máquina (Alan Turing: The Enigma, 1983).

El objetivo del juego es que el jurado(una persona) identifique quién es la máquina y quién la persona, donde ambos la otra persona como la máquina no están en la misma habitación que el jurado y este solo puede comunicarse con ambas mediante preguntas escritas, las cuales son respondidas en igual forma, donde la hipótesis a probar es que si el jurado no es capaz de identificar quién es la persona y quien es la máquina con base en sus respuestas, se puede considerar que la máquina piensa por sí sola al igual que un ser humano.

Luego de estas pruebas en los siguientes años se crearon los primeros chatbots:

- ELIZA, primer chatbot creado por Joseph Weinzebaum en 1966 en el MIT, se basó en un algoritmo que buscaba palabras clave en las frases del usuario y respondía de manera parecida a un psicólogo, reformulando la frase del usuario. Este fue la base para el siguiente modelo PARRY el cual utilizó palabras clave y patrones en su diseño, este fue utilizado para impersonar paranoia y esquizofrenia el cual sirvió para evaluar conversaciones con pacientes que sufrían paranoia (Kenneth Colby, 1972).
- A.L.I.C.E, este chatbot representó una versión mucho más completa de un chatbot ya que genera respuestas más complejas teniendo en cuenta patrones de entrada y plantillas en bases de datos. La mayor innovación es que este está escrito en AIML (Artificial Intelligence Markup Language), una extensión de XML que todavía hoy está en uso. A.L.I.C.E (Richard Wallace,1995).

Este último fue el ganador del premio en Premio Loebner de Inteligencia Artificial creado en 1990, este premio tiene como objetivo destacar trabajos que alcanzaron la primera instancia de un test

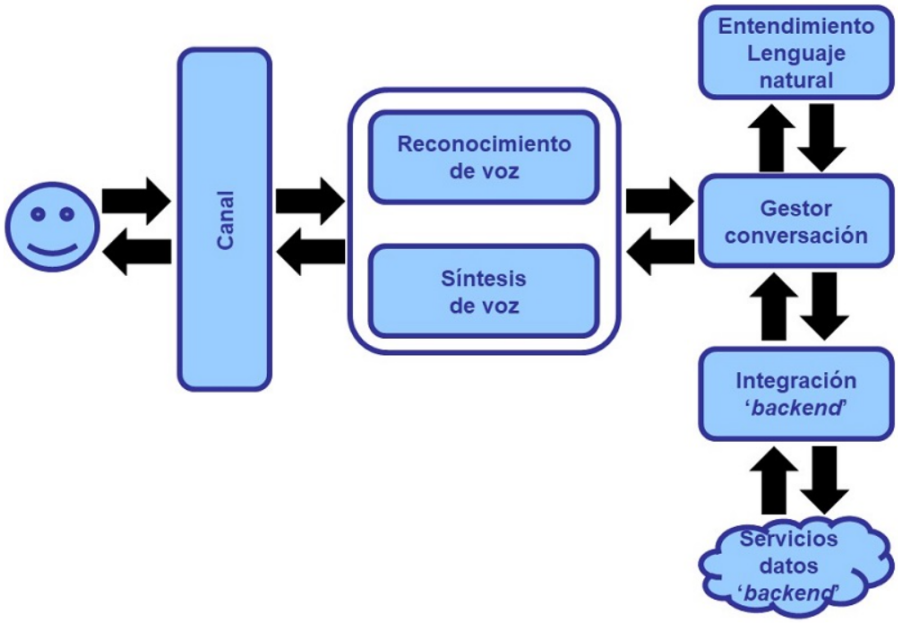
de Turing. Por ello se ofreció un premio de 100.000 dólares para el primer chatbot cuyas respuestas fueran indistinguibles de un ser humano. Los primeros años los premios fueron ganados por chatbots que implementan el funcionamiento básico de ELIZA. Actualmente el premio se sigue otorgando anualmente y destaca los mejores desarrollos de chatbots.

Su Arquitectura

La arquitectura descrita a continuación hace referencia a un chatbot que se comunica por medio de voz (Janarthanam, 2021).

- El canal: Es el medio que utiliza el chatbot para comunicarse con el usuario final. Canales habituales son sistemas de mensajería como Facebook Messenger, Skype o Slack, aplicaciones web, Twitter, SMSs, etc.
- Módulo de voz: Este se encuentra compuesto de dos sub-módulos. El módulo de reconocimiento de voz es necesario para recibir el mensaje del usuario y el módulo de síntesis de voz para que el chatbot pueda 'hablar' con el usuario.
- Módulo de Gestión de la conversación: Este es el módulo o elemento central ya que este decide el flujo de la conversación, la personalidad, el estilo y lo que el chatbot es en el fondo capaz de ofrecer.
- Módulo para el entendimiento del lenguaje natural (NLU, Natural Language Understanding): Es decir es donde se procesa lo que el usuario ha querido decir y se determina el sentido y objetivo del mensaje o pregunta.
- Módulo de integración con el backend: Este se encarga de interaccionar con las aplicaciones, bases de datos o sistemas utilizando las APIs que éstos ofrecen para ponerse a disposición del chatbot.

Figura 1. Arquitectura básica de una interfaz conversacional (Srini Janarthanam, 2021)



Características de los chatbots

Los chatbots pueden presentar diferentes funcionalidades, objetivos, complejidades y arquitecturas, pero en su mayoría comparten las siguientes caracteristicas (Juan Domingo Benate Mendoza, 2020):

- Autonomía: La mayoría de chatbots son capaces de adaptarse a los cambios y basados en estos pueden tener diferentes respuestas dependiendo del usuario, la pregunta realizada, el momento, las posibilidades más adecuadas a la posible respuesta, etc.
- Veracidad: El origen de la información que el chatbot utilizara en su análisis y en sus resultados deben ser de origen veraz, ya que el objetivo es brindar posibles respuestas adecuadas y no basadas en información u orígenes de información falsa o de dudosa procedencia.
- Racionalidad: Si el resultado obtenido del chatbot como resultado de la consulta es adecuado al momento, entorno, marco de referencia, se considera que este chatbot es racional en su interpretación.

- Reactividad: Si el entorno y sus cambios afectan el comportamiento de las respuestas del chatbot, este posee un comportamiento reactivo ya que no se limita solo a dar frases como respuestas, puede proporcionar imágenes, enlaces o distintos recursos como respuesta.
- Pro-actividad: Si el chatbot entiende los cambios en el entorno y depende en la situación que se encuentre es capaz de realizar el mismo las preguntas se considera pro-activo.

Las características antes mencionadas ayudan a determinar como un chatbot se comporta de acuerdo a la interacción humana, en el pasado como se mencionó anteriormente el TEST de Turing era muy utilizado como medida de evaluación ya que en dicho test un juez tendrá una conversación con un humano y un ordenador y deberá distinguir cuál es la persona y cuál es el ordenador, en caso que sean indistinguibles para él, la prueba será satisfactoria para la máquina y se puede decir que es inteligente.

En 1980, John Searle propuso un experimento para poner a prueba la veracidad del resultado del TEST de Turing, ya que el experimento se realizó en una habitación china, donde argumentaba que el experimento de Turing no podía usarse para determinar si una máquina es inteligente o no. Esto debido a que en esta prueba el resultado de este test sería si al chatbot es capaz de responder en chino, sin en efecto entender el significado de las palabras, es decir el programa permite a la persona en el espacio para pasar el Test de Turing para la comprensión de chino, pero él no entiende una palabra de chino." (Cole, 2004). Todo esto debido a que no se puede considerar como pensante si en realidad no comprende el significado de las respuestas que este proporcione y aun así pase en TEST de Turing.

Según (Walker et al., 1997) PARADISE (PARAdigm for Dialogue System Evaluation) es muy utilizado en la evaluación de chatbots ya que, es un framework que admite comparaciones entre estrategias de diálogo proporcionando una representación de tareas que desvincula lo que un agente necesita lograr en términos de los requisitos de la tarea de cómo el agente lleva a cabo la tarea a través del diálogo.

PARADISE utiliza una teoría de decisión marco para especificar la contribución relativa de varios factores para el desempeño general de un agente. El rendimiento se modela como una función ponderada de una medida de éxito basada en tareas y medidas de costo basadas en diálogo, donde los pesos se calculan correlacionando la satisfacción del usuario con el rendimiento, Además, el rendimiento se puede calcular tanto para los sub-diálogos como para diálogos completos.

¿Qué es el lenguaje natural?

El lenguaje natural no es más que la forma en que los seres humanos se comunican entre sí, esta comunicación puede ser mediante la voz o mediante el texto. Ambas formas se encuentran presentes en el día a día a través de señales, menús, correo electrónico, mensajes de texto, páginas web, etc.

Hoy en día la interacción que el ser humano tiene con los distintos dispositivos electrónicos a través de sus distintas interfaces representa la principal forma de interacción y de recopilación de datos, es por ello que fue necesario crear distintas formas de interpretar estos datos para poder utilizar esta información para que los chatbot fueran capaces de "entender" al ser humano.

El verdadero reto de la interpretación del lenguaje natural es que esta comunicación puede ser considerada desordenada, "Es difícil desde el punto de vista de un niño, que debe pasar muchos años adquiriendo un idioma... Es difícil para el adulto que aprende idiomas, es difícil para el científico que intenta modelar los fenómenos relevantes y es difícil para el ingeniero que intenta construir sistemas que se ocupen de la entrada o salida del lenguaje natural. Estas tareas son tan difíciles que Turing bien podría hacer de una conversación fluida en lenguaje natural la pieza central de su prueba de inteligencia." (Wladimir Serrano Gómez, 2005).

Por lo cual el lenguaje natural es muy ambiguo y continuamente está evolucionando, las personas son capaces de comprender el lenguaje, interpretar y expresar sus ideas en función del mensaje recibido de una persona, esto no significa que representan de manera adecuada las reglas del lenguaje y de la lingüística, es por ello que la interpretación de este puede volverse aún más complicado de comprender por los chatbots.

Lingüística computacional y el lenguaje natural

La lingüística es la ciencia que estudia el lenguaje, incluyendo su gramática, semántica y fonética. La lingüística clásica implicó idear y evaluar reglas del lenguaje además de métodos formales para la sintaxis y la semántica. Los matemáticos que trabajan con el lenguaje natural pueden referirse a su estudio como lingüística matemática, centrándose exclusivamente en el uso de formalismos matemáticos discretos y teoría del lenguaje natural, por ejemplo, lenguajes formales y teoría de autómatas (Lluís Payrató, 2003).

La lingüística computacional es el estudio moderno de la lingüística utilizando las herramientas de la informática, esto debido a que gracias a los diferentes medios digitales en la actualidad se cuenta con una gran cantidad de datos y a esto se le suman las computadoras rápidas que se pueden descubrir cosas nuevas y diferentes a partir de grandes conjuntos de datos de texto al escribir y ejecutar software. Los métodos estadísticos y el aprendizaje automático estadístico comenzaron a reemplazar los enfoques clásicos de arriba hacia abajo basados en reglas, principalmente debido a sus mejores resultados, velocidad y solidez (Lluís Payrató, 2003).

CAPÍTULO 3

3. CONCEPTUALIZACIÓN DE CHATBOTS PARA LA IMPLEMENTACIÓN DE CONVERSACIONES AUTOMATIZADAS

3.1 ¿CÓMO LOS CHATBOTS SON CAPACES DE INTERPRETAR EL LENGUAJE NATURAL?

Cuando se habla de chatbots es importante mencionar aquellas tecnologías que son utilizadas para interpretar el lenguaje natural, como la inteligencia artificial, el aprendizaje automático (machine learning) y el procesamiento del lenguaje natural (NPL).

Figura 2. Mapa Mental de cómo chatbots son capaces de interpretar el lenguaje natural

¿Qué es el procesamiento del lenguaje natural (NLP)?

El procesamiento del lenguaje natural (NLP) se refiere a la rama de la informática y más específicamente, la rama de la inteligencia artificial o IA, que se ocupa de brindar a las computadoras la capacidad de comprender textos y palabras habladas de la misma manera que los seres humanos, esto puede variar desde simples algoritmos de coincidencia de patrones de caracteres que usan expresiones regulares hasta sistemas inteligentes sofisticados que usan redes neuronales artificiales para traducir entre diferentes idiomas, algunos trabajos se encuentran entre lingüística y ciencias de la computación investigando métodos computacionales para modelar lenguajes y abordar cuestiones lingüísticas (Zhang & Teng, 2021).

NLP inició en 1950, como una parte central de la AI, algunos métodos seminales se basaron en gran medida en un conjunto de reglas lingüísticas (rule-based methods). Siendo la relación con las máquinas una de las tareas que captó mayor atención, pero con el paso del tiempo se empezó a entender que el desarrollar una serie de reglas para el procesamiento del lenguaje natural en general, era una tarea muy compleja siendo la razón principal la ambigüedad en los lenguajes (Zhang & Teng, 2021).

En 1980 los métodos estadísticos y el machine learning gradualmente fueron reemplazando a los métodos rule-based, la idea fue que los algoritmos aprendieran la distribución estadística del patrón lingüístico a través de los datos y usar estos para tomar decisiones. Desde finales del 2000 deep learning superó los métodos estadísticos tradicionales como el enfoque dominante, la idea consistió en entrenar redes neuronales artificiales de múltiples capas apiladas que tienen el poder de aprender arbitrariamente funciones complejas, empíricamente deep learning ha mostrado mejores resultados en comparación con los métodos estadísticos tradicionales para una amplia gama de tareas de procesamiento del lenguaje natural (Zhang & Teng, 2021).

La combinación de la lingüística computacional (rule-based methods) con modelos estadísticos, machine learning y deep learning que ofrece NLP permiten a las computadoras procesar el lenguaje humano en forma de texto o datos de voz y comprender su significado completo.

El lenguaje humano está lleno de ambigüedades que hacen que sea increíblemente difícil escribir software que determine con precisión el significado previsto del texto o los datos de voz, Por ejemplo Homónimos, homófonos, sarcasmo, modismos, metáforas, excepciones gramaticales y de uso, variaciones en la estructura de las oraciones y más, son solo algunas de las irregularidades del lenguaje humano que los humanos tardan años en aprender, pero que los programadores deben enseñar a las aplicaciones a reconocer y entender con precisión desde el principio (IBM Cloud Education, 2020).

Según Figueroa Sacoto (2021) para poder procesar el lenguaje natural las computadoras deben de hacer un preprocesamiento de los datos ingresados, este consiste en la limpieza y preparación de los datos ingresados según el tipo de dato y el formato que requiera el algoritmo para su correcta ejecución para luego desarrollar algoritmos o implementar algoritmos existentes que ayuden a completar el procesamiento del lenguaje natural.

Así mismo existen varias tareas en el proceso de NLP que desglosan el texto humano y los datos de voz de manera que ayuden a la computadora a dar sentido a lo que está ingiriendo. (IBM Cloud Education, 2020), Entre estas tareas están:

- Reconocimiento de voz (Speech Recognition): Es la tarea que consiste en convertir de manera confiable los datos de voz en datos de texto.
- Etiquetado gramatical (Grammatical Tagging): Es el proceso de determinar la parte del discurso de una palabra o fragmento de texto en particular en función de su uso y contexto.
- Desambiguación del sentido de la palabra (Word Sense disambiguation): Es la selección del significado de una palabra con múltiples significados a través de un proceso de análisis semántico que determina la palabra que tiene más sentido en el contexto dado.
- Reconocimiento de la entidad nombrada (Named entity recognition): Es la identificación de palabras o frases como entidades útiles.
- Resolución de correferencia (Coreference resolution): Es la tarea de identificar cuando existen dos palabras que se refieren a la misma entidad. El ejemplo más común es determinar la persona o el objeto al que se refiere un determinado pronombre (p. ej., 'ella' = 'María')

- Análisis de los sentimientos (Sentiment analysis): Es el intento de extraer cualidades subjetivas del texto (actitudes, emociones, sarcasmo, confusión, sospecha).
- Generación de lenguaje natural (Natural language generation): Es la tarea de poner información estructurada en lenguaje humano.
- En el caso de los chatbots NLP se puede utilizar para:
- Realizar resúmenes automáticos acortando inteligentemente los textos largos.
- Ayudar a evaluar la intención de la entrada de los usuarios para luego crear respuestas basadas en un análisis contextual similar al de un ser humano.
- Hacer sugerencias automáticas que pueden ser utilizadas para acelerar la escritura de mensajes y textos.
- Reconocimiento de intenciones, las entradas del usuario a través de un chatbot se dividen y compilan en una intención del usuario (intent) a través de pocas palabras, el NLP ayuda al análisis de las oraciones completas a través del desglose de la oracion o parrafo en uno más simple.
- Realizar relaciones semánticas entre entidades (Entity), los chatbots son capaces de identificar palabras ingresadas por los usuarios y asociarlas o relacionarlas con las entidades disponibles para completar la tarea solicitada. Las entidades pueden ser datos o palabras relacionadas a tiempo, fecha, lugares, ubicación, descripciones, sinónimos.
- Reconocimiento del habla.
- Segmentación temática.

¿Qué es la inteligencia artificial (ai)?

Las investigaciones iniciales sobre inteligencia artificial inician en la década de 1950 orientadas más a una inteligencia artificial simbólica, la cual aprende a entender el mundo formando representaciones simbólicas internas. Los símbolos son importantes para el pensamiento humano y el proceso de razonamiento ya que a partir de ellos se aprenden objetos y conceptos donde se crean reglas para identificarlos y manejarlos. La IA simbólica trata de imitar este mecanismo a través de un sistema imaginario que crea y modifica combinaciones de símbolos binarios representados por “0” y “1” (García Mendiola, 2008).

El término de inteligencia artificial fue adoptado en 1956 por John McCarthy, el cual expone que la AI necesita muchas ideas para que un robot tenga la capacidad de aprender de sus experiencias como lo hacen los humanos (García Mendiola, 2008). En el siglo XXI la Inteligencia artificial se puede definir como la habilidad que tienen los sistemas o máquinas para reproducir la inteligencia humana (como entendimiento, aprendizaje, razonamiento, lenguaje o resolución de problemas) para realizar tareas que pueden mejorar iterativamente a partir de la información que se recopila.

La AI se puede realizar a través de la combinación de algoritmos planteados con el propósito de crear máquinas que presenten la misma capacidad que el ser humano, según Stuart Russel y Peter Norving existen varios tipos de AI (Russell & Norvig, 1995):

- Sistemas que actúan como humanos (Acting Humanly): Son las máquinas que realizan tareas de forma similar a como las harían las personas (robots, el enfoque de la prueba de Turing).
- Sistemas que piensan como humanos (Thinking Humanly): Son las máquinas que pueden automatizar actividades como la toma de decisiones, la resolución de problemas y el aprendizaje (redes neuronales artificiales, el enfoque de modelado cognitivo).
- Sistemas con pensamiento racional (Thinking rationally): Son las máquinas que intentan imitar el pensamiento lógico de los humanos, se busca e investiga que estas máquinas puedan percibir, razonar y actuar en consecuencia a los eventos (sistemas expertos, el enfoque de las leyes del pensamiento).
- Sistemas que actúan racionalmente (Acting rationally): Son las máquinas que tratan de imitar de manera racional el comportamiento de los humanos (agentes inteligentes, el enfoque del agente racional).

En la actualidad la AI ayuda a mejorar el rendimiento y la productividad en las empresas mediante la automatización de procesos y tareas que antes eran realizadas por humanos. Por ejemplo, netflix utiliza machine learning en varias áreas de su empresa, aunque la personalización de las cuentas es el área en la que más se conoce, pero también la utilizan para dar forma a sus catálogos de películas y programas a través del aprendizaje de las características que hacen el contenido un éxito (Netflix Research, 2019).

Según un estudio hecho por IBM en el 2018 de 3000 organizaciones entrevistadas 4 de cada 10 utilizan funciones basadas en inteligencia artificial. En el caso de los chatbots la IA se puede utilizar para:

- Ayudar a la automatizar la interacción con sus clientes
- Ayudar a obtener información más precisa de los problemas de los clientes y ayudar a proporcionar respuestas más eficientes.
- Mediante los aspectos de búsqueda utilizados previamente por el usuario es posible determinar cuáles son sus intereses.
- Mediante aspectos de búsqueda colaborativa se puede determinar posibles sugerencias de información gracias a la colaboración de otros usuarios que han realizado consultas similares.
- Puede ayudar al aprendizaje de palabras o frases las cuales pueden ser relacionadas con consultas similares previas para así ir alimentando su base de conocimiento.

¿Qué es el aprendizaje automático (Machine Learning)?

Para poder entender sobre machine learning es importante saber la idea detrás del aprendizaje, esta idea consiste en que las percepciones deben usarse no solo para actuar, sino también para mejorar la capacidad del agente para actuar en el futuro. El aprendizaje se produce como resultado de la interacción entre el agente y el mundo (Russell & Norvig, 1995).

El Machine learning es una rama de la AI y las ciencias de la computación que se enfoca en el uso de datos y algoritmos para imitar el aprendizaje de los humanos, mejorando gradualmente su precisión. Esto permite que las aplicaciones de software sean más precisas en la predicción de resultados sin estar programadas explícitamente para hacerlo (Hurwitz & Kirsch, 2018).

Los algoritmos de aprendizaje automático utilizan datos históricos como entrada para predecir nuevos valores de salida. Se utilizan una variedad de algoritmos que iterativamente aprenden de los datos para mejorar, para describir otros datos y para predecir resultados. A medida que los algoritmos incorporan datos de entrenamiento, es posible producir modelos de aprendizaje más

precisos basados en esos datos. Un modelo es el resultado generado cuando se entrena un algoritmo y es capaz de dar resultados inteligentes al darle una entrada (Hurwitz & Kirsch, 2018).

Estos modelos de aprendizaje son entrenados en conjuntos de datos antes de ser desplegados. Algunos modelos están en línea (online models) y se adaptan continuamente a medida que se incorporan nuevos datos. Por otro lado, otros modelos están fuera de línea (offline models) y se derivan de algoritmos de aprendizaje automático pero una vez implementados no cambian (Hurwitz & Kirsch, 2018).

Este proceso de aprendizaje iterativo de los modelos en línea conduce a una mejora en los tipos de asociaciones que se hacen entre los elementos y los datos y que debido a su complejidad y tamaño podrían haber sido fácilmente pasado por alto por la observación humana. Después de haber sido entrenados estos modelos pueden ser utilizados en tiempo real para aprender de los datos en tiempo real (Hurwitz & Kirsch, 2018).

Muchos modelos de aprendizaje automático se definen por la presencia o ausencia de influencia humana en los datos sin procesar, ya sea que se ofrezca una recompensa, se proporcione retroalimentación específica o se utilicen etiquetas.

Según Tamir (2020) los sistemas de aprendizaje de un algoritmo de machine learning se dividen en tres partes principales:

- Un proceso de decisión (A decision process): Una receta de cálculos u otros pasos que toma los datos y devuelve una "suposición" sobre el tipo de patrón que su algoritmo está buscando. Por ejemplo, para hacer una predicción o clasificación en función de algunos datos de entrada, que se pueden etiquetar o no, el algoritmo producirá una estimación sobre un patrón en los datos.
- Una función de error (An error functions): Un método para medir qué tan buena fue la conjetura comparándola con ejemplos conocidos (cuando están disponibles). Esto significa que se debe lograr identificar si ¿El proceso de decisión lo hizo bien? Si no es así, ¿cómo cuantificar “qué tan grave” fue el fallo? Es decir, esta función sirve para evaluar la predicción del modelo si

existiera un ejemplo conocido esta función puede hacer una comparación para evaluar la precisión del modelo.

- Un proceso de actualización y optimización: Donde el algoritmo analiza el fallo y luego actualiza cómo el proceso de decisión llega a la decisión final para que la próxima vez el fallo no sea tan grande. Es decir que, si el modelo puede ajustarse mejor a ciertos puntos de los datos en el conjunto de entrenamiento, entonces los pesos se ajustan para reducir la discrepancia entre el ejemplo conocido y la estimación del modelo. El algoritmo repetirá este proceso de evaluación y optimización, actualizando los pesos de forma autónoma hasta alcanzar un umbral de precisión.

Dado que un algoritmo de machine learning se actualiza de forma autónoma, la precisión analítica mejora con cada ejecución, ya que aprende de sí mismo a partir de los datos que analiza. Esta naturaleza iterativa del aprendizaje es única y valiosa porque ocurre sin intervención humana, lo que brinda la capacidad de descubrir conocimientos ocultos sin estar específicamente programado para hacerlo.

Tomando como base lo anterior existen diferentes tipos de modelos de machine learning los cuales se dividen en 4 fundamentales (según la intervencion humano):

- Aprendizaje supervisado (Supervised learning): En este tipo el conjunto de datos que se utiliza ha sido pre etiquetado y clasificado por los usuarios para permitir que el algoritmo vea qué tan preciso es su desempeño. Esto significa que cada ejemplo en el conjunto de datos de entrenamiento está etiquetado con la respuesta que el algoritmo debe generar por sí solo. Por ejemplo, en un conjunto de datos etiquetados sobre imágenes de flores se le diría al modelo qué fotos son de rosas, margaritas y narcisos. Cuando se muestra una nueva imagen, el modelo la compara con los ejemplos de entrenamiento para predecir la etiqueta correcta (Salian, 2018) (Tamir, 2020).
- Aprendizaje no supervisado (unsupervised learning): En este tipo el conjunto de datos que se utiliza está sin procesar y no está etiquetado y un algoritmo identifica patrones y relaciones dentro de los datos sin la ayuda de los usuarios. Estos extraen automáticamente características y encuentran patrones en los datos. Debido a que no hay un elemento de "verdad fundamental"

en los datos, es difícil medir la precisión de un algoritmo entrenado con aprendizaje no supervisado. Pero hay muchas áreas de investigación donde los datos etiquetados son difíciles de alcanzar o demasiado costosos de obtener y es en estos casos donde dar rienda suelta a este modelo para encontrar patrones propios puede producir resultados de alta calidad (Salian, 2018) (Tamir, 2020).

- Aprendizaje semi supervisado (Semi Supervised learning): En este tipo el conjunto de datos contiene datos estructurados y no estructurados que guían al algoritmo en su camino hacia conclusiones independientes. La combinación de los dos tipos de datos en un conjunto de datos de entrenamiento permite que los algoritmos de machine learning aprendan a etiquetar datos sin etiqueta. Este método es particularmente útil cuando es difícil extraer características relevantes de los datos y el etiquetado de ejemplos es una tarea que requiere mucho tiempo para los expertos (Salian, 2018) (Tamir, 2020).
- Aprendizaje reforzado (Reinforcement learning): En este tipo el conjunto de datos utiliza un sistema de "recompensas/castigos", que ofrece retroalimentación al algoritmo para aprender de sus propias experiencias por ensayo y error. Los agentes de IA intentan encontrar la forma óptima de lograr un objetivo particular o mejorar el rendimiento en una tarea específica, a medida que el agente realiza una acción que va hacia la meta, recibe una recompensa, el objetivo general es predecir el mejor paso a seguir para ganar la mayor recompensa final. Para tomar sus decisiones, el agente se basa tanto en los aprendizajes de las retroalimentaciones anteriores como en la exploración de nuevas tácticas que pueden presentar una mayor recompensa, esto implica una estrategia a largo plazo. Este es un proceso iterativo, cuantas más rondas de retroalimentación se tenga se vuelve mejor la estrategia del agente (Salian, 2018) (Tamir, 2020).

Con base en la explicación anterior se podría decir que la aplicación de machine learning para el NLP en chatbots ayuda principalmente en:

- Ayuda a los bots a entrenarse y adquirir mayor conocimiento conforme se van dando las interacciones con los usuarios. Cuantos más datos reciben, más optimizado es su rendimiento. Entonces, a medida que pasa el tiempo, la “inteligencia” del bot aumenta.

- Ayuda para poder ofrecer una respuesta informativa según el contexto. Simplifica y maneja correctamente el análisis de grandes cantidades de datos, encontrando y brindándole la respuesta correcta.
- Mantiene el contexto del diálogo, ayuda a recopilar y analizar datos rápidamente mientras interactúa constantemente con los usuarios.
- Ayuda a que la interacción sea más fluida y así sea imperceptible para los humanos.

3.2 ALGORITMOS UTILIZADOS PARA INTERPRETACIÓN DE LENGUAJE NATURAL

Estos algoritmos están relacionados con algunas de las tareas que el procesamiento del lenguaje natural realiza como, por ejemplo: Segmentación, revisión gramatical y ortográfica, clasificación de texto, reconocimiento de entidades nombradas (NER), sumarización o interpretación del texto, generación de texto, modelado de temas.

El procesamiento del lenguaje natural (NLP) aplica el machine learning y otras técnicas al lenguaje. Estas suelen funcionar en matrices numéricas denominadas vectores que representan cada instancia (o fila) en el conjunto de datos. A la colección de todos estos arreglos se le llama matriz, cada fila en la matriz representa una instancia. Mirando la matriz por sus columnas, cada columna representa una característica (o atributo). Esto se trata de datos tabulares, a los que la mayoría de ingenieros de software ya han estado expuestos en forma de hojas de cálculo y bases de datos relacionales.

El primer problema que tienen que resolver estos algoritmos es convertir la colección de instancias de texto en una forma de matriz donde cada fila es una representación numérica de una instancia de texto es decir un vector. En otras palabras, se necesita desglosar los textos y las palabras de una manera que la máquina pueda entender (Krueger, 2022).

En NLP una sola instancia se denomina documento, mientras que una colección de instancias se denomina corpus. Dependiendo del problema en cuestión, un documento puede ser tan simple como una frase corta o un nombre o tan complejo como un libro. Luego se tiene que elegir cómo descomponer los documentos en partes más pequeñas, un proceso denominado tokenización este

proceso produce tokens. Los tokens son las unidades de significado que el algoritmo puede considerar. El conjunto de todos los tokens vistos en todo el corpus se llama vocabulario (Krueger, 2022).

El procesamiento del lenguaje natural realizado por las máquinas sigue un flujo de 7 pasos fundamentales:

- Ingreso del texto
- Segmentación
- Limpieza del texto
- Vectorización e ingeniería de características
- Lematización y derivación
- Aplicación de algoritmos y métodos de Machine learning para entrenar a los modelos
- Interpretación de los resultados.

Y para realizar estas tareas existen algoritmos que pueden ser utilizados y que se explicarán y describirán brevemente a continuación:

Tokenización

La Tokenización es el mecanismo por el cual el texto se segmenta en oraciones y frases. El trabajo principal es dividir un texto en partes más pequeñas (llamadas tokens) mientras se descartan ciertos caracteres, como la puntuación. Esta es una tarea común en el procesamiento del lenguaje natural y es un paso fundamental tanto en métodos tradicionales como en métodos más avanzados como el Deep Learning. La tokenización se puede ejecutar a nivel de palabra, carácter o subpalabra (Pai, 2021).

El algoritmo de tokenización de palabras es el utilizado más comúnmente, este consiste en separar un pedazo del texto en palabras individuales utilizando algún tipo de delimitador y los tokens son formados dependiendo de este delimitador. El algoritmo de tokenización de caracteres consiste en separar un texto en un set de caracteres.

El algoritmo de tokenización por subpalabra consiste en separar un pedazo de texto en subpalabras, es una solución entre la tokenización basada en palabras y caracteres. La idea principal es resolver los problemas que enfrenta la tokenización basada en palabras (tamaño de vocabulario muy grande, gran cantidad de tokens OOV (out of vocabulary) y diferentes significados de palabras muy similares) y la tokenización basada en caracteres (secuencias muy largas y tokens individuales menos significativos). Por ejemplo, "niño" no debe dividirse, pero "niños" debe dividirse en "niño" y "s". Esto ayudará al modelo a aprender que la palabra “niños” se forma usando la palabra “niño” con significados ligeramente diferentes, pero con la misma raíz.

Algoritmo BPE

Este algoritmo es una instancia de los algoritmos de compresión basados en macros, busca redundancias en el texto mediante la detección de patrones repetidos y comprime las secuencias reemplazando una ocurrencia de dicho patrón con punteros a una ocurrencia anterior. Es una versión simple del método de sustitución de patrones. Utiliza los códigos de caracteres que no aparecen en el texto para representar cadenas que aparecen con frecuencia.

En el algoritmo se crea un vocabulario de tokens y una tabla de combinación, el vocabulario de tokens se inicializa con el vocabulario de caracteres y la tabla de combinación se inicializa con una tabla vacía. En primer lugar, cada palabra se representa como una secuencia de fichas más un símbolo especial de fin de palabra. Luego, el método cuenta iterativamente todos los pares de tokens y fusiona el par más frecuente en un nuevo token (Shibata et al., 1999) (Gallé, 2019) (Provilkov et al., 2020).

Este token se agrega al vocabulario y la operación de combinación se agrega a la tabla de combinación. Esto se hace hasta que se alcanza el tamaño de vocabulario deseado. La tabla de fusión resultante, especifica qué subpalabras deben fusionarse en una subpalabra más grande. En primer lugar, una palabra se divide en distintos caracteres más el símbolo del final de la palabra. Luego, se fusiona el par de tokens adyacentes que tiene la prioridad más alta. Esto se hace de forma iterativa hasta que no haya ninguna combinación disponible en la tabla (Shibata et al., 1999) (Gallé, 2019) (Provilkov et al., 2020).

Por ejemplo, supongamos que tenemos los datos aaabdaaabac que deben codificarse (comprimirse). El par de bytes aa ocurre con mayor frecuencia, por lo que se reemplaza con Z ya que Z no aparece en los datos. Así que ahora se tiene ZabdZabac donde Z = aa. El siguiente par de bytes común es ab, así que se reemplaza con Y. Ahora se tiene ZYdZYac donde Z = aa e Y = ab. El único par de bytes que queda es ac, que aparece como uno solo, por lo que no se codifica.

Se puede utilizar la codificación recursiva de pares de bytes para codificar ZY como X. los datos ahora se han transformado en XdXac donde X = ZY, Y = ab y Z = aa. No se puede comprimir más ya que no hay pares de bytes que aparezcan más de una vez. Se descomprimen los datos realizando reemplazos en orden inverso.

BPE garantiza que las palabras más comunes se presentan en el vocabulario como un solo token, mientras que las palabras raras se dividen en dos o más tokens de subpalabras y esto concuerda con lo que hace un algoritmo de tokenización basado en subpalabras.

Para NLP los tokens son los componentes básicos, es así que la forma más común de procesar el texto sin formato ocurre a el nivel del token. En el caso de los chatbots la tokenización podría utilizarse para el procesamiento de grandes cantidades de entradas de texto, ayudando a dividir las entradas en palabras más sencillas que pueden ser procesadas más rápido y fácil por la máquina y utilizar estas palabras como una relación para ejecutar el intento con el que mejor pueda ser asociada y general una respuesta.

Algoritmo TF-IDF (Término de frecuencia inversa de la frecuencia de un documento)

La TF-IDF es una medida estadística que evalúa qué tan relevante es una palabra para un documento en una colección de documentos. Esto se lleva a cabo multiplicando dos métricas:

- Contando las veces que aparece una palabra en un documento
- Calculando la frecuencia inversa del documento de la palabra en un conjunto de documentos.

Dado un corpus de documento de texto "N" genera un vector de ponderaciones para cada documento en el corpus (tf-udf). Cada vector consta de "m" enteros, donde m es el número de palabras en el diccionario y cada entrada corresponde a una palabra en particular del diccionario. El valor de cada entrada se denomina ponderación (tf-odf) y es el producto de la frecuencia del término y una frecuencia inversa del documento.

La frecuencia del término es proporcional a la frecuencia de la palabra en el documento y la frecuencia inversa del documento es inversamente proporcional a la frecuencia de la palabra en el corpus. La idea detrás de la frecuencia inversa del documento es compensar el hecho de que algunas palabras comunes como artículos o preposiciones pueden aparecer muy a menudo en un artículo, pero no contribuyen mucho a la semántica del mismo (Ghoche, 2016).

La fórmula de TF-IDF está representado por:

$$w(t,d) = tf(t,d) * log\left(\frac{N}{Nt}\right)$$

Donde:

- tf(t,d) representa la frecuencia de aparición de la palabra clave t en el texto d.
- N representa el número de texto completo.
- Nt representa el número de textos que tienen la palabra t.

La parte principal de la fórmula es la frecuencia de término (TF) y la frecuencia inversa de documento (IDF). Aunque la fórmula considera la frecuencia de palabras y la frecuencia de documentos como dos factores al mismo tiempo (Wang & Tang, 2016).

La frecuencia de término (TF): Es la frecuencia de término de una palabra en un documento. Se utiliza para medir cuántas veces un término está presente en un documento. Por ejemplo, supongamos que tenemos un documento "T1" que contiene 5000 palabras y la palabra "Alpha" está presente en el documento exactamente 10 veces (Qaiser & Ali, 2018).

Es un hecho que la longitud total de los documentos puede variar desde muy pequeña hasta grande, por lo que existe la posibilidad de que cualquier término pueda aparecer con mayor frecuencia en un documento grande en comparación con un documento pequeño. Entonces, para corregir este problema, la ocurrencia de cualquier término en un documento se divide por el total de términos presentes en ese documento (Qaiser & Ali, 2018). Entonces, en este caso la frecuencia de término de la palabra "Alpha" en el documento "T1" será:

$$TF = 10/5000 = 0.002.$$

La frecuencia inversa de documento (IDF): Es la frecuencia inversa de la palabra en un conjunto de documentos, cuando se calcula la frecuencia de término de un documento, se puede observar que el algoritmo trata todas las palabras clave por igual, no importa si es una palabra vacía lo cual es incorrecto ya que todas las palabras clave tienen una importancia diferente (Qaiser & Ali, 2018).

Por ejemplo, si se tiene la palabra vacía "de" presente en un documento 2000 veces, pero esta no sirve o tiene un significado inferior. La frecuencia inversa del documento asignara menor peso a las palabras frecuentes y asigna mayor peso a las palabras poco frecuentes. Por ejemplo, si tenemos 10 documentos y el término "tecnología" está presente en 5 de esos documentos, la frecuencia inversa de documentos será:

$$IDF = log\_e\,(10/5) = 0.3010$$

Entonces, si la palabra es muy común y aparece en muchos documentos, este número se acercará a 0. De lo contrario, se acercará a 1.

La multiplicación de estos dos números (TF y IDF) da como resultado la puntuación TF-IDF de una palabra en un documento. Cuanto mayor sea la puntuación, más relevante será esa palabra en ese documento en particular.

TF-IDF fue inventado para la búsqueda de documentos y la recuperación de información, se utiliza en el análisis de texto automatizado y es muy útil para calificar palabras en los algoritmos de aprendizaje automático para NLP.

Pero puede aplicar se a los chatbots ya que la mayoría de los chatbots dependen en gran medida de un motor de búsqueda y algunos chatbots dependen exclusivamente de él como su único algoritmo para generar respuestas. Por lo tanto, se debe dar un paso adicional para convertir el motor de búsqueda simple en un chatbot. Ya que se necesita almacenar los datos de entrenamiento en pares de preguntas (o sentencias) y respuestas. Entonces se puede utilizar TF-IDF para buscar una pregunta (o sentencia) lo más parecida al texto ingresado por el usuario y devolver la respuesta asociada con esa declaración.

Eliminación de palabras vacías (Stop-word removal)

El algoritmo de stop-word removal es un algoritmo que se utiliza para optimizar el pre procesamiento de los datos en machine learning. El concepto de stop-words o palabras vacías tiene una larga historia y se atribuye a Hans Peter Luhn en 1960 (Luhn, 1960).

Por ejemplos en inglés se tiene estas palabras vacias "a", "the", "of" y "didn't". Estas son palabras muy comunes y por lo general no agregan mucho al significado de un texto, sino que aseguran que la estructura de una oración sea sólida. Según Hvitfeldt & Silge (2021) Una de las principales razones para eliminar las palabras vacías fue el disminuir el tiempo de cómputo para la minería de texto ya que se puede considerar como una reducción de la dimensionalidad de los datos de texto y se usaba comúnmente en los motores de búsqueda para dar mejores resultados.

Las palabras vacías son palabras que aparecen con frecuencia en un lenguaje natural, estas son las palabras más comunes que se encuentran en cualquier lenguaje natural y que llevan muy poco o ningún significado en la oración ya que sólo tienen una importancia semántica para ayudar a la formación de la oración. Las palabras vacías son principalmente clasificadas en: conjunciones, preposiciones, adverbios y artículos del idioma. Debido a esto se consideran sin importancia en ciertas aplicaciones de procesamiento de lenguaje natural como Agrupación, Resumen de texto, Recuperación de información, etc. Casi todas las aplicaciones de pre procesamiento de texto eliminan las palabras vacías antes de procesar documentos y consultas aumentando el rendimiento del sistema (Jaideepsinh K., 2016).

Las palabras vacías pueden tener diferentes roles en un corpus. Por lo general, se clasifican en tres grupos: palabras vacías globales, de tema y de documento. Las palabras vacías globales son palabras que casi siempre tienen un significado en un idioma determinado. Por ejemplo, en inglés "of" y "and" son palabras que se necesitan para unir el texto. Las palabras vacías de tema son palabras no informativas para un área temática dada, los temas pueden representar dominios como por ejemplo medicina, finanzas, etc. Entre mayor conocimiento del dominio se tenga se puede crear una mejor lista de palabras vacías de tema. Las palabras vacías de documento no proporcionan ninguna o mucha información para un documento determinado. Estas son difíciles de clasificar y no valdrá la pena identificarlas (Hvitfeldt & Silge, 2021).

Para explicar cómo se aplica este algoritmo es importante destacar que no existe una lista universal de palabras vacías y nunca se debe ignorar la relevancia contextual. Es decir, las palabras más comunes en un contexto médico probablemente sean muy diferentes de los términos más comunes en el contexto de la ingeniería.

El algoritmo inicia con la creación de una lista de palabras vacías específicas de un lenguaje, dominio o contexto, es útil tener una colección central de palabras vacías para empezar. Para crear esta lista es importante seleccionar aquellas palabras que al ser eliminadas no afectan el significado general del mensaje. Por ejemplo, en el idioma inglés palabras como: the, of, to, in, a, that, etc. Son conectores que pueden ser eliminados independientemente de su dominio.

Según Hvitfeldt & Silge (2021) cuando se selecciona una lista de palabras vacías, es importante tener en cuenta su tamaño y amplitud. Tener una lista pequeña y concisa de palabras puede reducir moderadamente el recuento de tokens sin tener una influencia demasiado grande en los modelos, suponiendo que se elegirán las palabras adecuadas. A medida que crece el tamaño de la lista de palabras vacías, cada palabra agregada tendrá un efecto positivo decreciente con el riesgo creciente de que una palabra significativa se colocará en la lista por error.

Ahora que se tiene la lista de palabras vacías, se puede seguir adelante con la eliminación de estas palabras. La forma particular en que se eliminan las palabras vacías dependerá de la forma en que se tengan los datos y el lenguaje de programación que se utiliza.

En la actualidad existen librerías prefabricadas las cuales se pueden utilizar para llevar este procedimiento por ejemplo Python ofrece el NLKT (Natural Language Toolkit) el cual contiene una lista predefinida de palabras vacías en 16 diferentes idiomas.

Stop words removal se puede utilizar en un chatbot para hacer una limpieza del texto y remover todas aquellas palabras que no tienen un significado profundo esto ayuda al bot a identificar las palabras importantes y así mejorar el modelado predictivo para poder brindar respuestas con mejor precisión reduciendo el tiempo computacional de las entradas.

Derivación y Lematización (Stemming and Lemmatization)

La derivación y la lematización son técnicas de modelado de lenguaje para métodos de reducción de datos.

Derivación (Stemming)

La derivación es un método de normalización de palabras para el procesamiento del lenguaje natural. Es una técnica en la que un conjunto de palabras de una oración se convierte en una secuencia para acortar su búsqueda. En este método, se normalizan las palabras que tienen el mismo significado, pero tienen algunas variaciones según el contexto o la oración. Ya que existe una palabra raíz, pero existen muchas variaciones de las mismas palabras.

El proceso de derivación produce variantes de una palabra raíz/base, reduce una palabra base a su palabra raíz (Stem), El Stem no necesariamente tiene que ser una palabra válida por sí misma, este se utiliza para acortar la búsqueda y normalizar las oraciones para una mejor comprensión, comúnmente se utiliza para eliminar prefijos y sufijos de las palabras (Boban et al., 2020).

Según Divya (2021) existen muchos algoritmos diferentes para realizar derivación, pero uno de ellos utiliza la llamada "bolsa de palabras" que contiene palabras que son semánticamente idénticas o similares, pero se escriben con variantes morfológicas diferentes. Al aplicar algoritmos de derivación, las palabras se reducen a su raíz, lo que permite representarlas mediante las raíces de las palabras en lugar de las palabras originales. Por ejemplo, en el idioma inglés tenemos las palabras "fishing", "fisher", "fished", etc. las cuales se pueden reducir a la palabra "fish".

Por ejemplo, si se toman las oraciones en inglés: “He was waiting at the car” y “He waited for two hours”.

El significado es el mismo, es el verbo “wait” conjugado en tiempo pasado, es decir la actividad de esperar en el pasado. Un humano puede entender fácilmente que ambos significados son iguales. Pero para las máquinas, ambas oraciones son diferentes. Por lo tanto, es difícil convertirlo en la misma fila de datos. Y en este caso ya que no se proporciona el mismo conjunto de datos, la máquina no puede predecir. Por lo tanto, es necesario diferenciar el significado de cada palabra para preparar el conjunto de datos para el aprendizaje automático. Y es aquí donde la derivación se utiliza para categorizar el mismo tipo de datos obteniendo su palabra raíz (wait en este caso).

Porter Stemmer Algorithm

Es uno de los algoritmos de derivación más comunes que está diseñado básicamente para eliminar y reemplazar sufijos conocidos de palabras en inglés. Este es un proceso para eliminar las terminaciones morfológicas y flexivas más comunes de las palabras en inglés. Su uso principal es parte de un proceso de normalización de términos que generalmente se realiza al configurar el sistema de recuperación de información (Porter, 2006).

Se basa en la idea de que los sufijos en el idioma inglés se componen de una combinación de sufijos más pequeños y simples. Este stemmer es conocido por su velocidad y simplicidad. Las principales aplicaciones de Porter Stemmer incluyen la minería de datos y la recuperación de información. Sin embargo, sus aplicaciones solo se limitan a palabras en inglés. Además, el grupo de raíces se asigna a la misma raíz y la raíz de salida no es necesariamente una palabra significativa.

Las reglas para eliminar un sufijo se darán en la forma (condición) S1 => S2

Esto significa que, si una palabra termina con el sufijo S1 y la raíz anterior a S1 cumple la condición dada, S1 se reemplaza por S2. La condición generalmente se da en términos de m, donde m es una parte de la palabra.

Por ejemplo:
(m > 0) EED => EE, esto significa que si la palabra tiene al menos una vocal y consonante más la terminación EED entonces se cambia el EED por EE.

- "feed" -> "fee"
- "agreed" -> "agree"
- "greed" -> "gree"
- "treed" -> "tree"
- "tweed" -> "twee"
- "guaranteed" -> "guarantee"

Figura 3. Algoritmo de Porter Stemmer (Porter, 2006)

```
2. THE ALGORITHM

To present the suffix stripping algorithm in its entirety we will need a few
difinitions.

A \consonant\ in a word is a letter other than A, E, I, O or U, and other
than Y preceded by a consonant. (The fact that the term `consonant' is
defined to some extent in terms of itself does not make it ambiguous.) So in
TOY the consonants are T and Y, and in SYZYGY they are S, Z and G. If a
letter is not a consonant it is a \vowel\.

A consonant will be denoted by c, a vowel by v. A list ccc... of length
greater than 0 will be denoted by C, and a list vvv... of length greater
than 0 will be denoted by V. Any word, or part of a word, therefore has one
of the four forms:

    CVCV ... C
    CVCV ... V
    VCVC ... C
    VCVC ... V

These may all be represented by the single form

    [C]VCVC ... [V]

where the square brackets denote arbitrary presence of their contents.
Using (VC){m} to denote VC repeated m times, this may again be written as

    [C](VC){m}[V].

m will be called the \measure\ of any word or word part when represented in
this form. The case m = 0 covers the null word. Here are some examples:

    m=0    TR,  EE,  TREE,  Y,  BY.
    m=1    TROUBLE,  OATS,  TREES,  IVY.
    m=2    TROUBLES,  PRIVATE,  OATEN,  ORRERY.

The \rules\ for removing a suffix will be given in the form

    (condition) S1 -> S2

This means that if a word ends with the suffix S1, and the stem before S1
satisfies the given condition, S1 is replaced by S2. The condition is
usually given in terms of m, e.g.
```

Lematización (Lemmatization)

La lematización es similar a la derivación (Stemming), ya que ambos reducen una variante de una palabra a su "raíz" en la derivación y a su "lema" en la lematización. Pero la diferencia sustancial es que la lematización utiliza vocabulario y análisis morfológico para devolver las palabras a su forma de diccionario. Tanto la derivación como la lematización tienen un papel importante para aumentar las capacidades de recuerdo.

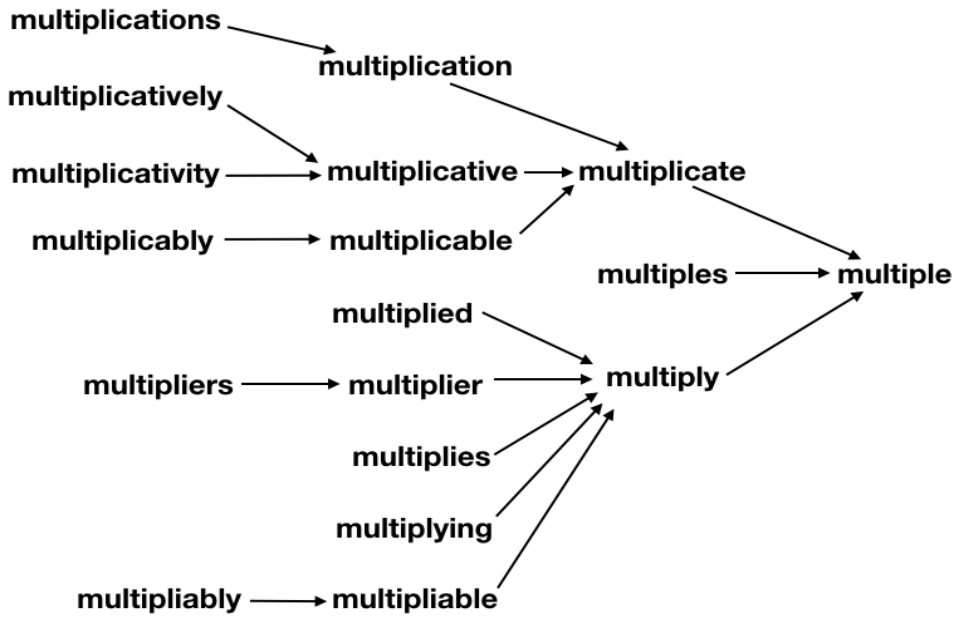
La lematización convierte cada palabra a su forma básica, el lema. En el idioma inglés, la lematización y la derivación a menudo producen los mismos resultados. A veces, la forma normalizada/básica de la palabra puede ser diferente a la raíz. Por ejemplo, para las palabras en inglés "computes", "computing", "computed" se derivan de "comput", pero el lema de esas palabras es "compute". (Boban et al., 2020).

En otras palabras, la lematización es el proceso de encontrar el lema de una palabra según su significado y contexto que generalmente se refiere al análisis morfológico de las palabras, cuyo objetivo es eliminar las terminaciones flexivas. Ayudando a devolver una palabra a la forma base o de diccionario conocida como lema y es un paso importante de preprocesamiento para muchas aplicaciones de minería de texto y también se utiliza en el procesamiento de lenguaje natural. Y aunque realizar el proceso de stemming es más fácil que el de la lematización, esta última resulta ser la mejor opción.

Para realizar el proceso de lematización se requiere una profunda comprensión lingüística para formar el glosario que permita al algoritmo buscar la parte significativa de la palabra. Una vez hecho esto, el resultado será más preciso, ya que al hacer uso del vocabulario y el análisis morfológico de las palabras se reciben resultados libres de afijos derivados. Por otro lado, la derivación simplemente corta el final de las palabras, independientemente del hecho de que el resultado transmita información significativa o no (Divya et al., 2021).

Para la simplificación del texto, ya sea Derivación o Lematización, ambos ayudan a reducir la dimensionalidad de un modelo sin un efecto significativamente negativo en la precisión del entrenamiento. Los modelos que utilizan textos que se someten a estas operaciones aún pueden lograr un resultado aceptable. La técnica de derivación es más rápida que la lematización, pero los textos pueden terminar con palabras sin sentido y sin significado de diccionario (Kurasinski, 2020).

Figura 4. Ejemplo de Lematización (DOBBINS, 2018)



Diferencia entre Derivación y Lematización

- La derivación es más rápida porque corta palabras sin conocer el contexto de la palabra en oraciones dadas y la lematización es más lenta, pero conoce el contexto de la palabra antes de continuar.
- El lema es el fundamento de todas sus partes flexivas y la raíz no lo es. Por eso los diccionarios son un registro de lemas y no de raíces.
- El algoritmo de derivación funciona cortando el sufijo de la palabra. En un sentido más amplio, corta el principio o el final de la palabra. Por el contrario, la Lematización es una operación más poderosa y toma en consideración el análisis morfológico de las palabras. Y devuelve el lema que es la forma base de todas sus formas flexivas.
- La derivación es un enfoque basado en reglas y la matización es un enfoque basado en diccionarios.
- La derivación convierte cualquier palabra en forma de raíz, pero esto puede crear el significado de inexistencia de una palabra (menos precisión), la lematización siempre da el significado de diccionario de la palabra mientras se convierte en forma de raíz (más precisión).

- Es preferible la derivación cuando el significado de la palabra no es importante para el análisis, se recomienda la lematización cuando el significado de la palabra es importante para el análisis.

En relación a los chatbots la lematización se puede utilizar para entrenar un bot, por ejemplo, si se quiere entrenar un bot tipo alexa para que ayude en casa, es importante que el bot sepa que las formas plural y singular se refieren a lo mismo. Como por ejemplo "¡enciende la luz!" ¡Enciende las luces!

Un aspecto importante a tener en cuenta es que muchos productos de inteligencia artificial como los chatbots requieren datos de entrenamiento para que funcionen de manera más eficiente. Cuantos más datos se tengan mejor funcionarán. Pero el entrenamiento lleva tiempo y el tiempo siempre es esencial, así que la incorporación de un lematizador podría resultar eficaz para ahorrar tiempo especialmente cuando se trata de lenguajes más sintéticos. Sin embargo, antes de lematizar se debería de considerar cuán detallados se quieren los datos de entrenamiento.

Naive Bayes (Clasificación Bayesiana Ingenua)

Los Naive Bayes son algoritmos supervisados de machine learning que se utilizan principalmente para la clasificación. Naive Bayes se basa en el Teorema de Bayes, que fue propuesto por Thomas Bayes en la década de 1760. El clasificador asume que las características de entrada que entran al modelo son independientes entre sí. Por lo tanto, cambiar una función de entrada no afectará a ninguna de las demás. Por lo tanto, es ingenuo (naive) en el sentido de que esta suposición puede o no ser cierta y lo más probable es que no lo sea.

Una de las ventajas significativas de Naive Bayes es que utiliza un enfoque probabilístico; todos los cálculos se realizan sobre la marcha en tiempo real y los resultados se generan instantáneamente y cuando se manejan grandes cantidades de datos, esto le da a Naive Bayes una ventaja sobre los algoritmos de clasificación tradicionales (TALUKDER, 2011) (SHENAVARI, 2018).

La clasificación Naive Bayes presenta una relación probabilística entre el conjunto de atributos, la frecuencia de aparición y la posición de aparición; con la variable de clase trabajando con el

teorema de probabilidad de Bayes para predecir la clase del conjunto de datos desconocido. Esto con una suposición en donde las características de las propiedades contribuyen de forma independiente a la probabilidad (SHENAVARI, 2018).

Por ejemplo, se puede considerar que una fruta es una manzana si es roja, redonda y tiene aproximadamente 4 pulgadas de diámetro. Incluso si estas características dependen unas de otras o de la existencia de otras características, un clasificador de naive bayes considera que todas estas propiedades contribuyen de forma independiente a la probabilidad de que esta fruta sea una manzana (TALUKDER, 2011).

Pero para entender cómo funciona Naive Bayes primero es importante entender el Teorema de Bayes. Pero para entender el Teorema de Bayes es importante refrescar los conceptos de Probabilidad y Probabilidad Condicional.

Probabilidad

La probabilidad es una de las ramas de las matemáticas que ayuda a predecir la probabilidad de que suceda un evento X considerando el total de resultados potenciales (Sánchez et al., 2015) (Irwin, 2008).

Probabilidad de un evento = Número de eventos favorables / Número total de resultados

Los "eventos favorables" denotan el (los) evento (s) para los que se desea que ocurra la probabilidad. La probabilidad siempre se encuentra en el rango de 0 a 1, donde 0 significa que no hay posibilidad de que ese evento suceda y 1 significa que hay un 100% de posibilidades de que suceda (Sánchez et al., 2015) (Irwin, 2008).

Probabilidad Condicional

La probabilidad condicional se calcula para dos o más eventos. Tome dos eventos A y B. La probabilidad condicional del evento B se define como la probabilidad de que ocurra el evento B dado el conocimiento de que el evento A ya sucedió. Se representa como P(B|A) mediante la fórmula matemática (Irwin, 2008).

$$P(B|A) = P(A \text{ and } B)/P(A)$$

Teorema de Bayes (Bayes Theorem)

Nombrado en honor al matemático inglés Thomas Bayes, quien estudió la probabilidad en el siglo XVIII y quien construyó este teorema y fue el primero en usarlo. Este teorema está basado en la probabilidad condicional donde se calcula la probabilidad de cierto evento A, dada la probabilidad de otro evento B. Es decir, encuentra la probabilidad de que ocurra un evento dada la probabilidad de que otro evento ya haya ocurrido (Norberg, 2013).

Es un principio estadístico para combinar el conocimiento previo de las clases con nueva evidencia recopilada a partir de los datos mediante la siguiente fórmula (SHENAVARI, 2018).

$$P(Y|X) = [P(Y|X)P(Y)] / P(X)$$

Donde Sea X el conjunto de atributos, Y la variable de clase, P(Y) es la probabilidad previa que se puede estimar a partir del conjunto de entrenamiento por la fracción de datos que pertenecen a cada clase, P(X|Y) la probabilidad condicional de clase a la que se le aplica la clasificación Naive Bayes, P(Y|X) las probabilidades posteriores para cada combinación de X e Y con base en la información recopilada de los datos de entrenamiento y P(X) la evidencia. (SHENAVARI, 2018).

En machine learning se suele tener datos de entrenamiento para enseñar le al modelo y datos de validación para evaluar el modelo y hacer nuevas predicciones. La meta es encontrar la probabilidad de un resultado con respecto a su entrada. Se llamará a las características de entrada "evidencia" y a las etiquetas como los "resultados" en los datos de entrenamiento. Usando la probabilidad condicional, se calcula la probabilidad de la evidencia dados los resultados, denotados como P(Evidencia|Resultado) (Norberg, 2013).

Ahora se define la Regla de Bayes para ambos, P(Evidencia|Resultado) y P(Resultado|Evidencia). Considerando X para denotar Evidencia e Y para denotar Resultado:

- P(Evidence|Resultado) es por lo tanto P(X|Y) y se representa de la siguiente manera: P(X|Y) = (P(Y|X) * P(X)) / P(Y) (Para ser estimada de los datos de entrenamiento.)
- P(Resultado|Evidencia) es P(Y|X) y se representa de la siguiente manera: P(Y|X) = (P(X|Y) * P(Y)) / P(X) (Para ser predecida a partir de los datos de prueba).

Si el problema en cuestión tiene dos resultados, entonces calculamos la probabilidad de cada resultado y se dice que gana el valor más alto. Pero, ¿qué pasa si se tiene múltiples características de entrada? Aquí es donde el teorema de Naive Bayes entra en juego.

Mientras que el teorema de Bayes calcula la probabilidad de que ocurra un evento dado que ya ha ocurrido otro evento, Naive Bayes modifica el método y asume "ingenuamente" que cada evento es condicionalmente independiente entre sí (Norberg, 2013).

Un clasificador de Naive Bayes asume que la presencia (o ausencia) de una característica particular de una clase no está relacionada con la presencia (o ausencia) de ninguna otra característica. Dependiendo de la naturaleza precisa del modelo de probabilidad, los clasificadores Naive Bayes pueden entrenarse de manera muy eficiente en un entorno de aprendizaje supervisado (TALUKDER, 2011).

Considerando el caso donde hay múltiples entradas (X1, X2, X3, ... Xn). Se predice el resultado (Y) utilizando la ecuación de Naive Bayes de la siguiente manera (Norberg, 2013):

$$P(Y \mid X1 \ldots Xn)$$

$$= \frac{P(Y \mid X1 \ldots Xn) \;=\; (\,P(X1 \mid Y) \;*\; P(X2 \mid Y) \;*\; P(X3 \mid Y) \;*\ldots.*\; P(Xn \mid Y)\,) \;*\; P(Y)}{P(X1) * P(X2) * P(X3) *\ldots.* P(Xn)}$$

Donde

- P(Y | X1...Xn) se llama Probabilidad Posterior, que es la probabilidad de un resultado dada la evidencia
- P(X1 | Y) * P(X2 | Y) * ... P(Xn | Y) es la probabilidad de la posibilidad de evidencia
- P(Y) es la probabilidad previa
- P(X1) *P(X2) *P(Xn) es la probabilidad de la evidencia.

Como se puede observar la ecuación solo crece linealmente para cada variable adicional, por lo que es posible procesar grandes cantidades de datos al mismo tiempo sin ningún problema.

En un chatbot, el algoritmo Naive Bayes podría utilizarse para intentar clasificar los muchos textos de entrada en ciertas categorías para que el chatbot pueda identificar la intención del usuario (intent) y, por lo tanto, reducir su posible rango de respuestas. Dado que la identificación de intenciones (intents) es uno de los primeros y más importantes pasos en las conversaciones de un chatbot, es imperativo que este algoritmo funcione correctamente. El algoritmo debería basarse en lo común, lo que esencialmente significa que ciertas palabras deberían tener más peso para categorías particulares en función de la frecuencia de sus apariciones en esa categoría.

Árbol de decisión (Decision Tree)

Un árbol de decisión es una técnica de clasificación que se centra en una forma de representación fácilmente comprensible y es uno de los métodos de aprendizaje más comunes. Los árboles de decisión utilizan conjuntos de datos que consisten en vectores de atributos, que a su vez contienen un conjunto de atributos de clasificación que describen el vector y un atributo de clase que asigna la entrada de datos a una determinada clase. Un árbol de decisión se crea dividiendo iterativamente el conjunto de datos en el atributo que separa los datos lo mejor posible en las diferentes clases existentes hasta que se alcanza un determinado criterio de stop. Los árboles de decisión se pueden visualizar fácilmente en un formato estructurado de árbol, que es fácil de entender para los humanos (Luckert & Schaefer-kehnert, 2015).

Los árboles de decisión son árboles dirigidos, que se utilizan como herramienta de apoyo a la toma de decisiones. Representan reglas de decisión e ilustran decisiones sucesivas. Los nodos se pueden separar en: nodo raíz, nodos internos y nodos finales, también llamados hojas (Gupta, 2018) (Peng et al., 2002).

- El nodo raíz representa el inicio del proceso de soporte de decisiones y no tiene aristas entrantes.
- Los nodos internos tienen exactamente un borde entrante y tienen al menos dos bordes salientes. Contienen una prueba basada en un atributo del conjunto de datos. Por ejemplo, una prueba de este tipo podría preguntar: "¿Tiene el cliente más de 35 años para el atributo edad?".
- Los nodos hoja consisten en una respuesta al problema de decisión, que en su mayoría está representado por una predicción de clase. Como ejemplo, un problema de decisión podría ser

la pregunta de si un cliente realizará una compra o no, siendo las predicciones de clase sí y no. Los nodos de hoja no tienen un borde saliente y solo uno entrante.

- Los bordes representan la decisión tomada desde el nodo anterior.

Dado un nodo n, todos los nodos siguientes que están separados exactamente por una arista de n se denominan hijos de n, mientras que n se denomina padre de todos sus nodos hijos. El entrenamiento de un árbol de decisión en un escenario supervisado se realiza mediante un conjunto de entrenamiento para encontrar patrones dentro de los datos y construir el árbol de decisiones. Posteriormente, se puede usar un conjunto de ejemplos nunca antes vistos para predecir el valor de su atributo de destino. Para entrenar un árbol de decisión y crear un clasificador, se necesita un conjunto de entrenamiento que contenga un atributo objetivo, atributos de entrada, un criterio de división y un criterio de stop.

En un nodo determinado, el criterio de división calcula un valor para todos los atributos. Este valor representa una medida de la cantidad de información que se obtiene al dividir el nodo utilizando este atributo. Posteriormente, se toma el mejor valor de todos los atributos y el nodo se divide en los diferentes resultados del atributo respectivo. En este punto, el proceso de encontrar la mejor división entre los atributos se aplica recursivamente a todos los subárboles generados hasta que se alcanza un criterio de stop, los criterios de stop comunes son:

- Se ha alcanzado la altura máxima del árbol.
- El número de registros en el nodo es menor que el mínimo permitido.
- El criterio de la mejor división no supera un cierto umbral en términos de información obtenida.

Entrenar un árbol de decisión con este proceso automatizado puede generar grandes árboles de decisión con secciones de muy poca potencia en cuanto a clasificación. Además, los árboles tienden a ser sobre ajustados (overfitted), lo que significa que se ajustan demasiado a las instancias de entrenamiento. Esto resulta en un mal rendimiento cuando estos árboles se aplican a datos no vistos.

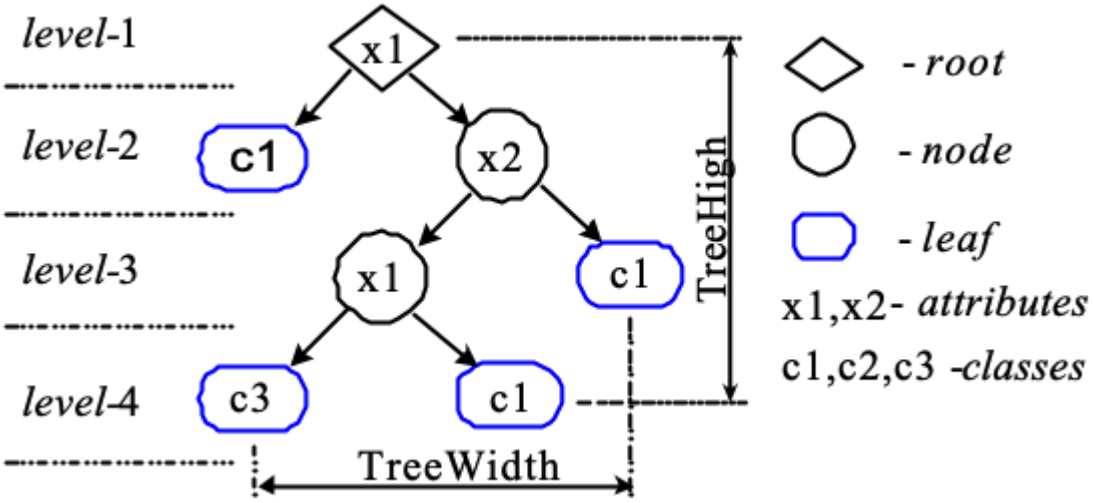
Por lo tanto, una técnica llamada pruning se ha desarrollado. El pruning consiste en eliminar las ramas que hacen uso de características que tienen poca importancia. De esta manera, se reduce la complejidad del árbol y por lo tanto, se aumenta su poder predictivo al reducir el sobreajuste. El pruning puede comenzar en la raíz o en las hojas. El método más simple comienza en las hojas y elimina cada nodo con la clase más popular en esa hoja, este cambio se mantiene si no deteriora la precisión (Gupta, 2018).

En relación a los chatbots, un árbol de decisión puede ser utilizado para ayudar a encontrar la respuesta exacta a la pregunta hecha por el usuario, por ejemplo el nodo inicial puede ser la pregunta del usuario y con base en esta pregunta se hace el enlace a un intent el cual para terminar podría necesitar información adicional y para obtenerla el bot hace una serie de preguntas al usuario cada respuesta o serie de respuestas pueden abrir o desencadenar a otra pregunta y así sucesivamente hasta completar la acción.

Por ejemplo, si nos encontramos con un bot de venta y la persona quiere comprar una camisa para llevar a cabo la compra es posible que el bot pregunte el color de la camisa y la talla de la camisa y con base en esas respuestas verificar si la compra puede realizarse o no.

Actualmente existe un tipo de chabot el cual es creado con base en un árbol de decisión, estos son conocimos como rule-based chatbots y consisten en una serie de reglas pre definidas que dirigen al usuario guiando la conversación a través de una serie de condiciones que les ayuda a los usuarios a obtener una respuesta más precisa a sus consultas, estos requieren un análisis exhaustivo de consultas y de datos históricos de servicio al cliente. Una vez que se determinan las preguntas frecuentes, los chatbots basados en reglas reducen lentamente cada conversación hasta que el visitante está satisfecho con su respuesta. A veces, los bots también los llevan a un agente en vivo si la persona del otro lado no está satisfecha con la respuesta.

Figura 5. Estructura de un árbol de decisión (Peng et al., 2002)



Redes Neuronales Artificiales (Artificial Neural Network - ANNs)

Las redes neuronales, también conocidas como redes neuronales artificiales (ANN) o redes neuronales simuladas (SNN), son un subconjunto del machine learning. Su nombre y estructura están inspirados en el cerebro humano, imitando la forma en que las neuronas biológicas envían señales entre sí (IBM Cloud Education, 2021).

Las ANNs se inspiraron en las ciencias biológicas, particularmente en las ciencias neurológicas y para entender cómo funcionan es importante mencionar brevemente cómo funciona el cerebro humano. Un cerebro humano contiene una enorme cantidad de células nerviosas, llamadas neuronas. La información o las señales se transmiten unidireccionalmente a través de las conexiones entre neuronas conocidas como axones, una neurona recibe la información a través de sus dendritas. El cerebro humano consta de alrededor de 100 mil millones de neuronas y más de 10^{14} sinapsis. Las neuronas se comunican entre sí a través de sinapsis, que son espacios o uniones entre las conexiones (Tan, 1997).

Esto crea una red muy compleja de transmisión de señales. Cada célula recopila entradas de todas las demás células neuronales a las que está conectada y si alcanza un cierto umbral envía una señal a todas las células a las que está conectada. El aprendizaje generalmente se realiza ajustando las sinapsis existentes, aunque algunas funciones de aprendizaje y memoria se llevan a cabo mediante la creación de nuevas sinapsis (Tan, 1997).

Es a través de estas sinapsis que la mayor parte del aprendizaje se lleva a cabo al excitar o inhibir la actividad de sus neuronas asociadas. El procesamiento en el cerebro biológico es altamente paralelo y también es muy tolerante a fallas. La característica de tolerancia a fallas es el resultado de que las vías neuronales son muy redundantes y que la información se propaga a través de las sinapsis en el cerebro. Esta amplia distribución de información también permite que las vías neuronales manejen bien los datos ruidosos (Tan, 1997).

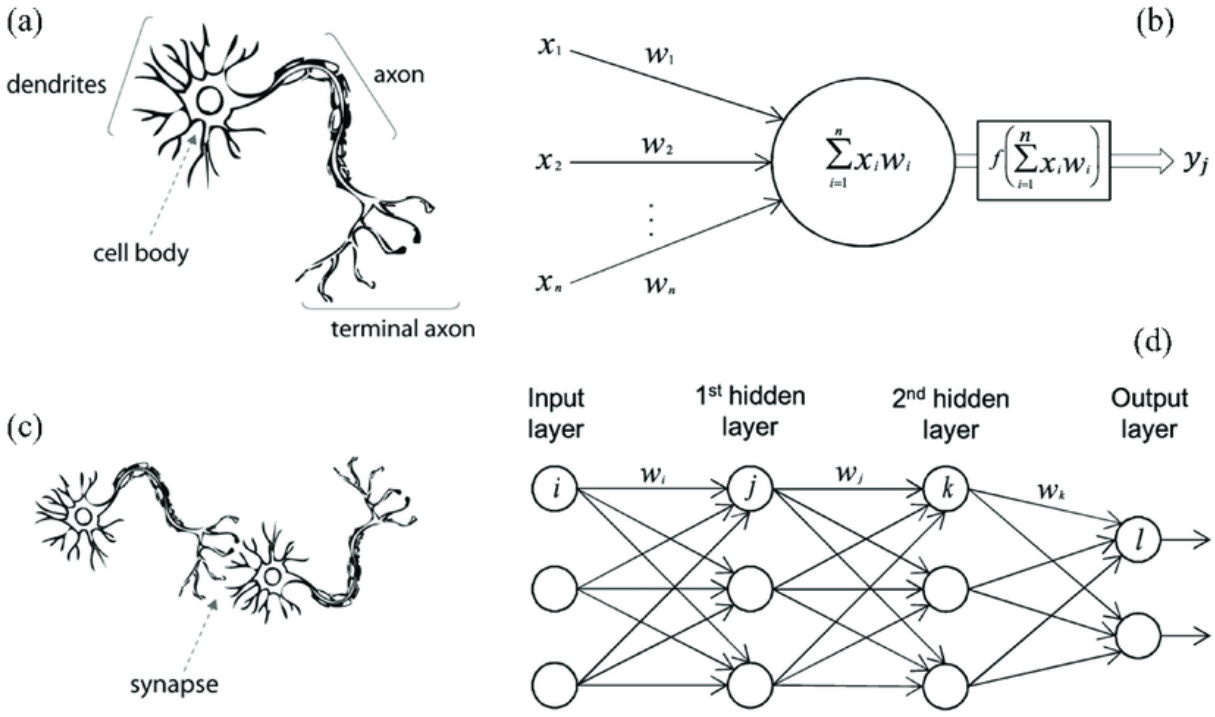
Las ANN todavía están lejos de igualarse al funcionamiento de las redes biológicas más simples, debido a la enorme complejidad de las redes biológicas. Una neurona biológica es tan compleja que las supercomputadoras actuales ni siquiera pueden modelar una sola neurona. Por lo tanto, los investigadores han simplificado los modelos de neuronas en el diseño de ANN.

Las ANNs se componen de capas de nodo, que contienen una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo o neurona artificial, se conecta con otro y tiene asociado un peso y un umbral. Si la salida de cualquier nodo individual está por encima del valor del umbral especificado, ese nodo se activa y envía datos a la siguiente capa de la red. De lo contrario, no se pasan datos a la siguiente capa (IBM Cloud Education, 2021).

En una ANN, los datos se reciben a través de la capa de entrada de las neuronas de y luego se transforman en la primera capa oculta de las neuronas a través de las conexiones ponderadas establecidas entre la capa de entrada y la primera capa oculta. Aquí, los datos en cada capa se procesan matemáticamente y luego el resultado se transforma a la siguiente capa (Sathelly, 2018).

En otras palabras, una ANN suele ser una red computacional basada en redes neuronales biológicas que construyen la estructura del cerebro humano. Al igual que un cerebro humano tiene neuronas interconectadas entre sí, las redes neuronales artificiales también tienen neuronas que están conectadas entre sí en varias capas de las redes. Estas neuronas se conocen como nodos. En el campo de la inteligencia artificial, donde se intenta imitar el cerebro humano para que las computadoras tengan la opción de comprender las cosas y tomar decisiones como un humano. La red neuronal artificial está diseñada programando computadoras para que se comporten simplemente como células cerebrales interconectadas.

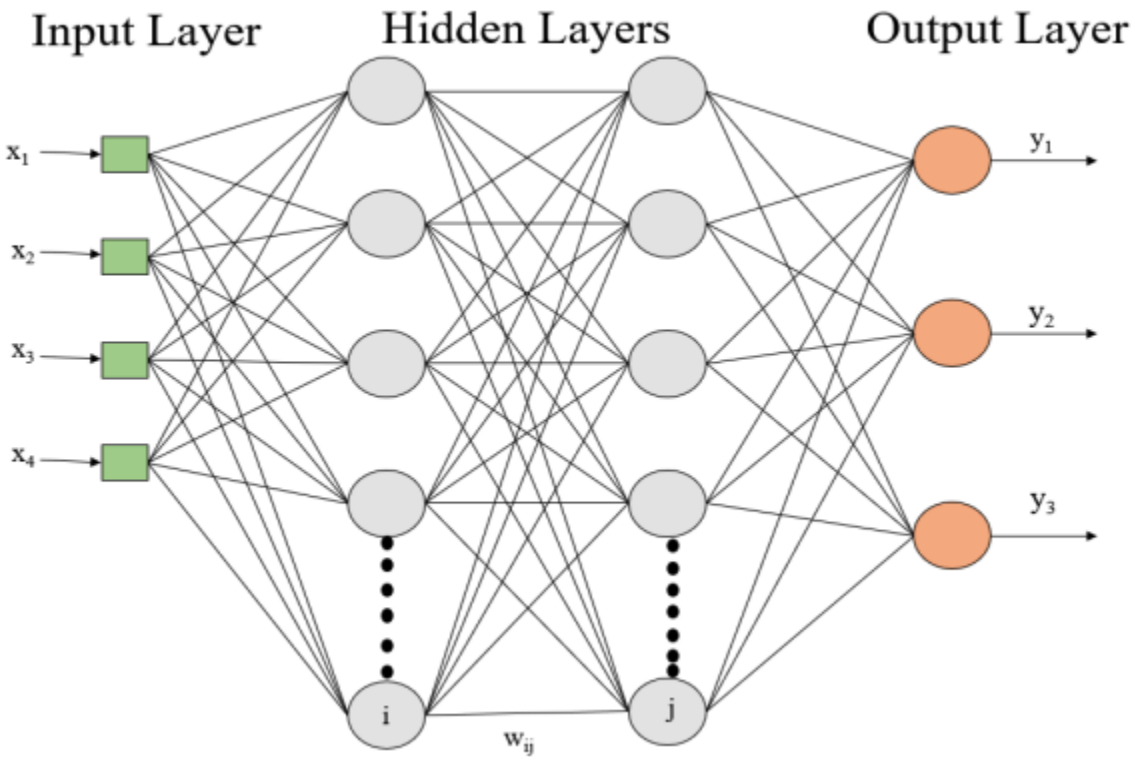
Figura 6. Red neuronal Biologica vs Red neuronal artificial (Zhenzhu Meng, 2020)



Por ejemplo, si se toma un ANN como una puerta lógica digital que toma una entrada y da una salida. Por ejemplo, la puerta "OR" que toma dos entradas. Si una o ambas entradas es "true", obtenemos "true" en la salida. Si ambas entradas están "false", entonces obtenemos "false" en la salida, aquí la salida depende de la entrada. El cerebro humano no realiza la misma tarea. La relación entre salidas y entradas sigue cambiando debido a las neuronas en el cerebro, que están

aprendiendo. Las dendritas de la red neuronal biológica representan entradas en redes neuronales artificiales, el núcleo celular representa nodos, la sinapsis representa pesos y Axon representa salida.

Figura 7. Estructura general de la red neuronal artificial con dos capas ocultas (Sathelly, 2018)



Modelo Matemático

Cada nodo individual tiene su propio modelo de regresión lineal, compuesto de datos de entrada, ponderaciones, un sesgo (o umbral) y una salida. La fórmula sería algo como esto (IBM, 2021):

$$\sum_{i=1}^{m} \quad W_i X_i \;\; + \; bias \; = \; W_1 X_1 + W_2 X_2 \; + W_3 X_3 \; + \; bias$$

Cuya salida será representada por la fórmula de activación:

$$f(x) = \begin{Bmatrix} 1 \rightarrow \sum W_1 X_1 + b \geq 0 \\ 0 \rightarrow \sum W_1 X_1 + b < 0 \end{Bmatrix}$$

Una vez determinada una capa de entrada, se asignan pesos. Estos pesos ayudan a determinar la importancia de cualquier variable dada y las más grandes contribuyen de manera más significativa a la salida en comparación con otras entradas. Luego, todas las entradas se multiplican por sus respectivos pesos y luego se suman (IBM, 2021).

Posteriormente, la salida pasa a través de una función de activación, que determina la salida. Si esa salida excede un umbral dado se activa el nodo, pasando datos a la siguiente capa en la red. Esto da como resultado que la salida de un nodo se convierta en la entrada del siguiente nodo, Este proceso de pasar datos de una capa a la siguiente define esta red neuronal como una red de retroalimentación (IBM, 2021).

Ejemplo de aplicación (IBM, 2021).
Se necesita decidir si Julio debería ir a surfear (Sí: 1, No: 0). La decisión de ir o no ir es el resultado previsto. Se asume que hay tres factores que influyen en la toma de decisiones.

- ¿Son buenas las olas? (X1) (Sí: 1, No: 0)
- ¿Está vacía la alineación? (X2) (Sí: 1, No: 0)
- ¿Ha habido un ataque de tiburón reciente? (X3) (Sí: 0, No: 1)

Entonces, se tienen las siguientes entradas:

- X1 = 1, ya que las olas están bombeando
- X2 = 0, ya que las multitudes están afuera
- X3 = 1, ya que no ha habido un ataque de tiburón reciente

Ahora, se necesita asignar algunos pesos para determinar la importancia, los pesos más grandes significan que las variables particulares son de mayor importancia para la decisión o el resultado.

- W1 = 5, ya que las olas grandes no ocurren con frecuencia
- W2 = 2, ya que Julio está acostumbrado a las multitudes
- W3 = 4, ya que Julio le tienes miedo a los tiburones

Finalmente, también se asumirá un valor de umbral de 3, lo que se traduciría en un valor de sesgo de -3. Con todas las diversas entradas, se puede comenzar a introducir valores en la fórmula para obtener el resultado.

$$f(x) = (1 * 5) + (0 * 2) + (1 * 4) - 3 \Rightarrow 6$$

Utilizando la función de activación se puede determinar que la salida de este nodo sería 1, ya que 6 es mayor que 0. Para esta instancia, se tendría la respuesta que Julio si irá a surfear.

En el caso de los chatbots ANNs pueden ser utilizadas para la clasificación del texto, para el entrenamiento del bot, para determinar cuál es la mejor respuesta a una pregunta dada.

Reconocimiento De Entidades Nombradas (Ner - Named Entity Recognition)

El reconocimiento de entidades nombradas, también conocido como identificación de entidades, fragmentación de entidades o extracción de entidades. Es una subtarea de extracción de información que se enfoca en reconocer unidades de información llamadas "entidades" como, por ejemplo: nombres de personas, organizaciones, ubicaciones, productos, grupos, fecha y hora, porcentajes, etc (Zhang, 2019).

La categorización de una entidad nombrada puede ser diferente según el propósito de la tarea. NER es una tarea de NLP a menudo realizada como un paso de preprocesamiento de tareas más complejas. Como por ejemplo se necesita como un paso de preprocesamiento para tareas de NLP como traducción automática, respuesta a preguntas, recuperación de información, etc. (Zhang, 2019).

En otras palabras, es la tarea de identificar y categorizar la información clave (entidades) en el texto. Una entidad puede ser cualquier palabra o serie de palabras que se refiera consistentemente a la misma cosa. Cada entidad detectada se clasifica en una categoría predeterminada. Por ejemplo, un modelo de aprendizaje automático (ML) de NER podría detectar la palabra "Microsoft" en un texto y clasificarlo como una "Empresa".

Estas entidades también a veces se denominan nombres propios en lenguaje natural. La tarea generalmente implica la clasificación de estas entidades identificadas en un conjunto de clases predefinidas. Por lo tanto, la tarea de NER a veces se subdivide en dos subtareas: identificación de

entidades nombradas (la tarea es identificar correctamente un intervalo de entidades nombradas sin clasificación adicional, es decir, recuperar todos los tokens de entidades nombradas) y clasificación de entidades nombradas (la tarea es para clasificar correctamente los tokens de entidad con nombre recuperados en un conjunto de clases predefinidas). Por lo general ambas tareas son realizadas conjuntamente (Straková, 2017).

Un dominio típico para NER es la recuperación de nombres de personas, ubicaciones y organizaciones en artículos periodísticos. Sin embargo, esto implica obviamente el reconocimiento de cualquier secuencia destacada o importante. La complejidad de la tarea particular de reconocimiento depende de las características morfosintácticas del idioma, la cantidad y calidad de los datos supervisados disponibles y obviamente del número y la jerarquía de las clases de entidades nombradas (Straková, 2017).

Los primeros sistemas para tareas NER tienden a utilizar algoritmos basados en reglas hechos a mano, mientras que los sistemas modernos suelen utilizar métodos de aprendizaje automático supervisados, semi supervisados y no supervisados. El aprendizaje supervisado requiere una gran cantidad de datos anotados que pueden ser costosos de recopilar. Se propone el aprendizaje semi supervisado y no supervisado para abordar este problema mediante el uso de datos no etiquetados. Los enfoques dominantes hoy en día son la combinación de CRF y diferentes métodos de redes neuronales (Zhang, 2019).

Enfoques basados en reglas

Las herramientas de NER que utilizan enfoques basados en reglas suelen generar patrones de estado finito manualmente. Los patrones que son similares a la expresión regular tienen como objetivo hacer coincidir una secuencia de palabras. Mikheev (1998) utilizó XML para simplificar el proceso. Los tokenizadores que se utilizan aquí no solo dividen palabras por espacios, sino que también identifican tokens de acuerdo con alguna definición acordada. Por ejemplo, este tokenizador puede identificar a "Robert Downey Jr" como un token único. En comparación con identificar tiempo y números, identificar nombres es más complicado y depende más del contexto (Zhang, 2019).

Enfoques basados en datos

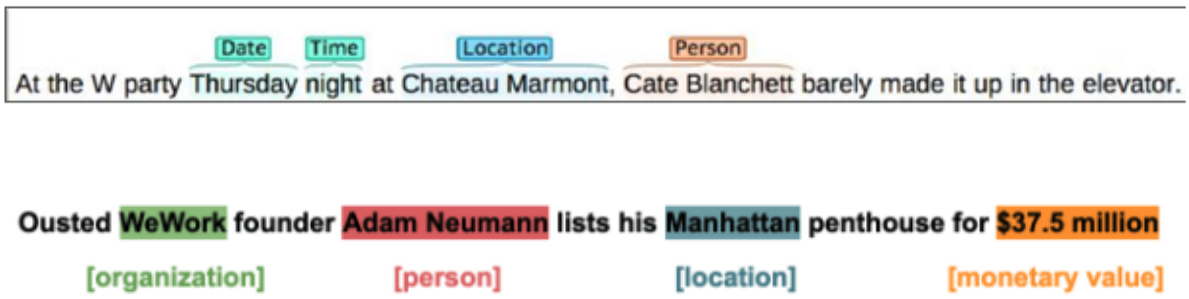
Un problema importante con los enfoques basados en reglas es que las reglas generalmente se crean para un dominio específico, lo que dificulta su aplicación en otros lugares. Como mejora, los métodos basados en datos se adoptan ampliamente hoy en día para crear reglas automáticamente utilizando aprendizaje supervisado, semi supervisado y no supervisado. El aprendizaje supervisado requiere una gran cantidad de datos anotados que contienen ejemplos positivos y negativos de entidades nombradas. El modelo estudia las características del corpus anotado para producir las reglas automáticamente para que coincidan con los tipos dados de entidades nombradas. El aprendizaje semi supervisado utiliza datos anotados y datos no etiquetados. Comienza usando algunos datos anotados bajo supervisión para "arrancar" el proceso de aprendizaje. El aprendizaje no supervisado sólo requiere datos no etiquetados y el enfoque típico es el agrupamiento (Zhang, 2019).

Se puede decir que los pasos principales para un NER son detectar una entidad nombrada y categorizar la entidad, pero un modelo típico de NER consta de 3 bloques (Goyal, 2021):

- Identificación de frases nominales: Este paso trata de extraer todas las frases nominales de un texto con la ayuda del análisis de dependencia y el etiquetado de partes del discurso.
- Clasificación de frases: En este paso de clasificación, se clasifican todas las frases nominales extraídas del paso anterior en sus respectivas categorías. Un ejemplo de categorías puede ser ubicación, hora y fecha, organización, persona, etc.
- Desambiguación de entidades: A veces, lo que sucede es que las entidades están mal clasificadas, por lo que resulta útil crear una capa de validación sobre los resultados.

Con la ayuda del reconocimiento de entidades nombradas, se puede extraer información clave para comprender el texto o simplemente usarla para extraer información importante para almacenar en una base de datos. En el caso de los chatbots NER se puede utilizar para la creación y detección de entidades predefinidas dentro del contexto del bot y con base en su identificación tomar decisiones sobre qué acciones ejecutar sobre la data.

Figura 8. Ejemplo de reconocimiento de entidades nombradas en una oración (Goyal, 2021)



Los algoritmos y técnicas mencionados anteriormente se pueden resumir en la siguiente tabla.

Tabla 2. Tabla Resumen de algoritmos y técnicas

Algoritmo Método Técnica	Descripción	Aplicación Típica	Aplicación en Chatbots
Tokenización	La Tokenización es el mecanismo por el cual el texto se segmenta en oraciones y frases. El trabajo principal es dividir un texto en partes más pequeñas (llamadas tokens).	Tratamiento de datos, en compiladores, para generar códigos de seguridad,	La tokenización es una tarea común en el procesamiento del lenguaje natural (NLP). Es un paso fundamental dado que los tokens son los componentes básicos del lenguaje natural, la forma más común de procesar el texto sin formato ocurre en el nivel del token.
BPE	Este algoritmo es una instancia de los algoritmos de compresión basados en macros, busca redundancias en el texto mediante la detección de patrones repetidos y comprime las secuencias reemplazando una ocurrencia de dicho patrón con punteros a una ocurrencia anterior.	utilizado en NLP para encontrar la mejor representación de texto usando la menor cantidad de tokens.	Podría utilizarse para el procesamiento de grandes cantidades de entradas de texto, ayudando a dividir las entradas en palabras más sencillas que pueden ser procesadas más rápido y fácil por la máquina.
TF-IDF	Es una medida estadística que evalúa qué tan relevante es una palabra para un documento en una colección de documentos. Esto se lleva a cabo multiplicando dos métricas: Contando las veces que aparece una palabra en un documento y Calculando la frecuencia inversa del documento de la palabra en un conjunto de documentos.	Para motores de búsqueda, en resumen, de texto y extracción de palabras clave, para vectorizar texto en un formato más agradable para NLP.	Ya que se necesita almacenar los datos de entrenamiento del chatbot en pares de preguntas (o sentencias) y respuestas, se puede utilizar TF-IDF para buscar una pregunta (o sentencia) lo más parecida al texto ingresado por el usuario y devolver la respuesta asociada con esa declaración.
Stop-word removal	Se utiliza para optimizar el pre procesamiento de los datos en machine learning, ya que consiste en remover las palabras vacías de un texto, las palabras	Clasificación de lenguaje, filtro de spam, Generación de subtítulos, Generación	Se puede utilizar en un chatbot para hacer una limpieza del texto y remover todas aquellas palabras que no tienen un significado profundo esto ayuda al bot a

	vacías son palabras que no agregan mucho significado a un texto y el removerlas ayuda a disminuir el tiempo de cómputo	automática de etiquetas, pre procesamiento de texto para análisis o predicciones hechas por la computadora	identificar las palabras importantes y así mejorar el modelado predictivo para poder brindar respuestas con mejor precisión reduciendo el tiempo computacional de las entradas
Derivación	Es una técnica en la que un conjunto de palabras de una oración se convierte en una secuencia para acortar su búsqueda. En este método, se normalizan las palabras que tienen el mismo significado, pero tienen algunas variaciones según el contexto o la oración. El proceso de derivación produce variantes de una palabra raíz/base, reduce una palabra base a su palabra raíz (Stem), El Stem no necesariamente tiene que ser una palabra válida por sí misma	Comúnmente se utiliza para eliminar prefijos y sufijos de las palabras.	Puede ser utilizado para el entrenamiento de un chatbot y para la identificar similitud de palabras claves para relacionarlas a un intent
Porter Stemmer	algoritmos de derivación diseñado para eliminar y reemplazar sufijos conocidos de palabras en inglés. Proceso para eliminar las terminaciones morfológicas y flexivas más comunes de las palabras en inglés. Se basa en la idea de que los sufijos en el idioma inglés se componen de una combinación de sufijos más pequeños y simples.	recuperación de información, motores de búsqueda, determinar vocabularios de dominio en análisis de dominio, tokenizador	Puede ser utilizado para el entrenamiento de un chatbot en idioma inglés ayudando al bot a entender mejor el contexto del mensaje
Lematización	Es el proceso de encontrar el lema de una palabra según su significado y contexto que generalmente se refiere al análisis morfológico de las palabras, cuyo objetivo es eliminar las terminaciones flexivas, Debido a que implica derivar del significado de una palabra de algo como un diccionario, su tiempo de procesamiento es más lento pero el resultado es más preciso.	Preprocesamiento de texto para NLP, normalización de texto, big data analytics, inteligencia artificial, sentiment analysis, entornos de recuperación de información, biomedicina, agrupación de documentos, motores de búsqueda	Puede ser utilizado para el entrenamiento del chatbot ayudando al bot a entender mejor el contexto y el mensaje del usuario y así funcionar de manera más eficiente para relacionar el mensaje con los intents.
Naive Bayes	Son algoritmos supervisados de machine learning que se utilizan principalmente para la clasificación, presentan una relación probabilística entre el conjunto de atributos, la frecuencia de aparición y la posición de aparición. Asume que la presencia (o ausencia) de una característica particular de una clase no está relacionada con la presencia (o ausencia) de ninguna otra característica. Por ejemplo, se puede considerar que una fruta es una manzana si es roja, redonda y tiene aproximadamente 4 pulgadas de	predicción en tiempo real, predicción multiclase, clasificación de texto, filtrado de spam, análisis de sentimientos, sistemas de recomendación.	Podría utilizarse para intentar clasificar los muchos textos de entrada en ciertas categorías para que el chatbot pueda identificar la intención del usuario y, por lo tanto, reducir su posible rango de respuestas.

	diámetro. Incluso si estas características dependen unas de otras o de la existencia de otras características, un clasificador de naive bayes considera que todas estas propiedades contribuyen de forma independiente a la probabilidad de que esta fruta sea una manzana.		
Árbol de decisión	Es una técnica de clasificación en un formato estructurado de árbol, utiliza conjuntos de datos en forma de vectores de atributos, que a su vez contienen un conjunto de atributos de clasificación que describen el vector y un atributo de clase que asigna la entrada de datos a una determinada clase. Son árboles dirigidos que poseen un nodo raíz, nodos internos(con un borde entrante y al menos dos de salida), nodos finales (o hojas que consiste en una respuesta al problema).	clasificación y regresión, análisis de decisiones para representar visualmente las decisiones y la toma de decisiones, en minería de datos para derivar una estrategia para alcanzar un objetivo particular.	Puede ser utilizado para ayudar a encontrar la respuesta exacta a la pregunta hecha por el usuario. Actualmente existe un tipo de chabot el cual es creado con base en un árbol de decisión, estos son conocimos como rule-based chatbots y consisten en una serie de reglas pre definidas que dirigen al usuario guiando la conversación a través de una serie de condiciones que les ayuda a los usuarios a obtener una respuesta más precisa a sus consultas.
ANNs	Se inspiraron en las ciencias neurológicas y en cómo funciona el cerebro humano y las neuronas. Las ANNs se componen de capas nodos que contienen una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo o neurona artificial, se conecta con otro y tiene asociado un peso y un umbral. Si la salida de cualquier nodo individual está por encima del valor del umbral especificado, ese nodo se activa y envía datos a la siguiente capa de la red. De lo contrario, no se pasan datos a la siguiente capa.	Clasificación, agrupamiento, reconocimiento y predicción de patrones, reconocimiento de imágenes, NLP, para aproximación de funciones universales en paradigmas numéricos, análisis de datos con grandes entradas.	Se pueden utilizar para la clasificación del texto, para el entrenamiento del bot, para determinar cuál es la mejor respuesta a una pregunta dada.
NER	subtarea de extracción de información que identifica y categoriza información clave en el texto en unidades llamadas entidades, cada entidad se clasifica en categoría predeterminada.	Motores de búsqueda y recomendación, clasificación de contenido, extracción de información esencial, cualquier situación en la que sea útil una visión general de alto nivel de una gran cantidad de texto.	Se puede utilizar para la creación y detección de entidades predefinidas dentro del contexto del bot.

CAPÍTULO 4

4. ENFOQUE TÉCNICO DE CHATBOTS PARA LA IMPLEMENTACIÓN DE CONVERSACIONES AUTOMATIZADAS

4.1 TECNOLOGÍAS EXISTENTES DE CHATBOTS

En la actualidad existen una gran cantidad de proveedores para la creación de chatbots, pero para efectos de esta tesis sólo se mencionará dos proveedores (Google y Microsoft) debido al posicionamiento de estas dos empresas en el mundo de la tecnología actual.

Enfoque técnico de tecnologías ofrecidas por Google - Dialogflow

Dialogflow es una plataforma de comprensión del lenguaje natural que facilita el diseño y la integración de una interfaz de usuario conversacional en diferentes formatos (aplicación móvil, aplicación web, bot, sistema de respuesta de voz interactivo, etc). Puede analizar múltiples tipos de entradas, incluidas entradas de texto o audio (como desde un teléfono o una grabación de voz). También puede responder de varias maneras, ya sea a través de texto o con voz sintética y además es parte de la oferta de IA conversacional dentro de GCP (Google Cloud Platform) (Google, 2022).

Google proporciona dos servicios de agente virtual, cada uno con su propio tipo de agente, interfaz de usuario, API, bibliotecas de clientes y documentación (Google, 2022):

- Dialogflow ES
- Dialogflow CX

Y tiene tres ediciones (Google, 2022):

- Dialogflow Trial Edition: Es la edición gratuita que proporciona la mayoría de las funciones del tipo de agente ES estándar. Ofrece cupo limitado y soporte para la comunidad y correo electrónico. Esta edición es adecuada para experimentar con Dialogflow.
- Dialogflow ES Edition: La edición Dialogflow Essentials (ES) es una edición de pago por uso (pay-as-you-go) que proporciona el tipo de agente ES estándar. Esta edición ofrece cuotas listas para producción y soporte de Google Cloud.

- Dialogflow CX Edition: La edición Dialogflow Customer Experience (CX) es una edición de pago por uso (pay-as-you-go) que proporciona el tipo de agente CX avanzado. La edición CX ofrece cuotas listas para producción y soporte de Google Cloud.

Dialogflow proporcionar servicios de agente virtual para chatbots y centros de contacto que emplean agentes humanos con Agent Assist. Agent Assist proporciona sugerencias en tiempo real para agentes humanos mientras están en conversaciones con los clientes. La API de Agent Assist se implementa como una extensión de la API Dialogflow ES. Aunque Agent Assist es una extensión de Dialogflow ES API, se puede usar un tipo de agente Dialogflow CX como agente virtual para Agent Assist (Google, 2022).

Algunos de los beneficios que se pueden mencionar sobre Dialogflow son:

- Soporta conversaciones ricas e intuitivas con los clientes, impulsadas por la principal IA de Google.
- Es una plataforma de desarrollo integral para chatbots y voicebots.
- Tiene una comunidad de más de 1,5 millones de desarrolladores que construyen con Dialogflow.
- Permite experiencias de cliente más naturales con agentes virtuales que admiten conversaciones de varios turnos con preguntas complementarias y que están construidos con las tecnologías de aprendizaje profundo (deep learning) que potencian el Asistente de Google.
- Reduce el tiempo de desarrollo con un constructor visual y agentes prediseñados que se implementan fácilmente a través de canales digitales, incluidos servicios web, móviles y de mensajería.
- Administra fácilmente los agentes virtuales con CI/CD end-to-end a través del control de versiones y la evaluación continua, con módulos basados en flujo que permiten escalar hasta 20 flujos independientes y 40,000 intentos para cada agente.
- Posee una gran cantidad de documentación y videos tutoriales.

¿Qué es un Agente?

Un agente de Dialogflow es un agente virtual que maneja conversaciones simultáneas con sus usuarios finales. Es un módulo de comprensión del lenguaje natural que comprende los matices del lenguaje humano. Dialogflow traduce el texto o el audio del usuario final durante una conversación en datos estructurados para que las aplicaciones y servicios puedan comprenderlo. El developer diseña y construye un agente para manejar los tipos de conversaciones que requiere un sistema. Un agente es similar a un agente de un centro de llamadas humano. Se entrena para manejar los escenarios de conversación esperados y su entrenamiento no necesita ser demasiado explícito (Google, 2022).

Los agentes soportan 32 idiomas y se pueden programar en 9 lenguajes de programación y a través de cloud Functions o proyectos individuales alojados en la nube utilizando webhook, los lenguajes soportados son: REST, RPC, C#, Go, Java, Node.js, PHP, Python y Ruby.

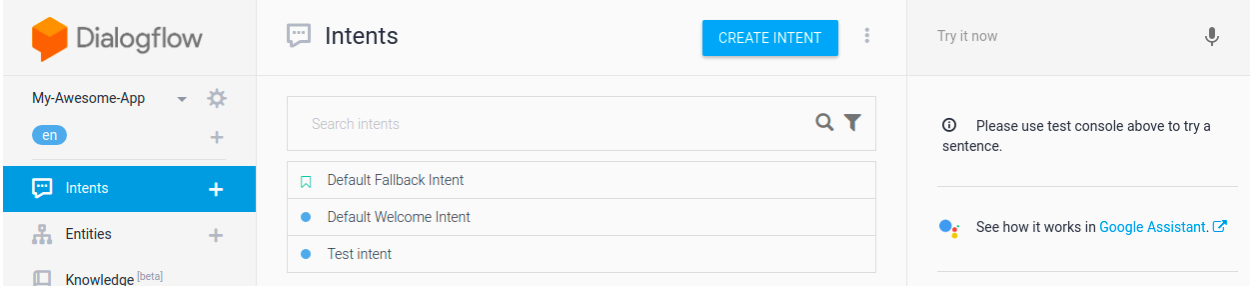
¿Qué es Dialogflow Console?

Dialogflow proporciona una interfaz de usuario web llamada Dialogflow Console. Esta consola se utiliza para crear, compilar y probar agentes. La consola de Dialogflow es diferente de la consola de GCP. Esta se utiliza para administrar los agentes de Dialogflow, mientras que la Consola de GCP se utiliza para administrar la configuración de Dialogflow específica de GCP (por ejemplo, la facturación) y otros recursos de GCP. En la mayoría de los casos se debe usar la Consola de Dialogflow para crear agentes, pero también se puede usar la API de Dialogflow para crear agentes para escenarios avanzados (Google, 2022).

Cuando se inicia sesión en la consola de Dialogflow, se inicia sesión con la cuenta de Google utilizada para acceder a los servicios de GCP. Cuando se inicia sesión por primera vez, se solicitará permisos para que Dialogflow acceda a la cuenta de Google para:

- Ver y administrar la data en GCP.
- Ver y administrar los comandos de voz, el diálogo y la gramática del Asistente de Google.
- Ver y administrar las Google actions.

Figura 9. Dialogflow Console (Google, 2022)



¿Qué son los Webhooks?

Los webhooks son servicios que alojan la lógica del negocio. Durante una sesión, los webhooks permiten usar los datos extraídos por el procesamiento de lenguaje natural de Dialogflow para generar respuestas dinámicas, validar los datos recopilados o activar acciones en el backend. Los webhooks de CX son similares a los webhooks de ES, excepto que los campos de solicitud y respuesta se han cambiado para admitir funciones de CX (Google, 2022).

Los servicios de webhook deben cumplir los siguientes requisitos:

- Debe manejar solicitudes HTTPS. HTTP no es compatible.
- La URL para solicitudes debe ser de acceso público, a menos que sea una cloud function.
- Debe manejar solicitudes POST con un cuerpo JSON de WebhookRequest.
- Debe responder a las solicitudes de WebhookRequest con un cuerpo JSON de WebhookResponse.
- Si el agente no se integra con el acceso a la red privada de Service Directory, las llamadas de webhook se consideran fuera del perímetro del servicio y se bloquean al habilitar los Controles de servicio de VPC. Service Directory admite puntos finales limitados.
- Se debe de tener una forma de seguridad para el webhook.

Dialogflow ES

Este es el tipo de agente estándar que es adecuado para agentes pequeños o medianos de simples a moderadamente complejos. Las intenciones son los componentes básicos del diseño de la conversación y los contextos se utilizan para controlar las rutas de la conversación (Google, 2022).

El agente ES tiene los siguientes componentes:

Intent (Intención)

Un intent (intención) clasifica la intención de un usuario final para un turno de conversación. Para cada agente, se definen muchas intenciones, donde sus intenciones combinadas pueden manejar una conversación completa. Cuando un usuario final escribe o dice algo, lo que se conoce como una expresión de usuario final, Dialogflow hace coincidir la expresión del usuario final con la mejor intención en el agente. Hacer coincidir una intención también se conoce como clasificación de intención (Google, 2022).

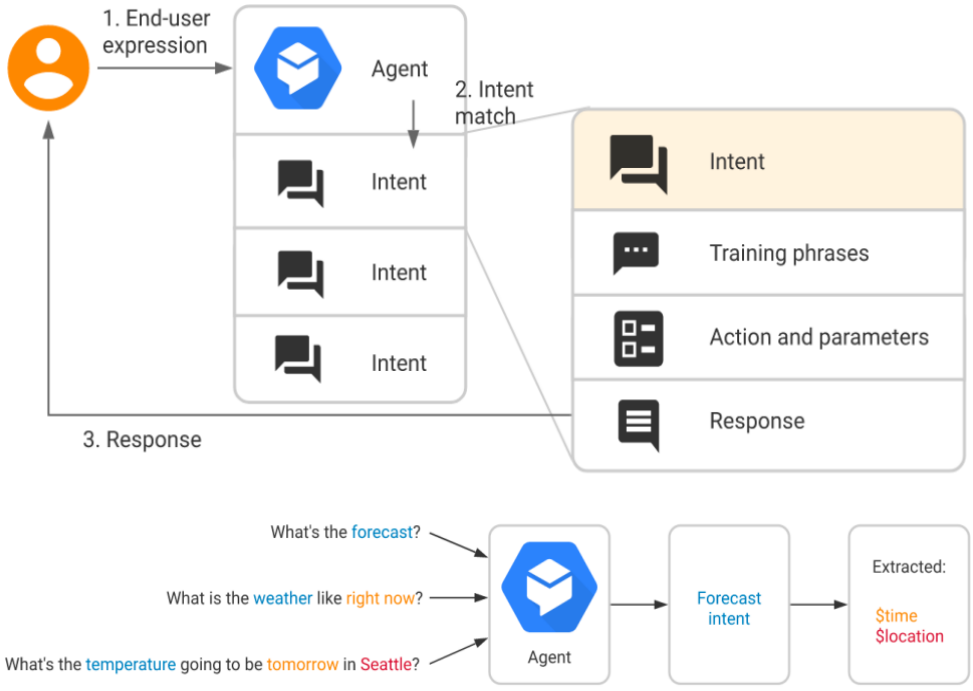
Por ejemplo, podría crear un agente meteorológico que reconozca y responda a las preguntas del usuario final sobre el clima. Es probable que defina una intención para las preguntas sobre el pronóstico del tiempo. Si un usuario final dice "¿Cuál es el pronóstico?", Dialogflow haría coincidir esa expresión del usuario final con la intención del pronóstico. También puede definir su intención de extraer información útil de la expresión del usuario final, como una hora o una ubicación para el pronóstico del tiempo deseado. Estos datos extraídos son importantes para que el sistema realice una consulta meteorológica para el usuario final.

Un intent contiene las siguientes partes:

- Training Phrase (Frase de entrenamiento): Estas son frases de ejemplo de lo que podrían decir los usuarios finales. Cuando una expresión de usuario final se parece a una de estas frases, Dialogflow ES coincide con la intención. No se tiene que definir todos los ejemplos posibles, porque el aprendizaje automático integrado de Dialogflow se expande en su lista con otras frases similares (Google, 2022).

- Action (acción): Se puede definir una acción para cada intención. Cuando una intención coincide, Dialogflow ES proporciona la acción a su sistema y puede usar la acción para desencadenar ciertas acciones definidas en el sistema (Google, 2022).
- Parameters (Parámetros): Cuando se compara una intención en tiempo de ejecución, Dialogflow proporciona los valores extraídos de la expresión del usuario final como parámetros. Cada parámetro tiene un tipo, denominado tipo de entidad (entity type), que dicta exactamente cómo se extraen los datos. A diferencia de la entrada sin procesar del usuario final, los parámetros son datos estructurados que se pueden usar fácilmente para realizar alguna lógica o generar respuestas (Google, 2022).
- Responses (respuestas): Se puede definir respuestas de texto, voz o visuales para devolver al usuario final. Estos pueden proporcionar respuestas al usuario final, solicitar más información al usuario final o finalizar la conversación. Existen dos tipos de respuestas, para generar una respuesta en el intento es posible dejar el texto quemado o generar el texto de forma dinámica a través de un fulfillment (Google, 2022).

Figura 10. Flujo básico de un intent (Google, 2022)



Entities (Entidades)

Cada parámetro de intención tiene un tipo, llamado entity type (tipo de entidad), que dicta exactamente cómo se extraen los datos de una expresión de usuario final. Dialogflow proporciona entidades de sistema predefinidas que pueden coincidir con muchos tipos comunes de datos. Por ejemplo, existen entidades del sistema para hacer coincidir fechas, horas, colores, direcciones de correo electrónico, etc. También se puede crear entidades propias personalizadas para hacer coincidir datos personalizados. Por ejemplo, se podría definir una entidad vegetal que pueda hacer coincidir los tipos de verduras disponibles para comprar con un agente de supermercado (Google, 2022).

El término entidad se utiliza en Dialogflow Console para describir el concepto general de entidades. Al analizar los detalles de la entidad, es importante comprender algunos términos más específicos, como:

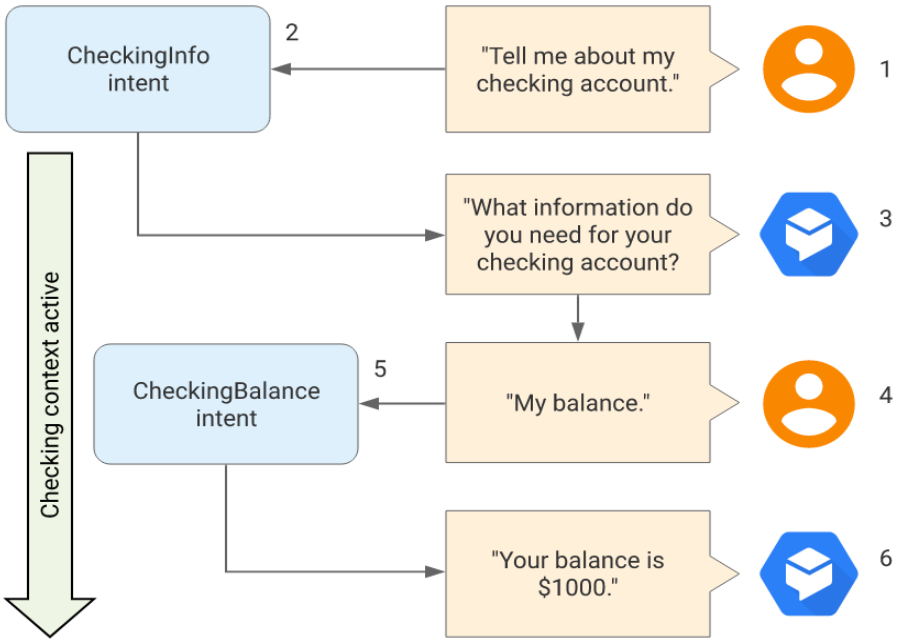
- Entity Type (Tipo de entidad): Define el tipo de información que desea extraer de la entrada del usuario. Por ejemplo, vegetal podría ser el nombre de un tipo de entidad (Google, 2022).
- Entity entry (entrada de una entidad): Para cada tipo de entidad, hay muchas entradas de entidad. Cada entrada de entidad proporciona un conjunto de palabras o frases que se consideran equivalentes. Por ejemplo, si el vegetal es un tipo de entidad, se podrían definir estas entradas de entidad: zanahoria, cebollín, cebolla verde, chile morrón, chile dulce (Google, 2022).
- Entity reference value and synonyms (referencia al valor o sinónimo de una entidad): Algunas entradas de entidad tienen varias palabras o frases que se consideran equivalentes (Google, 2022).

Context (Contexto)

Los contextos de Dialogflow son similares al contexto del lenguaje natural. Si una persona dice "ellos son naranjas", se necesita contexto para entender a qué se refiere "ellos". De manera similar, para que Dialogflow maneje una expresión de usuario final como esa, debe proporcionarse un contexto para que coincida correctamente con una intención. Utilizando el contexto, se puede

controlar el flujo de una conversación. Se puede configurar contextos para una intención estableciendo contextos de entrada y salida (Google, 2022).

Figura 11. Ejemplo de contexto para agente bancario (Google, 2022)



Explicación:

El usuario solicita información sobre su cuenta corriente. Dialogflow hace coincidir esta expresión de usuario final con la intención de CheckingInfo. Esta intención tiene un contexto de salida de verificación, por lo que ese contexto se vuelve activo. El agente le pregunta al usuario final el tipo de información que desea sobre su cuenta corriente.

El usuario final responde con "mi saldo". Dialogflow hace coincidir esta expresión de usuario final con la intención de CheckingBalance. Esta intención tiene un contexto de entrada de verificación, que debe estar activo para coincidir con esta intención. Después de que el sistema realiza las consultas en la base de datos y realiza los procesos necesarios, el agente responde con el saldo de la cuenta corriente.

Follow-up intents (intención de seguimiento)

Se puede utilizar intenciones de seguimiento para establecer automáticamente contextos para pares de intenciones. Un intent de seguimiento es un elemento secundario asociado a un intent principal. Cuando se crea una intención de seguimiento, se agrega automáticamente un contexto de salida a la intención principal y un contexto de entrada con el mismo nombre es agregado a la intención de seguimiento. Una intención de seguimiento sólo coincide cuando la intención principal si coincide en el turno de conversación anterior. También se puede crear múltiples niveles de intenciones de seguimiento anidadas. Dialogflow posee varios follow-up intentes predefinidos para respuestas comunes como yes, no y cancel. Pero también se pueden crear follow-up intentes personalizados (Google, 2022).

Figura 12. Ejemplo de follow-up intent, para un agente de salon de belleza (Google, 2022)

Intent name	Training phrase	Input context	Output context	Intent response
Appointment	Hello		appointment-followup	Would you like to make an appointment?
↳ Appointment - yes	Yes	appointment-followup	appointment-yes-followup	Would you like a haircut?
↳ Haircut - yes	Yes	appointment-yes-followup		Your appointment is set.
↳ Haircut - no	No	appointment-yes-followup		Goodbye.
↳ Appointment - no	No	appointment-followup		Goodbye.

Parameters (Parámetros)

Los parámetros se utilizan para capturar y hacer referencia a valores proporcionados por el usuario final durante una sesión. Cada parámetro tiene un nombre y un tipo de entidad. A diferencia de la entrada sin procesar del usuario final, los parámetros son datos estructurados que se pueden usar fácilmente para realizar alguna lógica o generar respuestas (Google, 2022).

Los parámetros están sujetos a una convención de nombres la cual cumple dos reglas principales:

- Son case insensitive

- Se puede utilizar los siguientes caracteres: [A-Z], [a-z], [0-9], punto (.), guion (-), guion bajo (_)
- También pueden ser de tres tipos:
- Escalar (Scalar): Un único valor numérico o carácter
- Compuesto (Composite): Un objeto JSON que se completa al hacer coincidir una entidad compuesta o al completar un parámetro de intención, que contiene campos originales y resueltos.
- Lista (List): Una lista de valores escalares o compuestos
- Se puede setear los valores a "", que significa un string vacío
- Se puede setear el valor a null, que significa que no ha sido seteado

Existen 4 formas generales de utilizar los parámetros:

- Definidos en design-time: Durante el diseño del agente se puede definir parámetros utilizando la consola o la API de Dialogflow.
- Referenciados en design-time: Las referencias de parámetros son variables que contienen valores de parámetros que se extraerán en tiempo de ejecución. Durante el tiempo de diseño, utiliza la consola o la API para hacer referencia a parámetros en varios tipos de datos.
- Seteados en runtime: En tiempo de ejecución, el servicio de Dialogflow, el servicio que llama a la API y el servicio de webhook pueden establecer valores de parámetros.
- Obtenidos en runtime: En tiempo de ejecución, las referencias de parámetros contienen los valores de los parámetros que se han establecido y puede usar la API o un webhook para obtener los valores de los parámetros.

Al crear un agente, se controla la extracción de los parámetros a través de anotaciones (annotation) en partes de las frases de entrenamiento y se pueden configurar los parámetros asociados a esas anotaciones (Google, 2022).

A Continuación, se presenta la lista de configuraciones para los campos de un parámetro:

- Required (Obligatorio): Se marca esta casilla si se requiere el parámetro para que se complete la intención.

- Parameter Name (nombre): Es el nombre de identificación del parámetro.
- Entity (entidad): El tipo de entidad (entity type) asociada al parámetro.
- Value (valor): En la mayoría de los casos se establece en una referencia de parámetro como $parameter-name, que se usa como marcador de posición para el valor extraído en tiempo de ejecución. Sin embargo, este campo también se puede utilizar para seleccionar valores alternativos.
- Is List (es lista): Se marca esta casilla si los valores deben devolverse como una lista.
- Prompts (indicaciones): Pregunta que el agente hará al usuario final si no se ingresó este parámetro. Este campo sólo se utiliza si el campo obligatorio está marcado.
- Default value (valor por defecto): Este es el valor predeterminado para el parámetro cuando el usuario final no proporciona un valor.

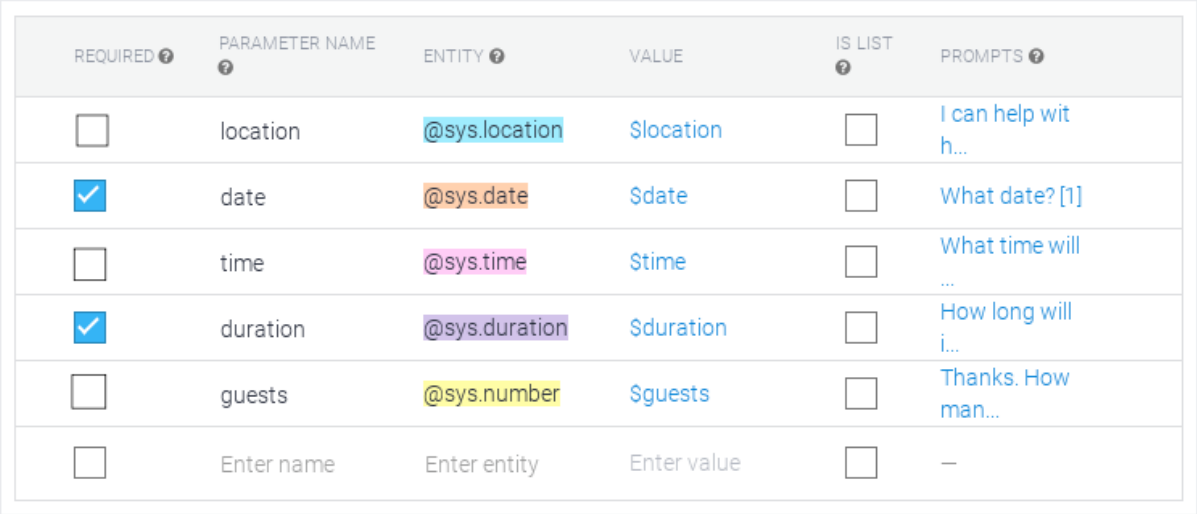
Al crear un agente se puede usar referencias de parámetros en respuestas de intenciones, solicitudes de parámetros y en el campo valor del parámetro. Las referencias de parámetros son variables que contienen valores de parámetros extraídos en tiempo de ejecución (Google, 2022).

Existen diferentes formas de referenciar un parámetro:

- Referencia básica: Es el valor extraído por la entidad asociada.
    - $parameter-name
- Referencia para un valor original: Cuando el texto se compara con una entidad en particular, a menudo se convierte en texto que es más conveniente para el procesamiento.
    - $parameter-name.original
- Referencia para fechas ambiguas: Cuando un parámetro está asociado con la entidad del sistema @sys.date y el usuario final proporciona una fecha parcial (sin especificar el mes, el día y el año), Dialogflow coincidirá con la fecha más cercana en el futuro. Sin embargo, también se puede recuperar variaciones de esta fecha.
    - $parameter-name.partial
    - $parameter-name.recent

- Referencia para entidad compuesta: Las entidades compuestas son entidades que contienen otras subentidades.
    - $parameter-name.sub-entity-name
- Referencia para un contexto activo: Los contextos pueden servir como almacenamiento temporal para los valores de los parámetros.
    - #context-name.parameter-name
- Referencia para un parámetro de evento: Las intenciones normalmente coinciden cuando una expresión de usuario final coincide con una frase de entrenamiento de intención. Sin embargo, también se puede desencadenar intentos mediante eventos. Existen dos tipos de eventos los de plataforma que son eventos built-in proporcionados por la plataforma de integración y eventos personalizados que pueden ser invocados a través del fulfillment o la API.
    - #event-name.parameter-name

Figura 13. Ejemplo de parámetros en intent, slot filling (Google, 2022)



Fulfillment

Por defecto el agente responde a una intención coincidente con una respuesta estática. Si se está utilizando una de las opciones de integración, se puede proporcionar una respuesta más dinámica utilizando el fulfillment. Cuando se habilita el fulfillment Dialogflow responde a esa intención

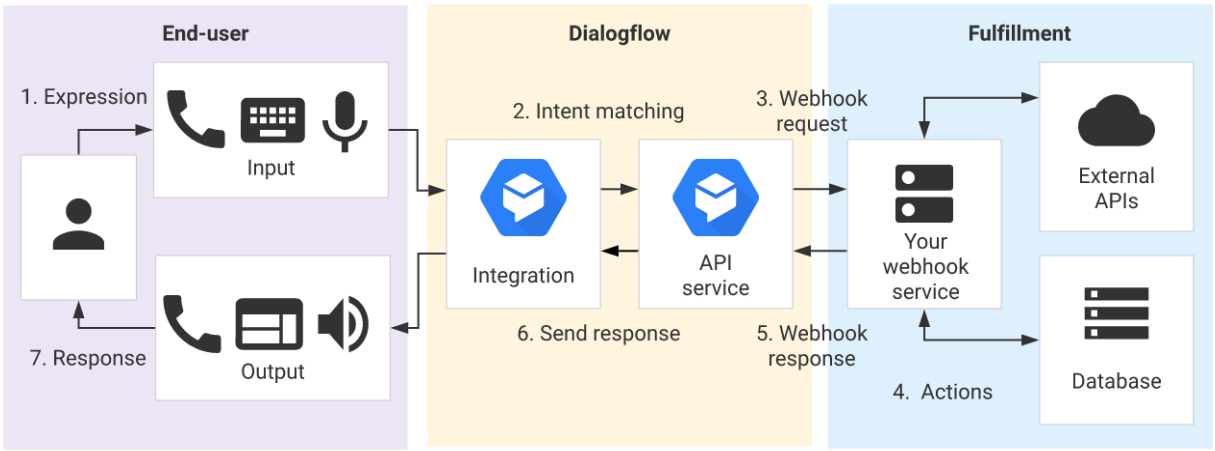
llamando a un servicio que será definido. Por ejemplo, para un chat de soporte para un salón de belleza, si un usuario final desea programar un corte de cabello para el viernes, el servicio puede verificar la base de datos y responder al usuario final con información de disponibilidad para ese día (Google, 2022).

Cada intención tiene una configuración para habilitar el fulfillment. Si una intención requiere alguna acción por parte del sistema o una respuesta dinámica, se debe habilitar el fulfillment de la intención. Si coincide una intención sin fulfillment habilitado, Dialogflow utiliza la respuesta estática que se definió para la intención. Cuando se compara una intención con fulfillment habilitado, Dialogflow envía una solicitud a su servicio de webhook con información sobre la intención coincidente. El sistema puede realizar cualquier acción requerida y responder a Dialogflow con información sobre cómo proceder. Cuando el fulfillment está habilitado, la respuesta estática que se definió para la intención solo se utiliza si el servicio de webhook falla (Google, 2022).

Un fulfillment puede contener cualquiera de los siguientes:

- Mensajes de respuesta estáticos.
- Webhook requiere respuestas dinámicas y/o acciones.
- Ajustes preestablecidos de parámetros para establecer o anular los valores de los parámetros.

Figura 14. Flujo de intent con fulfillment (Google, 2022)



Explicación:

El usuario final escribe o habla una expresión, Dialogflow hace coincidir la expresión del usuario final con una intención y extrae los parámetros. Dialogflow envía un mensaje de solicitud de webhook al servicio de webhook. Este mensaje contiene información sobre la intención coincidente, la acción, los parámetros y la respuesta definida para la intención. El servicio realiza acciones según sea necesario, como consultas a bases de datos o llamadas a API externas, etc. El servicio envía un mensaje de respuesta de webhook a Dialogflow este mensaje contiene la respuesta que debe enviarse al usuario final. Dialogflow envía la respuesta al usuario final. El usuario final ve o escucha la respuesta.

Integraciones

Por cada turno de conversación tiene lugar una interacción. Durante una interacción, un usuario final envía información a Dialogflow y Dialogflow envía una respuesta. Se tienen dos opciones al implementar el sistema para manejar interacciones: usar una integración o usar la API.

Ambos agentes de Dialogflow (ES y CX) soportan fulfillment e interacciones con la API, además pueden ser integrados con las siguientes aplicaciones:

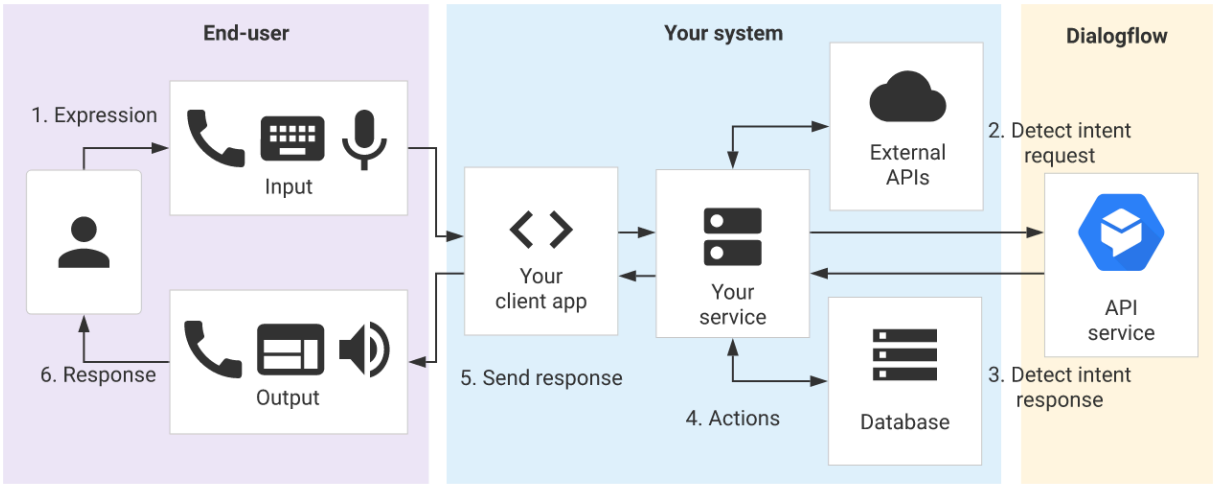
- Google Assistant
- Slack
- Facebook Messenger
- LINE
- Whatsapp
- Sistemas o sitios Web

Se puede crear un agente para una de estas plataformas, pero se debe usar una de las muchas opciones de integración. Las interacciones directas con el usuario final se manejan por Dialogflow, por lo que el developer solo debe concentrarse en crear el agente. Cada integración maneja las interacciones del usuario final de una manera específica de la plataforma.

Interacciones del usuario con la API

Si no se está utilizando una de las opciones de integración incorporadas, se debe escribir el código que interactúe directamente con el usuario final. También se debe interactuar directamente con la API de Dialogflow para cada turno de conversación para enviar expresiones de usuario final y recibir coincidencias de intenciones (Google, 2022).

Figura 15. Flujo de integración con la API (Google, 2022)



Explicación:

El usuario final escribe o habla una expresión. El servicio envía esta expresión de usuario final a Dialogflow en un mensaje de solicitud de intención de detección. Dialogflow envía un mensaje de respuesta de intención de detección al servicio. Este mensaje contiene información sobre la intención coincidente, la acción, los parámetros y la respuesta definida para la intención. El servicio realiza acciones según sea necesario, como consultas a bases de datos o llamadas a API externas. El servicio envía una respuesta al usuario final. El usuario final ve o escucha la respuesta.

Dialogflow CX

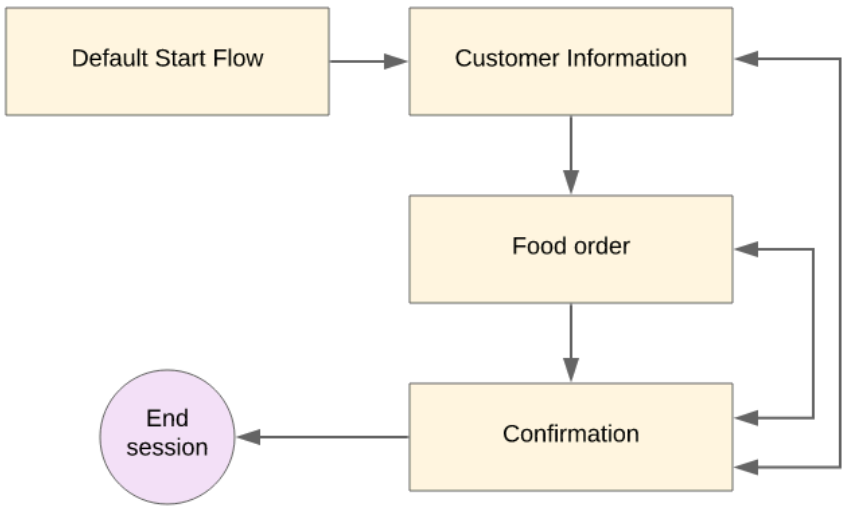
Este es un tipo de agente avanzado que es adecuado para agentes grandes o muy complejos. Los flujos y las páginas son los componentes básicos del diseño de conversaciones y los controladores de estado se utilizan para controlar las rutas de la conversación (Google, 2022).

El agente CX además de tener los mismos componentes que un agente ES posee los siguientes componentes adicionales:

Flow (Flujos)

Las conversaciones complejas a menudo involucran múltiples temas. Cada tema requiere múltiples turnos de conversación para que un agente adquiera la información relevante del usuario final. Los flujos se utilizan para definir estos temas y las rutas de conversación asociadas. Cada agente tiene un flujo llamado flujo de inicio predeterminado. Este flujo único puede ser todo lo que se necesita para un agente simple. Los agentes más complicados pueden requerir flujos adicionales y diferentes miembros del equipo de desarrollo pueden ser responsables de crear y mantener estos flujos, los flujos de Dialogflow CX tienen un propósito similar como un subagente (Google, 2022).

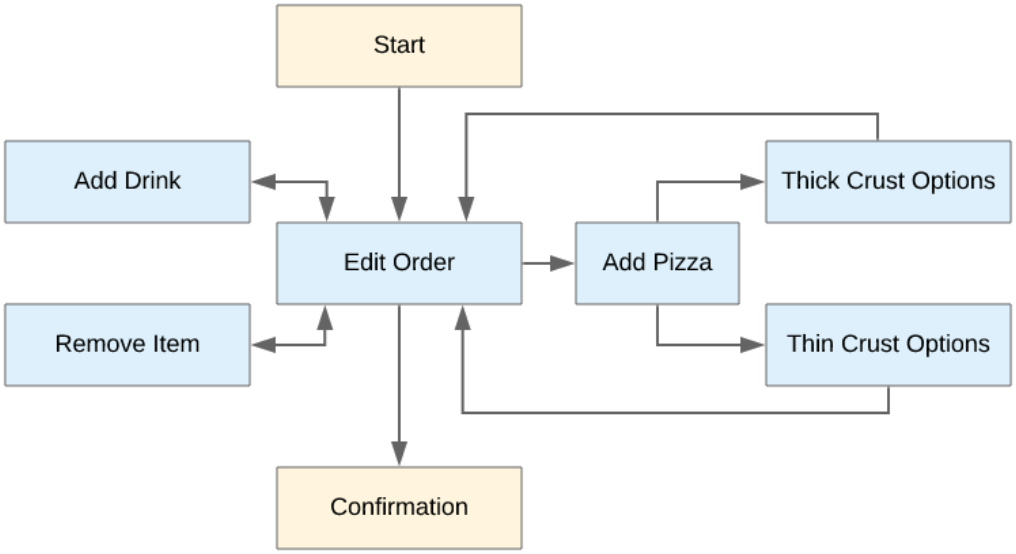
Figura 16. Ejemplo de flow, para un agente de pizza delivery (Google, 2022)



Pages (Páginas)

Una conversación (sesión) de Dialogflow CX se puede describir y visualizar como una máquina de estado. Los estados de una sesión CX están representados por páginas. Para cada flujo se definen muchas páginas, donde sus páginas combinadas pueden manejar una conversación completa sobre los temas para los que está diseñado el flujo. En un momento dado, exactamente una página es la página actual, la página actual se considera activa y el flujo asociado con esa página se considera activo. Cada flujo tiene una página de inicio especial. Cuando un flujo se activa inicialmente, la página de inicio se convierte en la página actual. Para cada turno de conversación, la página actual permanecerá igual o pasará a otra página (Google, 2022).

Figura 17. Ejemplo de pages, para un agente de pizza delivery (Google, 2022)



Forms (Formularios)

Para cada página se puede definir un formulario, que es una lista de parámetros que se deben recopilar del usuario final de la página. El agente interactúa con el usuario final durante varios turnos de conversación, hasta que haya recopilado todos los parámetros del formulario requerido, que también se conocen como parámetros de página. El agente recopila estos parámetros en el

orden definido en la página. Para cada parámetro de formulario obligatorio, también proporciona mensajes que el agente utiliza para solicitar esa información al usuario final. Este proceso se denomina llenado de formularios (Google, 2022). El llenado de los formularios para CX es similar al de ES.

Los parámetros de los formularios se definen en tiempo de diseño cuando se está creando la página. Las referencias de parámetros de formulario no se usan directamente. Solo puede verificarse el estado de llenado de los parámetros de formularios individuales o del formulario en su conjunto.

Intens (Intenciones)

Una intención clasifica la intención de un usuario final para un turno de conversación. En comparación con los intentos de ES, los intentos de CX se han simplificado para convertirlos en un recurso más reutilizable (Google, 2022).

Para CX los intent contiene la siguiente información:

- Training phrases (frases de entrenamiento).
- Parameters (parámetros).

State Handler (Controladores de estado)

Se utilizan para controlar la conversación mediante la creación de respuestas para los usuarios finales y/o la transición de la página actual. Para cada turno de conversación, los controladores son evaluados y pueden afectar la sesión (Google, 2022). Tiene 3 tipos de datos:

- Handler requirements (controlador de requisitos): Estos son los requisitos que deben cumplirse para que el controlador tenga algún efecto en la sesión. Se llama a un controlador cuando satisface sus requisitos y afecta la sesión de alguna manera (Google, 2022).
- Handler fulfillment (controlador de fulfillment): Si se llama a un controlador, se utiliza un cumplimiento opcional para crear respuestas para los usuarios finales. Estas respuestas se definen en datos de agente estático o se recuperan dinámicamente desde su servicio de webhook (Google, 2022).

- Handler transition target (controlador de objetivo de transición): Si se llama a un controlador, se utiliza un destino de transición opcional para cambiar la página actual. La página siguiente solo puede ser una página de inicio de flujo o una página dentro del flujo actualmente activo (Google, 2022).

Hay tres pasos para procesar un controlador de estado:

- Scope (Alcance): Un controlador debe estar dentro del alcance para tener algún efecto en la sesión. El alcance está determinado por si se aplica un controlador a un flujo, una página o un parámetro de formulario; y por si el flujo asociado está activo, la página asociada está activa o el agente está intentando completar el parámetro de formulario asociado (Google, 2022).
- Evaluation (Evaluación): Cada controlador en el scope se evalúa en orden. Si se cumplen los requisitos de un controlador, para la evaluación.
- Call (llamada): Si un controlador está dentro del alcance y pasa la evaluación. Se llama a cualquier cumplimiento asociado y se aplica cualquier objetivo de transición asociado a la sesión.

Fulfillment

El fulfillment de ES se limita a conectarse a un servicio de webhook. El alcance del cumplimiento se ha aumentado para CX, por lo que cubre todos los tipos de avisos y respuestas.

Integraciones

Adicionalmente el agente CX tiene estas integraciones adicionales: AudioCodes, Avaya, Voximplant. Son creadas por socios de Google en colaboración con Google. Google no brinda soporte para estas integraciones. Si se desean utilizar se debe poner en contacto con el propietario de la integración para obtener asistencia (Google, 2022).

Enfoque técnico de tecnologías ofrecidas por microsoft

Bot Framework Composer es un canvas de código abierto para que los desarrolladores y equipos multidisciplinarios puedan diseñar y crear experiencias conversacionales con Language Understanding, QnA Maker y una composición sofisticada de respuestas de bot (Language Generation), que permite crear un bot con la capacidad de hablar, escuchar, comprender y aprender de los usuarios con la integración nativa de Azure Cognitive Services. Azure Bot Service permite crear bots inteligentes de nivel empresarial con propiedad y control de los datos. Da la capacidad de crear un simple bot de preguntas y respuestas o crear un asistente virtual sofisticado (Microsoft, 2022).

Bot Framework Composer está basado en Bot Framework SDK y es un IDE de código abierto para que los desarrolladores creen, prueben y administren experiencias conversacionales. Proporciona un poderoso lienzo de creación visual que permite la creación de diálogos, modelos de comprensión del lenguaje, bases de conocimiento de QnAMaker y respuestas de generación de lenguaje desde un solo lienzo y permite que estas experiencias se amplíen con código para tareas más complejas, como la integración del sistema. Las experiencias resultantes se pueden probar en Composer y aprovisionar en Azure junto con los recursos dependientes. Composer está disponible como aplicación de escritorio para Windows, macOS y Linux. Si la aplicación de escritorio no se adapta también se puede compilar Composer desde el origen o alojar Composer en la nube (Microsoft, 2022).

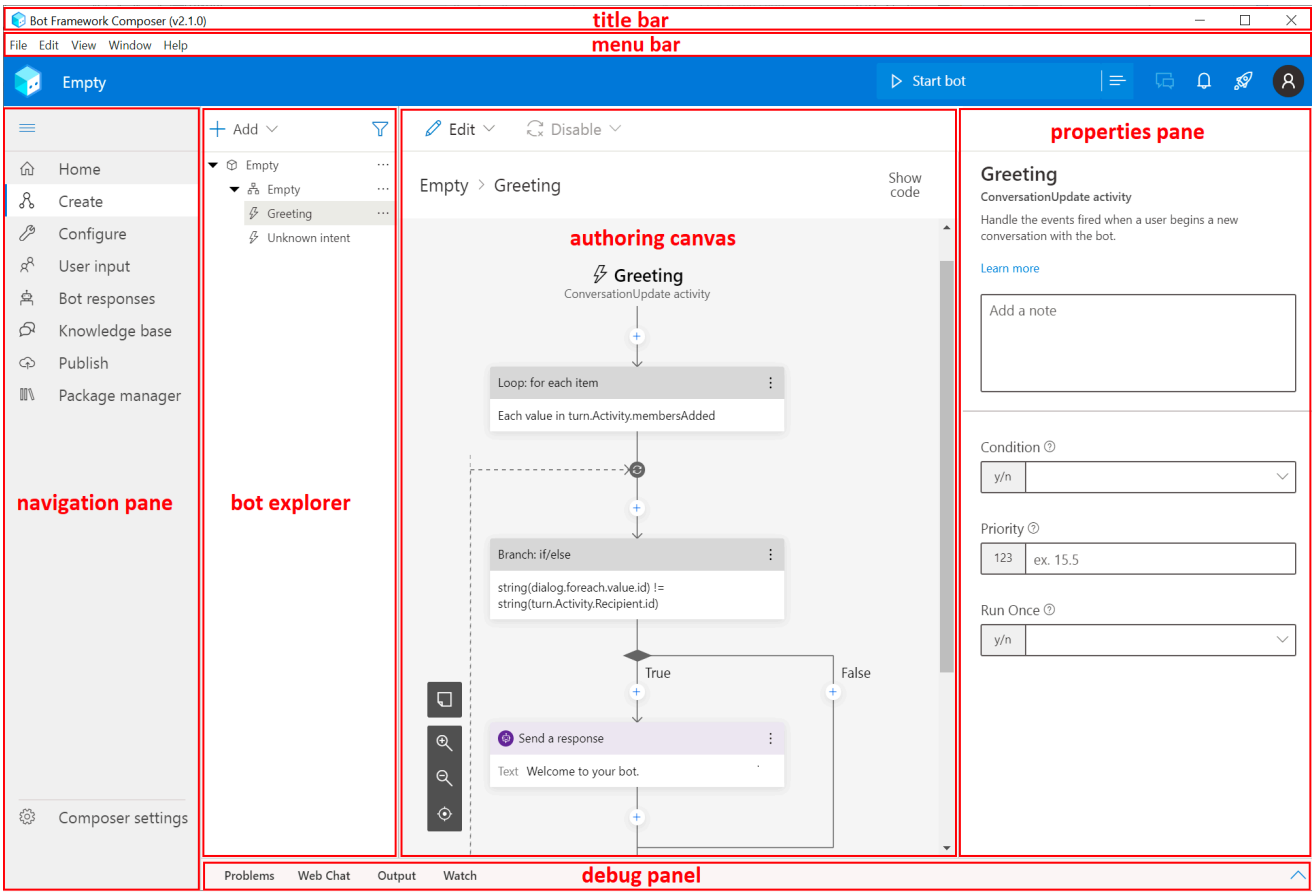
Composer es un lienzo de edición visual para construir bots que permite realizar las siguientes acciones:

- Crear un nuevo bot usando una plantilla, que incorpora las capacidades del Asistente virtual directamente en Composer.
- Agregar capacidades de comprensión del lenguaje natural al Bot mediante LUIS y QnA.
- Crear texto y si es necesario, respuestas de variación de voz para el Bot utilizando plantillas de generación de lenguaje.
- Crear bots en varios idiomas.
- Probar directamente dentro de Composer utilizando un Web Chat incorporado.

- Publicar bots en Azure App Service y Azure Functions.
- Ampliar Power Virtual Agents con Composer (versión preliminar).
- Integrar servicios externos como la base de conocimiento de QnA Maker.
- Importar y exportar activos de diálogo para compartir con otros desarrolladores.
- El administrador de paquetes proporciona una variedad de activos y códigos conversacionales reutilizables creados por Microsoft y terceros. Estos activos pueden agregar funcionalidad al proyecto.
- Hacer que cualquier Bot esté disponible como skill para que otros Bots lo llamen.
- Conectarse a una skill.
- Ampliar el canvas de creación de diálogos creando acciones personalizadas.
- Integrar Orchestrator.
- Host Composer en la nube.
- Extender Composer con plugins.

(da Silva, ¿Cuántos niveles de soporte técnico deben tener las empresas?, 2021) (da Silva, ¿Que es un ticket de soporte?, 2021; innovacionate.com, 2020; Portillo, 2020).

Figura 18. Interfaz gráfica de Bot Framework V2 (Microsoft, 2022)



Bot Framework tiene los siguientes componentes:

Adaptive dialogs (diálogos adaptables)

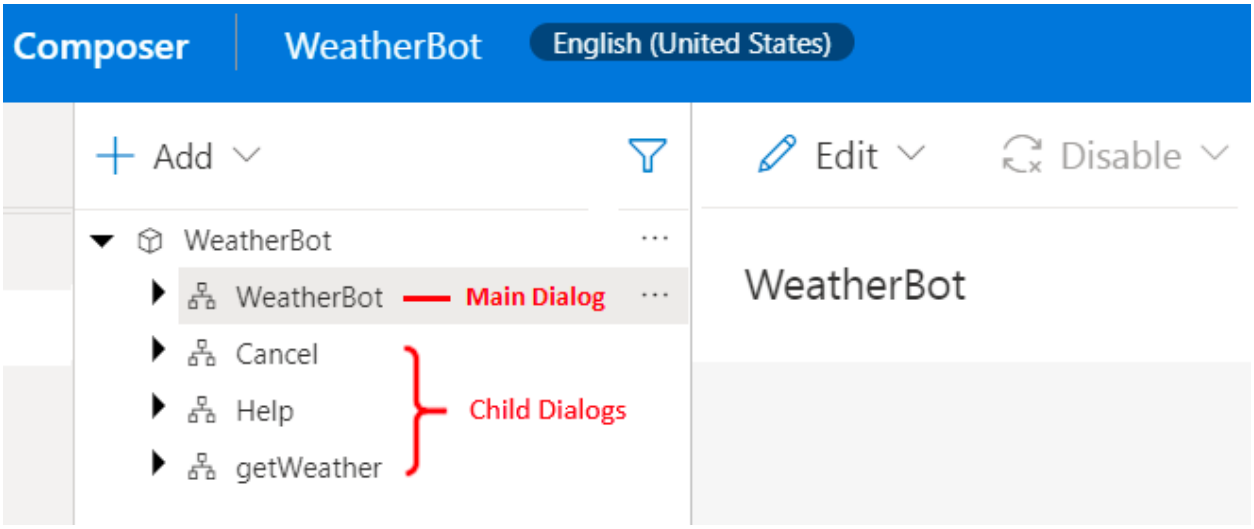
El software conversacional moderno tiene muchos componentes diferentes. Bot Framework Composer integra estas piezas en cuadros de diálogo, una interfaz única para construir los componentes básicos de la funcionalidad del bot. Cada cuadro de diálogo representa una parte de la funcionalidad del bot y contiene instrucciones sobre cómo reacciona el bot a la entrada (Microsoft, 2022).

Los diálogos pueden incluir lógica comercial personalizada, llamadas a API en la nube, datos de capacitación para sistemas de procesamiento de lenguaje y contenido real utilizado en las conversaciones con los usuarios finales del bot. Los bots simples tendrán solo unos pocos cuadros

de diálogo. Los bots sofisticados pueden tener docenas o cientos de diálogos individuales. Los diálogos son componentes funcionales que se ofrecen en una interfaz visual que reduce la necesidad de escribir código. El sistema de diálogo admite la creación de un modelo extensible que integra todos los componentes básicos de la funcionalidad de un bot (Microsoft, 2022).

Existen dos tipos de diálogos en Composer: diálogo principal (main dialog) y diálogo secundario (child dialog). El diálogo principal se inicia de forma predeterminada cuando se crea un nuevo bot. Se puede crear uno o más diálogos secundarios para mantener el sistema de diálogo organizado. Cada bot tiene un diálogo principal y puede tener cero o más diálogos secundarios (Microsoft, 2022).

Figura 19. Ejemplo de diálogo principal y secundario (Microsoft, 2022)



Explicación:

En tiempo de ejecución el cuadro de diálogo principal entra en acción y se convierte en el cuadro de diálogo activo, lo que activa los controladores de eventos con las acciones que se definieron durante la creación del bot. A medida que fluye la conversación, el cuadro de diálogo principal puede llamar a un cuadro de diálogo secundario y un cuadro de diálogo secundario puede, a su vez, llamar al cuadro de diálogo principal o a otros cuadros de diálogo secundarios.

Un dialog está compuesto por las siguientes partes:

Recognizers (reconocedores)

Un reconocedor interpreta lo que el usuario quiere en función de la entrada. Cuando se invoca un diálogo, su reconocedor comenzará a procesar el mensaje e intentará extraer la intención principal y cualquier valor de entidad que incluya el mensaje. Después de procesar el mensaje, tanto la intención como los valores de la entidad se pasan a los activadores del cuadro de diálogo. Actualmente, Composer admite tres reconocedores: el reconocedor LUIS, que es el predeterminado, el reconocedor de expresiones regulares y el reconocedor personalizado que le permite usar un propio reconocedor personalizado. Solo se puede elegir un reconocedor por cuadro de diálogo o elegir no tener ningún reconocedor. Los reconocedores le dan al bot la capacidad de comprender y extraer información significativa de la entrada del usuario. Todos los reconocedores emiten eventos cuando el reconocedor detecta una intención (o extrae entidades) de una expresión de usuario dada. El reconocedor de un diálogo no siempre entra en juego cuando se invoca un diálogo. Depende de cómo se diseñe el sistema de diálogo (Microsoft, 2022).

Triggers

Cada cuadro de diálogo incluye uno o más controladores de eventos llamados triggers. Cada trigger contiene una o más acciones. Los triggers son reglas que le dicen al bot cómo procesar los mensajes entrantes. Los triggers se utilizan para definir varios comportamientos de bot, desde cómo cumplir con una solicitud de usuario hasta cómo manejar interrupciones y cómo manejar eventos definidos por el desarrollador. Existen diferentes tipos de disparadores que funcionan de manera similar y en algunos casos, se pueden intercambiar (triggers de intención, triggers de eventos de diálogo, triggers de actividades, eventos personalizados) (Microsoft, 2022).

Actions (acciones)

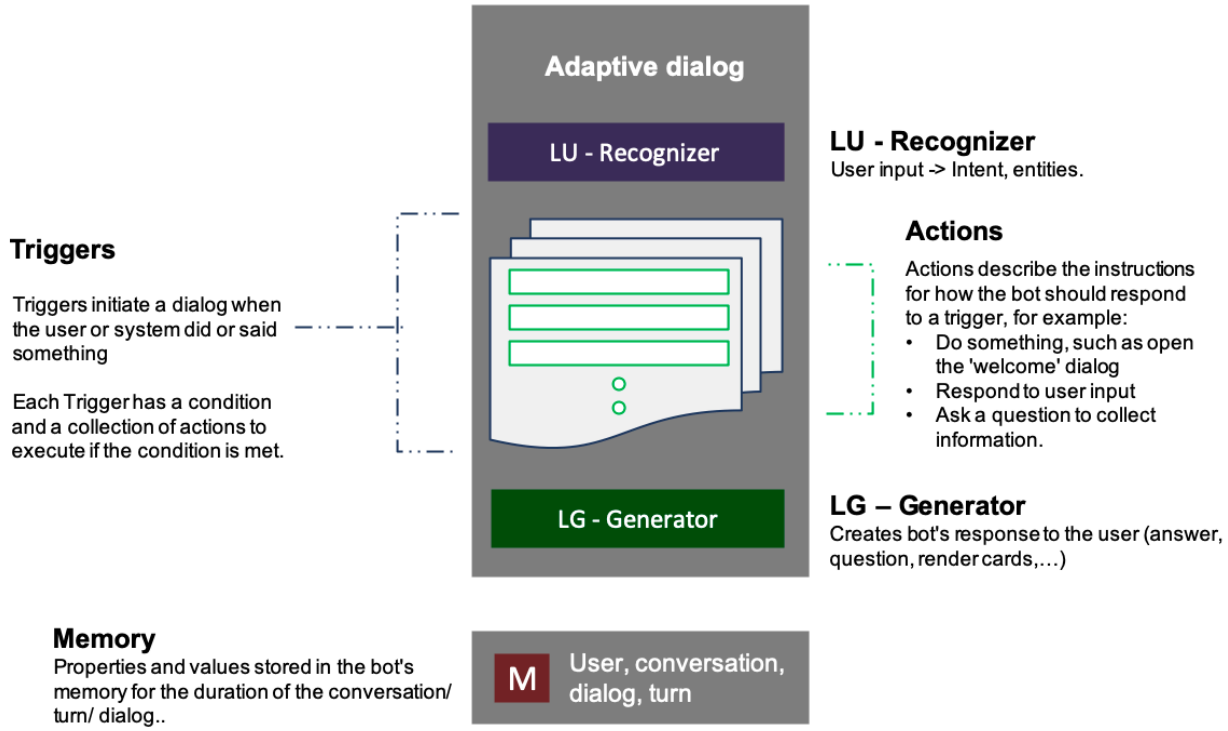
Los triggers contienen una serie de acciones que el bot llevará a cabo para cumplir con la solicitud de un usuario. Las acciones son las instrucciones que el bot ejecutará cuando el cuadro de diálogo reciba cualquier evento que tenga un trigger definido. Acciones como enviar mensajes, responder a las preguntas de los usuarios utilizando una base de conocimiento, hacer cálculos y realizar tareas

computacionales para el usuario. La ruta que sigue el bot a través de un diálogo puede bifurcarse y hacer un bucle. El bot puede hacer o responder preguntas, validar entradas, manipular y almacenar valores en la memoria y tomar decisiones (Microsoft, 2022).

Memory (memoria)

Las propiedades y valores son almacenados en la memoria del bot, esta tiene el tiempo de duración de la conversación, turno o diálogo.

Figura 20. Anatomía de un dialog (Microsoft, 2022)



Language understanding (Comprensión del lenguaje)

Un bot puede utilizar la comprensión del lenguaje para interpretar la entrada de un usuario y determinar contextualmente qué hacer a continuación en una conversación. Se puede asociar un reconocedor (recognizer) con un cuadro de diálogo y agregar datos de entrenamiento para permitir que el bot extraiga intenciones(intents) y entidades(entities). Se puede usar triggers para definir cómo responde el bot a intenciones específicas (Microsoft, 2022).

La comprensión del idioma es un componente central de Composer que permite a los desarrolladores y diseñadores entrenar modelos de comprensión del idioma directamente en el contexto de la edición de un diálogo. A medida que los diálogos se editan en Composer, los desarrolladores pueden agregar continuamente capacidades de lenguaje natural a sus bots (Microsoft, 2022).

La comprensión del lenguaje en Bot Framework Composer contiene los siguientes elementos:

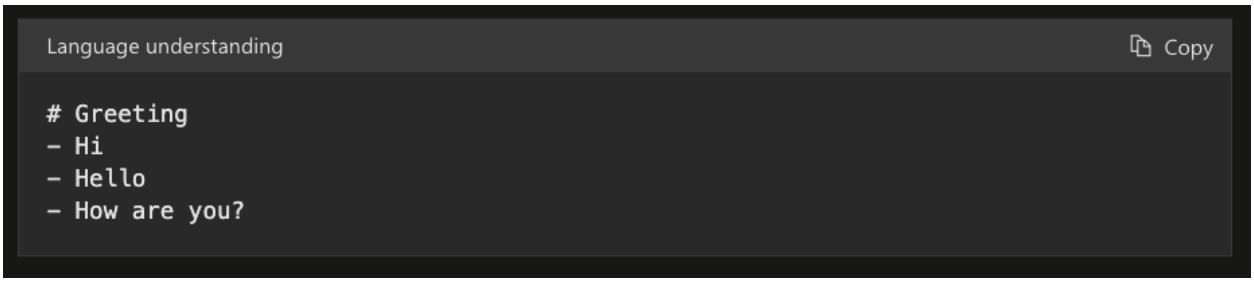
Utterances (Enunciados)

Son entradas de los usuarios y pueden tener muchas variaciones. Dado que las expresiones no siempre están bien formadas, se debe proporcionar expresiones de ejemplo para intenciones específicas a fin de entrenar a los bots para que reconozcan las intenciones de diferentes expresiones. Al hacer esto los bots tendrán algo de "inteligencia" para comprender los lenguajes humanos (Microsoft, 2022).

Intents (Intenciones)

Son categorías o clasificaciones de las intenciones del usuario. Una intención representa una acción que el usuario desea realizar. Es un propósito u objetivo expresado en la entrada del usuario, cómo reservar un vuelo, pagar una factura o encontrar un artículo de noticias. El developer define y nombra las intenciones que corresponden a estas acciones (Microsoft, 2022).

Figura 21. Ejemplo de un archivo .lu que representa una intención de saludo (Microsoft, 2022)



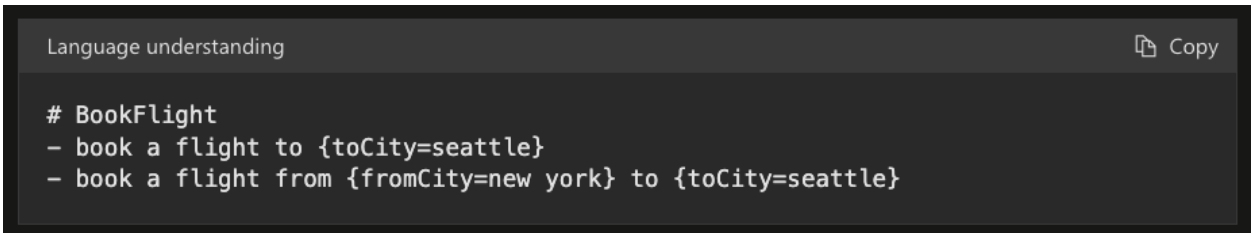
Explicación:

#<intent-name> define una nueva intención. Cada línea después de la definición de la intención contiene una declaración de ejemplo que describe esa intención. Se pueden unir varias definiciones de intención en un editor de comprensión de idiomas en Composer. Cada sección se identifica mediante la notación #<intent-name>. Las líneas en blanco se omiten al analizar el archivo.

Entities (entidades)

Las entidades son una colección de objetos, cada uno de los cuales consta de datos extraídos de un enunciado, como lugares, horas y personas. Tanto las entidades como las intenciones son datos importantes extraídos de las declaraciones. Un enunciado puede incluir cero o más entidades, mientras que un enunciado generalmente representa una intención. En Composer, todas las entidades se definen y gestionan inline (Microsoft, 2022).

Figura 22. Ejemplo de un archivo .lu utilizando entidades (Microsoft, 2022)



Explicación:

Las entidades en el formato de archivo .lu se indican mediante {\<entityName\>=\<labelled value\>}. El ejemplo anterior muestra la definición de una intención BookFlight con dos expresiones de ejemplo y dos definiciones de entidad: toCity y fromCity. Cuando se activa, si se puede identificar una ciudad de destino, el nombre de la ciudad estará disponible como @toCity dentro de las acciones activadas o una ciudad de salida con @fromCity como valores de entidad disponibles. Los valores de entidad se pueden usar directamente en expresiones y plantillas LG, o se pueden almacenar en una propiedad en la memoria para su uso posterior.

LUIS

Es un servicio de IA conversacional alojado en la nube que aplica inteligencia de aprendizaje automático personalizado al texto conversacional en lenguaje natural del usuario para predecir el

significado general y extraer información relevante y detallada. Este servicio ofrece una serie de ventajas como (Microsoft, 2022):

- Simplicidad: Ya que no se requiere experiencia interna en IA o cualquier conocimiento previo de aprendizaje automático. Con solo unos pocos clics, se puede crear una aplicación de IA conversacional. Se puede crear una aplicación personalizada siguiendo una guía de inicio rápido o se puede usar una de las aplicaciones de dominio pre diseñadas.
- Seguridad privacidad y cumplimiento: Respaldado por la infraestructura de Azure, LUIS ofrece seguridad, privacidad y cumplimiento a nivel empresarial. Los datos siguen siendo de la empresa, se puede eliminar los datos en cualquier momento además los datos están encriptados mientras están almacenados.
- Integración: Se integra fácilmente con otros servicios de Microsoft, como Microsoft Bot Framework, QnA Maker y Speech Service.

LUIS junto Azure Bot Service permiten a los desarrolladores crear interfaces conversacionales para varios escenarios como banca, viajes y entretenimiento, permite crear Bot informativos, permite crear interfaces de conversación fluidas con todos los dispositivos accesibles por Internet (Microsoft, 2022).

Language generator (generador de lenguaje)

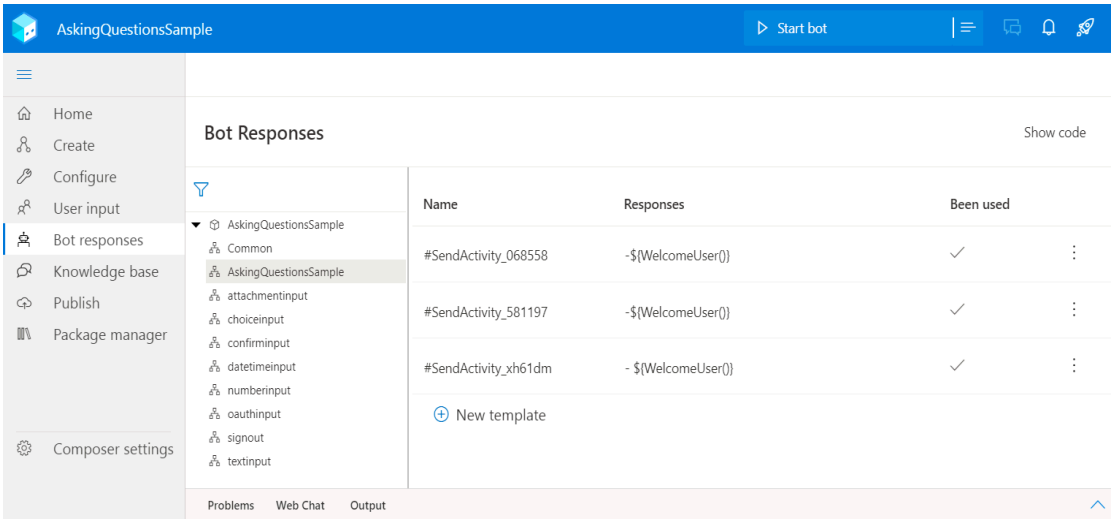
El generador de lenguaje permite definir múltiples variaciones de una frase, ejecutar expresiones simples basadas en el contexto y consultar la memoria conversacional. A medida que el bot realiza acciones y envía mensajes, se utiliza un generador de lenguaje para crear esos mensajes a partir de variables y plantillas. Los generadores de idiomas pueden crear componentes reutilizables, mensajes variables, macros y mensajes dinámicos que son gramaticalmente correctos (Microsoft, 2022).

Con los generadores de lenguaje se puede realizar las siguientes tareas:

- Incluir elementos dinámicos en los mensajes.
- Generar listas, pronombres, artículos gramaticalmente correctos.
- Proporcionar una variación sensible al contexto en los mensajes.

- Crear archivos adjuntos de tarjetas adaptables.
- Proporcionar variaciones de voz para cada respuesta, incluidas las modificaciones del lenguaje de marcado de síntesis de voz (SSML).
- Separar la lógica empresarial de la presentación.
- Crear tarjetas, acciones sugeridas y archivos adjuntos utilizando una plantilla de respuesta estructurada.

Figura 23. Generador de lenguaje (Microsoft, 2022)



QnA Maker

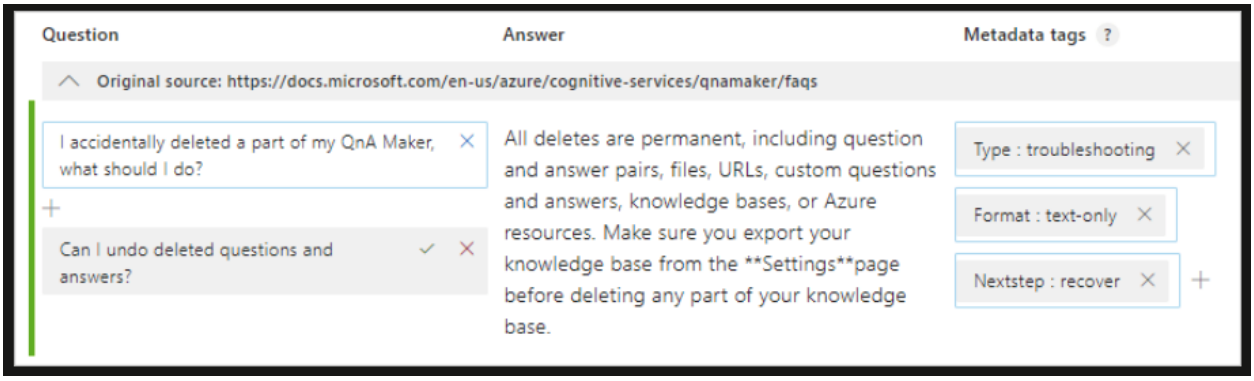
QnA Maker es un servicio de procesamiento de lenguaje natural (NLP) basado en la nube que permite crear una capa de conversación natural sobre los datos. Se utiliza para encontrar la respuesta más adecuada para cualquier entrada de la base de conocimiento personalizada de información. Se utiliza comúnmente para crear aplicaciones cliente conversacionales, que incluyen aplicaciones de redes sociales, chatbots y aplicaciones de escritorio habilitadas para voz. Todos los datos del cliente (respuestas de preguntas y registros de chat) se almacenan en la región en la que el cliente implementa las instancias de servicio dependientes (Microsoft, 2022).

QnA Maker importa el contenido a una base de conocimiento de pares de preguntas y respuestas. El proceso de importación extrae información sobre la relación entre las partes del contenido

estructurado y semi estructurado para implicar relaciones entre los pares de preguntas y respuestas. Se pueden editar estos pares de preguntas y respuestas o agregar nuevos pares. El contenido del par de preguntas y respuestas incluye (Microsoft, 2022):

- Todas las formas alternativas de la pregunta.
- Etiquetas de metadatos utilizadas para filtrar las opciones de respuesta durante la búsqueda.
- Indicaciones de seguimiento para continuar con el refinamiento de la búsqueda.

Figura 24. Ejemplo del contenido para preguntas y respuestas (Microsoft, 2022)



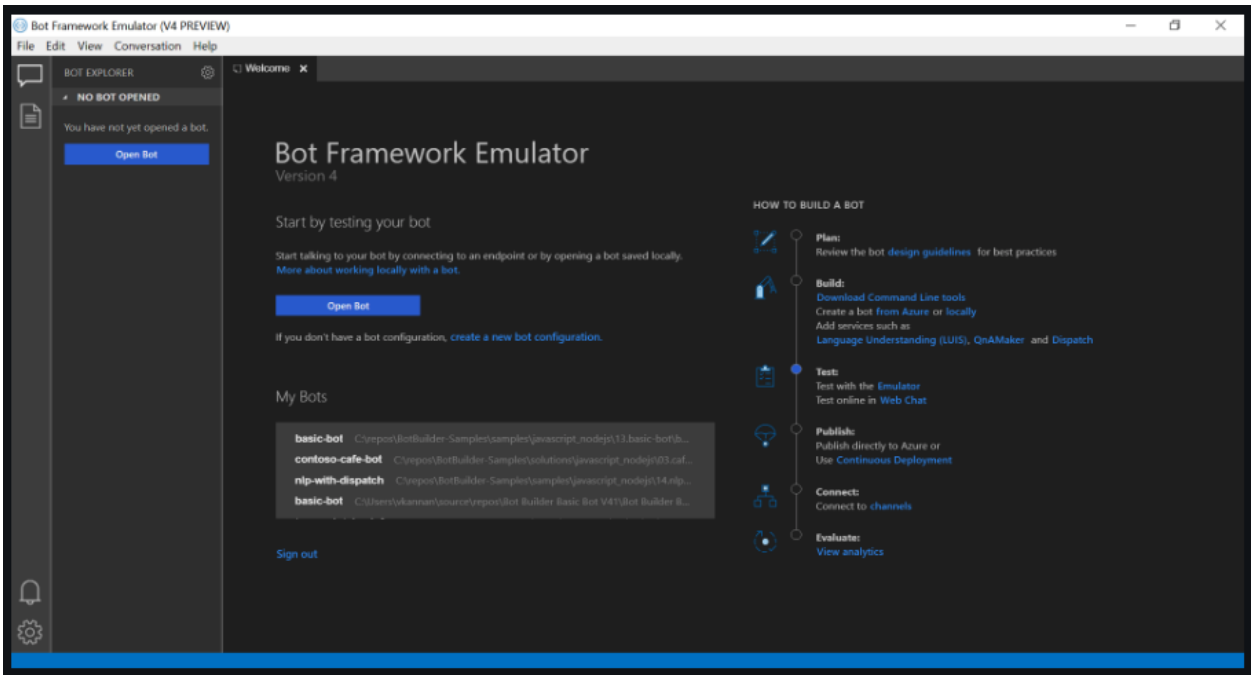
Emulador de Bot Framework

Es una aplicación de escritorio que permite a los desarrolladores de bots probar y depurar bots creados con Composer. Esta herramienta permite escenarios más avanzados como la autenticación, que la función de chat web integrado de Composer no admite en este momento (Microsoft, 2022).

Microsoft Bot Framework Emulator es una herramienta multiplataforma que puede ser instalada en Windows, OS X y Linux, este emulador ayuda en las siguientes tareas:

- Probar y depurar bots conversacionales creados con Microsoft Bot Framework SDK
- Conectar y realizar un seguimiento de los servicios de los que depende el bot.
- Refinar las aplicaciones de LUIS y QnA Maker de las que depende el bot.

Figura 25. Emulador de Bot Framework (Microsoft, 2022)



Enfoque técnico de nodejs

Node.js es un entorno de ejecución de código abierto y multiplataforma de JavaScript que ejecuta el motor de JavaScript V8, el núcleo de Google Chrome, fuera del navegador.

Una aplicación de Node.js se ejecuta en un solo proceso, sin crear un nuevo subproceso para cada solicitud, proporciona un conjunto de primitivas de I/O asíncronas en su biblioteca estándar que evita que se bloquee el código JavaScript. Cuando Node.js realiza una operación de I/O como leer de la red, acceder a una base de datos o al sistema de archivos, en lugar de bloquear el subproceso y desperdiciar ciclos de CPU en espera, Node.js reanudará las operaciones cuando regrese la respuesta. Esto permite que Node.js maneje miles de conexiones simultáneas con un solo servidor sin presentar la carga de administrar la concurrencia de subprocesos, lo que podría ser una fuente importante de errores (nodejs.org, 2022).

La ventaja más grande de nodejs es que millones de developers que escriben JavaScript para el navegador ahora pueden escribir el código del lado del servidor además del código del lado del

cliente sin necesidad de aprender un lenguaje completamente diferente. En Node.js los nuevos estándares de ECMAScript se pueden usar sin problema, ya que no se tiene que esperar a que todos los usuarios actualicen sus navegadores, el developer es el encargado de decidir qué versión de ECMAScript usar cambiando la versión de Node.js y también puede habilitar funciones experimentales específicas ejecutando Node.js con banderas. Nodejs tiene un gran número de librerías alojadas en npm que aloja más de 1,000,000 de paquetes de código abierto que pueden ser usados libremente.

Figura 26. Ejemplo de "Hello World" con Nodejs (nodejs.org, 2022)

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

¿Qué es ECMAScript?

ECMAScript es un lenguaje de programación orientado a objetos para realizar cálculos y manipular objetos computacionales dentro de un entorno host. ECMAScript no pretende ser computacionalmente autosuficiente; de hecho, no hay disposiciones en la especificación para la entrada de datos externos o la salida de resultados calculados. Se diseñó originalmente para usarse como lenguaje de secuencias de comandos, pero se ha vuelto ampliamente utilizado como lenguaje de programación de propósito general. Un lenguaje de secuencias de comandos es un lenguaje de programación que se utiliza para manipular, personalizar y automatizar las instalaciones de un

sistema existente. En dichos sistemas, la funcionalidad útil ya está disponible a través de una interfaz de usuario y el lenguaje de secuencias de comandos es un mecanismo para exponer esa funcionalidad al control del programa. De esta forma, se dice que el sistema existente proporciona un entorno anfitrión de objetos e instalaciones, lo que completa las capacidades del lenguaje de programación. Un lenguaje de secuencias de comandos está destinado a ser utilizado por programadores profesionales y no profesionales (ecma International, 2022).

ECMAScript se diseñó originalmente para ser un lenguaje de secuencias de comandos web, proporcionando un mecanismo para animar las páginas web en los navegadores y realizar el cálculo del servidor como parte de una arquitectura cliente-servidor basada en la web. Ahora se usa para proporcionar capacidades básicas de secuencias de comandos para una variedad de entornos de host. Y su uso ha ido más allá de la simple secuencia de comandos, ahora se usa para el espectro completo de tareas de programación en muchos entornos y escalas diferentes. A medida que se ha expandido el uso de ECMAScript, también lo han hecho las funciones y facilidades que proporciona. ECMAScript es ahora un lenguaje de programación de propósito general con todas las funciones (ecma International, 2022).

Que ECMAScript esté basado en objetos significa que los objetos proporcionan el lenguaje básico y las funciones de host. Un programa ECMAScript es un grupo de objetos que se comunican. En ECMAScript, un objeto es una colección de cero o más propiedades, cada una con atributos que determinan cómo se puede usar cada propiedad; por ejemplo, cuando el atributo Escribible para una propiedad se establece en falso, cualquier intento del código ECMAScript ejecutado para asignar una propiedad diferente el valor de la propiedad falla (ecma International, 2022).

Las propiedades son contenedores que contienen otros objetos, valores primitivos o funciones. Un valor primitivo es miembro de uno de los siguientes tipos integrados: Undefined, Null, Boolean, Number, BigInt, String y Symbol; un objeto es un miembro del tipo incorporado Objeto; y una función es un objeto invocable. Una función que está asociada con un objeto a través de una propiedad se llama método (ecma International, 2022).

ECMAScript define una colección de objetos integrados que completan la definición de entidades de ECMAScript. Estos objetos integrados incluyen el objeto global; objetos que son fundamentales para la semántica de tiempo de ejecución del lenguaje, incluidos objetos, funciones, booleanos, símbolos y varios objetos de error; objetos que representan y manipulan valores numéricos, incluidas operaciones matemáticas, números y fechas; los objetos de procesamiento de texto String y RegExp; objetos que son colecciones indexadas de valores que incluyen Array y nueve tipos diferentes de Typed Arrays cuyos elementos tienen una representación de datos numéricos específica; colecciones con clave que incluyen objetos Map y Set; objetos que admiten datos estructurados, incluido el objeto JSON, ArrayBuffer, SharedArrayBuffer y DataView; objetos que admiten abstracciones de control, incluidas funciones de generador y objetos Promise; y objetos de reflexión, incluidos Proxy y Reflect (ecma International, 2022).

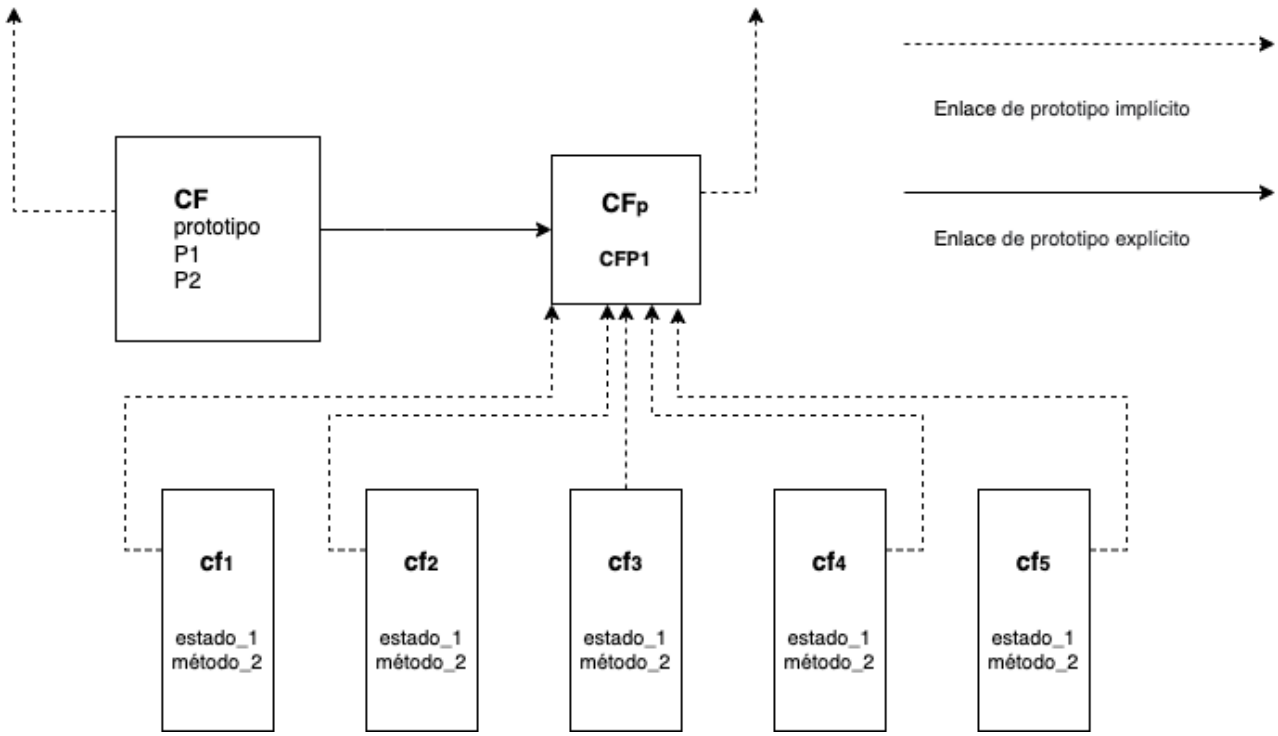
ECMAScript también define un conjunto de operadores integrados. Los operadores ECMAScript incluyen varias operaciones unarias, operadores multiplicativos, operadores aditivos, operadores de desplazamiento bit a bit, operadores relacionales, operadores de igualdad, operadores binarios bit a bit, operadores lógicos binarios, operadores de asignación y el operador de coma (ecma International, 2022).

Los programas grandes de ECMAScript están respaldados por módulos que permiten que un programa se divida en múltiples secuencias de sentencias y declaraciones. Cada módulo identifica explícitamente las declaraciones que deben utilizar y que deben proporcionar otros módulos y cuáles de sus declaraciones están disponibles para que las utilicen los otros módulos. La sintaxis de ECMAScript se relaja para permitir que sirva como un lenguaje de secuencias de comandos fácil de usar. Por ejemplo, no se requiere que una variable tenga declarado su tipo ni los tipos asociados con propiedades y las funciones definidas no necesitan que sus declaraciones aparezcan textualmente antes de llamarlas (ecma International, 2022).

Node.js se basa en versiones modernas de V8. Y se mantiene actualizado con las últimas versiones de ese motor, asegurándose de que las nuevas características de la especificación ECMA-262 de JavaScript se integren a Node.js (MDN web docs, 2022).

V8 es el nombre del motor de JavaScript que impulsa Google Chrome (es el motor que toma nuestro JavaScript y lo ejecuta mientras navegas con Chrome). V8 proporciona el entorno de tiempo de ejecución en el que se ejecuta JavaScript mientras el navegador proporciona el DOM y las demás API de la plataforma web. Lo bueno es que el motor de JavaScript es independiente del navegador en el que está alojado y es esta característica clave la que permitió el surgimiento de Node.js. Es así como V8 fue elegido para ser el motor que impulsó Node.js en 2009 y a medida que la popularidad de Node.js explotó, V8 se convirtió en el motor que ahora impulsa una cantidad increíble de código del lado del servidor escrito en JavaScript (MDN web docs, 2022).

Figura 27. Relación entre objeto y prototipo (Ecma International, 2022)



Explicación:

En un lenguaje orientado a objetos basado en clases, en general, el estado lo llevan las instancias, los métodos los llevan las clases y la herencia es solo de estructura y comportamiento. En ECMAScript, el estado y los métodos los llevan los objetos, mientras que la estructura, el

comportamiento y el estado se heredan. Todos los objetos que no contienen directamente una propiedad particular que contiene su prototipo comparten esa propiedad y su valor (ecma International, 2022).

¿Qué es Javascript?

JavaScript (JS) es un lenguaje de programación ligero, interpretado o compilado just-in-time con funciones de primera clase. Si bien es más conocido como el lenguaje de secuencias de comandos para páginas web, muchos entornos que no son de navegador también lo utilizan, como Node.js, Apache CouchDB y Adobe Acrobat. JavaScript es un lenguaje dinámico basado en prototipos, multiparadigma, de un solo subproceso, que admite estilos orientados a objetos, imperativos, declarativos (por ejemplo, programación funcional) (MDN web docs, 2022).

JavaScript se ejecuta en el lado del cliente web y se utiliza más comúnmente como parte de las páginas web, cuyas implementaciones permiten que el script del lado del cliente interactúe con el usuario y cree páginas dinámicas. Es un lenguaje de programación interpretado con capacidades orientadas a objetos que se puede utilizar para diseñar/programar cómo se comportan las páginas web ante la ocurrencia de un evento. JavaScript es un lenguaje de secuencias de comandos fácil de aprender y también potente, ampliamente utilizado para controlar el comportamiento de la página web.

JavaScript se conoció primero como LiveScript, pero Netscape cambió su nombre a JavaScript, JavaScript hizo su primera aparición en Netscape 2.0 en 1995 con el nombre LiveScript. El núcleo de propósito general del lenguaje se ha integrado en Netscape, Internet Explorer y otros navegadores web utilizando las especificaciones de ECMAScript ECMA-262 y ECMA-402 para definir la versión estándar del core del lenguaje (MDN web docs, 2022).

JavaScript es un lenguaje de secuencias de comandos dinámicos que admite la construcción de objetos basados en prototipos y que puede funcionar como un lenguaje procedimental y orientado a objetos. Los objetos se crean mediante programación en JavaScript, adjuntando métodos y propiedades a objetos vacíos en tiempo de ejecución, a diferencia de las definiciones de clases

sintácticas comunes en lenguajes compilados como C + + y Java. Una vez que se ha construido un objeto, se puede utilizar como modelo (o prototipo) para crear objetos similares. Las capacidades dinámicas de JavaScript incluyen la construcción de objetos en tiempo de ejecución, listas de parámetros variables, variables de funciones, creación de secuencias de comandos dinámicas, introspección de objetos (a través de for... in) y recuperación de código fuente (MDN web docs, 2022).

Utilizar JavaScript del lado del cliente (cliente-side) es la forma más común del lenguaje. La secuencia de comandos debe incluirse o hacer la referencia en un documento HTML para que el navegador interprete el código. Esto significa que una página web no necesita ser un HTML estático, sino que puede incluir programas que interactúan con el usuario, controlando el navegador y creando dinámicamente contenido HTML. Client-side proporciona muchas ventajas sobre los scripts CGI tradicionales del lado del servidor. Por ejemplo, se puede usar JavaScript para verificar si el usuario ha ingresado una dirección de correo electrónico válida en un campo de formulario. El código JavaScript se ejecuta cuando el usuario envía el formulario y sólo si todas las entradas son válidas, se enviarán al servidor web. adicionalmente JavaScript se puede utilizar para atrapar eventos iniciados por el usuario, como clics en botones, navegación por enlaces y otras acciones que el usuario inicia explícita o implícitamente.

Ventajas y Desventajas de Javascript

Ventajas:

- Velocidad, es rápido ya que se ejecuta inmediatamente en el navegador.
- Simplicidad, La sintaxis de JavaScript está inspirada en lenguajes de programación conocidos anteriormente por tanto es relativamente sencillo de aprender.
- Popularidad, ya que es utilizado por toda la web y en los últimos años se ha incrementado su uso en el backend lo que ayuda a que se encuentren más documentación y recursos.
- Compatibilidad, ya que puede ser utilizado en cualquier página y diferente tipo de aplicaciones.
- Server Load, es client-side lo que ayuda a reducir la demanda del servidor en general.
- Interfaces sencillas, puede ser usado para crear características vistosas en las interfaces a través de librerías como JQuery, react, vue, entre otras.

- Funcionalidad extendida, ya que los desarrolladores lo pueden utilizar para extender la funcionalidad de las páginas web mediante archivos .js o fragmentos de código.
- Versatilidad, Existen varios métodos para utilizar Javascript en la actualidad
- Actualizaciones, Desde la llegada de ECMAScript 5 (las mejores escritas en que se basa JavaScript), ECMA International se ha dedicado a actualizar JavaScript anualmente.

Desventajas

- Seguridad Client-side, ya que el código en JavaScript se ejecuta en el lado del cliente, los errores descubiertos pueden ser explotados algunas veces para malos propósitos.
- Soporte del navegador, Mientras que el script del lado del servidor siempre produce el mismo resultado, algunas veces diferentes navegadores interpretan el código JavaScript de manera distinta.
- El código es visible en el frontend por lo que el código puede ser leído por cualquier usuario.
- Muchas veces se debe ingresar una gran cantidad de código en los sitios webs y esto genera una carga pesada para el sitio.
- Las opciones 3D son limitadas.
- Los usuarios tienen la opción de desactivar JavaScript desde el navegador.

Conocimiento básico de JavaScript

La sintaxis básica es intencionalmente similar a lenguajes de programación previos como C, C + + para reducir la cantidad de conceptos nuevos necesarios para aprender del lenguaje. Las construcciones del lenguaje, como declaraciones if, bucles for y while, los bloques switch y try... catch funcionan igual que en estos lenguajes (o casi) (MDN web docs, 2022).

JavaScript puede ser escrito en archivos .js o dentro HTML utilizando los tags <script> especificado de la siguiente manera:

```
<script language = "javascript" type = "text/javascript">
// JavaScript code
</script>
```

JavaScript ignora los espacios, las tabulaciones y las líneas nuevas que aparecen en los programas de JavaScript. Se puede usar espacios, tabulaciones y saltos de línea libremente en los programas y se puede formatear y sangrar de una manera ordenada y consistente que hace que el código sea fácil de leer y comprender.

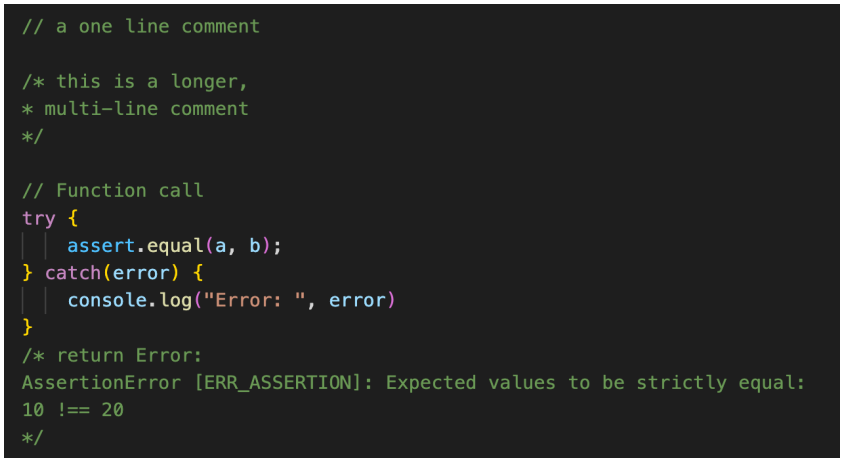
Las declaraciones simples en JavaScript generalmente van seguidas de un carácter de punto y coma, al igual que en C, C + + y Java. Sin embargo, JavaScript le permite omitir este punto y coma si cada una de sus declaraciones se coloca en una línea separada. Es case sensitive distingue entre mayúsculas y minúsculas esto significa que las palabras clave del idioma, las variables, los nombres de las funciones y cualquier otro identificador siempre deben escribirse con mayúsculas y minúsculas.

Resumen de Elemento básicos de programación en JavaScript

Comments

Los comentarios se utilizan para documentar el código y lógica y por lo general son ignorados en tiempo de ejecución.

Figura 28. Ejemplo de comentarios



Declaraciones de variables

Las variables se utilizan como nombres simbólicos para valores en la aplicación. Los nombres de las variables, llamados identificadores, se ajustan a ciertas reglas. Un identificador de JavaScript debe comenzar con una letra, un guión bajo (_) o un signo de dólar ($). Los caracteres subsiguientes también pueden ser dígitos (0–9).

Figura 29. Ejemplo de variables

```
var name = 'jorge'; // Declara una variable
let name = 'jorge'; //Declara una variable local con ámbito de bloque
const name = 'jorge'; // Declara una constante con nombre de solo lectura y ámbito de bloque.
```

Objetos

En JS todo es un objeto. Los objetos están compuestos de atributos. Si un atributo contiene una función, se considera que es un método del objeto; de lo contrario, el atributo se considera una propiedad. Existen diferentes tipos de objetos como los numbers, boolean, strings, arrays, dates, math, regExp, html DOM. Cada uno con un conjunto de propiedades y funciones.

Figura 30. Ejemplo de un objeto tipo array

```
// ejemplo objeto array
var fruits = [ "apple", "orange", "mango" ];
console.log(fruits.length) //retorna 3

const numbers = [4, 9, 16, 25];
const newArr = numbers.map(Math.sqrt);
// retorna [2,3,4,5]
```

Funciones

Es un bloque de código diseñado para realizar una tarea en particular. Una función de JavaScript se ejecuta cuando "algo" la invoca (lo llama). Existen dos formas para crear una función, utilizando

la palabra clave function o utilizando una arrow function. Los nombres de funciones pueden contener letras, dígitos, guión bajo y signos de dólar (las mismas reglas que las variables).

Figura 31. Ejemplo de funciones

```
function plusTen(val){
  return val + 10;
}

let sum = plusTen(5); // retorna 15

const plusTenArrow = (val)=> {
  return val + 10;
}

let sum2 = plusTenArrow(5); //retorna 15
```

Bloque (Block Statements)

El carácter de punto y coma (;) se utiliza para separar declaraciones en código JavaScript. Cualquier expresión de JavaScript también es una declaración. La sentencia más básica es una sentencia de bloque, que se utiliza para agrupar sentencias. El bloque está delimitado por un par de corchetes. Las declaraciones de bloque se usan comúnmente con declaraciones de flujo de control (if, for, while, switch).

Figura 32. Ejemplo de bloque

```
if (condition) {
  // bloque #1
  statement_1;
} else {
  // bloque #2
  statement_2;
}

const val = 100;
if (val > 10) {
  console.log('Mayor que 10');
} else {
  console.log ('menor que 10');
}

// imprime: Mayor que 10
```

Errores y Manejo de Excepciones (Exception Handling)

Existen tres tipos de errores en la programación:

- Errores de sintaxis: También llamados errores de análisis, ocurren en tiempo de compilación en lenguajes de programación tradicionales, pero en JS ocurren en tiempo de interpretación. Por ejemplo, la siguiente línea genera un error de sintaxis porque falta un paréntesis de cierre.
    - console.log(;
- Errores de tiempo de ejecución: También llamados excepciones, ocurren durante la ejecución (después de la compilación/interpretación).
- Errores lógicos: Los errores lógicos pueden ser el tipo de error más difícil de rastrear. Estos errores no son el resultado de un error de sintaxis o tiempo de ejecución. En cambio, ocurren cuando se comete un error en la lógica que impulsa la secuencia de comandos y no obtiene el resultado que esperaba.

En las últimas versiones de JS se agregó el manejo de excepciones. JavaScript implementa la construcción try...catch...finally, try…catch, así como el operador throw para manejar las excepciones. Se puede detectar excepciones de tiempo de ejecución y generadas por el programador, pero no se puede detectar errores de sintaxis de JavaScript.

Figura 33. Ejemplo de manejo de excepciones.

```
const a = 10;
const b = 5;

try {
console.log(rest);
}
catch ( e ) {
console.log("Error: " + e.description );
}
// retorna "Error: undefined"
```

Bucles (Loops) e Iteraciones

Los bucles ofrecen una forma rápida y fácil de hacer algo repetidamente a través de iteraciones. En JS hay muchos tipos diferentes de bucles, pero todos esencialmente hacen lo mismo, repiten una acción varias veces. Algunas formas de hacer bucles son: for, while do ... while.

Figura 34. Ejemplo de bucle

```
for (let step = 0; step < 5; step++) {
  console.log(`Walking step ${step}`);
}
/* Se ejecuta 5 veces, con valores de step 0 hasta 4.
  "Walking step 0"
  "Walking step 1"
  "Walking step 2"
  "Walking step 3"
  "Walking step 4"
*/
```

Operadores

JavaScript tiene operadores binarios y unarios, estos operadores son:

- Operadores aritméticos: suma (+), resta (-), multiplicación (*), división (/), módulo (%), incrementó (++), etc.
- Operadores de comparación: igual (==), no igual (!=), mayor que(>), menor que (<), mayor igual que (>=), etc.
- Operadores lógicos (o relacionales): y lógico (&&), o lógico (||), no lógico (!).
- Operadores de Asignación: simple (=), suma y asignación (+=), multiplicación y asignación (*=), etc.
- Operadores condicionales (o ternarios): ? : (Condicional).
- Operadores de bit: bit and (&), bit or (|), bit xor (^), etc.

Figura 35. Ejemplo de operadores.

```
const num = 5 + 10; // num es 15
const i = i++; // i = i + 1;
const b = 20; // b es 20
let result = (num > b) ? 100 : 200; // retorna 200
result += b; // result es igual a 200 + 20
```

Scope

El scope determina la accesibilidad (visibilidad) de las variables.

JavaScript tiene 4 tipos de scopes:

- Global scope: Las variables declaradas Globalmente (fuera de cualquier función) tienen Alcance Global. Se puede acceder a las variables globales desde cualquier lugar en un programa de JavaScript
- Local scope: Las variables declaradas dentro de una función se vuelven LOCALES para la función.
- Function scope: Cada función crea un nuevo scope. Las variables definidas dentro de una función no son accesibles (visibles) desde fuera de la función.
- Block scope: Las variables dentro de un bloque no pueden ser accesadas fuera del bloque.

Figura 36. Ejemplo de scope.

```
var num = 0; // global scope

function sumNumber(val){
const five = 5; // function y local scope
num = val +10 + five;
console.log(num); // 17

if(num > 10){
let rest = 10; // block scope
console.log(rest); // 10
}

console.log(rest); // rest not defined
}

sumNumber(2);
console.log(num); //17
console.log(five); // five is not defined
```

¿Qué es NPM?

NPM (Node Package Manager) es un registro de software, es uno de los más utilizados a nivel mundial para paquetes creados en javascript. Los desarrolladores de código abierto de todos los continentes usan npm para compartir y tomar prestados paquetes y muchas organizaciones también usan npm para administrar el desarrollo privado (npm, 2022).

npm consta de tres componentes distintos:

- El sitio web: El sitio web se utiliza para descubrir paquetes, configurar perfiles y administrar otros aspectos de la experiencia npm. Por ejemplo, se pueden configurar organizaciones para administrar el acceso a paquetes públicos o privados.
- La interfaz de línea de comandos (CLI): La CLI se ejecuta desde una terminal y es la forma en la que la mayoría de los desarrolladores interactúan con npm.

- El registro: El registro es una gran base de datos pública de software JavaScript y de la metainformación que lo rodea.

El registro npm contiene paquetes (packages), muchos de los cuales también son módulos de node o contienen módulos de node. Un paquete (package) es un archivo o directorio descrito por un archivo package.json. Un paquete debe contener un archivo package.json para que se publique en el registro de npm. Los paquetes pueden estar sin ámbito (unscoped) o con ámbito (scoped) para un usuario u organización y los paquetes con ámbito pueden ser privados o públicos (npm, 2022).

El registro público de npm es una base de datos de paquetes de JavaScript, cada uno compuesto por software y metadatos. Los desarrolladores de código abierto y los desarrolladores de las empresas utilizan el registro npm para aportar paquetes a toda la comunidad o miembros de sus organizaciones y descargar paquetes para usarlos en sus proyectos. Para poder descargar un paquete y utilizarlo es necesario tener instalada el CLI en el ambiente donde se correrán los comandos (npm, 2022). Como se mencionó anteriormente se pueden crear y publicar paquetes públicos que cualquiera puede descargar y usar en sus proyectos.

Los paquetes públicos sin ámbito (unscoped) existen en el espacio de nombres del registro público global y se puede hacer referencia a ellos en un archivo package.json solo con el nombre del paquete, por ejemplo nombre-paquete. Los paquetes públicos con alcance (scooped) pertenecen a un usuario u organización y deben estar precedidos por el nombre del usuario o de la organización cuando se incluyen como una dependencia en un package.json, por ejemplo @username/package-name (npm, 2022).

Un módulo es cualquier archivo o directorio en el directorio node_modules que la función require() de Node.js puede cargar. Dado que no se requiere que los módulos tengan un archivo `package.json`, no todos los módulos son paquetes. Solo los módulos que tienen un archivo `package.json` también son paquetes. En el contexto de un programa Node, el módulo también es lo que se carga desde un archivo. Para ser cargado por la función require() de Node.js, un módulo debe ser uno de los siguientes (npm, 2022):

- Una carpeta con un archivo package.json que contiene un campo "principal".
- Un archivo JavaScript.

En conclusión, se podría decir que npm es un administrador de paquetes para el lenguaje de programación JavaScript que se utiliza como el administrador de paquetes predeterminado para el entorno de ejecución de JavaScript Node.js.

¿Cuál es la diferencia entre Nodejs y el Browser?

Tanto el browser como Node.js utilizan JavaScript como lenguaje de programación. Crear aplicaciones que se ejecutan en el navegador es algo completamente diferente a la creación de una aplicación Node.js. En tiempo de ejecución al igual que javascript que se ejecuta en un navegador, node.js tiene un solo hilo para ejecutar javascript y utiliza la cola de eventos, al bloquear el subproceso principal no congela ninguna interfaz de usuario, pero sigue siendo una mala práctica porque evita que se manejan tareas asíncronas como las solicitudes web. Las excepciones, el flujo y el alcance funcionan de forma idéntica a los módulos de JavaScript. Pero hay muchas diferencias significativas:

Tabla 3. Análisis comparativo entre JS en navegador y NodeJS

Browser JS	Node JS
La mayoría de las veces lo que se está haciendo es interactuar con el DOM u otras API de la plataforma web como las cookies.	Node.js, no tiene el windows document y todos los demás objetos que proporciona el navegador. Por otro lado, como ya se ha mencionado node.js es un runtime de JavaScript basado en el motor de JavaScript de Chrome llamado V8.
el navegador no se tienen todas las buenas API que proporciona Node.js a través de sus módulos, como la funcionalidad de acceso al sistema de archivos.	Sus creadores extrajeron el motor JavaScript de Chrome y lo hicieron capaz de ejecutarse de forma independiente. Pero no es tan simple ya que no hay DOM, no hay interfaz de usuario y hay algunas diferencias de tiempo de ejecución.

En comparación con el entorno del navegador, donde no se puede elegir qué navegador utilizan los usuarios.	El desarrollador controla el entorno. A menos que esté creando una aplicación de código abierto que cualquiera pueda implementar en cualquier lugar, pero por lo general el desarrollador sabe en qué versión de Node.js se ejecutará la aplicación, esto significa que se puede escribir todo en el JavaScript moderno de ES6-7-8-9 que admita la versión de Node.js.
Desde la perspectiva del navegador dado que JavaScript se mueve tan rápido y los navegadores pueden ser lentos para actualizarse a la misma velocidad, a veces se queda atascado con el uso de versiones anteriores de JavaScript/ECMAScript es por eso que se puede utilizar Babel para transformar el código para que sea compatible con ES5 antes de ser enviado al navegador.	Esto no es necesario con Node.js.
Ocupa solo import.	Node.js es compatible con CommonJS y los sistemas de módulo ES (desde Node.js v12). En la práctica, esto significa que se puede usar tanto require() como import en Node.js, mientras que el navegador está limitado a import. Esto quiere decir que a la hora de cargar los módulos Node.js tiene un sistema de importación de módulos con reglas predefinidas de uso, donde la importación requerida es sincrónica y de bloqueo, lo que significa que el código después de la solicitud no se ejecutará hasta que se complete todo el proceso de importación.

Javascript está aislado por su seguridad.	node.js tiene acceso total al sistema como cualquier otra aplicación nativa. Esto significa que puede leer y escribir directamente en/desde el sistema de archivos, tener acceso ilimitado a la red, ejecutar software y más. Esto significa que es posible escribir software de escritorio completo con node.js incluso incluyendo una interfaz de usuario a través de módulos (Santos, 2019).
Los navegadores que tienen varios subprocesos para admitir la ejecución de javascript.	node.js se utilizan un conjunto de subprocesos para una I/O super rápida. Cuando se utilizan módulos nativos, se pueden aprovechar el grupo de subprocesos, generalmente para hacer cosas como leer desde el disco o enviar/recibir datos a través de la red en segundo plano sin desperdiciar valiosos ciclos de CPU del subproceso principal que están reservados para su código javascript. Esto significa que siempre que se usen las versiones asíncronas de los métodos de operación de I/O, el costo de I/O es básicamente cero para el subproceso de javascript porque la carga es manejada por otro subproceso en segundo plano. Esto es parte de lo que hace posible tener servidores basados en javascript de alto rendimiento, aunque javascript solo se ejecuta en un subproceso (Santos, 2019).

Desde la perspectiva de un desarrollador que usa mucho JavaScript, las aplicaciones de Node.js traen consigo una gran ventaja que es la comodidad de programar todo, el frontend y el backend, en un solo lenguaje. Esto presenta una gran oportunidad porque sabemos lo difícil que es aprender un lenguaje de programación en profundidad y por completo y al usar el mismo lenguaje para realizar todo el trabajo en la web, tanto en el cliente como en el servidor, se encuentra en una posición única de ventaja.

¿Cuáles son los módulos incorporados en nodejs?

En Node.js, los módulos son los bloques de código encapsulado que se comunican con una aplicación externa en función de su funcionalidad relacionada. Los módulos pueden ser un solo archivo o una colección de múltiples archivos/carpetas. La razón por la que los programadores

dependen en gran medida de los módulos es por su reutilización, así como por la capacidad de dividir una pieza compleja de código en fragmentos manejables.

Los módulos son de tres tipos:

- Módulos centrales (Core): Node.js tiene muchos módulos integrados que forman parte de la plataforma y viene con la instalación de Node.js. Estos módulos se pueden cargar en el programa usando la función require.
- módulos locales: A diferencia de los módulos integrados y externos, los módulos locales se crean localmente en la aplicación Node.js.
- Módulos de terceros: Los módulos de terceros son módulos que están disponibles en línea mediante NPM. Estos módulos se pueden instalar en la carpeta del proyecto o globalmente. Algunos de los módulos de terceros más populares son mongoose, express, angular y react.

Los modulus a ser abarcado en esta sección son los módulos centrales o core modules, para invocar estos modulus en el código se utiliza la sintaxis:

$$let\ mod\ =\ require('NOMBRE\ MÓDULO')$$

En donde la función require() devolverá un tipo de JavaScript dependiendo de lo que devuelva el módulo en particular. Algunos de los módulos core son:

Módulo File System (FS)

Proporciona una gran cantidad de funciones muy útiles para acceder e interactuar con el sistema de archivos.

Algunos de sus métodos más representativos son: fs.access(), fs.close(), fs.copyFile(), fs.createReadStream(), fs.createWriteStream(), fs.mkdir(), fs.open(), fs.readFile(), fs.rename(), fs.rmdir(), fs.truncate(),fs.writeFile(), etc.

Figura 37. Ejemplo de uso módulo fs con base en documentación

```
const fs = require('fs');

try {
  fs.renameSync('oldFileName.txt', 'newFileName.txt');
  // done
} catch (err) {
  console.error(err);
}
```

Módulo http

Es un módulo clave para la red de Node.js, está diseñado para admitir muchas características del protocolo que tradicionalmente han sido difíciles de usar, la interfaz tiene cuidado de nunca almacenar en búfer solicitudes o respuestas completas, por lo que el usuario puede transmitir datos.

Algunos de sus métodos más representativos son: http.createServer(), http.request(), http.get(), http.Agent, http.ClientRequest, http.Server, http.ServerResponse, etc

Figura 38. Ejemplo de uso módulo http según documentación

```
const http = require('node:http');

const options = {
  host: '127.0.0.1',
  port: 8080,
  path: '/length_request'
};

// Make a request
const req = http.request(options);
req.end();

req.on('information', (info) => {
  console.log(`Got information prior to main response: ${info.statusCode}`);
});
```

Módulo path

Proporciona utilidades para trabajar con rutas de archivos y directorios.

Algunos de sus métodos más representativos son: path.basename(), path.dirname(), path.extname(), path.format(), path.isAbsolute(), path.join(), path.normalize(), path.parse(), path.relative(), path.resolve().

Figura 39. Ejemplo de uso módulo path según documentación

```
const path = require('path');
path.basename('/test/something.txt'); // something.text
path.dirname('/test/something/file.txt'); //test
```

Módulo querystring

Proporciona utilidades para analizar y formatear cadenas de consulta de URL.

Algunos de sus métodos más representativos son: querystring.decode(), querystring.encode(), querystring.escape(str), querystring.parse(str[, sep[, eq[, options]]]), querystring.stringify(obj[, sep[, eq[, options]]]), querystring.unescape(str).

Figura 40. Ejemplo de uso módulo querystring según documentación.

```
const querystring = require('node:querystring');
querystring.stringify({ foo: 'bar', baz: ['qux', 'quux'], corge: '' });
// Returns 'foo=bar&baz=qux&baz=quux&corge='
```

Módulo os

Proporciona propiedades y métodos de utilidad relacionados con el sistema operativo.

Algunos de sus métodos más representativos son: os.EOL, os.arch(), os.constants, os.cpus(), os.devNull, os.endianness(), os.freemem(), os.getPriority([pid]), os.homedir(), os.hostname(), os.loadavg(), os.networkInterfaces(), os.platform(), os.release(), etc.

Figura 41. Ejemplo de uso módulo os según documentación

```
const os = require('os');
os.homedir();
//return '/Users/joe';
```

Módulo url

Proporciona utilidades para la resolución y el análisis de URL.

Algunos de sus métodos más representativos son: new URL (input[, base]), url.hash, url.host, url.hostname, url.href, url.origin, url.password, url.pathname, url.port, url.protocol, etc.

Figura 42. Ejemplo de uso módulo url según documentación

```
import { URL } from 'node:url';
const myURL = new URL('https://example.org:81/foo');
console.log(myURL.hostname);
// Prints example.org
```

Módulo assert

Proporciona facilidades que son útiles para la afirmación de la función. El módulo de aserción proporciona un conjunto de funciones de aserción para verificar invariantes, si la condición es verdadera, no generará nada más, la consola proporciona un error de afirmación. Por lo general utilizado para unit test

Algunos de sus métodos más representativos son: assert.assert(), assert.deepStrictEqual(actual, expected[, message]), assert.deepStrictEqual(actual, expected[, message]), assert.equal(actual, expected[, message]), assert.equal(actual, expected[, message]), assert.equal(actual, expected[, message]).

Figura 43. Ejemplo de uso módulo assert según documentación

```
const assert = require('assert').strict;

var a = 10;
var b = 20;

// Function call
try {
    assert.equal(a, b);
} catch(error) {
    console.log("Error: ", error)
}
// return Error: AssertionError [ERR_ASSERTION]: Expected values to be strictly equal:
// 10 !== 20
```

Enfoque técnico de MySQL

MySQL es un sistema de gestión de bases de datos relacionales de código abierto (RDBMS, por sus siglas en inglés) con un modelo cliente-servidor. RDBMS es un software o servicio utilizado para crear y administrar bases de datos basadas en un modelo relacional. Las computadoras que tienen instalado y ejecutan el software RDBMS se llaman clientes.

Un sistema de gestión de bases de datos relacionales (RDBMS) es una colección de programas y capacidades que permiten a los equipos de TI y a otros crear, actualizar, administrar e interactuar con una base de datos relacional. Los RDBMS almacenan datos en forma de tablas, y la mayoría de los sistemas comerciales de administración de bases de datos relacionales utilizan el lenguaje de consulta estructurado (SQL) para acceder a la base de datos.

Ahora la comunicación entre cliente servidor se realiza mediante un lenguaje específico del dominio: lenguaje de consulta estructurado (SQL, Structured Query Language). PostgreSQL y el servidor Microsoft SQL contienen “SQL” ya que es muy probable que sean marcas que también utilizan la sintaxis SQL. El software RDBMS a menudo se escribe en otros lenguajes de programación, pero siempre usa SQL como lenguaje principal para interactuar con la base de datos. MySQL como tal está escrito en C y C ++. Pasa como con los países sudamericanos, todos son geográficamente diferentes y tienen historias diferentes, pero todos hablan principalmente español.

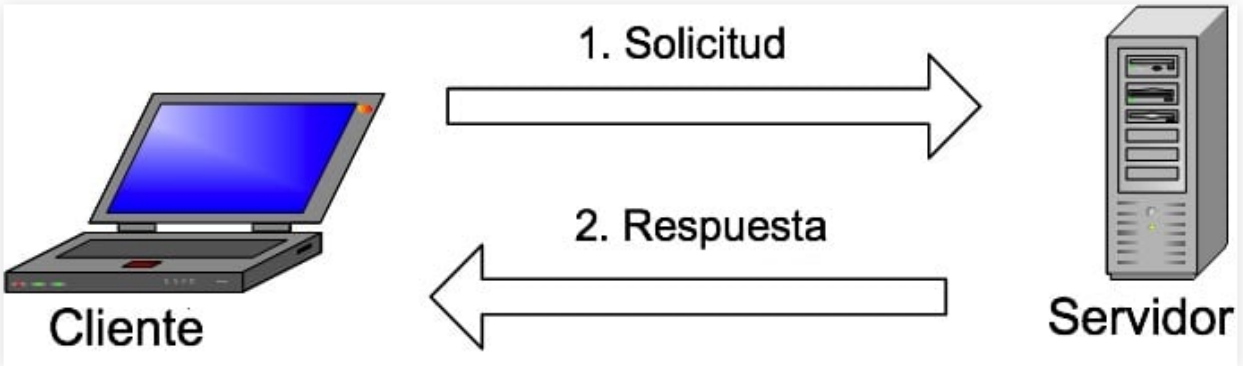
Siempre que necesitan acceder a los datos, se conectan al servidor RDBMS. Esa es la parte «cliente-servidor». MySQL es una de las muchas opciones de software RDBMS.

Las operaciones que se pueden crear utilizando SQL son las siguientes:

- Consulta de datos: solicitar información específica de la base de datos existente.
- Manipulación de datos: agregar, eliminar, cambiar, ordenar y otras operaciones para modificar los datos, los valores o los elementos visuales.
- Identidad de datos: definir tipos de datos, por ejemplo, cambiar datos numéricos a números enteros.
- Control de acceso a los datos: proporcionar técnicas de seguridad para proteger los datos, lo que incluye decidir quién puede ver o usar cualquier información almacenada en la base de datos.

¿Cómo funciona MySQL?

Figura 44. Flujo funcional de Mysql (Gustavo B, 2022)



Explicación:

La figura 43 muestra cómo se realiza la interacción de una estructura básica cliente-servidor. Uno o más dispositivos (clientes) se conectan a un servidor a través de una red específica. Cada cliente

puede realizar una solicitud desde la interfaz gráfica de usuario (GUI) en sus pantallas, y el servidor producirá el output deseado, siempre que ambas partes entiendan la instrucción.
MySQL crea una base de datos donde se pueden almacenar y manipular datos además de también definir la relación de cada tabla, los clientes pueden realizar solicitudes a través de instrucciones SQL específicas para la herramienta y en ese momento la aplicación del servidor responderá con la información solicitada y esta será desplegada según el formato que se haya establecido en la instrucción SQL (Gustavo B, 2022).

¿Cómo se maneja MySQL a través de GCP (Google Cloud Platform)?

Esto es posible gracias a Cloud SQL para MySQL. Este es un servicio de base de datos completamente administrado que permite configurar, mantener, controlar y administrar bases de datos relacionales de MySQL en Google Cloud Platform.

Cloud SQL proporciona una alternativa basada en la nube a las bases de datos locales de MySQL, PostgreSQL y SQL Server.

Análisis comparativo entre Dialog Flow vs Bot Framework citadas.

Los Chatbot frameworks son espacios digitales donde los desarrolladores crean bots desde cero a partir de funciones y clases predefinidas. Estos hacen que sea más fácil crear un chatbot personalizado que comenzar desde cero, pero se requieren conocimientos avanzados de codificación para usarlos, pero aún más importante saber identificar cual es la mejor opción de implementación, por ello a continuación se realizará un análisis comparativo de las dos opciones técnicas antes descritas: DialogFlow vs Bot Framework.

La decisión sobre qué Framework utilizar se basará en lo que quiere hacer con su bot. La mayor distinción entre los Frameworks es el ecosistema de desarrollo y el asistente que mejor admiten. Lex es mejor para Alexa, DialogFlow es mejor para Google Assistant y Microsoft Bot Framework es mejor para Cortana.

DialogFlow versus Bot Framework

A continuación, se muestra la tabla 3 un resumen de las principales diferencias que se deben tomar en cuenta al momento de seleccionar una de estas opciones:

Tabla 4. Tabla sobre principales diferencias entre Dialogflow y Bot Framework

DialogFlow	Bot Framework
Su principal diferencia	
Dialogflow se conocía inicialmente como API.AI antes de que Google lo adquiriera. Proporciona una interfaz web que permite tanto a los que no son desarrolladores como a los desarrolladores crear bots. Es un servicio de código cerrado con API y una interfaz web. Los usuarios pueden interactuar con el bot mediante interfaces conversacionales basadas en voz y texto.	Bot Service funciona con Bot Framework como una plataforma para crear bots de chat utilizando SDK, herramientas y servicios de código abierto (como el emulador de bot) que le permitirán desarrollar de forma rápida y sencilla. También hay una interfaz visual para que la utilicen los no desarrolladores.
Ambos disponibles en Cloud en 2016	
Google adquirió Dialogflow y luego lo agregó como un servicio en Google Cloud Platform. La consola web y de interfaz de usuario hace que sea muy fácil crear un chatbot básico que luego se puede agregar a cualquier aplicación web, lo que permite a los usuarios comprender rápidamente las diferencias entre intenciones y entidades sin tener conocimientos previos. Existe una opción de demostración web para la integración en un sitio web, pero es muy rudimentaria. Los desarrolladores deberán crear su propia interfaz de usuario e integrarla con las API de Dialogflow. La interfaz de usuario se puede transformar fácilmente con CSS y HTML simples.	Microsoft lanzó Azure Bot Service para permitir que los creadores de chatbots se trasladaran a la nube y permitir que Microsoft administrara las consideraciones de servidor y almacenamiento. Microsoft tiene el deseo de crear un directorio de bots sólido y grande y un motor de búsqueda de chatbots con esta plataforma, por lo que los bots que utilizan Azure Bot Service se agregarán automáticamente al directorio de bots de Microsoft. El servicio proporciona plantillas y SDK con Bot Framework y funciona bien. con otros servicios de Azure como QNAMaker y LUIS.
Integracions disponibles	
Las integraciones de Google Dialogflow incluyen Google Assistant, Slack, , Skype, Spark, Facebook Messenger, Twitter y Twilio. Ademas de	Las integraciones de Bot Service incluyen Cortana, Facebook Messenger, Skype, Kik, Alexa, Direct Line, Email, LINE, Microsof teams, Skype, Telegram y Twilio

integraciones de telefonia incorporradas de socios como: AudioCodes, Avaya, Genesys, SignalWire.	
Integracions web y moviles	
Google Dialogflow tiene una integración web básica incorporada sin código (consulte el tutorial de WordPress para integrar un chatbot con Dialogflow).	Azure Bot Service admite widgets de chat web de código abierto que están disponibles en Github.
Idiomas disponibles	
Google Dialogflow admite más de 20 idiomas, que incluyen inglés, español, portugués, francés, hindi y chino.	Azure Bot Service en más de 15 idiomas, incluidos inglés, español, portugués, francés, hindi, árabe y chino.
Costos	
Tiene un plan estándar gratuito que pueden usar las pequeñas y medianas empresas. La edición Google Dialogflow Enterprise cuesta $0.0002 por solicitud.	Azure Bot Service es gratuito hasta 10 000 mensajes por mes. Los planes pagos comienzan desde $0.50 por cada 1000 mensajes con cargos adicionales por consumir otros servicios como Azure Functions y Azure Web Apps.

¿Cómo seleccionar la opción más adecuada?

Existen dos perspectivas que se deben tomar en cuenta en cuanto a la propuesta de valor que el chatbot como tal beneficia a los involucrados:

- Desde el punto de vista de los resultados de negocio
- Desde el punto de vista del usuario/consumidor

Desde el punto de vista de los resultados de negocio: Para una multinacional o una pequeña empresa, los chatbots se pueden usar para proporcionar información analítica, toma de decisiones basada en datos y mejorar la eficiencia dentro del negocio.

Si bien no es común que un chatbot contribuya directamente al flujo de ingresos (a menos que sea parte de la mitad del embudo de ventas), es posible que no sea factible calcular el ROI directo para identificar sus beneficios. Algunos puntos de valor para el negocio incluyen:

- Incrementar la satisfacción del usuario.
- Incrementar la lealtad de los usuarios y por consiguiente ventas/compras.

- Manejo de la capacidad del soporte y costos de dicho soporte.
- Proveer soluciones en el menor tiempo posible.
- Proporciona un canal alternativo de ventas.

Desde el punto de vista del usuario/consumidor: Los usuarios y clientes que utilicen el chatbot de una empresa asumirán una experiencia sin fricciones cuando interactúen con la marca. Desde la disponibilidad las 24 horas, las transacciones instantáneas, las visitas del usuario serán cada vez más personalizadas e interactivas, las empresas descubrirán que los usuarios estarán más satisfechos en escenarios en los que originalmente habrían tenido que contratar a un representante humano. Algunos puntos de valor para el negocio incluyen:

- Incrementar la satisfacción del usuario y del cliente.
- Incrementar la lealtad de los usuarios, clientes y potenciales clientes.
- Interacción proactiva con el cliente.
- Permite analizar las respuestas de los clientes.
- Ayuda a abrir nuevas oportunidades en distintos mercados sin importar los idiomas.

4.2 ¿CÓMO LLEVAR A CABO UNA IMPLEMENTACIÓN DE CHATBOTS UTILIZANDO LAS TECNOLOGÍAS OFRECIDAS POR GOOGLE?

Perfil técnico necesario

Para la implementación de chatbots utilizando GCP es necesario contar con un perfil técnico en el cual el desarrollador posea el conocimiento necesario para aprovechar todas las opciones que la herramienta proporciona, por ello se mencionara algunas de las habilidades técnicas que un Junior Cloud Engineer debe poseer:

- Ingeniería Informática, Telecomunicaciones, o equivalente.
- Experiencia trabajando con Google Cloud Platform.
- Conocimientos de programación y/o experiencia profesional en desarrollo de software o de sistemas (Java, Python, Nodej, PHP, Go, JavaScript).
- Conocimientos en API’s, plataformas de contenedores o Serverless.

- Pasión por aprender y adaptarse a un ecosistema tecnológico cambiante.
- Excelentes habilidades de diagnóstico y resolución de problemas en todas las tecnologías.
- Experiencia en el debug de aplicaciones.
- Nivel alto de inglés, escrito y hablado (B2 o superior).

Infraestructura necesaria

Google Cloud Platform GCP

Google Cloud Platform (GCP) ofrecido por Google es un conjunto de servicios de computación en la nube que se ejecuta en la misma infraestructura que Google utiliza internamente para sus productos de usuario final, como Google Search, Gmail, Google Drive y YouTube. Junto con un conjunto de herramientas de administración, proporciona una serie de servicios modulares en la nube que incluyen computación, almacenamiento de datos, análisis de datos y aprendizaje automático. Proporciona infraestructura como servicio, plataforma como servicio y entornos informáticos sin servidor. Ofrece los servicios de: compute (Google Kubernetes, Cloud Functions, Cloud Run y mas), Storage and Databases (Cloud SQL, Cloud Storage, Cloud Datasources y más), Networking (Cloud load balancing, VPC, Cloud Armor, DNS y más), Big Data (BigQuery, Cloud Data Studio y más), Cloud AI(Cloud speech-to-text, Cloud text-to-speech, Cloud machine learning engine y más), Management Tools (Cloud Console, Cloud Shell, Cloud API's y más), Identity y Security (Cloud IAM, Security Key Enforcement, Cloud Security Scanner y más), IoT (Cloud IoT Core, Edge TPU y cloud IoT edge) y API Platform (Maps platform, Developer portal, Service infrastructure y más) (Google, 2022).

El registro requiere una tarjeta de crédito o detalles de una cuenta bancaria a la cual hacer los cargos de los servicios que se utilizan.

Servicios necesarios

Cloud Functions

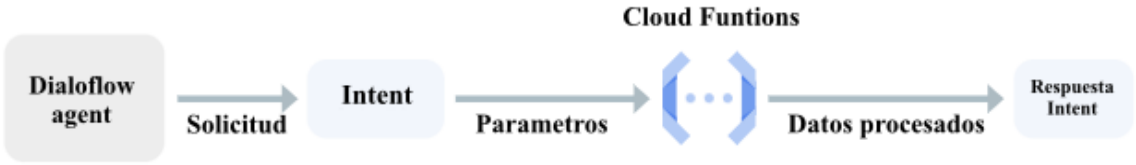
Cloud Functions es la opción que presenta Google dentro de GCP para ejecutar código en la nube sin servidores ni contenedores que administrar. Cloud Functions es un producto escalable de

funciones de pago por uso como servicio (FaaS) para ayudar a crear y conectar servicios basados en eventos con un código simple y de un solo propósito (Google, 2022).

Cloud Functions brinda una experiencia de desarrollador simple e intuitiva solo se necesita escribir el código y dejar que GCP maneje la infraestructura operativa. Ayuda a desarrollar más rápido pequeños fragmentos de código que responden a eventos. Optimiza los problemas de orquestación desafiantes conectando los productos de GCP entre sí o con servicios de terceros mediante eventos. Y lo mejor que solo se factura por el tiempo de ejecución de la función, medido con precisión de 100 milisegundos. No se paga nada cuando la función está inactiva. Cloud Functions gira y retrocede automáticamente en respuesta a los eventos (Google, 2022).

Cloud Functions se pueden utilizar para mostrar microservicios propios a través de API HTTP o integrarse con servicios de terceros que ofrecen integraciones vía webhook para ampliar rápidamente las capacidades de la aplicación (Google, 2022).

Figura 45. Flujo entre Dialogflow y Cloud Functions (Google, 2022).



Cloud SQL

Cloud SQL es un servicio conocido como una base de datos en la nube, este es un servicio de base de datos creado para ser accedido a través de una plataforma en la nube. Cumple muchas de las mismas funciones que una base de datos tradicional con la flexibilidad adicional de la computación en la nube. Los usuarios instalan el software en una infraestructura de nube para implementar la base de datos. Hay dos modelos de implementación comunes (IBM, 2022):

- Los usuarios pueden ejecutar bases de datos en la nube de forma independiente, utilizando una imagen de máquina virtual.
- Los usuarios pueden comprar acceso a un servicio de base de datos, mantenido por un proveedor de bases de datos en la nube de las bases de datos disponibles. Algunas están basadas en SQL y otras utilizan un modelo de datos NoSQL.

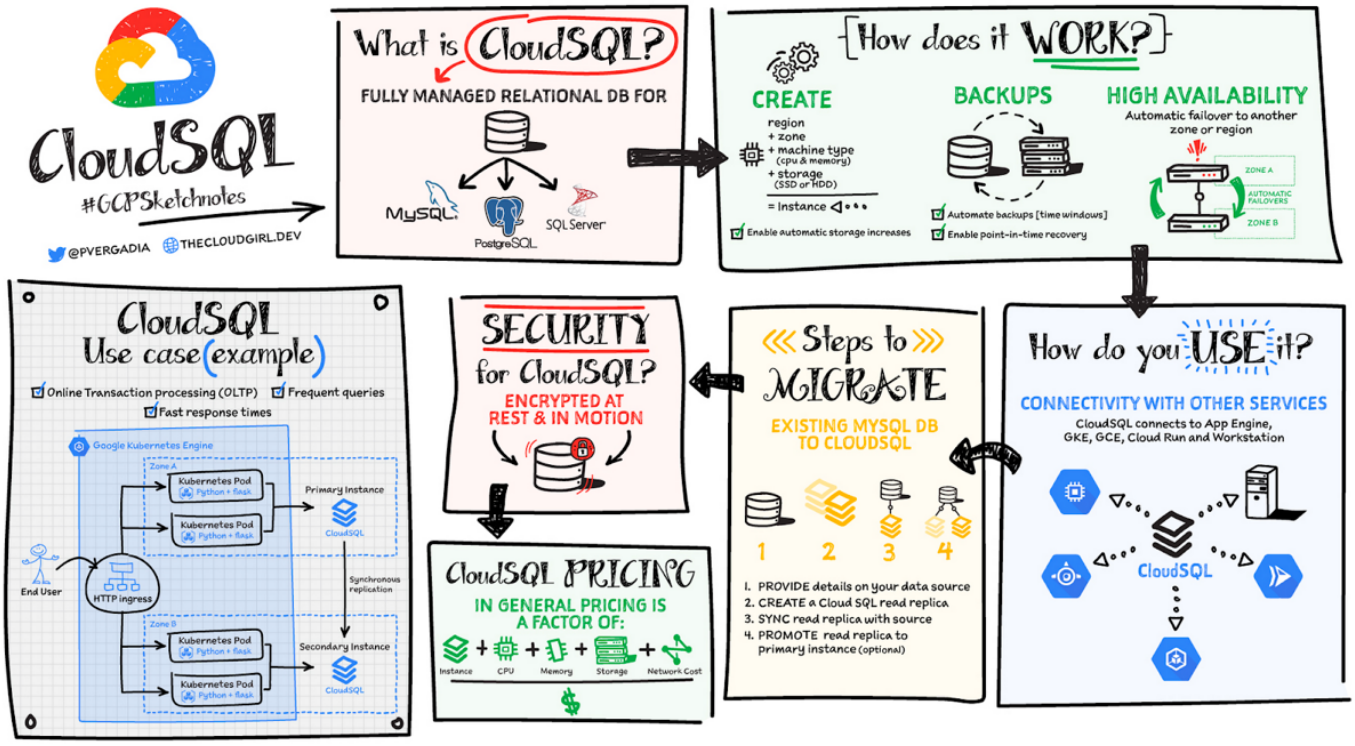
Este servicio presenta algunos beneficios como los siguientes:

- Acceso a las bases de datos en la nube desde prácticamente cualquier lugar, utilizando la API o la interfaz web de un proveedor.
- Las bases de datos en la nube pueden expandir sus capacidades de almacenamiento en tiempo de ejecución para adaptarse a las necesidades cambiantes. Las organizaciones solo pagan por lo que usan.
- Poseen recuperación de desastres, esto quiere decir que, si hay una falla del equipo o corte de energía, los datos se mantienen seguros mediante copias de seguridad en servidores remotos.

El servicio que se utilizará será el denominado Database-as-a-service(DBaaS), el cual es un modelo de base de datos como servicio, los propietarios de las aplicaciones no tienen que instalar ni mantener la base de datos ellos mismos. En cambio, el proveedor de servicios de la base de datos asume la responsabilidad de instalar y mantener la base de datos y los propietarios de la aplicación pagan de acuerdo con el uso que hacen del servicio (IBM, 2022).

Cloud SQL brinda el servicio de base de datos relacional totalmente administrado para MySQL, PostgreSQL y SQL Server. Ejecuta las mismas bases de datos relacionales que se conoce con sus colecciones de extensiones, indicadores de configuración y ecosistema de desarrolladores, pero sin la molestia de la autogestión en un servidor (Google, 2022). Para este prototipo se utilizará el servicio para MySQL.

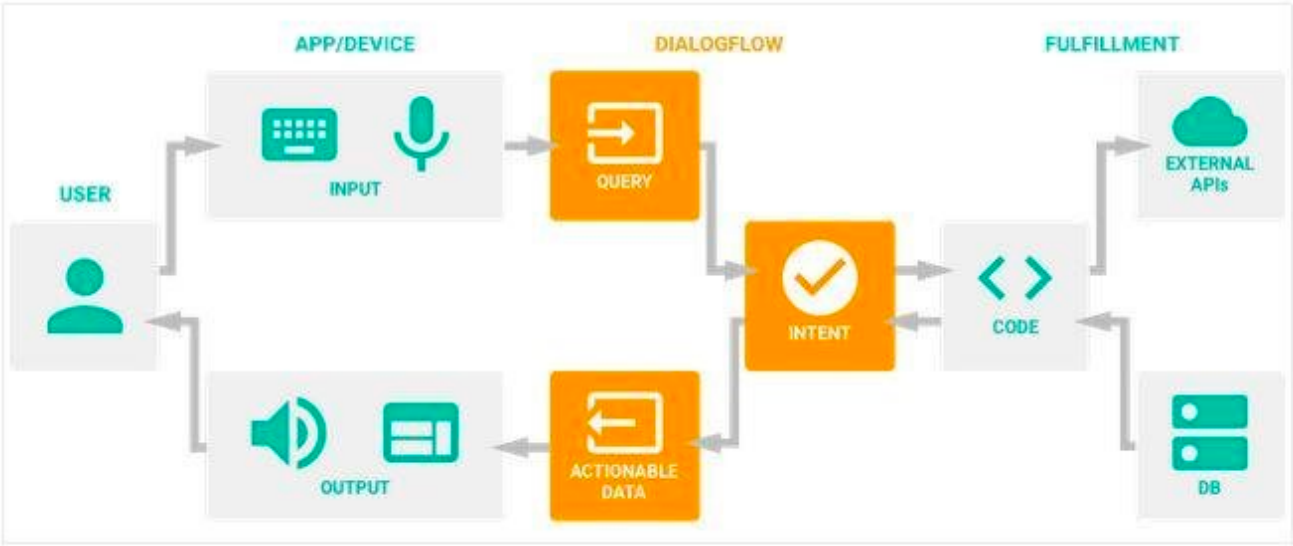
Figura 46. Resumen de Cloud SQL (Google, 2022).



Dialogflow

Como se mencionó en el capítulo 4 en la sección de tecnologías ofrecidas por Google, Dialogflow es una plataforma de comprensión del lenguaje natural que facilita el diseño y la integración de una interfaz de usuario conversacional en diferentes formatos que proporciona servicios de agente virtual para chatbots. Para la creación de este prototipo se utilizará Dialogflow trial edition que es la edición gratuita que proporciona la mayoría de las funciones del tipo de agente ES estándar (mayor información ver capítulo 4). Por ser la edición trial ofrece cupo limitado.

Figura 47. Flujo básico de un agente de Dialogflow (Dennis Gannon, 2018)



Lenguajes de programación necesarios

Como lenguaje de programación para este prototipo se utilizará Node.js como ya se en el capítulo 4 en la sección de enfoque técnico de Nodejs, este es un entorno de ejecución de código abierto y multiplataforma de JavaScript que ejecuta el motor de JavaScript V8, que es el núcleo de Google Chrome, fuera del navegador.

Node.js es un ambiente de ejecución de JavaScript. Utiliza un modelo de entrada y salida sin bloqueo controlado por eventos, de esta manera lo hace un entorno ligero y eficiente, gracias a esto Node.js cambió la forma en que se programa JavaScript ya que ahora todo el código debe funcionar de manera asíncrona a partir de eventos, prácticamente se puede correr JavaScript en cualquier sistema operativo, desde servidores hasta dispositivos móviles.

4.3 CREACIÓN DE PROTOTIPO EN EL CONTEXTO DE SERVICIO AL CLIENTE

El prototipo de chatbot a crear estará basado en un caso muy básico en el dominio de soporte técnico para usuarios que utilizan un sistema de tickets, el bot ayudará a los usuarios para poder consultar la siguiente información:

- Estado de los tickets
- Conocer la fecha de la última actualización
- Conocer la resolución o comentarios sobre el ticket

Definición del dominio en el que se implementa el prototipo

Definición del negocio

El soporte técnico es un rango de servicios por medio del cual se proporciona asistencia a los usuarios al tener algún problema al utilizar un producto o un servicio, ya sea de hardware o software ya sea de una computadora, de un servicio en internet, periféricos, artículos electrónicos, maquinaria o cualquier tipo de dispositivo. El soporte técnico se puede dar por distintos medios, incluyendo correo electrónico, chat, telefónicos y presencial. Por lo general se utiliza un sistema de tickets para monitorear el progreso de la consulta.

Se pueden dar varios niveles de soporte según (da Silva, 2021) (Portillo, 2020):

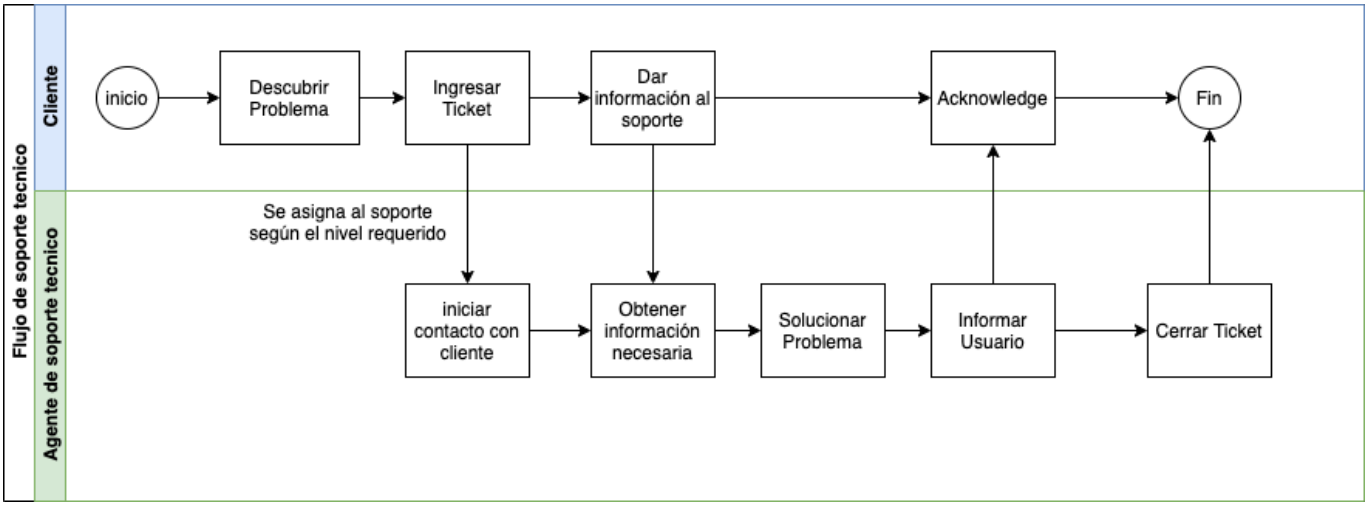
- Soporte Nivel 1 (Tier 1, T1): Consiste en reunir toda la información del cliente y hacer un análisis de los síntomas, en este nivel se incluyen algunos métodos para resolver problemas como: verificar incidencias en las líneas básicas, resolución de problemas de usuarios y contraseñas, desinstalación, instalación y reinstalación de aplicaciones de software, asistencia navegando en menús de aplicación.
- Nivel 2 (Tier 2, T2): Denominado también help desk, este soporte pasa especialmente en el grupo de ayuda en escritorio, este se realiza por personas especializadas en redes de comunicación, sistemas de información, sistemas operativos, bases de datos, etc. para este es necesario contar con manuales o guías donde se encuentre paso a paso con los procesos a seguir para resolver dicho problema.
- Nivel 3 (Tier 3, T3): Denominado soporte de backend, este soporte se considera ya a nivel avanzado o nivel experto con un análisis avanzado, los técnicos a este nivel son expertos y son

responsables no solo de ayudar al personal de los niveles 1 y 2 sino también de investigar y desarrollar soluciones a los problemas nuevos desconocidos.

- Nivel 4 (Tier 4, T4): Este tipo de soporte debe tener conocimientos de nivel 1, 2, 3 y adicionalmente debe manejar servidores Microsoft y Linux temas con instalación, configuración, interconexión, administración y operación de servidores está dentro de las responsabilidades, este suele ser un nivel de soporte presencial o remoto.
- Nivel 5 (Tier 5, T5): Este es el nivel de soporte más avanzado y está conformado por especialistas externos a la empresa, en este nivel los especialistas están capacitados para en todos los niveles anteriores y además manejan sistemas complejos, programación en varios lenguajes y hablan por lo menos una lengua extranjera.

Lo que se busca con un sistema de tickets para soporte técnico es centralizar todos los incidentes de los clientes en relación a una plataforma, hardware o infraestructura para recaudar la mayor cantidad posible de datos para mejorar la atención y soporte técnico.

Figura 48. Diagrama de flujo de soporte técnico



Definición de términos

- *Soporte Técnico:* Es una asistencia que brinda una empresa para que sus clientes puedan hacer uso de los productos o servicios, la finalidad es ayudar a los usuarios para que puedan resolver

ciertos problemas, como por ejemplo “no poder conectarse a internet”, “No poder guardar información o consultar información en un sistema”, “fallo de una computadora”, “fallo de un dispositivo de hardware”, etc (de Silva, 2021).

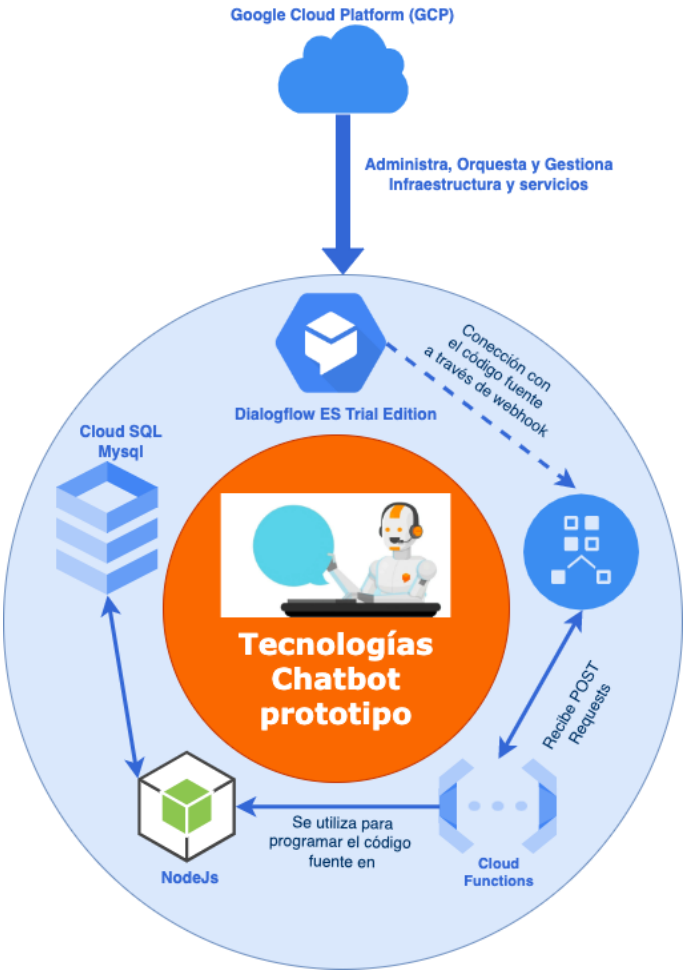
- *Ticket:* Es un término inglés que llegó al idioma español como tiquete, aunque por lo general se emplea en inglés, este puede ser un boleto, un pasaje, un recibo o una factura que sirve como una constancia de pago (Pérez Porto & Merino, 2016). En informática es la forma de hacer llegar por escrito una consulta o de reportar un problema relacionado a cualquiera de los servicios brindados (Linube, 2018).
- *Sistema de Tickets:* Es una solución tecnológica que permite abrir y centralizar en un solo lugar todos los servicios de soporte que brinda una empresa (de Silva, 2021). Sirve para abrir, organizar, priorizar y rastrear todos y cada uno de los tickets de soporte que se han generado.
- *Ticket de Soporte Técnico:* Es un boleto digital generado por un sistema de tickets a partir de las solicitudes entrantes realizadas por los usuarios, sin importar el canal que hayan utilizado para comunicarse. El ticket de soporte técnico permite organizar las incidencias y recopilar la mayor cantidad de datos de clientes, lo que posibilita resolverlas en un periodo de tiempo más breve e identificar aquellas repetidas para solucionarlas en conjunto (de Silva, 2021).

Selección de tecnología a utilizar

Para el prototipo de esta tesis de chatbot se utilizará las tecnologías de:

- Google Cloud Platform (GCP)
- Google Dialogflow ES Trial
- SQL Cloud Mysql como motor de base de datos
- NodeJs como lenguaje de programación
- Webhooks
- Cloud Function.

Figura 49. Interacción entre tecnologías seleccionadas



Identificación de dependencias

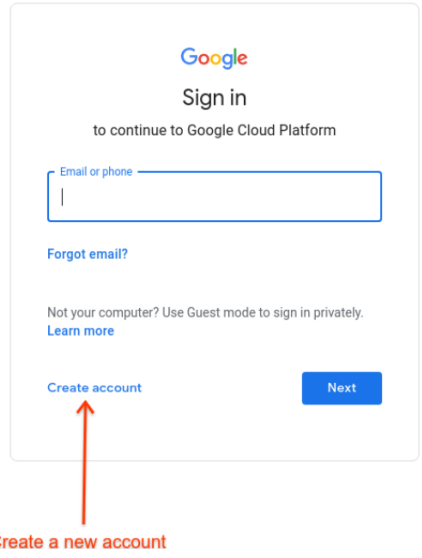
Para el prototipo de este chatbot se tiene dependencia de Google Cloud Platform (GCP) por lo tanto se necesitará una cuenta de gmail, esta cuenta se utilizará para tener acceso a GCP, Dialogflow, cloud functions y cloud SQL**.**

Google Cloud Platform (GCP)

Para la creación de una cuenta en GCP se deben seguir los siguientes pasos:

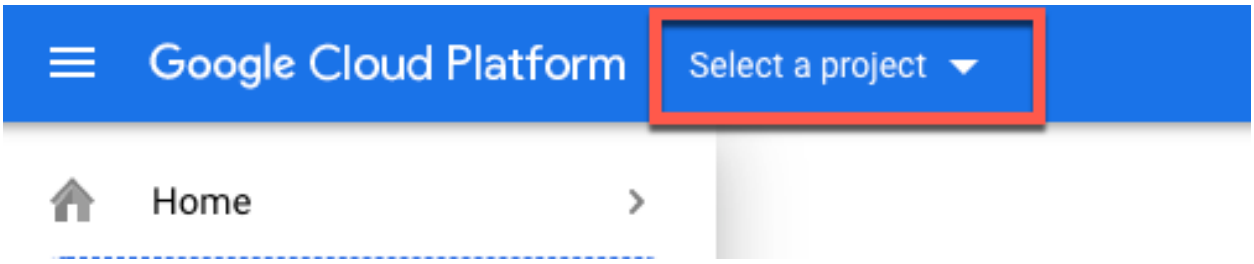
- Ir a Google **Cloud Console** (https://console.cloud.google.com/) en un navegador.
- Cuando se le solicite acceder, hacer clic en Crear cuenta para crear una cuenta nueva como se muestra en la figura 48:

Figura 50. Opción Crear cuenta en página principal de GCP



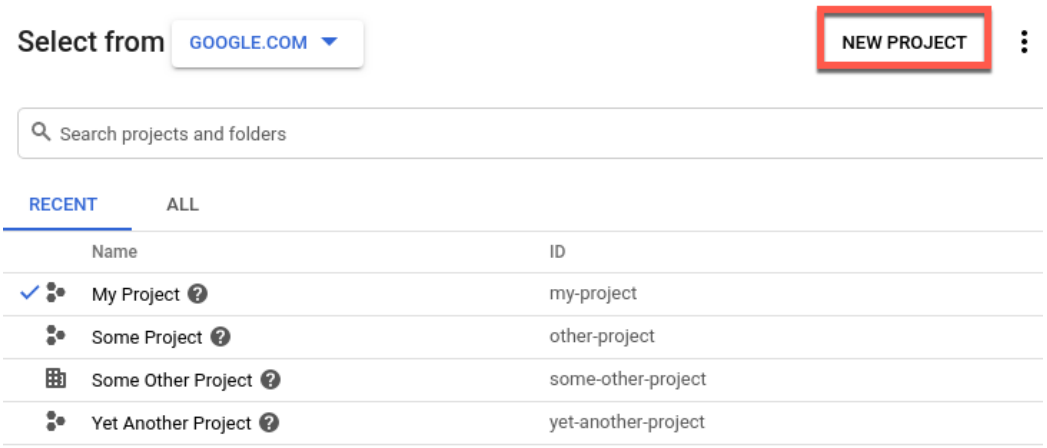
- Se debe seguir las instrucciones para registrar la dirección de correo electrónico corporativo como una Cuenta de Google. Como alternativa, se puede usar una Cuenta de Gmail o cualquier otra Cuenta de Google.
- Continuar en Google Cloud Console y aceptar las condiciones de Google Cloud que se presentan. (Google Cloud, 2022)
- Luego de crear la cuenta es necesario crear un proyecto en donde se trabajará el prototipo. Se debe hacer clic en el selector de proyectos “Select a project” como en figura 49:

Figura 51. Seleccionar un proyecto en página principal de GCP



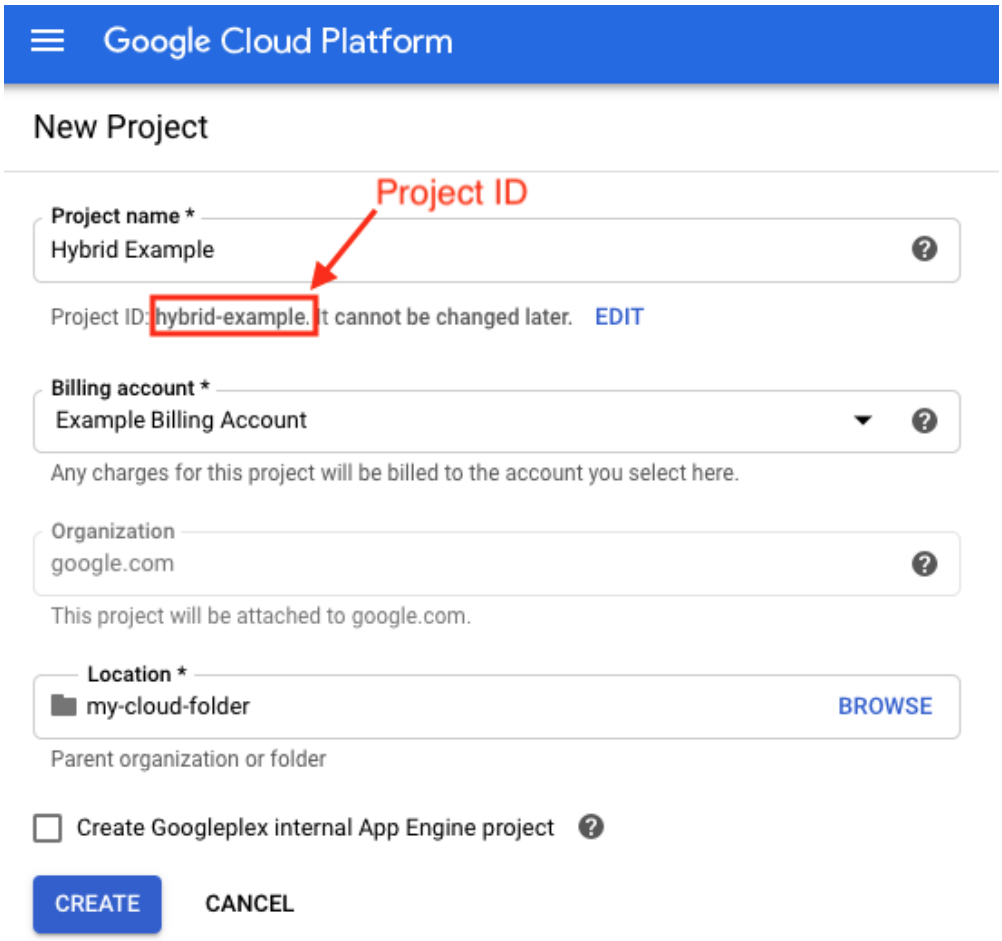
- En el cuadro de diálogo Selector de proyectos, se muestra la siguiente pantalla, donde se debe seleccionar “New Project”, figura 50.

Figura 52. Crear un nuevo proyecto en página principal de GCP



- Ingresar un nombre descriptivo del proyecto a crear en el campo Nombre del proyecto (Project name). El nombre del proyecto solo puede contener letras, números, comillas simples, guiones, espacios o signos de exclamación.
- Google Cloud genera un ID del proyecto correspondiente debajo del campo “Project name”, figura 51.

Figura 53. Formulario de creación de un proyecto nuevo en GCP



- En el campo “Billing account”, se debe seleccionar la cuenta de Facturación de Google Cloud que se usará para pagar los proyectos.
- En el campo “Location”, se debe seleccionar una carpeta para el proyecto. Las carpetas son una parte opcional del árbol de recursos del proyecto de Google Cloud.
- Hacer clic en “Create” y Google Cloud creará el proyecto nuevo.

Cloud SQL de MySQL

Para el desarrollo del prototipo es necesario crear una instancia de MySQL utilizando Cloud SQL de GCP, esto se debe implementar través de los pasos siguientes:

- En Google Cloud Console, ir a la página Instancias de Cloud SQL.
- Hacer clic en “Create an instance” para crear una instancia de base de datos.

- En el panel “Choose your database engine” de la página Crear una instancia, hacer clic u elegir MySQL y luego, en Siguiente.
- En el campo ID de instancia del panel de información de la instancia, ingresar un ID para la instancia.
- No incluir información sensible o de identificación personal en el nombre de la instancia, ya que es visible de forma externa.
- No se debe incluir el ID de proyecto en el nombre de la instancia. Esto se hace de manera automática cuando es apropiado (por ejemplo, en los archivos de registro).
- Establecer una contraseña para el usuario raíz. Se puede ingresar la contraseña de forma manual o hacer clic en “Generate” para que Cloud SQL cree una contraseña de forma automática.
- Sugerencia: Para ver la contraseña en texto no encriptado, hacer clic en el ícono Mostrar contraseña.
- Aunque existe la opción sin contraseña (No password), esta no se recomienda por razones de seguridad.
- Seleccionar la versión de la base de datos para la instancia: MySQL 8.0 (predeterminada), MySQL 5.7 o MySQL 5.6.
- La versión de la base de datos no se puede editar después de que se crea la instancia.
- En la sección elegir la región y la disponibilidad de zona, seleccionar la región y la zona para la instancia.
- Ubicar la instancia en la misma región que los recursos que acceden a ella. No se podrá modificar la región seleccionada en un futuro. En la mayoría de los casos, no se necesita especificar una zona.
- En la sección “Customize your instace”, actualizar la configuración de la instancia. Primero, hacer clic en “show configuration options” para mostrar los grupos de opciones de configuración. Luego, expandir los grupos para revisar y personalizar la configuración. A la derecha, se muestra un informe de resumen de todas las opciones que seleccionadas. Personalizar esta configuración de instancia es opcional. Los valores predeterminados se asignan en todos los casos en los que no se realizan personalizaciones.

Figura 54. Creación de una instancia de bases de datos en GCP

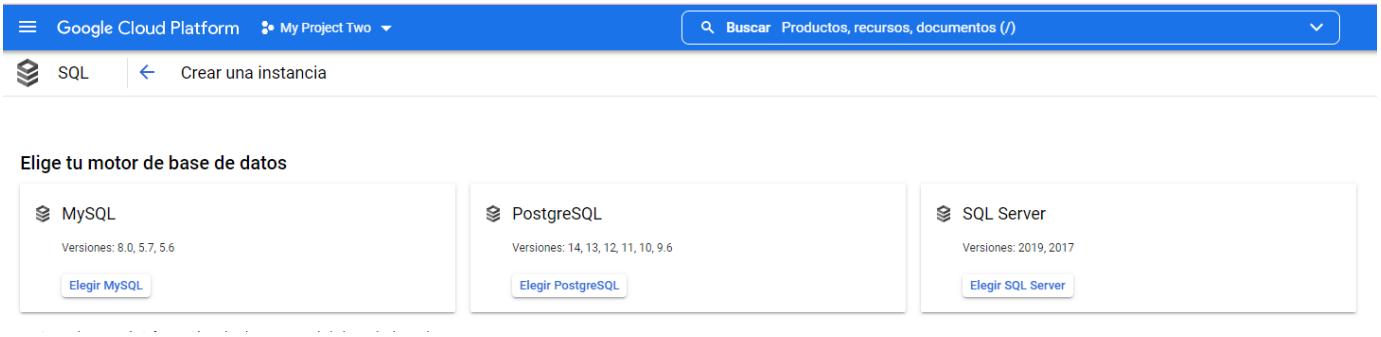
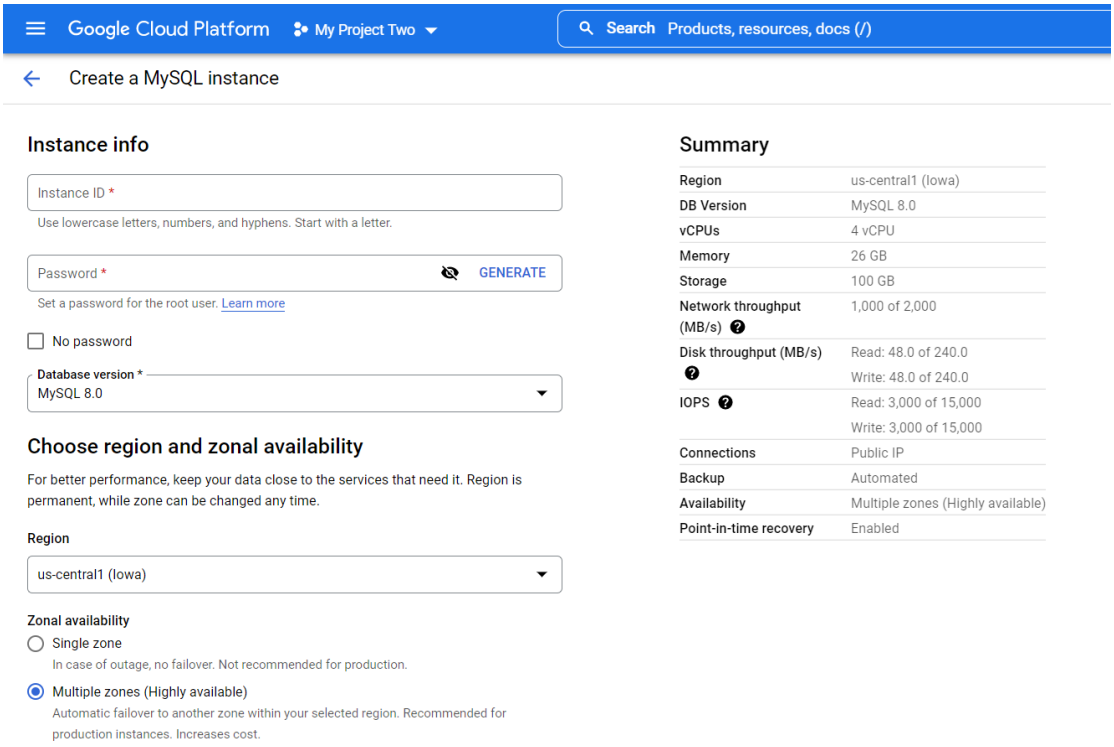


Figura 55 Configuración de la instancia para base de datos

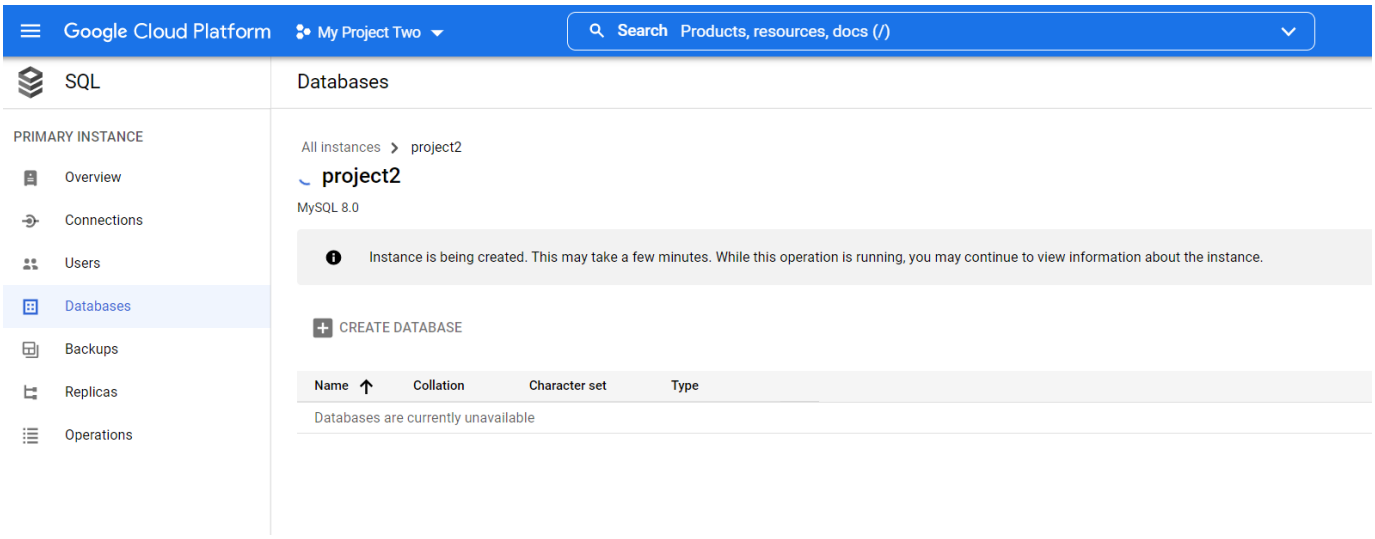


Luego de crear la instancia se debe crear la base de datos con la que se trabajara, para esto se deben seguir los siguientes pasos:

- Visitar la página de instancias de Cloud SQL en la consola de Google Cloud Platform.
- Ir a la página de instancias de Cloud SQL

- Seleccionar la instancia a la que se añadirá la base de datos.
- Seleccionar la pestaña “Databases”.
- Hacer clic en “créate database”.
- En el cuadro de diálogo, especificar el nombre de la base de datos y el conjunto de caracteres y la recopilación.
- Para obtener más información sobre los conjuntos de caracteres, consultar la documentación de MySQL referente a los conjuntos de caracteres. Para obtener más información sobre la recopilación, consultar la documentación de MySQL referente a la recopilación.
- Haz clic en “Create”.

Figura 56 Creación de la base de datos



Cloud Functions

Para crear una función se debe hacer bajo un proyecto previamente creado o crear un nuevo proyecto.

- En el “navigation menú” buscar los elementos de serverless y seleccionar cloud functions, figura 53.
- Esto abrirá la página de cloud function que muestra la lista de cloud functions existentes, figura 54.

- Para crear la cloud function dar click en el botón “créate function”, esto abrira la página de configuración, la creación tiene dos pasos la configuración de la cloud function y el código.
- En la sección de configuración se debe asignar un environment, nombre, región y trigger Figura 55.
- Dar clic en el botón “Save”
- Luego dar clic en el botón “Next”, esto abre el paso dos que es agregar el código de la función, la función se puede crear en los siguientes lenguajes de programación: Nodejs, Java, PHP, Python, Go, Ruby y .Net Core. Figura 56.
- Para este prototipo se utilizará Nodejs como Runtime e Inline Editor como Source code.
- Para esta configuración de código se presentan dos archivos un package.json el cual manejara todas las dependencias y un index,js en el cual se debe de agregar el código de la función.
- Luego se debe dar clic en el boton “deploy” y esto hará el deploy de la función, al haber deployado con éxito se mostrar un check verde en la paina principal en la primera columna de la lista.
- Para ver información de la cloud function se debe dar clic en el nombre en la lista de la página principal, esto abrirá la página de detalles, esta página presenta varias opciones para ver información de la cloud function selecciona (metric, details, source, trigger, permissions, logs, testing, edit, delete, copy) Figura 57.

Figura 57. Navigation Menu de GCP

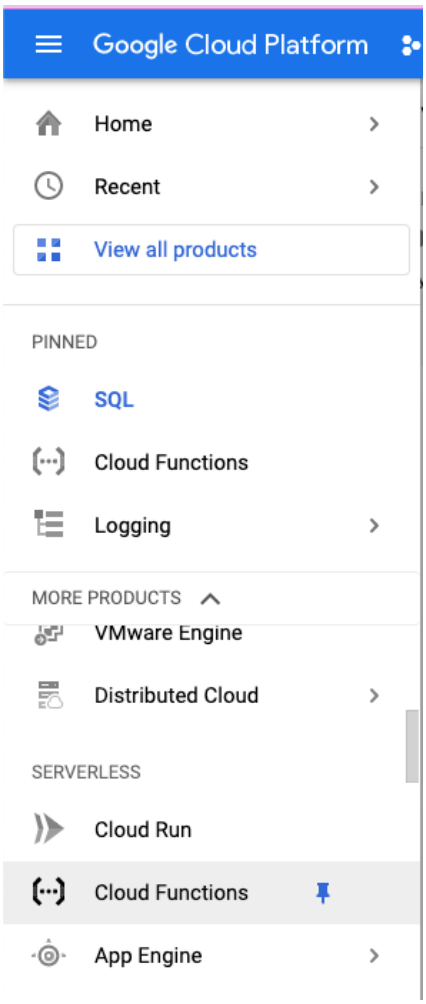


Figura 58. Página principal de Cloud Functions en GCP

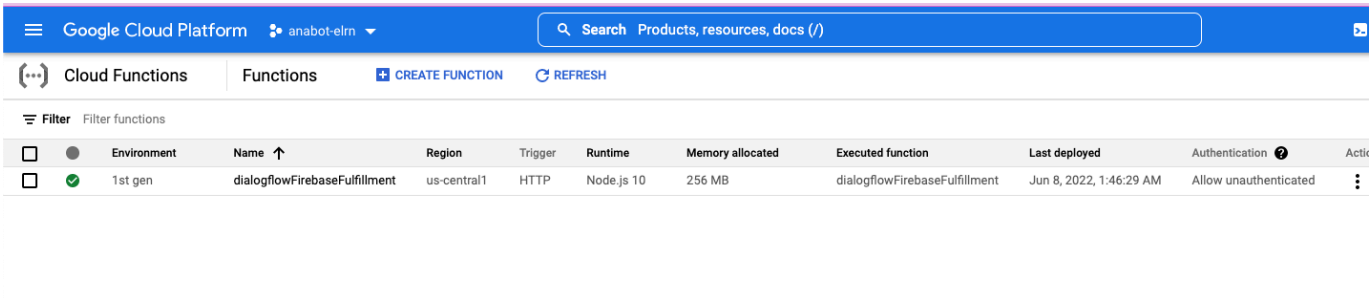


Figura 59. Página de configuración de Cloud Functions en GCP

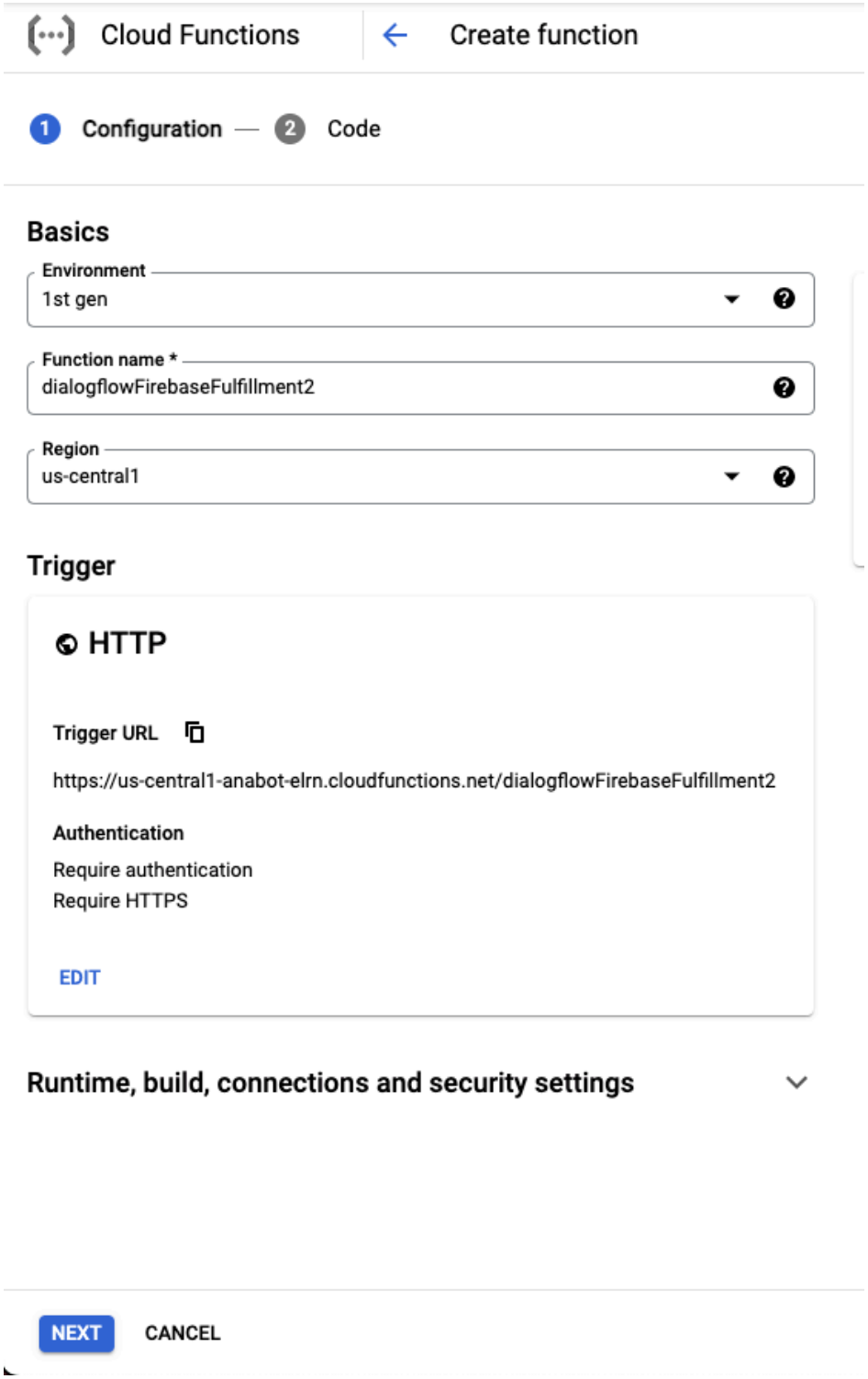


Figura 60. Página de código de Cloud Functions en GCP

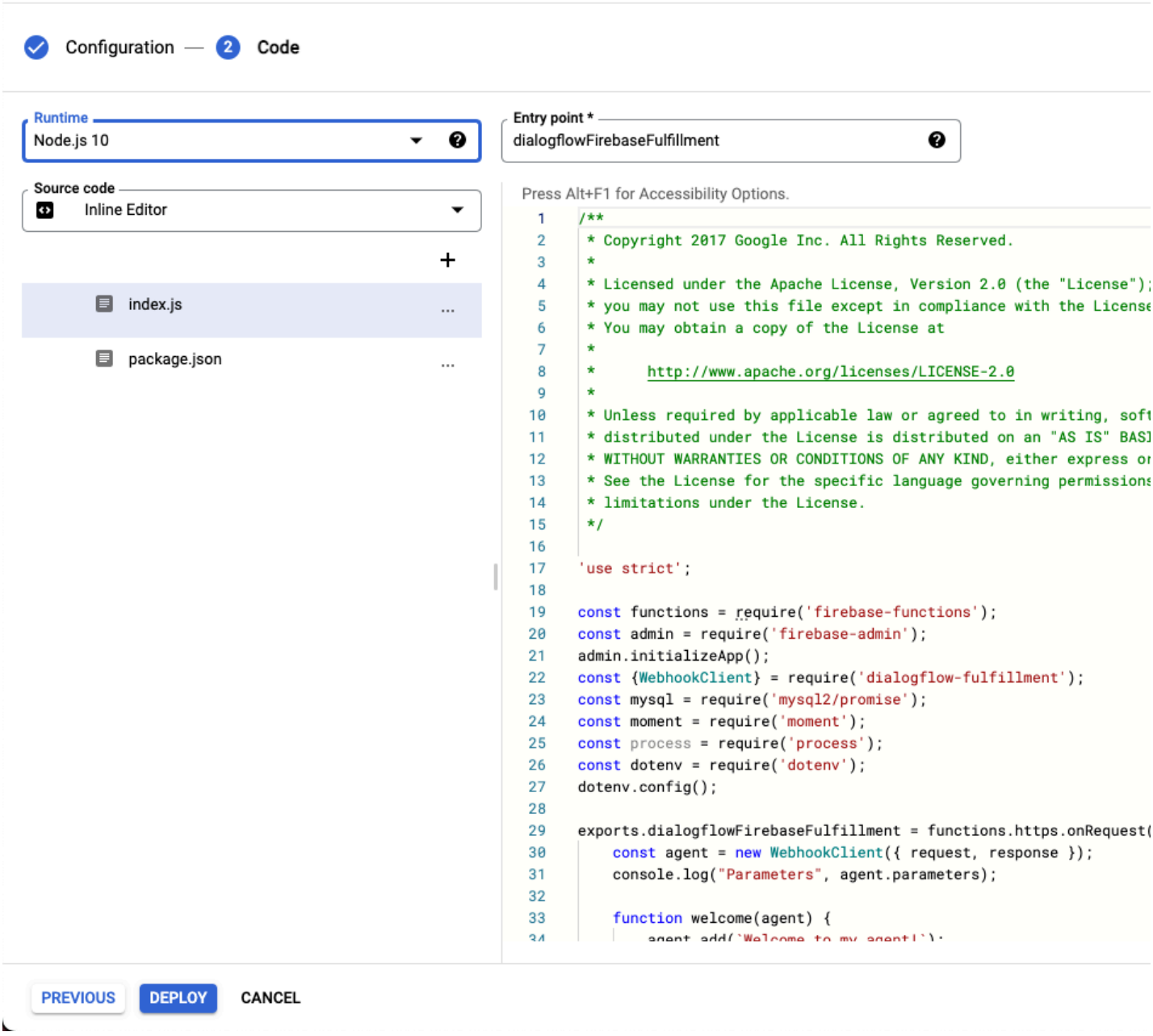
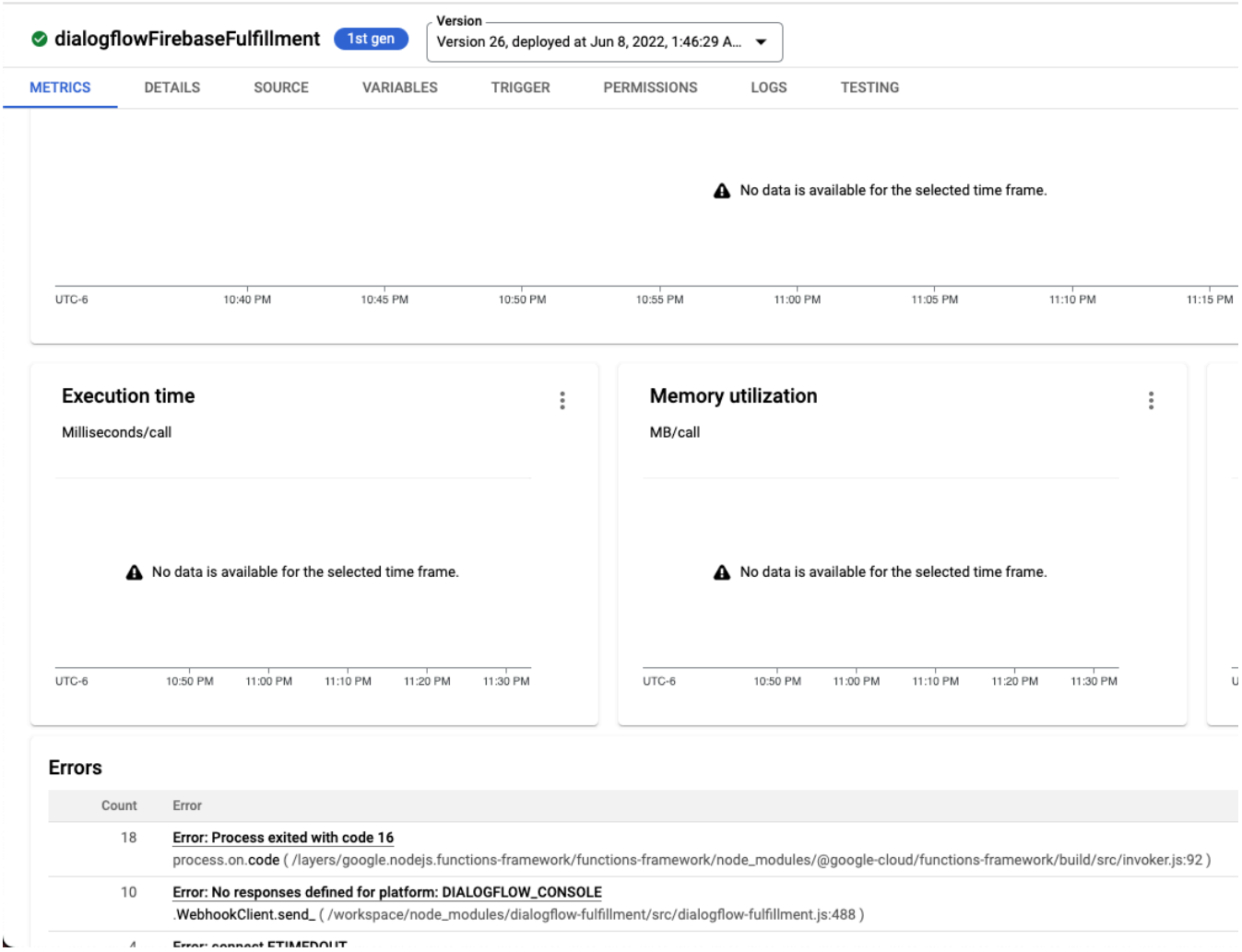


Figura 61. Página de detalle de la Cloud Functions en GCP



Pasos para la implementación de un prototipo

Creación de la cuenta de Dialogflow

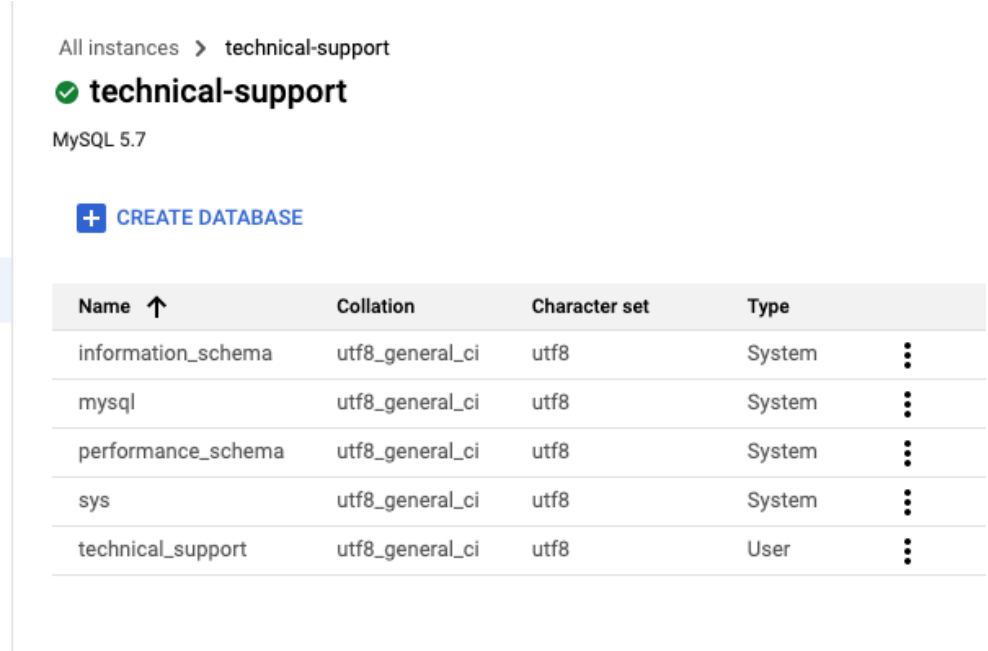
Para la creación de una cuenta en DialogFlow se debe poseer un proyecto de GCP previamente creado sobre el cual se trabajará, luego de seleccionar el proyecto se debe seguir los siguientes acceder a la consola de dialogflow https://dialogflow.cloud.google.com/.

Creación de la base de datos

Para esta prueba de concepto se asume que se tiene una instancia de SQL Cloud para MySQL con solo una tabla con todos los campos necesarios para la creación del bot.

- Crear una base de datos llamada technical_support
- Crear una tabla llamada tickets con los siguientes campos: id (numérico, autogenerado), , description (text), notes (text), reporter(varchar, 150), contact_email (varchar, 150), contact_phone (varchar, 50), status (varchar, 50), resolver (varchar, 150), resolver_email (varchar, 50), created_at (timestamp), updated_at(timestamp),
- Se debe correr el script de creación de base de datos anexo 1
- Se debe correr el script de creación de tabla anexo 2
- Se debe correr el script de inserts anexo 3

Figura 62. SQL Cloud para MySQL en GCP.



Creación de Cloud Function

Para este prototipo se creará una cloud function con el nombre “dialogflowFirebaseFulfillment

". Se utilizará Nodejs 10 como Runtime e Inline Editor como Source code. Y se utilizara el url del trigger.

- Para ver el código del index.js ver anexo 5.
- Para ver el código del package.json ver anexo 4.
- Habilitar la Cloud SQL Admin API.
- Para poder conectarse a la base de datos se debe de utilizar el tipo de conexión socketPath, ver anexo 6.
- La cuenta de GCP debe tener permisos de IAM o Developer.
- El template de la cloud function para comunicarse con dialogflow está en el anexo 5.
- El codigo del para el intent de ticket-status está en el anexo 7.
- El codigo del para el intent de ticket-last-update está en el anexo 8.
- El codigo del para el intent de ticket-notes está en el anexo 9.

Creación del agente

Para esta prueba de concepto se asume que ya se cuenta con una cuenta de Dialogflow y con un proyecto de GCP (Google Project).

- Ir a dialogflow console https://dialogflow.cloud.google.com/
- En la tuerquita dar click en la opción "Create new agent" como en la Figura 59.
- Se abrirá un formulario para llenar la información de: nombre del agente, descripción, google project id, log setting como en la Figura 60.
- Al dar click en save se creará el agente y aparecerá al lado izquierdo de la Dialogflow console como en la Figura61.
- Una vez creado el agente este podrá ser configurado, como en la Figura 62.
- Del lado izquierdo de la consola aparecerán todos los elementos del agente, la consola cuenta con tres secciones como en la Figura 63.
    - Elementos: presenta el listado de todos los elementos del agente
    - Área de trabajo: Área donde según el elemento seleccionado se presentan las opciones y configuraciones a setear
    - Área de test: Área para testear el agente

Figura 63. Crear nuevo agente

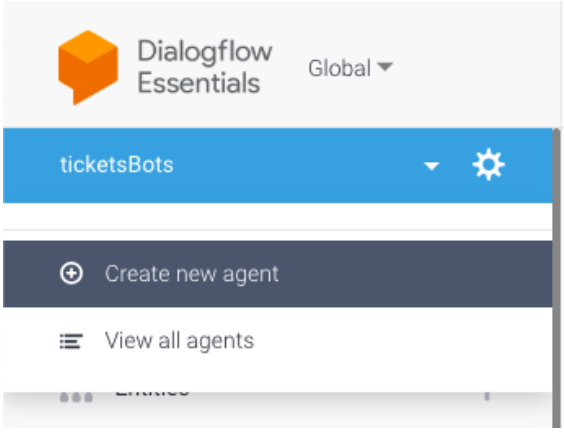


Figura 64. Creación de agente

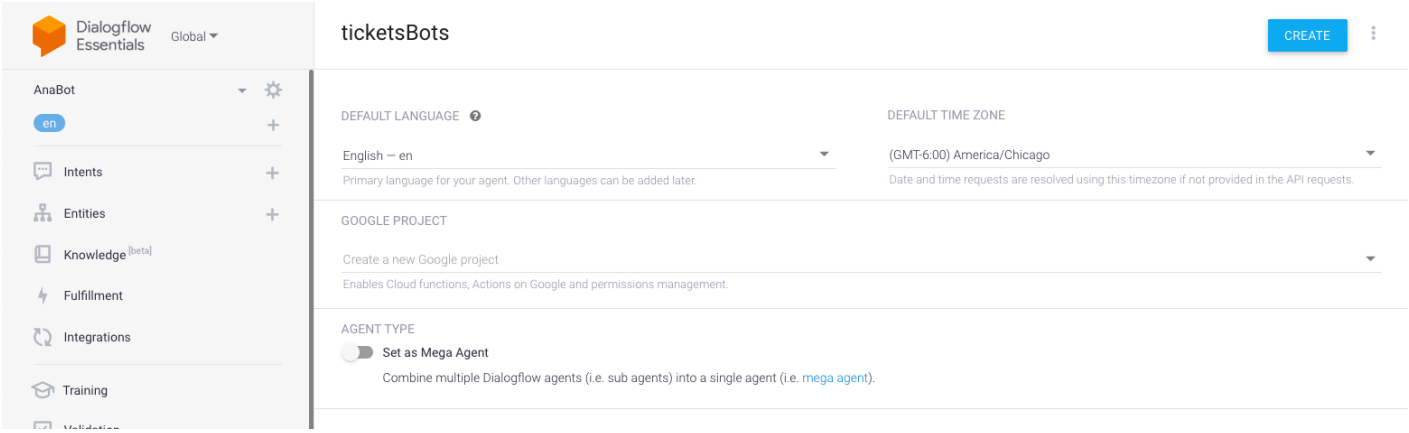


Figura 65. Agente en la sección de elementos

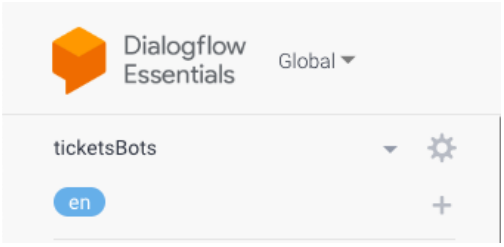


Figura 66. Configuración de agente

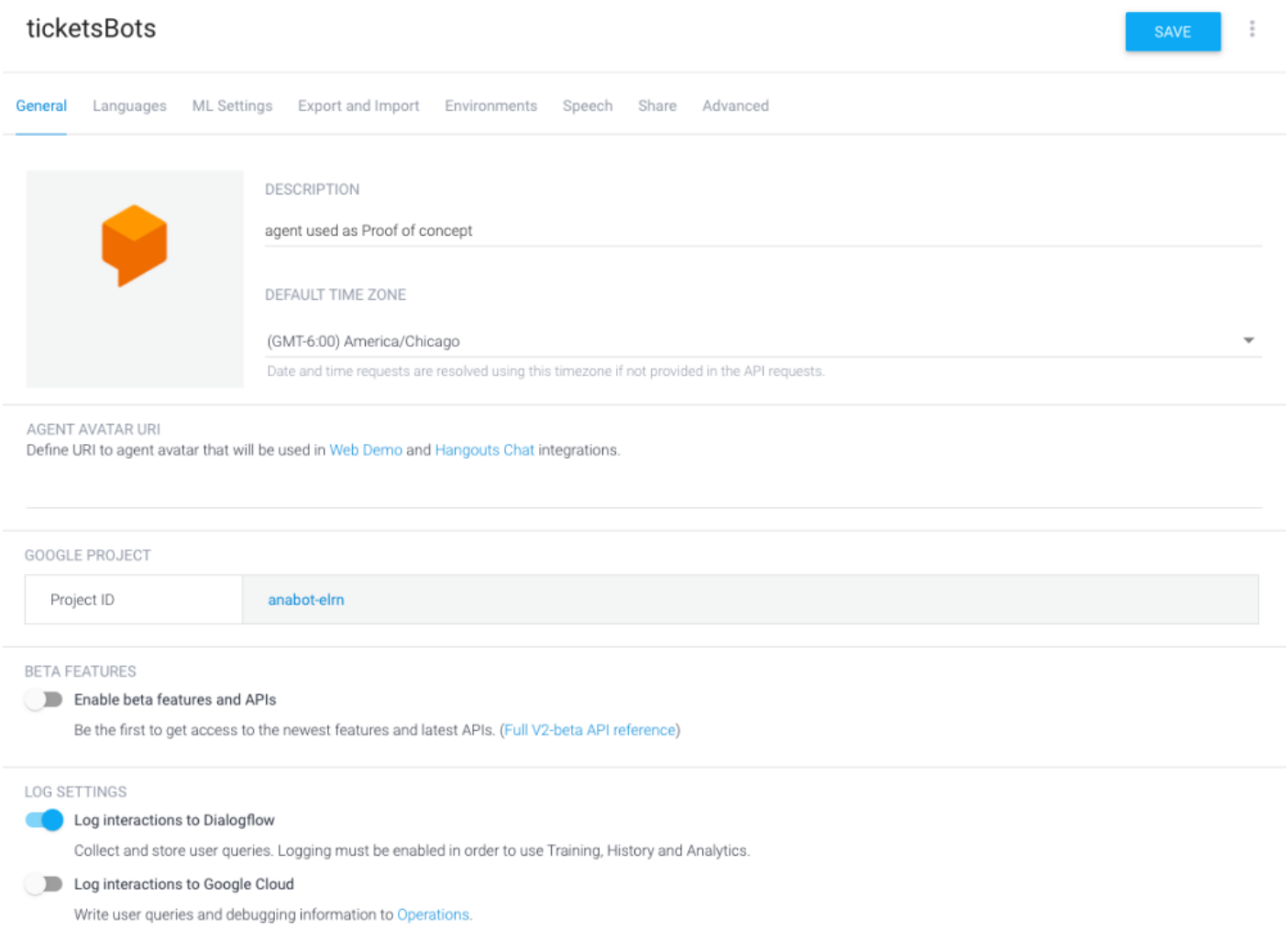


Figura 67. Dialogflow Console

Creación de Entities

- Para este prototipo se utiliza una entity del sistema @sys.number, pero para efectos de documentación de muestra la creación de una nueva entity llamada ticket-id.
- Se debe seleccionar el elemento Entities en la sección de elementos de Dialogflow console, como en la Figura 64.
- Dar click en el botón “Create entity” y llenar la información de la entidad para este caso, la entidad será ticket-id, como en la Figura 65.
- Esta entity se utilizará para ser asociada en las training phrases del intent.

Figura 68. Elemento entidad (Entity)

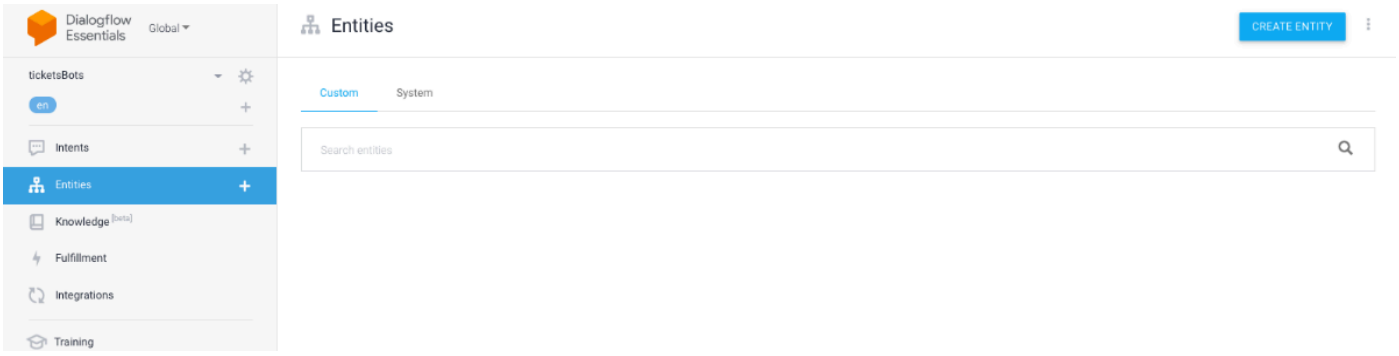
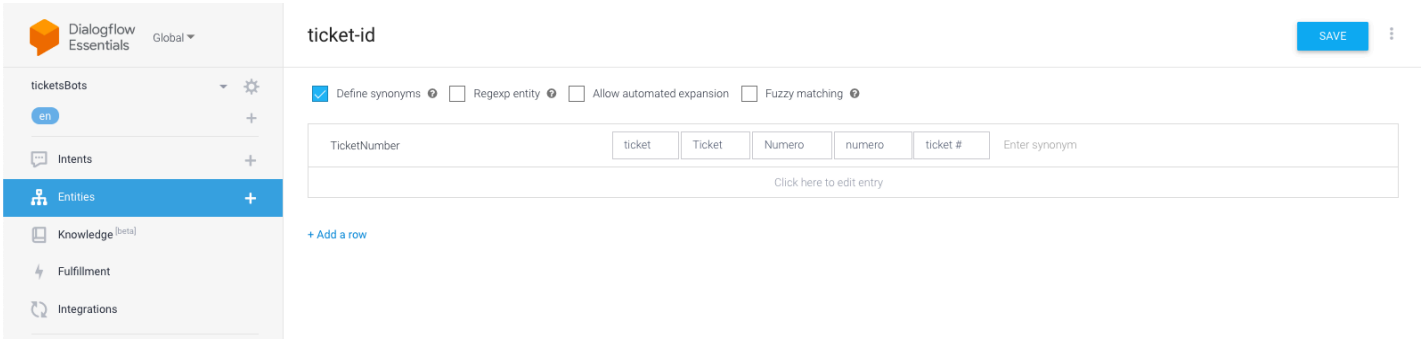


Figura 69. Creación de entidad (Entity)



Creación de intens

Para este prototipo se crearán tres intenciones (Intents):

- ticket-status: Para consultar el estado del ticket.
- ticket-last-update: Para consultar la fecha de la fecha de la última actualización del ticket
- ticket-notes: Para consultar comentarios sobre el ticket.

Los pasos a seguir para crear un intent son los siguiente:

- Se debe seleccionar el elemento Intents en la sección de elementos de Dialogflow console
- Dar click en el botón de “Create Intent”.
    - Se agrega el nombre del intent.
    - Se agregan las training phrases.

- Se asocian los números a la entity ticket-id, sección de action and parameters
- Se agrega una respuesta por defecto.
- Si se debe agregar procesamiento de datos por ejemplo búsqueda en una base de datos o algún procesamiento de datos como cálculos, etc se debe de habilitar la opción de fulfillment.
- Cuando se guarda el intent este aparece en la lista de intenciones creadas para el agente.

Figura 70. Elemento de una intención (Intent)

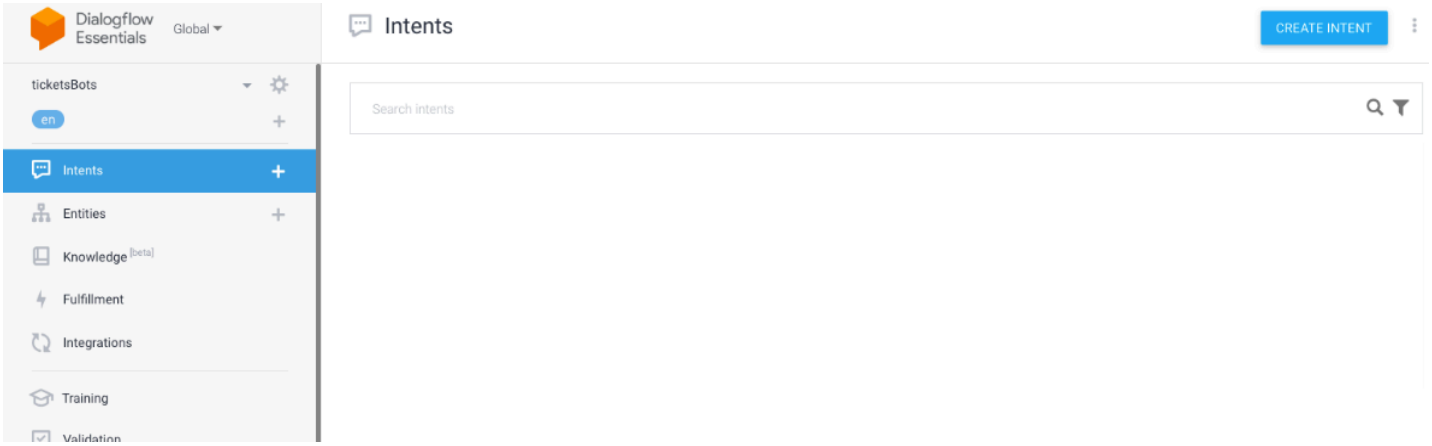
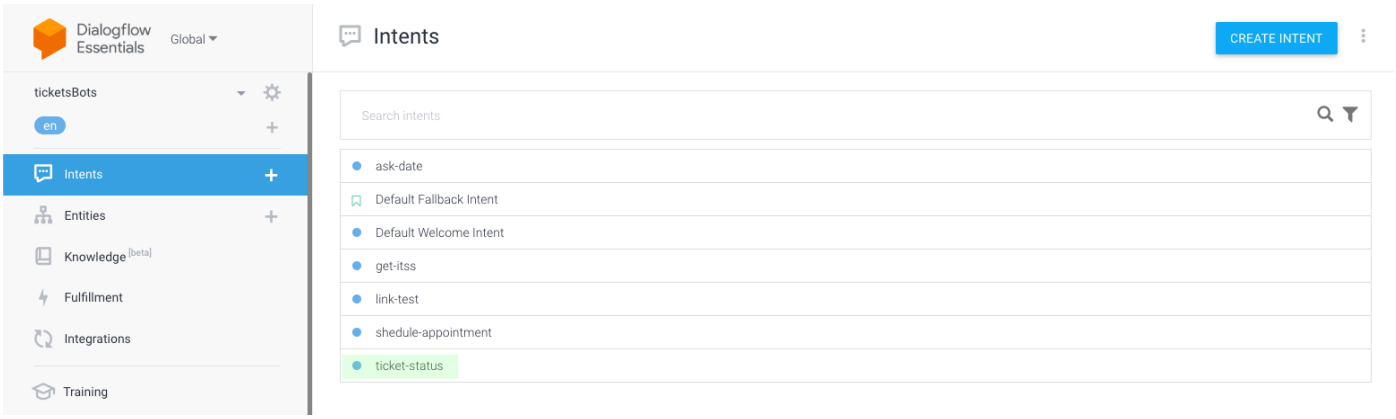


Figura 71. Ejemplo de listado de intenciones para un agente

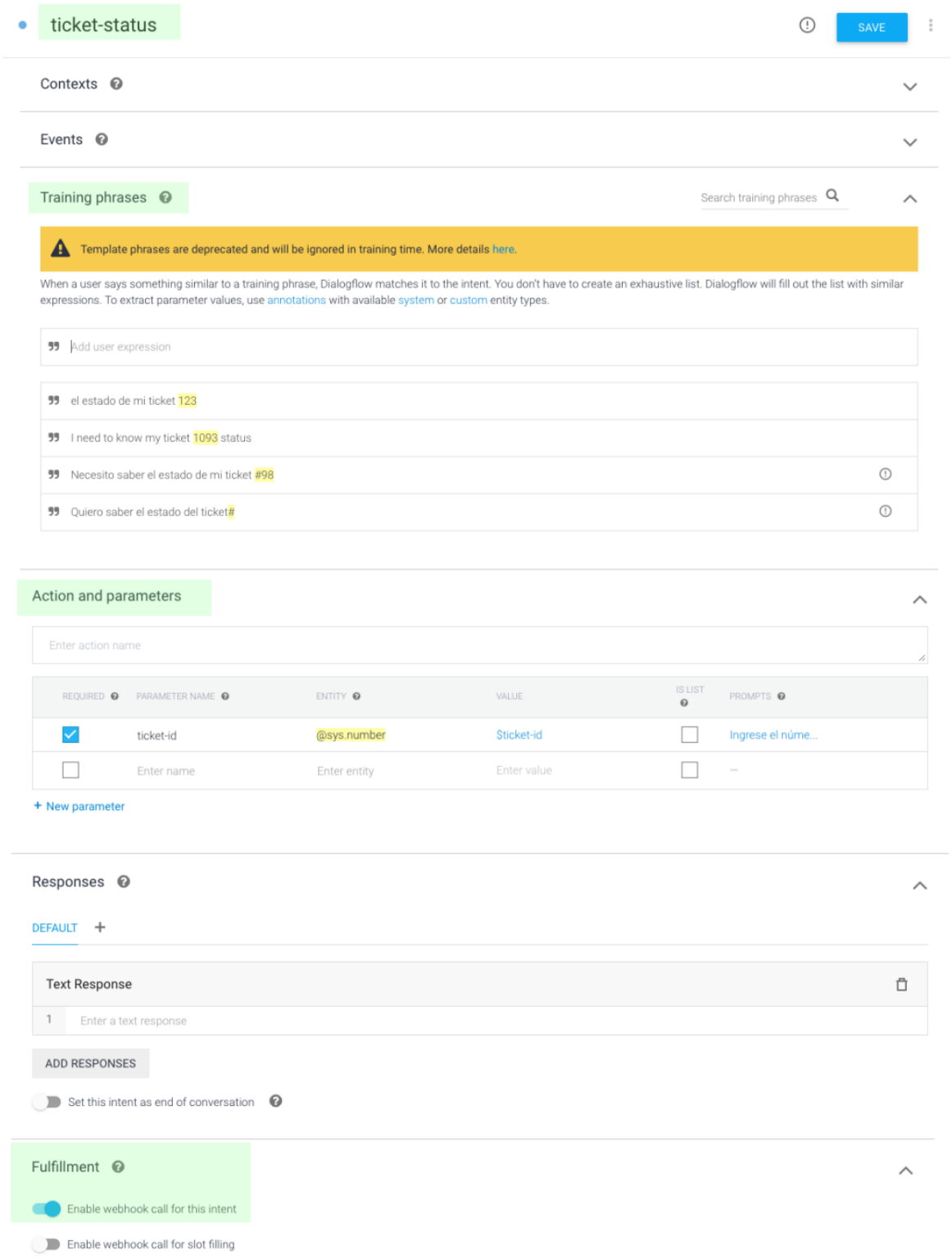


Creación del intent de ticket-status

Los pasos para Crear el intent ticket-status son:

- Se debe setear el nombre de ticket-status
- En la sección "Trainin Phrases" se debe agregar las siguientes frases de entrenamiento (training phrases), asociando los números al parametron ticket-id.
    - el estado de mi ticket 123
    - Necesito saber el estado de mi ticket 756
    - Quiero saber el estado del ticket #876
    - I need to know my ticket 123 status
- En la sección de "Action and Parameters", se debe crear un parámetros requerido asociado a la entity @sys.number con el nombre de ticket-id con el prompt "Ingrese el número de ticket".
- En la sección "Response" NO se debe agregar una respuesta por defecto, ya que se usará fulfillment.
- En la sección de "Fulfillment" se debe habilitar la opción "Enable webhook call for this intent", eso hará que el intent se relacione con el webhook que tendrá las llamadas a las funciones encargadas de procesar la data, se explicará más adelante esta funcionalidad.
- Referencia en la Figura 70.

Figura 72. Creación y configuración de una intención Intent ticket-status

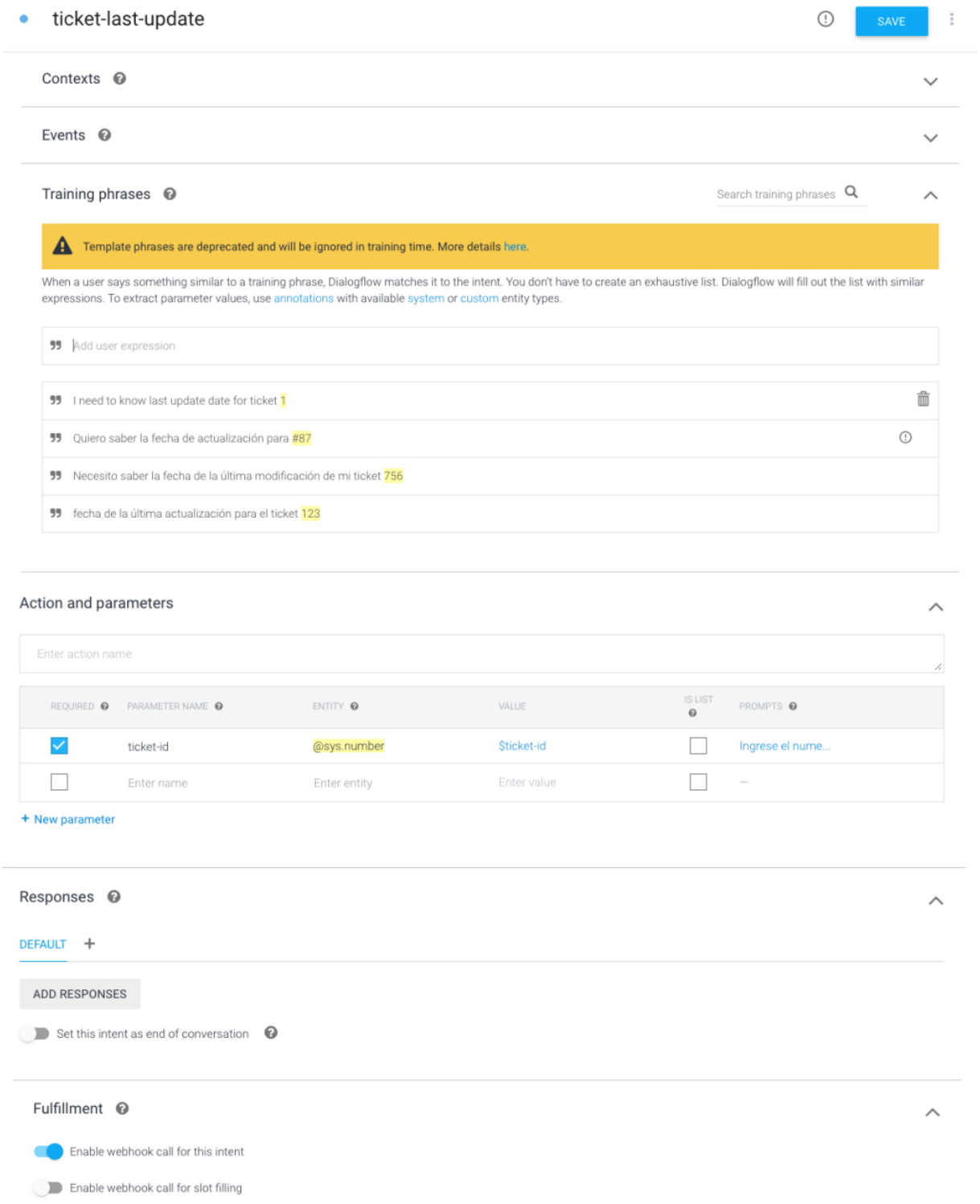


Creación del intent de ticket-last-update

Los pasos para Crear el intent ticket-last-update son:

- Se debe setear el nombre de ticket-last-update
- En la sección "Trainin Phrases" se debe agregar las siguientes frases de entrenamiento (training phrases), asociando los números al parámetro ticket-id.
    - fecha de la última actualización para el ticket 123
    - Necesito saber la fecha de la última modificación de mi ticket 756
    - Quiero saber la fecha de actualización para #876
    - I need to know last update date for ticket 123
- En la sección de "Action and Parameters", se debe crear un parámetros requerido asociado a la entity @sys.number con el nombre de ticket-id con el prompt "Ingrese el número de ticket".
- En la sección "Response" NO se debe agregar una respuesta por defecto, ya que se usará fulfillment.
- En la sección de "Fulfillment" se debe habilitar la opción "Enable webhook call for this intent", eso hará que el intent se relacione con el webhook que tendrá las llamadas a las funciones encargadas de procesar la data, se explicará más adelante esta funcionalidad.
- Referencia en la Figura 71.

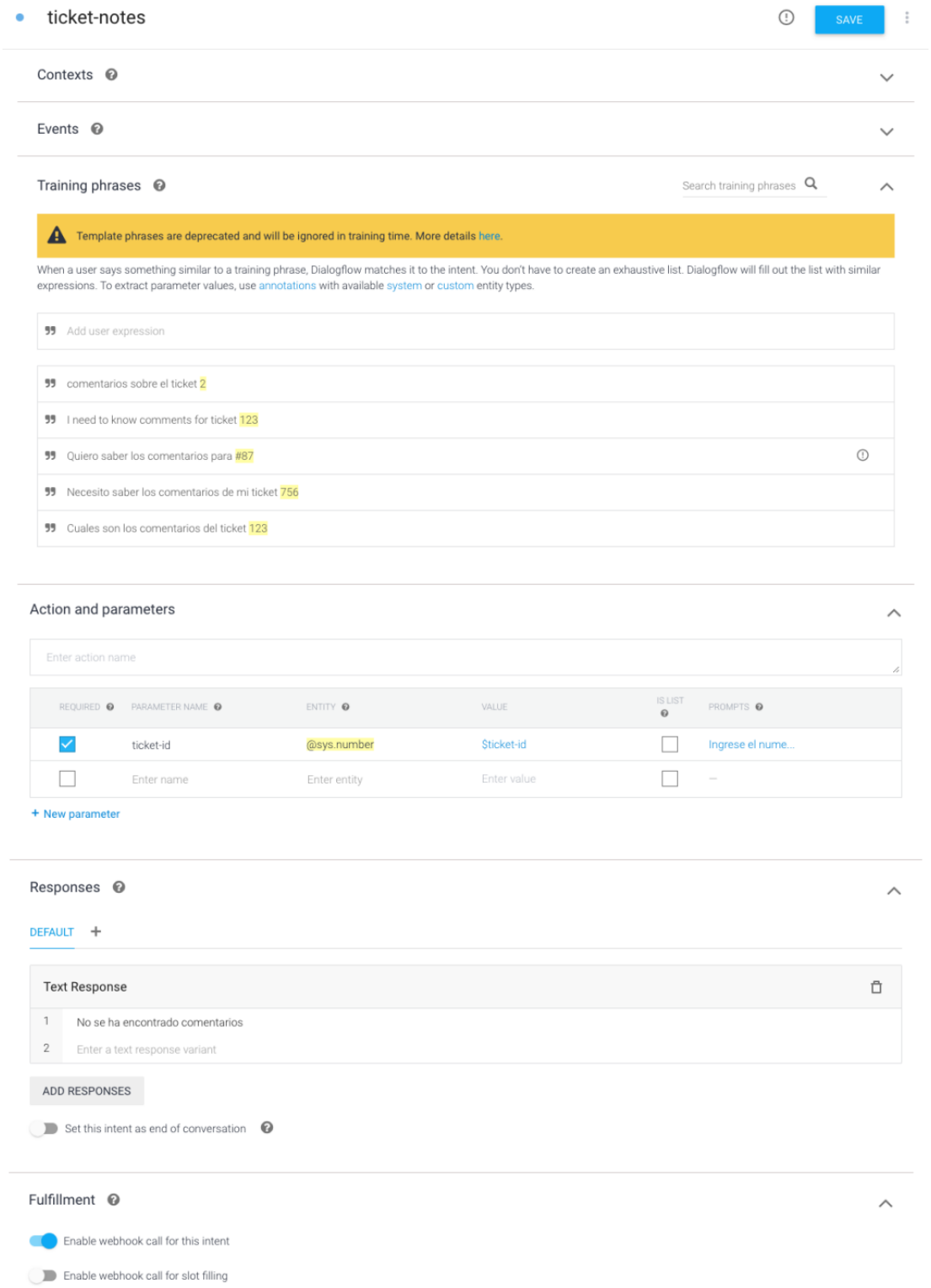
Figura 73. Creación y configuración de una intención Intent ticket-last-update



Creación del intent de ticket-notes

- Los pasos para Crear el intent ticket-notes son:
- Se debe setear el nombre de ticket-last-update
- En la sección "Training Phrases" se debe agregar las siguientes frases de entrenamiento (training phrases), asociando los números al parámetro ticket-id.
    - comentarios sobre el ticket 2
    - Quiero saber los comentarios para #87
    - I need to know comments for ticket 123
    - Necesito saber los comentarios #756
    - Cuáles son los comentarios para 1
- En la sección de "Action and Parameters", se debe crear un parámetros requerido asociado a la entity @sys.number con el nombre de ticket-id con el prompt "Ingrese el número de ticket".
- En la sección "Response" NO se debe agregar una respuesta por defecto, ya que se usará fulfillment.
- En la sección de "Fulfillment" se debe habilitar la opción "Enable webhook call for this intent", eso hará que el intent se relacione con el webhook que tendrá las llamadas a las funciones encargadas de procesar la data, se explicará más adelante esta funcionalidad.
- Referencia en la Figura 72.

Figura 74. Creación y configuración de una intención Intent ticket-notes



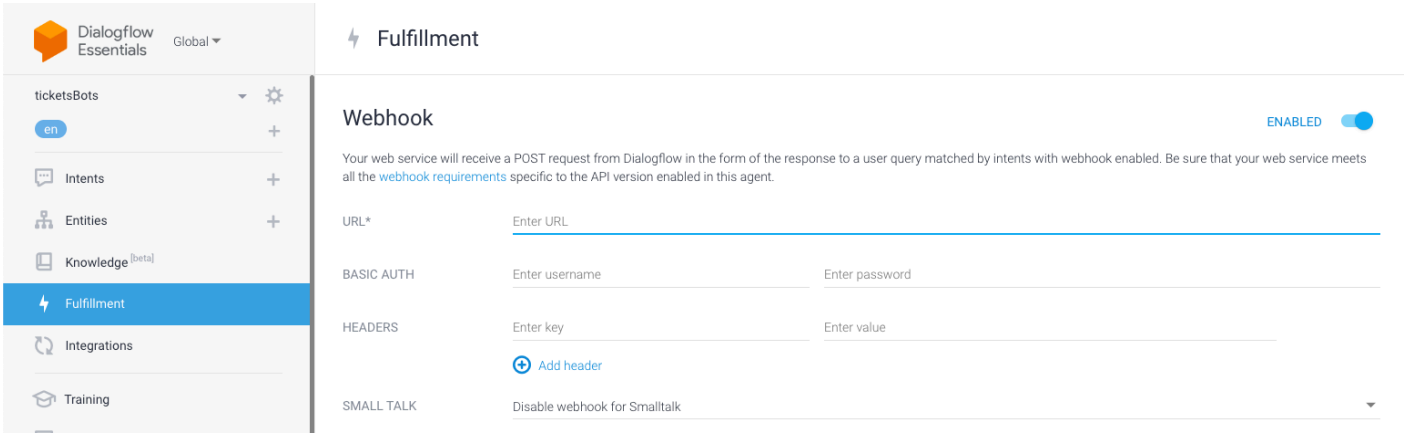
Configuration del webhook

Para configurar el webhook se necesita optener el "Trigger URL" de la cloud function, ese se puede ver abriendo la página de details de la cloud functions.

Para configurar le webhook se debe seguir los siguientes pasos:

- Ir a la sección fulfillment del agente.
- Habilitar la opción webhook.
- Copiar el trigger url de la cloud function.
- Dar clic en el botón "save"
- Para testear el bot referirse a la siguiente sección "test cases para el prototipo implementado".

Figura 75. Configuración de Webhook

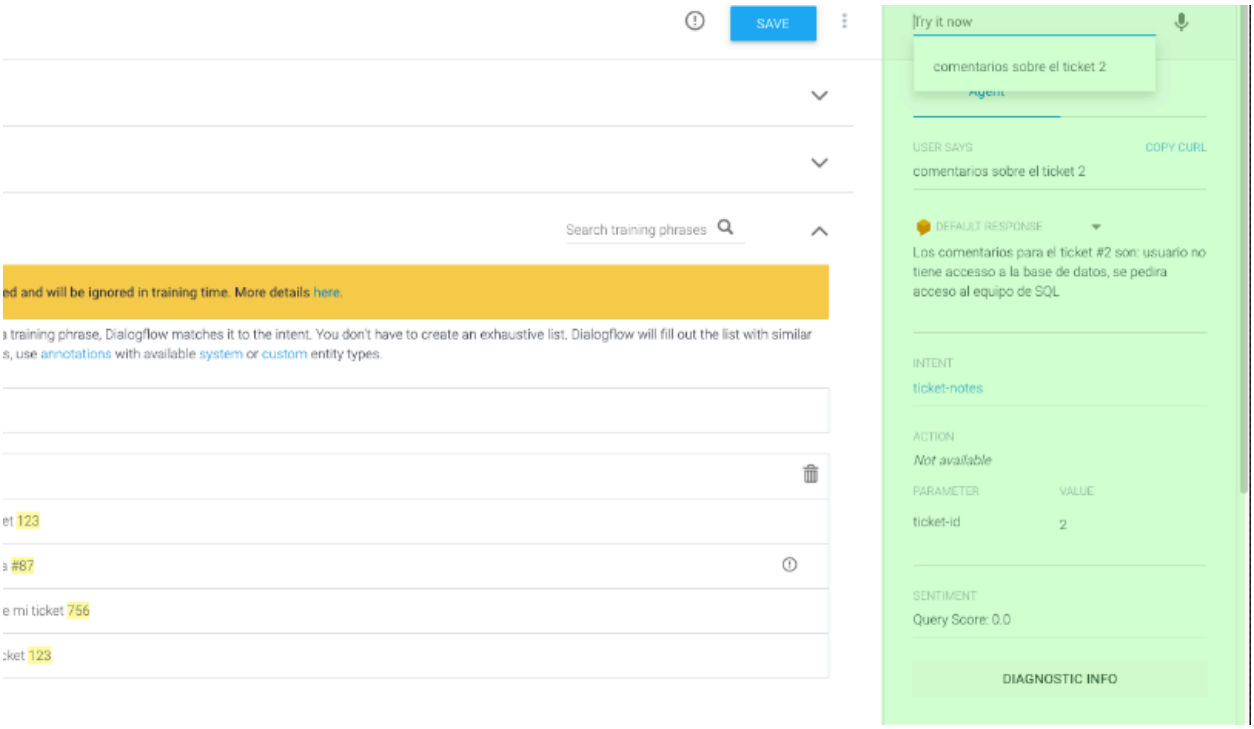


Para testear el chatbot

Para testear el chat se tienen dos opciones:

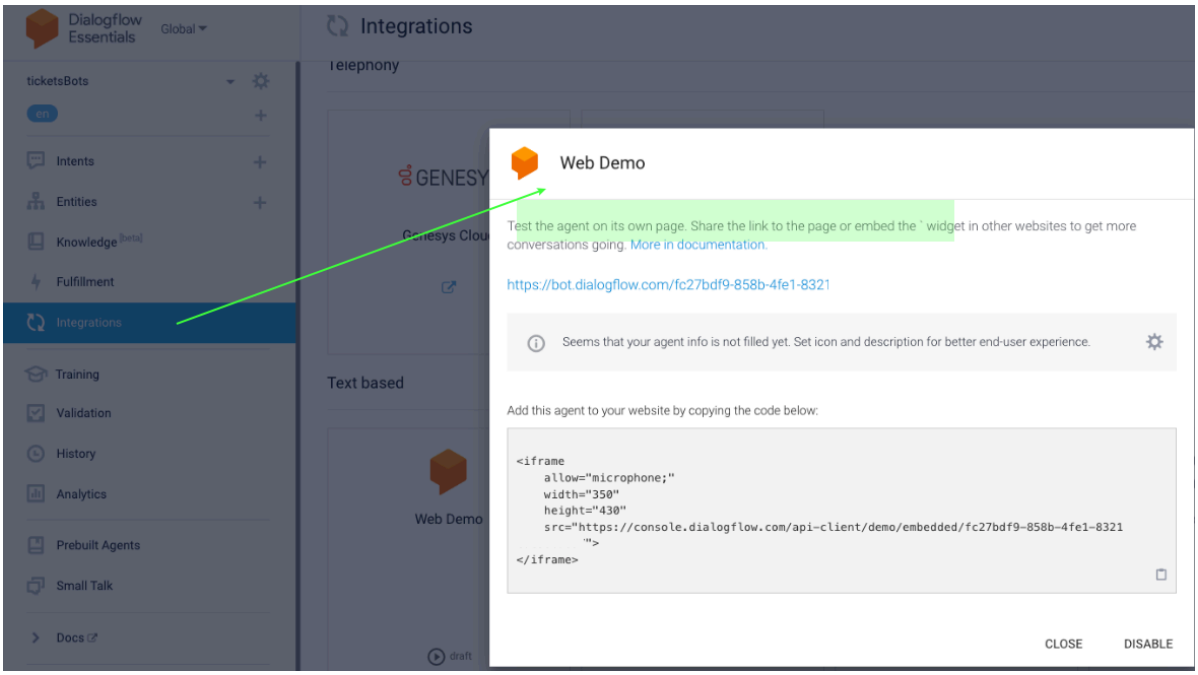
- Utilizar el panel izquierdo incorporado en la dialogflow console

Figura 76. Configuración de Webhook



- Utilizar el link proporcionado por la web demo

Figura 77. Configuración de Webhook



Test Cases para el prototipo de agente implementado (tickestBots)

Para el prototipo implementado se realizaron las validaciones de los siguientes escenarios basados en las "Training Phrases", para determinar ciertos parámetros de consulta a un ticket previamente almacenados en la base de datos, esto con el objetivo de simular el comportamiento de una persona realizando consultas acerca de un ticket.

A continuación, se muestran los casos de prueba ejecutados para validar la calidad de respuesta del prototipo del Chatbot "ticketsBots" por cada intent configurado.

Test Case 1. Consultar estado de un ticket

Caso de prueba 1. Consultar estado de un ticket
Precondiciones: Haber creado un ticket en la base de datos con un status Fixed.
Pasos del caso de prueba: 1. Ejecutar "ticketsBots" en GCP 2. Digitar: **el estado de mi ticket** Respuesta "ticketsBots": Ingrese el número de su ticket 1. Digitar: **1 (un número de ticket previamente creado)** Respuesta "ticketsBots": The status for ticket #1 is Fixed
Resultado esperado: **Para el caso consultar estado de ticket el chatbot debe consultar en la base de datos el estatus del número de ticket proporcionado por el usuario y devolver el estatus que para este caso es Fixed. (Anexo 10)**

Test Case 2. Consultar estado de un ticket que no tiene estado asignado

Caso de prueba 2. Consultar estado de un ticket que no tiene estado asignado
Precondiciones: Haber creado un ticket en la base de datos sin un status definido.
Pasos del caso de prueba: 1. Ejecutar “ticketsBots” en GCP 2. Digitar: **el estado de mi ticket** Respuesta “ticketsBots”: Ingrese el número de su ticket 1. Digitar: **9 (un número de ticket previamente creado sin estatus)** Respuesta “ticketsBots”: El estatus para el ticket #9 no fue encontrado
Resultado esperado: **Para el caso consultar estado de ticket el chatbot debe consultar en la base de datos el estatus del número de ticket proporcionado por el usuario y devolver el estatus que para este caso es no fue encontrado (Anexo 10).**

Test Case 3. Consultar estado de un ticket que no existe

Caso de prueba 3. Consultar estado de un ticket que no existe
Precondiciones: Utilizar un número de ticket que no existe
Pasos del caso de prueba: 1. Ejecutar “ticketsBots” en GCP 2. Digitar: **el estado de mi ticket** Respuesta “ticketsBots”: Ingrese el número de su ticket 1. Digitar: **123 (un número de ticket que no existe)** Respuesta “ticketsBots”: El ticket #123 no existe
Resultado esperado: **Para el caso consultar estado de un ticket que no existe el chatbot debe consultar en la base de datos el estatus del número de ticket proporcionado por el usuario y devolver que el número de ticket a consultar no existe (Anexo 10).**

Test Case 4. Consultar última actualización de un ticket

Caso de prueba 4. Consultar última actualización de un ticket
Precondiciones: Utilizar un número de ticket existente
Pasos del caso de prueba: 1. Ejecutar "ticketsBots" en GCP 2. Digitar: **fecha de última actualización para el ticket** Respuesta "ticketsBots": Ingrese el número de su ticket 1. Digitar: **1 (un número de ticket previamente creado)** Respuesta "ticketsBots": La última fecha de actualización para el ticket #1 fue 04-08-21 14:53:27
Resultado esperado: **Para el caso consultar la última actualización de un ticket el chatbot debe consultar en la base de datos la última actualización del ticket proporcionado por el usuario y devolver la fecha y hora de esta actualización (Anexo 11).**

Test Case 5. Consultar última actualización de un ticket que no existe

Caso de prueba 5. Consultar última actualización de un ticket que no existe
Precondiciones: Utilizar un número de ticket que no existe
Pasos del caso de prueba: 1. Ejecutar "ticketsBots" en GCP 2. Digitar: **fecha de última actualización para el ticket** Respuesta "ticketsBots": Ingrese el número de su ticket 1. Digitar: **123 (un número de ticket que no existe)** Respuesta "ticketsBots": El ticket #123 no existe
Resultado esperado: **Para el caso consultar la última actualización de un ticket que no existe el chatbot debe consultar en la base de datos la última actualización del ticket proporcionado por el usuario y devolver en este caso que el ticket no existe (Anexo 11).**

Test Case 6. Consultar comentarios sobre un ticket

Caso de prueba 6. Consultar comentarios sobre un ticket
Precondiciones: Utilizar un número de ticket que existente con comentarios
Pasos del caso de prueba: 1. Ejecutar “ticketsBots” en GCP 2. Digitar: **comentarios sobre el ticket** Respuesta “ticketsBots”: Ingrese el número de su ticket 1. Digitar: **1 (un número de ticket previamente creado)** Respuesta “ticketsBots”: Los comentarios para el ticket #1 son: “user was inactive...”
Resultado esperado: **Para el caso consultar los comentarios de un ticket el chatbot debe consultar en la base de datos los comentarios del ticket proporcionado por el usuario y devolver en este caso los comentarios hechos sobre ese ticket (Anexo 12).**

Test Case 7. Consultar comentarios sobre un ticket que no existe

Caso de prueba 7. Consultar comentarios sobre un ticket que no existe
Precondiciones: Utilizar un número de ticket que no existe
Pasos del caso de prueba: 1. Ejecutar “ticketsBots” en GCP 2. Digitar: **comentarios sobre el ticket** Respuesta “ticketsBots”: Ingrese el número de su ticket 1. Digitar: **123 (un número de ticket que no existe)** Respuesta “ticketsBots”: El ticket #123 no existe
Resultado esperado: **Para el caso consultar los comentarios de un ticket que no existe el chatbot debe consultar en la base de datos los comentarios del ticket proporcionado por el usuario y devolver en este caso que el ticket no existe (Anexo 12).**

Test Case 8. Consultar comentarios sobre un ticket que no tiene comentarios

Caso de prueba 8. Consultar comentarios sobre un ticket que no tiene comentarios
Precondiciones: Utilizar un número de ticket existente
Pasos del caso de prueba: 1. Ejecutar "ticketsBots" en GCP 2. Digitar: **comentarios sobre el ticket** Respuesta "ticketsBots": Ingrese el número de su ticket 1. Digitar: **9 (un número de ticket que no tiene comentarios)** Respuesta "ticketsBots": No se encontraron comentarios para el ticket #9
Resultado esperado: **Para el caso consultar los comentarios de un ticket que no no tiene comentarios el chatbot debe consultar en la base de datos los comentarios del ticket proporcionado por el usuario y devolver en este caso que el ticket si existe, pero no posee comentarios (Anexo 12).**

CAPÍTULO 5
5. ANÁLISIS ECONÓMICO

5.1 ANÁLISIS COSTO-BENEFICIO EN PROCESOS DE NEGOCIO

Aplicando los costos promedios determinados en la sección 2.5 del presente documento, se conoce cuánto es el pago que se le realiza a cada agente dependiendo de la línea de negocio que maneja. Para las 4 empresas seleccionadas, se tiene la siguiente distribución de agentes por cuenta:

- Promedio de agentes en cuentas o líneas de soporte: 308
- Promedio de agentes en cuentas o líneas de servicio al cliente: 338
- Promedio de agentes en cuentas o líneas de ventas: 108

Adicionalmente, cada línea de negocio posee un tiempo promedio por llamada en la cual pasan los agentes. Cabe destacar que dicho tiempo promedio puede ser menor o mayor a la métrica de "tiempo de interacción" solicitada por la misma cuenta. Para las 4 empresas estudiadas se han obtenido los siguientes tiempos promedios para cada rubro de negocio:

- Promedio de minutos en cuentas o líneas de soporte: 30
- Promedio de minutos en cuentas o líneas de servicio al cliente: 10
- Promedio de minutos en cuentas o líneas de ventas: 15

Con los datos anteriores y con base en una semana laboral de 5 días a 8 horas diarias de trabajo por agente, se construye la siguiente tabla de costos anuales promedio en procesos de servicio al cliente para las empresas objetivo:

Tabla 5. Tabla de costos anuales promedio en procesos de servicio al cliente

Proceso Actual						
Proceso	**Minutos**	**Llamadas por día**	**Minutos por semana**	**Empleados**	**Hrs por proceso Semanales**	**Costo anual**
Llamadas tomadas por agentes de soporte	30	16	2400	308	12320	$3,273,670.40
Llamadas tomadas por agentes de servicio al cliente	10	48	2400	338	13520	$2,791,068.80
Llamadas tomadas por agentes de ventas	15	32	2400	108	4320	$828,921.60
Totales					30160	**$6,893,660.80**

Haciendo un análisis en cuanto a las categorías de servicio al cliente que se brindan en todas las áreas, en promedio, son un aproximado de 43 en total, de las cuales se ha determinado que solo 11 requieren una intervención humana directamente, es decir, los 32 restantes pueden resolverse a través de un chatbot debido a su sencilla complejidad.

Tomando en cuenta que el chatbot podría mitigar el 75% de las categorías de llamada y manteniendo el mismo número de empleados en cada cuenta, se ha construido la siguiente tabla de costos estimados de procesos de servicio al cliente después de la implementación de chatbots:

Tabla 6. Tabla de costos estimados de procesos de servicio al cliente después de la implementación de chatbots.

Proceso Esperado						
Proceso	**Minutos**	**Llamadas por día**	**Minutos por semana**	**Empleados**	**Hrs por proceso Semanales**	**Costo anual**
Llamadas tomadas por agentes de soporte	30	4	600	308	3080	$818,417.60
Llamadas tomadas por agentes de servicio al cliente	10	12	600	338	3380	$697,767.20
Llamadas tomadas por agentes de ventas	15	8	600	108	1080	$207,230.40
Totales					7540	**$1,723,415.20**

Para hacer la cuantificación de valor se ha realizado una diferencia entre el costo anual del proceso actual y del proceso esperado, resumidos en la siguiente tabla:

Tabla 7. Tabla de cuantificación de valor esperado

Cuantificación de valor	Horas anuales	Costo anual
Proceso Actual	30160	$6,893,660.80
Proceso Esperado	7540	$1,723,415.20
Ahorros estimados	**22620**	**$5,170,245.60**

Por tanto, se puede deducir que, en promedio, las empresas se estarían ahorrando un aproximado de 5 millones de dólares anuales en cuanto a procesos de servicio al cliente.

5.2 ANÁLISIS COSTO-BENEFICIO EN INFRAESTRUCTURA TECNOLÓGICA

Para poder realizar una implementación efectiva de un chatbot, se debe contar con una infraestructura tecnológica que permita una disponibilidad de 24 horas y que sea robusta, a fin de que pueda soportar una amplia cantidad de transacciones simultáneas sin incurrir en latencia ni en pérdida de rendimiento. Por tanto, para la implementación de chatbots por cuenta se sugiere el uso de Google Cloud Platform (GCP) y los componentes para creación y personalización de los mismos.

Con base en el punto 5.1, se estima que un chatbot pueda resolver un caso en no más de 15 pasos para cada solicitud que se realice, lo cual si implica la resolución del 75% de casos, se tiene que al día un chatbot tendría un aproximado de 276,840 pasos (o transacciones) o su equivalente de 8,305,200 de transacciones al mes.

Para poder procesar y almacenar todas estas transacciones, se requiere una máquina virtual de GCP tanto para el chatbot, como para el servicio de base de datos de MySQL (conocido como Cloud SQL en GCP), cuyas capacidades deben de ser de 128GB de RAM y 8 procesadores virtuales para él HA (High Availability o disponibilidad alta), y una capacidad de almacenar por lo menos 9 GB de información al mes. Adicionalmente para protección de la información y de la carga del servidor se contrata el servicio de Cloud Armor, a fin de apoyar a la disponibilidad del mismo.
Para mejoras, mantenimiento y soporte del chatbot, lo recomendable en este caso es que se contrate a dos desarrolladores con conocimientos del área e integraciones, un ingeniero de control de calidad y un Scrum Master para la gestión de los diversos proyectos de mejora o de soporte que puedan surgir.

Con la información anterior se construye la siguiente tabla de costos estimados de infraestructura y soporte de chatbot:

Tabla 8. Tabla de costos estimados de infraestructura y soporte de chatbot

Recurso	Costo unitario	Unidades	Costo Mensual	Costo Anual
GCP DialogFlow Service	$0.007	8305200	$58,136.40	$697,636.80
GCP Cloud SQL Service HA Memory	$2.45	128	$313.60	$3,763.20
GCP Cloud SQL Service HA CPU	$14.47	8	$115.76	$1,389.12
GCP Cloud SQL HA Storage Service	$0.34	9	$3.06	$36.72
GCP Cloud Functions	$0.40	6	$2.40	$28.80
GCP Cloud Storage Service	$0.02	9	$0.18	$2.16
GCP Cloud Armor Service	$12.05	1	$12.05	$144.60
Data Integration Specialist	$2,000.00	2	$4,000.00	$48,000.00
QA Engineer	$1,600.00	1	$1,600.00	$19,200.00
Scrum Master	$1,600.00	1	$1,600.00	$19,200.00
Total			$65,783.45	$789,401.40

Por tanto, se puede deducir que, en promedio, las empresas estarían gastando un aproximado de $789 mil dólares anuales en costos de chatbot, con los cuales los ahorros mismos que se consigan estarían pagando estos costos y se tendría todavía un ahorro anual en cuanto a procesos.

5.3 ANÁLISIS DE IMPACTO ECONÓMICO EN EL CLIENTE

A la hora de hacer un prorrateo de los costos totales, se ha determinado que el costo para la cuenta como tal de mantener un chatbot es de aproximadamente $0.12 por caso (tomando en cuenta que se usaron las 15 transacciones máximas del caso), logrando que la cuenta pueda ofrecer planes de servicio de chatbot a un precio igual o más bajo que los planes de voz a fin de fomentar competitividad en el mercado y garantizar la buena experiencia del cliente, dado que el hecho de poder solventar algún problema en un tiempo menor que una llamada y con una menor cantidad de pasos, ayudando a que la empresa cliente posea una retroalimentación positiva de sus clientes en cuanto al servicio.

En cuanto a las empresas, el impacto en la reducción de costos de operación es visible, lo que también ayuda a que los recursos humanos puedan ser destinados a otras áreas que se necesiten sin incurrir en gastos adicionales y adicionalmente, debido a que la carga laboral puede ser más ligera para los agentes, se garantiza una menor rotación de personal dentro de la compañía, fomentando también que la empresa misma posea más recursos económicos para destinarlos al crecimiento y expansión de la misma.

CAPÍTULO 6
6. CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

- La investigación ayudó a obtener una respuesta de cómo las máquinas son capaces de entender e interpretar el lenguaje humano a través del procesamiento del lenguaje natural, la inteligencia artificial y el aprendizaje automático presentando algunas técnicas y algoritmos que formar parte del core de las plataformas que ofrecen servicios y soporte para la creación de conversaciones automatizadas (chatbots).
- La investigación del enfoque técnico además de identificar distintos proveedores que proporcionan herramientas para la implementación de chatbots, también permitió identificar que las nuevas tecnologías que apuntan al posicionamiento de la Nube actualmente están incluyendo la IA (chatbots) como un elemento más haciendo notar la importancia de estas nuevas tecnologías.
- La investigación ayudó a proporcionar una guía rápida para la creación de un chatbot de servicio al cliente básico utilizando las tecnologías ofrecidas por Google como GCP y Dialogflow, en conjunto con el lenguaje de programación NodeJS para el procesamiento del backend a través de Google Cloud Function y el motor de base de datos MySql a mediante Cloud SQL.
- La investigación ayudó a dar a conocer el enfoque tecnológico de dos proveedores para la creación de chatbots con un gran posicionamiento en el mercado actual, presentando los elementos, servicios y tecnologías utilizadas por ambos, así como también un análisis comparativo entre las dos tecnologías presentadas.
- La investigación de este trabajo de graduación permitió conocer fundamentos teóricos de algunas de las tecnologías utilizadas para la creación de chatbots ayudando a tener una mejor comprensión de sus elementos, arquitectura, funcionamiento, utilización e implementación.
- La investigación ayudó a identificar la necesidad principal de las empresas de servicio al cliente en El Salvador, la cual es cumplir satisfactoriamente las demandas de las empresas a las cuales les brindan soporte para garantizar su continuidad y la retribución económica que implica en la empresa. Para medir el nivel de cumplimiento se usan diversos indicadores, los más

relevantes vienen siendo tiempos de interacción, existencia de escalaciones y satisfacción del cliente o usuario final.

- El costo de mantenimiento de un chatbot a gran escala como es en una empresa BPO es bastante accesible a comparación de los costos de estar pagando a agentes para poder realizar tareas o solventar casos sencillos que el mismo solicitante puede realizar con ayudas automatizadas.
- Es importante conocer el estimado de transacciones esperadas a la hora de implementar las tecnologías de chatbots para poder dimensionar la arquitectura sobre la cual estarán montados y sobre todo los costos asociados a infraestructura.

6.2 RECOMENDACIONES

- Para una tesis investigativa sobre tecnología se recomienda consultar fuentes con información y descubrimientos recientes para poder brindar conocimiento actual sobre los temas a ser expuestos en el documento entre algunas de las fuentes están: libros, tesis doctorales, tesis de maestrías, artículos científicos, artículos tecnológicos de investigadores, artículos tecnológicos de empresas con un buen posicionamiento en el mundo de la tecnología, documentaciones oficiales de las tecnologías y más.
- Se debe considerar que la prueba de concepto presentada se creó tomando como base un caso de uso de un escenario real delimitando a su forma más simple.
- Los profesionales deben considerar los conocimientos de nuevas tecnologías basadas en la nube para la implementación de chatbots de última generación.
- Es importante seleccionar las tecnologías a utilizar para implementar chatbots tomando en cuenta el presupuesto y la infraestructura tecnológica que posea la organización objetivo.
- Cuando se tienen diferentes costos en un mismo rubro, es necesario promediar para poder tener una estimación que pueda abarcar todo el espectro del mercado objetivo, que en este caso ha sido el negocio de BPO.

REFERENCIAS

Alvarez, A. (2020). Modelo de Chatbot de Inteligencia Artificial articulado con el Business Process Management (BPM) del Ministerio de Tecnologías de la Información y las Comunicaciones (MinTIC) para el área de la Subdirección para la Industria de Comunicaciones (SICom). Universidad EAN. From https://repository.ean.edu.co/bitstream/handle/10882/10107/FloridoAlberto2020.pdf;jsessionid=DA50660126EBA4AB3E11C1C45ABEAF9A?sequence=1

Bello, E. (2021, 09). Qué es un Chatbot y por qué implementarlos en tu negocio. From https://www.iebschool.com/blog/implementar-chatbots-negocios-tecnologia/

Benate Mendoza, J. (2020, 09). CHATBOT PEDIÁTRICO PARA LA ORIENTACIÓN SOBRE APENDICITIS AGUDA BASADO EN NLP Y MODELOS DE CLASIFICACIÓN SUPERVISADA. Universidad de Lima. From https://repositorio.ulima.edu.pe/bitstream/handle/20.500.12724/14046/Benate_Mendoza_Juan_Domingo.pdf?sequence=1&isAllowed=y

Boban, I., Doko, A., & Gotovac, S. (2020). Sentence retrieval using Stemming and Lemmatization with Different Length of the Queries. ASTES. From https://elib.dlr.de/139260/1/ASTESJ_050345.pdf

Boden, M. (2017). Inteligencia Artificial. Turner.

Borglin, J. (2011). Classification of hand movements using multi-channel EMG. Chalmers university of technology. Sweden. From https://publications.lib.chalmers.se/records/fulltext/159500.pdf

csub. (2013). LA LINGÜÍSTICA. California State University. From https://www.csub.edu/modlang/department/Spanish/LINGUISTICS/TEMA%201%20MA.pdf

Dharmendra, S., & Suresh, J. (2015). Evaluation of Stemming and Stop Word Techniques on Text Classification Problem. Mewar University. From https://www.isroset.org/pub_paper/IJSRCSE/ISROSET-IJSRCSE-NCETITM-IT-1.pdf

Díaz, A. (2017). El costo-beneficio como herramienta de decisión en la inversión en actividades científicas. Universidad de la Habana. From http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2073-60612017000200022

Divya, K., Niveditha, N., Divya, B., & Siddhartha, B. (2021). An Interpretation of Lemmatization and Stemming in Natural Language Processing. Journal of University of Shanghai for Science and Technology. From https://www.researchgate.net/publication/348306833_An_Interpretation_of_Lemmatization_and_Stemming_in_Natural_Language_Processing

DOBBINS, S. (2018, 01). Re-learning English: Pluralization and Lemmatization. Searching for 伯樂. From https://searchingforbole.com/2018/01/10/lemmatizer/

Drashtikumari, K. (2021). Multi-Label Emotion Classification Using Machine Learning and Deep Learning Methods. Laurentian University. From https://zone.biblio.laurentian.ca/bitstream/10219/3662/1/Thesis%20FINAL%20-%20Drashtikumari%20Kher_Feb22.pdf

ecma International. (2022). ECMAScript® 2023 Language Specification. ecmac262. From https://tc39.es/ecma262

Fernández, Y. (2017, May 27). *Así era ELIZA, el primer bot conversacional de la historia.* From Xataka: https://www.xataka.com/historia-tecnologica/asi-era-eliza-el-primer-bot-conversacional-de-la-historia

Figueroa Sacoto, S. (2021, July 14). Diseño y desarrollo de un chatbot usando redes neuronales recurentes y procesamiento de lenguaje natural para tiendas virtuales en comercio electronico. Universidad politécnica salesiana. From https://dspace.ups.edu.ec/bitstream/123456789/21195/1/UPS-CT009315.pdf

Gallé, M. (2019, 11). Investigating the Effectiveness of BPE: The Power of Shorter Sequences. NAVER LABS Europe. From https://aclanthology.org/D19-1141.pdf

GAO, H. (2008). Business process outsourcing industry: An innovative enterprise perspective - ProQuest. UNIVERSITY OF MASSACHUSETTS LOWELL. From https://www.proquest.com/openview/1d7e0cdc8ac4d446a27db19f2a9bd3c0/1?pq-origsite=gscholar&cbl=18750

García Mendiola, J. (2008). Sistemas de símbolos en inteligencia artificial. Universidad Autónoma de Querétaro.

Ghoche, S. (2015, 05). Real-Time Tf-idf clustering using Simhash, approximate nearest neighbors, and DBCAN. Harvard University. From

https://dash.harvard.edu/bitstream/handle/1/38811431/GHOCHE-SENIORTHESIS-2016.pdf?sequence=3&isAllowed=y

Rodrigo, G. (2007). El Test de Turing: dos mitos, un dogma1. e Universiteit Leuven. From https://repositorio.uchile.cl/bitstream/handle/2250/143626/El%20Test%20de%20Turing.pdf?sequence=1&isAllowed=y

Google. (2022). Dialogflow. Google. From https://cloud.google.com/dialogflow

Goyal, C. (2021, 06). Named Entity Recognition | Guide to Master NLP (Part 10). Analytics Vidhya. From https://www.analyticsvidhya.com/blog/2021/06/part-10-step-by-step-guide-to-master-nlp-named-entity-recognition/

Gupta, P. (2018, 06). Decision Trees in Machine Learning - Towards Data Science. towards data science.

Gutiérrez Siliceo, J. (2019, 06). DESARRROLLO DE CHATBOTS CON ENTORNOS DE CÓDIGO ABIERTO. UNIVERSIDAD DE CANTABRIA. From https://docplayer.es/171766315-Trabajo-fin-de-grado-desarrollo-de-chatbots-con-entornos-de-codigo-abierto.html

Hurwitz, J., & Kirsch, D. (2018). Machine Learning For Dummies®, IBM Limited Edition.

Hvitfeldt, E., & Silge, J. (2021). Supervised Machine Learning for Text Analysis in R. CRC Press.

IBM. (2018). La evolución de la automatización de procesos. From https://www.ibm.com/downloads/cas/RJGDMZ2D

IBM. (2021, 08). Neural Networks. IBM Cloud Education. From https://www.ibm.com/cloud/learn/neural-networks

IBM. (2021, 11). Machine Learning. From https://www.ibm.com/cloud/learn/machine-learning

IBM. (2020). Natural Language Processing (NLP). IBM Cloud Education. From https://www.ibm.com/cloud/learn/natural-language-processing

Irwin, M. (2008). Probabilidad Y Estadística Para Ingenieros. *1*. Editorial Reverté.

Jaideepsinh K., R. (2016, 09). Stop-Word Removal Algorithm and its Implementation for Sanskrit Language. University Ahmedabad. From https://www.ijcaonline.org/archives/volume150/number2/raulji-2016-ijca-911462.pdf

Janarthanam, S. (2021, 12). Architecture of a Conversational AI system — 5 essential building blocks. Analytics Vidhya. From https://medium.com/analytics-vidhya/architecture-of-a-conversational-ai-system-5-essential-building-blocks-2f149466da1d

Jassova, B. (2020). Chatbots y Procesamiento de Lenguaje Natural: La Guía Definitiva. From https://landbot.io/es/blog/chatbots-procesamiento-lenguaje-natural

Jurafsky, D. (2022). Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Pearson India.

Krueger, E. (2022, 01). Natural Language Processing First Steps: How Algorithms Understand Text. NVIDIA Developer Blog. From https://developer.nvidia.com/blog/natural-language-processing-first-steps-how-algorithms-understand-text/

Kurasinski, L. (2020). Machine Learning explainability in text classification for Fake News detection. Malmö University. From https://www.diva-portal.org/smash/get/diva2:1479925/FULLTEXT01.pdf

Luckert, M., & Schaefer-kehnert, M. (2015). Using Machine Learning Methods for Evaluating the Quality of Technical Documents. From https://www.diva-portal.org/smash/get/diva2:920202/FULLTEXT01.pdf

MDN. (2022, 04). JavaScript | MDN. MDN web docs. From https://developer.mozilla.org/en-US/docs/Web/JavaScript

Microsoft. (2022). Bot Framework Composer documentation. Microsoft. From https://docs.microsoft.com/en-us/composer/

Microsoft. (2022). LUIS. From https://www.luis.ai/

Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing Order into Texts. University of North Texas. From https://aclanthology.org/W04-3252.pdf

Mueller, J., & Massaron, L. (2021). Machine Learning For Dummies. Wiley.

Munarriz, L. (1994). fundamentos de inteligencia artificial. Universidad de Murcia.

Machine Learning. (2019). Netflix Research. From https://research.netflix.com/research-area/machine-learning

NLTK :: Natural Language Toolkit. (2022). NLTK Documentation. From https://www.nltk.org/

Nodejs. (2022). nodejs.org. From https://nodejs.org/en/

Norberg, R. (2013). Information Data Management for the Future of Communication. Luleå University of Technology. From https://www.diva-portal.org/smash/get/diva2:1018159/FULLTEXT02

npm. (2022). npm Docs. From https://docs.npmjs.com

Pai, A. (2021, 07). What is Tokenization | Tokenization In NLP. Analytics Vidhya. From https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/#:%7E:text=Tokenization%20is%20a%20common%20task%20in%20Natural%20Language%20Processing%20(NLP).&text=Tokenization%20is%20a%20way%20of,words%2C%20characters%2C%20or%20subwords.

Peng, Y., A Flach, P., & Soares, C. (2002). Decision Tree-Based Data Characterization for MetaLearning. Department of Computer Science, University of Bristol, UK. From http://kt.ijs.si/Branax/IDDM-2002_proceedings/Peng_new.pdf

Pérez León, E., & Rojas Arévalo, D. (2019). Impacto de la inteligencia artificial en las empresas con un enfoque global. UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS.

Pluralsight. (2016). Learn the JavaScript basics. From https://www.javascript.com/learn

Porter, M. (2006). Porter Stemming Algorithm. tartarus.org. From https://tartarus.org/martin/PorterStemmer/

Provilkov, I., Emelianenko, D., & Voita, E. (2020, 07). BPE-Dropout: Simple and Effective Subword Regularization. Moscow Institute of Physics and Technology, Russia. From https://aclanthology.org/2020.acl-main.170.pdf

Qaiser, S., & Ali, R. (2018, 07). Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. Universiti Utara Malaysia. From https://www.researchgate.net/publication/326425709_Text_Mining_Use_of_TF-IDF_to_Examine_the_Relevance_of_Words_to_Documents

Ramos, J. (2021, 09). Using TF-IDF to Determine Word Relevance in Document Queries. Rutgers University. From https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.1424&rep=rep1&type=pdf

Rantakari, L. (2010). Governance in business process outsourcing: case study on call center outsourcing. HELSINGIN KAUPPAKORKEAKOULU. From

https://aaltodoc.aalto.fi/bitstream/handle/123456789/392/hse_ethesis_12260.pdf?sequence=1&isAllowed=y

Russell, S., & Norvig, P. (1995). Artificial Intelligence: A Modern Approach.

Salian, I. (2018). SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning? nvidia. From https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/

Sánchez, E., Cazares, S., & Antuna, R. (2015). Probabilidad y Estadística 1. Grupo Editorial Patria.

Santos, J. (2019, 08). What is Node.js and how does it differ from a browser. From https://medium.com/swlh/what-is-node-js-and-how-does-it-differ-from-a-browser-ddebef00cbd9#:~:text=Unlike%20the%20browser%20where%20Javascript,can%20execute%20software%20and%20more.

Sathelly, B. (2018). An Artificial Neural Network Approach to Predict Liver Failure Likelihood. The University of Toledo. From https://etd.ohiolink.edu/apexprod/rws_etd/send_file/send?accession=toledo1535112414430542&disposition=inline

Serrano Gómez, W. (205, 06). ¿Qué constituye a los lenguajes natural y matemático? UPEL. From http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S1317-58152005000100004

Shalev-Shwart, S., & David, B. (2014). understanding machine learning from theory to algorithms. Cambridge University Press.

SHENAVARI, A. (2018, 08). Machine Learning methods to detect improper and irrelevant citations. UNIVERSITY OF STAVANGER. From https://uis.brage.unit.no/uis-xmlui/bitstream/handle/11250/2564360/Masters_Thesis%20.pdf?sequence=1&isAllowed=y

Shibata, Y., Kida, T., Fukamachi, S., Takeda, M., Shinohara, T., & Shinohara, A. (1999, 09). Byte Pair Encoding: A Text Compression Scheme That Accelerates Pattern Matching. Kyushu University.

Straková, J. (2017). Neural Network Based Named Entity Recognition. Charles University. From https://ufal.mff.cuni.cz/~strakova/doctoral_thesis.pdf

TALUKDER, A. (2011). Face characterisation based on Gabor filters, Bayes Rule and AdaBoost. UNIVERSITAT ROVIRA I VIRGILI. From

https://deim.urv.cat/~francesc.serratosa/2011_06_20_Abmnurul_Taluckder_MESISI_MasterThesis.pdf

Tamir, D. (2020). What Is Machine Learning (ML)? berkeley School of Information. From https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/

Tan, C. (1997, 02). Artificial Neural Networks: A Financial Tool As Applied in the Australian Market. Bond University. From https://pure.bond.edu.au/ws/portalfiles/portal/30065798/TAN_THESIS97.PDF

Walker, M., Litman, D., Kamm, C., & Abella, A. (1997, 07). PARADISE: A Framework for Evaluating Spoken Dialogue Agents. AT&T Labs Research. From https://aclanthology.org/P97-1035.pdf

Wang, W., & Tang, Y. (2016). Improvement and Application of TF-IDF Algorithm in Text Orientation Analysis. Hebei University of Engineering.

Widjaja, M., & Hansun, S. (2015). Implementation of Modified Porter Stemming Algorithm to Indonesian Word Error Detection Plugin Application. International Journal of Technology. From https://www.researchgate.net/publication/277346460_Implementation_of_Modified_Porter_Stemming_Algorithm_to_Indonesian_Word_Error_Detection_Plugin_Application

Yang, D., Stewart, I., Chen, J., Singh, N., & Yang, J. (2020). Natural Language Processing. Georgia Tech College of Computing. From https://www.cc.gatech.edu/classes/AY2020/cs7650_spring/

Zhang, Y. (2019, 10). Named Entity Recognition for Social Media Text. Uppsala University. From https://uu.diva-portal.org/smash/get/diva2:1366031/FULLTEXT01.pdf

Zhang, Y., & Teng, Z. (2021). Natural Language Processing, A Machine Learning Perspective. Cambridge University Press.

innovacionate.com. (2020, 05). From Niveles de soporte informáticos: https://www.innovacionate.com/centro-de-ayuda/sabias-que/ref-hardware/128-niveles-soporte

da Silva, D. (2021, 02). *¿Cuántos niveles de soporte técnico deben tener las empresas?* From Zendesk: https://www.zendesk.com.mx/blog/niveles-soporte-tecnico/

da Silva, D. (2021, 02). *¿Que es un ticket de soporte?* From Zendesk: https://www.zendesk.com.mx/blog/ticket-de-soporte-que-es/

Portillo, E. (2020, 05). *Niveles de soporte informáticos.* From innovacionate.com: https://www.innovacionate.com/centro-de-ayuda/sabias-que/ref-hardware/128-niveles-soporte

IBM. (2022). *What is a cloud database*. From IBM.com: https://www.ibm.com/cloud/learn/what-is-cloud-database

Peris, R. (2021). *Chatbot: ¿Qué es, para qué sirve y cómo funcionan?* From Bloo.media: https://bloo.media/blog/por-que-implementar-chatbot-en-tu-estrategia-de-marketing/

Doudchitzky, E. (2021, 02 05). *Chatbot: qué es y cómo puede ayudar a las empresas*. From chattigo: https://blog.chattigo.com/chatbot-que-es-y-como-puede-ayudar-a-las-empresas

Mathieson. (2018). *Chatbots demonstrate unexpected range of uses, from mental health to education*. From https://www.computerweekly.com/feature/Chatbots-demonstrate-unexpected- range-of-uses-from-mental-health-to-education

ANEXOS

Anexo 1. Script de creación de base de datos technical_support

```
CREATE DATABASE technical_support;
```

Anexo 2. Script de creación de tabla Tickets

```
CREATE TABLE Tickets (
   id int NOT NULL AUTO_INCREMENT,
   description text,
   notes text,
   reporter varchar(150),
   contact_email varchar(150),
   contact_phone varchar(50),
   status varchar(50),
   resolver varchar(150),
   resolver_email varchar(50),
   created_at timestamp DEFAULT NOW(),
   updated_at timestamp DEFAULT NOW(),
   PRIMARY KEY (id)
);
```

Anexo 3. Script de inserción para tabla Tickets

```
INSERT INTO Tickets (description, notes, reporter, contact_email, contact_phone,
status, resolver, resolver_email, created_at, updated_at )
VALUES (
   'I cant login to my email', 'user was inactive by inactivity, account was acitivated again', 'lolo rodriguez',
   ' lolo.test@gmail.com', ' 1-800-310-8189', 'Fixed', 'ana rodriguez', 'ana.suport@gmail.com',
```

```
   (SELECT timestamp('2022-01-01 14:53:27') - INTERVAL FLOOR( RAND( ) * 366) DAY), (SELECT timestamp('2022-01-01 14:53:27') - INTERVAL FLOOR( RAND( ) * 366) DAY)
   );

INSERT INTO Tickets (description, notes, reporter, contact_email, contact_phone, status, resolver, resolver_email, created_at, updated_at )
VALUES (
   'No me puedo conectar a la base de datos sales_advertisers', 'usuario no tiene accesso a la base de datos, se pedira acceso al equipo de SQL', 'karen cornejo',
   'karen.test@gmail.com', '1-800-310-8189', 'In Progress', 'ana rodriguez', 'ana.suport@gmail.com',
   (SELECT timestamp('2022-01-01 14:53:27') - INTERVAL FLOOR( RAND( ) * 366) DAY), (SELECT timestamp('2022-01-01 14:53:27') - INTERVAL FLOOR( RAND( ) * 366) DAY)
 );

INSERT INTO Tickets (description, notes, reporter, contact_email, contact_phone, status, resolver, resolver_email, created_at, updated_at )
VALUES (
      'prod environment is crashing for salesAdvertisers system', null, 'Evelyn Montalvo',
      'evelyn.test@gmail.com', '1-800-412-8189', null, 'ana rodriguez', 'ana.suport@gmail.com',
      (SELECT timestamp('2022-01-01 14:53:27') - INTERVAL FLOOR( RAND( ) * 366) DAY), null
   );
```

Anexo 4. Código del package.json para la cloud function que se conecta con Dialogflow

```
{
  "name": "dialogflowFirebaseFulfillment",
  "description": "This is the default fulfillment for a Dialogflow agents using Cloud Functions for Firebase",
  "version": "0.0.1",
  "private": true,
  "license": "Apache Version 2.0",
  "author": "Google Inc.",
  "engines": {
    "node": "10"
  },
  "scripts": {
    "lint": "semistandard --fix \"**/*.js\"",
    "start": "firebase serve --only functions:dialogflowFirebaseFulfillment",
    "deploy": "firebase deploy --only functions:dialogflowFirebaseFulfillment"
  },
  "dependencies": {
```

```
    "actions-on-google": "^2.2.0",
    "firebase-admin": "^5.13.1",
    "firebase-functions": "^2.0.2",
    "dialogflow": "^0.6.0",
    "dialogflow-fulfillment": "^0.5.0",
    "dotenv": "^9.0.2",
    "express": "^4.17.1",
    "moment": "^2.29.0",
    "mysql2": "^2.3.3",
    "nodemon": "^2.0.7"
  }
}
```

Anexo 5. Template de cloud function para conectarse con Dialogflow via webhook

```
/**
* Copyright 2017 Google Inc. All Rights Reserved.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*      http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

'use strict';

const functions = require('firebase-functions');
const admin = require('firebase-admin');
admin.initializeApp();
const {WebhookClient} = require('dialogflow-fulfillment');
const mysql = require('mysql2/promise');
const moment = require('moment');
const process = require('process');
```

```
const dotenv = require('dotenv');
dotenv.config();

exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request, response) => {
   const agent = new WebhookClient({ request, response });
   console.log("Parameters", agent.parameters);

   function welcome(agent) {
      agent.add(`Welcome to my agent!`);
   }

   function fallback(agent) {
      agent.add(`I didn't understand`);
      agent.add(`I'm sorry, can you try again?`);
   }

   let intentMap = new Map();
  // format
  //  intentMap.set('#INTENT_NAME', #FUNCTION_NAME);
   intentMap.set('Default Welcome Intent', welcome);
   intentMap.set('Default Fallback Intent', fallback);
   intentMap.set('ticket-status', getStatus);
   intentMap.set('ticket-last-update', getLastUpdate);
   intentMap.set('ticket-notes', getNotes);
   agent.handleRequest(intentMap);
 });
```

Anexo 6. Código de conexión a la base de datos alojada en Cloud SQL de GCP

```
const mysqlConnect = async ()=> {
      try {
         const con = await mysql.createConnection({
            socketPath: '#CONNECTION_NAME',
            user: '#USER_NAME',
            password: '#CLAVE',
            database: 'technical_support',
         });

         return con;
```

```
    } catch (error) {
      console.log(`Error: ${error}`);
      return null;
    }
  }
```

Anexo 7. Código para la el intent ticket-status

```
const getStatus = async (agent) => {
    const id = agent.parameters['ticket-id'];//parameter name config in intent
    const con = await mysqlConnect();
    if (con != null){
      try {
        const [rows, fields] = await con.execute('SELECT status FROM Tickets WHERE id = ?', [id]);

        if(rows == null || rows == undefined || rows == [] || rows.length == 0){
          const msg = `El ticket #${id} no existe`;
          agent.add(msg);
        } else if(rows[0].status == null){
          const msg = `El ESTADO para el ticket #${id}  no fue encontrado`;
          agent.add(msg);
        }else {
          const status = rows[0].status;
          agent.add(`The status for ticket #${id} is:${status}`);
        }
      } catch (error) {
        console.log(error);
      }
    }

    con.destroy
  }
```

Anexo 8. Código para la el intent ticket-last-update

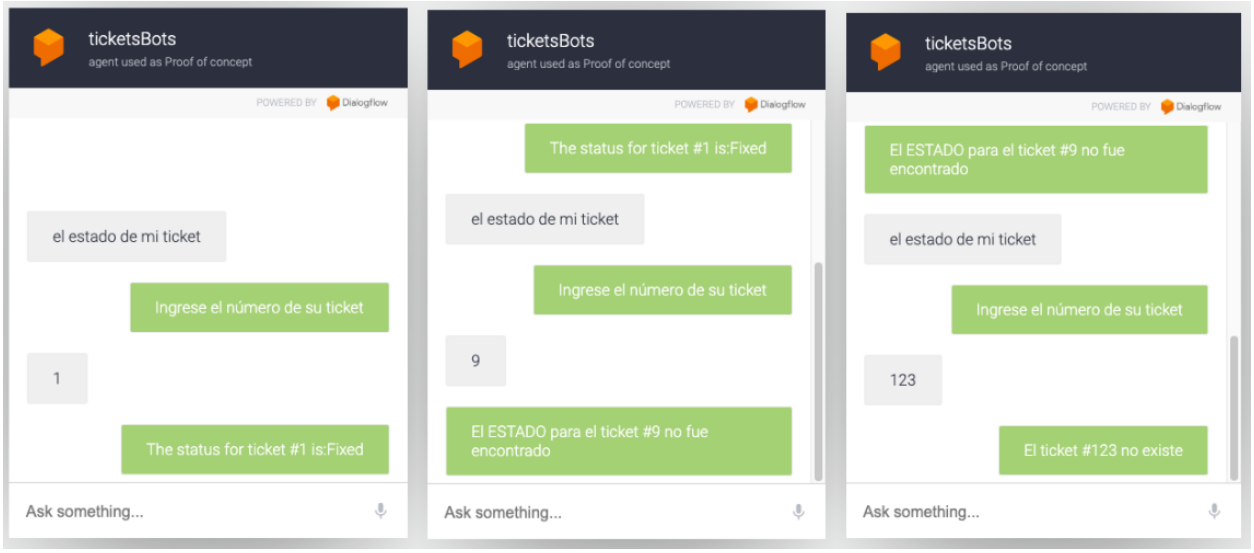
```
const getLastUpdate = async (agent) => {
    const id = agent.parameters['ticket-id'];//1;
    const con = await mysqlConnect();
    if (con != null){
      try {
        const [rows, fields] = await con.execute('SELECT updated_at FROM Tickets WHERE id = ?', [id]);

        if(rows == null || rows == undefined || rows == [] || rows.length == 0){
          const msg = `El ticket #${id} no existe`;
          agent.add(msg);

        }else if(rows[0].updated_at == null){
          const msg = `No se encontro fecha de actualizacion para el ticket #${id}`;
          agent.add(msg);

        }else {
          const dateStr = rows[0].updated_at;
          const lastUpdatedDate = moment(dateStr).format('DD-MM-YY HH:mm:ss');
          agent.add(`La ultima fecha de actualizacion para el ticket #${id} fue: ${lastUpdatedDate}`);

        }
      } catch (error) {
        console.log(error);
      }
    }

    con.destroy
  }
```

Anexo 9. Código para la el intent ticket-notes

```
const getNotes = async (agent) => {
    const id = agent.parameters['ticket-id'];//8;
    const con = await mysqlConnect();
    if (con != null){
      try {
        const [rows, fields] = await con.execute('SELECT notes FROM Tickets WHERE id = ?', [id]);
        console.log(rows);
        console.log(rows.length);

        if(rows == null || rows == undefined || rows == [] || rows.length == 0){
          const msg = `El ticket #${id} no existe`;
```

```
            agent.add(msg);
        }else if(rows[0].notes == null){
            const msg = `No se encontraron comentarios para el ticket #${id}`;
            agent.add(msg);
        }else{
            const notes = rows[0].notes;
            agent.add(`Los comentarios para el ticket #${id} son:
                ${notes} `
                );
        }

    } catch (error) {
        console.log(error);
    }
  }

  con.destroy
}
```

Anexo 10. Pruebas ticket-status



Anexo 11. Pruebas ticket-last-update

Anexo 12. Pruebas ticket-notes