Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Desarrollo de un sistema de interacción basado en Google Home y Dialog Flow.

Autor:

Santamarta Martín, Jaime

Tutores:

Zalama Casanova, Eduardo

Ingeniería de Sistemas y Automática

Valladolid, Abril de 2021.

Trabajo de Fin de Grado

Resumen

Un *Agente Conversacional* es una entidad de software diseñada para mantener una interacción con seres humanos, utilizando la conversación como interfaz, ya sea de forma escrita o de forma oral; sumando la capacidad del entendimiento del lenguaje humano y la síntesis del habla gracias al procesamiento del lenguaje natural.

El comedido de este trabajo es el análisis de las soluciones actuales para la realización de un *agente conversacional*, explicación de la tecnología común utilizada, muestra de una aplicación real con todas las posibilidades actuales mediante una *arquitectura serverless* y su posterior *cálculo económico* para determinar la viabilidad del desarrollo e implementación de un agente conversacional en un negocio.
Para la demostración práctica, se realizará un agente conversacional para la reserva de mesas y pedidos en un restaurante

Palabras Clave: Agente Conversacional, Dialogflow, Google Assistant, chatbot, Firebase.

Abstract:

A *Conversational Agent* is a software entity designed to maintain an interaction with human beings, using the conversation as an interface, either in writing or orally; adding the ability to understand human language and speech synthesis thanks to natural language processing.

The restraint of this work is the analysis of current solutions for the realization of a conversational agent, explanation of the common technology used, sample of a real application with all the current possibilities through a serverless architecture and its subsequent economic calculation to determine the feasibility of development and implementation. of a conversational agent in a business.

Keywords: Conversational Agent, Dialogflow, Google Assistant, Chatbot, Firebase.

Índice General

1.- Introducción

Motivación:

La idea por la que empecé con este trabajo de fin de grado es debido a mi atracción por nuevas tecnologías que suponen puntos de inflexión en el tratamiento social, cómo pueden ser los agentes conversacionales, estableciendo nuevas vías de comunicación entre el ser humano y las máquinas, haciendo que se desarrolle una relación más natural.

A su vez, el desarrollo de cada vez más soluciones SaaS[1] y PaaS[2] permiten la realización de aplicaciones más complejas con un ahorro de costes y tiempo inimaginable hace diez años, haciendo que cualquier pequeña empresa y organización pueda permitirse invertir en esta tecnología pudiendo reducir costes y tareas más repetitivas.

Por último, también es debido a mi incorporación en una empresa innovadora de educación online donde me encargo de trasladar los requerimientos de negocio de la empresa mediante un agente conversacional en texto, entendiendo las necesidades de los usuarios y desarrollando un flujo óptimo para resolverlas.

En muchas ocasiones, los avances tecnológicos no se traducen directamente en aplicaciones reales, por eso la propuesta no es solamente entender el mercado actual de esta solución, ni el diseño e implementación de un agente conversacional básico, sino entender cómo desplegarlo en un entorno real para que sea económicamente viable en un negocio.

[1] SaaS: Software as a Service, es un modelo de distribución de software donde existe una empresa externa que ofrece una aplicación propia en sus servidores a los que el contratante puede acceder con un coste por uso. Empresas como Zendesk, Salesforce, etc lo utilizan brindando toda la lógica de negocio y el soporte

[2] PaaS: Platform as a Service, este modelo de distribución ofrece el acceso a un entorno de desarrollo en la nube donde se provee una infraestructura escalable y herramientas de desarrollo, gestión de BBDD e inteligencia artificial y de negocio.

Objetivos:

El objetivo principal de este trabajo es el diseño completo de un agente conversacional basado en una arquitectura sin servidor (serverless[3]), beneficiándonos de avances significativos de grandes empresas tras años de investigación especializada, utilizando herramientas de lenguaje natural, servidores en la nube y síntesis del habla. Incluirá el diseño, desarrollo, implementación y entrenamiento, comenzando con el análisis de requerimientos por parte de una empresa ficticia llamado "Restaurante Robotizado".

Para ello, dividiremos esta sección en objetivos para garantizar el seguimiento de cada uno y satisfacer los requisitos globales.

- Estudio de las soluciones del mercado actual
- Crear una interfaz de usuario o '*front-end*' que permita la interacción entre el humano y la máquina y resulte cómodo para la persona con la que habla.
- Implementar un servidor o *'backend'* que responsa a los requerimientos de negocio estipulados
- Integración del *backend* con otros servicios externos que permitan ampliar funcionalidades futuras en el agente conversacional.
- Entrenamiento de una herramienta de análisis del lenguaje natural para conectar la entrada del usuario a la lógica de negocio y devolver una respuesta coherente
- Realizar un análisis económico de dicha solución atendiendo a la demanda real de peticiones.

Impacto esperado

De este trabajo se espera como resultado un análisis de las opciones actuales, una guía de su desarrollo mediante arquitectura sin servidor y una análisis económico para promover la utilización de estos agentes conversacionales en todo tipo de entornos profesionales.

[3] Serverless: Es un modelo de computación en la nube donde el proveedor ejecuta un código que tu aportes, ocupándose de la infraestructura que existe detrás. Este modelo permite el lanzamiento rápido de lanzamiento a producción sin tener que preocuparse de la escalabilidad de los recursos empleados. Google Cloud Functions, AWS Lambda y Azure Functions son algunas de estas herramientas.

Estructura de la memoria:

Esta memoria comienza con una introducción en el capítulo 1, donde se presentan los objetivos de este trabajo de fin de grado, una explicación de las motivaciones para realizarlo.
Continua en el capítulo 2 con el análisis del estado del arte de las tecnologías y herramientas implicadas en la creación de un agente conversacional, así como ejemplos de uso reales.

En el capítulo 3 se explicarán las distintas herramientas utilizadas durante el proceso relacionadas con el proyecto para así profundizar en su uso.
Posteriormente, en el capítulo 4 se analizará el problema, explicando los conceptos clave, los requerimientos de negocio y los retos que se deben afrontar a la hora de crear un agente conversacional, así como una propuesta de diseño de la aplicación, utilizando metodologías y procedimientos creativos utilizados en el mercado actual. Con él se conseguirá que el usuario se encuentre a gusto cuando interaccione con el asistente. Para ello realizaremos un estudio de experiencia conversacional enfocado en un público objetivo concreto.

Posteriormente, en el capítulo 5, se procederá al desarrollo e implementación del proyecto, añadiendo una guía paso a paso de un ejemplo real. Esta guía estará creada para que cualquier persona pueda replicar este concepto.

En el capítulo 6, se mostrarán los resultados y la encuesta realizada.
En el capítulo 7, se realizará un estudio económico tanto del coste de la realización del proyecto, así como también se realizará un plan de expansión para determinar la viabilidad de su implementación en cualquier tipo de negocio, así proponer mejoras que no se han acometido y comentar futuras líneas de desarrollo.
Por último en el capítulo 8 se mostrarán las conclusiones y futuras líneas de desarrollo.

Para entender bien el desarrollo de este trabajo de fin de grado, se adjuntará un documento en Notion[4] donde se aborda este trabajo siguiendo una metodología basada en proyectos. Acceso.

[4] Notion es una herramienta gratuita en la nube que te permite organizar todo tipo de proyectos en los que se requiera cierto orden. Este software incluye una interfaz tipo Google Docs para crear tu propia wiki, además de tablones Kanban y BBDD internas.

2.- Estado de los Agentes Conversacionales y mercado

Ecosistema de los agentes conversacionales:

Un agente conversacional, también conocido como *chatbot*, es una entidad artificial diseñada para mantener conversaciones con un ser humano, ya sea mediante texto escrito o a través de interfaces de usuario multimedia como el audio. Este chatbot provee de respuestas automáticas inteligentes según distintas entradas que pueda hacer el usuario y unas reglas definidas, en ocasiones apoyada en inteligencia artificial.
Debido a que un agente conversacional no puede entender todo lo que se le diga y contestar fidedignamente, normalmente se preparan para que actúen en un ecosistema cerrado con objetivos limitados, como por ejemplo una reserva en un restaurante, una consulta de soporte o venta de productos. Así se puede atender a todas las formas de petición posibles y cumplir con los requerimientos de negocio que se piden.

Lenguaje Natural. Definición y Conceptos

Para poder entender cómo se ha llegado al punto en el que existe una inteligencia artificial capaz de procesar nuestra habla y "entender" a qué nos referimos tenemos que remontarnos a la propia definición de lenguaje.

Según *Wikipedia*[5], el **lenguaje** es un sistema de comunicación estructurado para el que existe un contexto de uso y ciertos procesos combinatorios formales. Es decir, un lenguaje es una estructura que nos permite comunicarnos entre nosotros o con otro receptor. Como ejemplos de lenguaje existen muchos, los idiomas, el braille, los lenguajes de programación, etc..

Lo más importante es que la información fluya y sea entendida entre las partes implicadas en la conversación.

Entonces, ¿Cómo funciona el lenguaje entre un ser humano y la máquina?

Para que exista la comunicación es necesario que existan tres componentes básicos: Un emisor, un receptor y un canal. Dentro de un lenguaje de programación, se establece la persona que escribe el código como emisor, un ordenador que lee el mensaje escrito como receptor y la propia escritura del código como canal. Para que exista la comunicación es necesario que un mensaje fluya con el canal.

El acto de la comunicación puede constar de varios mensajes, y estos a su vez pueden tener varios atributos. En el mensaje de "quiero comer espárragos", transmitimos el mensaje de querer comer, pero también los atributos referidos a quien tiene ese deseo, que soy "Yo" y el atributo de "espárragos", que es el objeto de la acción.

Aquí entra el concepto de semántica, es decir, que para que el mensaje se entienda necesitamos conocer qué significa su contenido.

Un lenguaje de programación tiene una estructura muy definida, donde el ordenador puede dejar de entender el mensaje simplemente porque una letra esté en mayúscula.

Con un lenguaje humano todo es aún más complejo ya que son sistemas mucho más abiertos. Podemos hablar mal, desordenando las palabras

[5] https://es.wikipedia.org/wiki/Lenguaje

según hablamos, utilizar expresiones, vocablos locales y aún así nos podremos entender, incluso solo por el contexto de la conversación.

Suponiendo que un ordenador recibe información según la dictas, pasando de voz a texto, palabra por palabra, quedaría la barrera de conocer que se está queriendo comunicar.

Esta funcionalidad lleva existiendo desde hace tiempo, por ejemplo en el BMW serie 7 de 2002[6], donde a través de unos comandos de voz podías preguntar acerca del tiempo. Tenias que decir exactamente "¿Que tiempo hace hoy?" para que lo reconociera. Es decir, se necesitaba comunicar un mensaje fijo y definido para ser utilizado, ya que esa era exactamente la señal de voz guardada en el coche. No recibiria correctamente el mensaje "¿Cual es el tiempo hoy?" ni "¿Qué tal tiempo hay hoy?", si no solamente para los que se le haya programado.

No era en realidad un lenguaje natural donde exista comunicación, solo unos comandos dictados por voz.

Gracias a nuevos avances en computación, un campo conocido desde hace muchos años como es el de la inteligencia artificial ha desarrollado un crecimiento exponencial, permitiendo aplicaciones inesperadas hace siglos debido al volumen de procesado de datos necesario.

De la mano de estos avances surge una especialización dentro del machine learning encargada del procesamiento del lenguaje natural, permitiendo que una máquina pueda entender el contexto en el que se desarrolla una conversación.

[6] Manual de usuario del BMW, rescatado de una página web.

Procesamiento del lenguaje natural

El **procesamiento del lenguaje natural** es un campo de conocimiento de la inteligencia artificial que se encarga de que las personas puedan comunicarse con las máquinas a través de lenguajes naturales, como puede ser el español, inglés o cualquier otro idioma.

Como las máquinas solamente entienden bits es necesario que exista un proceso de modelización matemática, tarea que se han encargado los llamado lingüistas computacionales, que preparan modelos lingüísticos para que los ingenieros informáticos puedan implementar un código eficiente y funcional.

Durante el desarrollo, se utilizaron dos aproximaciones al problema de la modelización lingüística, un modelo lógico y un modelo probabilístico.

Modelo lógico gramatical

Este modelo almacena información en diccionarios computacionales que definen los patrones que hay que reconocer para poder resolver una tarea, como por ejemplo una traducción o responder a una acción. Estos diccionarios son construidos por lingüistas que definen unas reglas de reconocimiento de patrones estructurales en el lenguaje, basándose en las teorías del lenguaje de Noam Chomsky[7].
Este modelo fue la primera opción debido a limitaciones del pasado.

Modelo probabilístico del lenguaje natural[8]

En este caso, los lingüistas recopilan colecciones de datos para poder calcular las frecuencias de distintas unidades lingüísticas (como letras, frases, oraciones) y así calcular la probabilidad de aparecer según un contexto determinado. Esto permite predecir la siguiente unidad lingüística según las unidades anteriores. Estos avances han sido posibles gracias al aumento de poder de cómputo, y nos ha permitido avanzar en este campo sin tener que definir unas reglas gramaticales explícitas.

[7] Artículo científico explicativo acerca de la Teoría de Lenguajes Natural de Noam Chomsky. Creado por Franco M. Lugue
[8] https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/

Arquitectura de un agente conversacional

Un chatbot proporciona una interfaz entre el usuario y la lógica de negocio, entre medias, se ejecutan varios procesos que permiten transformar el habla del usuario en acciones que la lógica interna del bot pueda procesar y realizar su posterior respuesta.

Todo comienza cuando una persona habla, dichas ondas son captadas por el sensor electroacústico del aparato en cuestión, que las transforma en señales digitales que pueden son transformadas gracias a algoritmos de voz a texto, para posteriormente introducir este texto obtenido a un procesado de lenguaje natural, donde se contextualiza y se analiza para dar forma a las palabras recibidas. Después del procesamiento, el resultado puede ser utilizado por la lógica interna del bot, almacenados en Bases de Datos, ejecutando acciones en consecuencia o estableciendo una comunicación con terceras aplicaciones.

Una representación de la arquitectura utilizada para el diseño de un chatbot puede ser la referida a la Figura 1 a continuación:

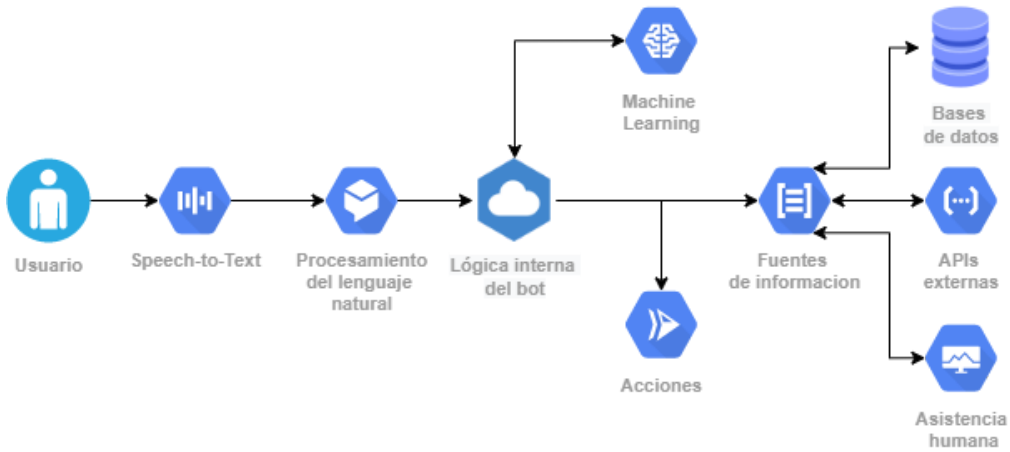


Figura 1.

Cómo apreciamos en el esquema, se requiere de diversos pasos para llevar a cabo una aplicación, para los cuales existen herramientas de todo tipo de proveedores capaces de asumir esta tarea. Para poder abarcar cualquier proyecto, es necesario dividir y modularizar el todo en segmentos más pequeños que podamos tratarlos como aplicaciones separadas:

Aquí listaremos las distintas partes existentes:

- Reconocimiento de voz a texto
- Interfaz de usuario
- Procesamiento de lenguaje natural
- Ejecución de la lógica interna
- Bases de datos
- API (Application Programming Interfaces) externas

En este proyecto, utilizaremos diversas herramientas ya existentes para poder llevar a cabo los requerimientos de negocio necesarios.

Cómo la creación desde cero de un agente conversacional puede ser muy complejo, nos centraremos en analizar herramientas que podemos utilizar para recrear una conversación con funcionalidades limitadas, haciendo uso de bases de inteligencia artificial ya creadas y herramientas *serverless*.

Analizaremos herramientas encargadas de las siguientes partes:

- Reconocimiento de voz a texto
- Generación de texto a voz
- Interfaz de usuario
- Procesamiento de lenguaje natural
- Bases de datos y APIs externas.

Estado de la tecnología

Herramientas de reconocimiento de voz:

El reconocimiento de voz consiste en la conversión de audio de diferentes fuentes en texto, gracias a potentes modelos de redes neuronales entrenados en diferentes idiomas, que permiten detectar las palabras habladas y la intención de estas, dependiendo del volumen y la entonación del dictante. Los productos más utilizados para el reconocimiento de voz han sido desarrollados por grandes empresas tecnológicas que han podido crear redes neuronales gracias a su capacidad de cómputo.
Las más conocidas y utilizadas son:

- **Amazon Transcribe**[9]: La herramienta de *Amazon* para el reconocimiento de voz que emplea un proceso de aprendizaje profundo llamado *ASR*[10] que permite la conversión de voz en tiempo real para transcribir llamadas, automatizar subtítulos, generar metadatos y búsqueda por video inteligente. Su coste es gratuito durante 12 meses con una cuenta de *Amazon*, permitiendo 60 minutos al mes, y aproximadamente 2.4 céntimos por minuto.
  Esta empresa también cuenta con una versión especializada en medicina, llamada *Amazon Transcribe Medical*[11].
- **Google Cloud Speech-to-Text**[12]: Alternativa de *Google* para el reconocimiento de voz, incluye una API fácil de utilizar, más de 120 idiomas con sus respectivas variantes y reconocimiento automático de idioma. Como novedad, añade el reconocimiento de voz en conversaciones de hasta 4 personas, pudiendo asignar cada parte del diálogo y con reconocimientos de nombres propios.
  Su precio[13] es de 0,004 USD por cada 15 segundos en su versión estándar, con 60 minutos gratuitos por mes.
- **Microsoft speech to text**[14]: Alternativa de *Microsoft* que destaca por su privacidad, sin almacenar ningún audio dentro de sus servidores e integraciones profundas con su suite Office 365. Su precio es de 0,844€/hora con una capa gratuita de 5 horas al mes.

[9] https://aws.amazon.com/es/transcribe/
[10] ASR: Automatic Speech Recognition
[11] https://aws.amazon.com/es/transcribe/medical/
[12] https://cloud.google.com/speech-to-text
[13] Precio para dispositivos personales, consultar para dispositivos integrados como coches, robots, televisores, etc..
[14] https://azure.microsoft.com/es-es/services/cognitive-services/speech-to-text/#features

Herramienta de generación de voz mediante texto:

La síntesis de texto a voz ha permitido mejorar la accesibilidad y aumentar la participación en aplicaciones que la utilizan. Se puede dividir en dos partes principales, la conversión de texto a palabras escritas con significado, llamado pre-procesamiento y posteriormente la transcripción fonética de cada palabra con una entonación, fase y duración determinadas para generar la forma de ondas conveniente

- **Amazon Polly**[15]: Es la herramienta de *Amazon* para sintetizar texto a un habla realista con un sonido natural. Permite el almacenamiento y redistribución del contenido oral generado para su posterior uso sin coste adicional y su precio aproximado es de 4,80 USD por cada millón de caracteres.
- **Cloud Text-to-Speech**[16]: Sintetiza texto a mas de 180 voces distintas en más de 30 idiomas, utilizando la tecnología *WaveNet* que permiten una cercanía extraordinaria al habla humana, imitando la voz de actores famosos. Su precio en es de 4 USD por millón de caracteres en nivel estándar, incluyendo 4 millones gratuitos y de 16 *USD* por millón de caracteres con las voces del modelo WaveNet, incluyendo 1 millón gratuitos
- **Microsoft Text to Speech**[17]: La alternativa de Microsoft que permite sintetizar texto en más de 40 idiomas con más de 100 voces a elegir. Su precio es de 4 USD por millón de caracteres en nivel estándar, incluyendo 5 millones gratuitos y de 16 USD por millón de caracteres con las voces del modelo WaveNet, incluyendo 500.000 gratuitos.

[15] https://aws.amazon.com/es/polly/
[16] https://cloud.google.com/text-to-speech
[17] https://azure.microsoft.com/es-es/services/cognitive-services/text-to-speech/

Interfaz de usuario:

La interfaz de usuario es el medio por el que el usuario puede comunicarse con una máquina o dispositivo, donde se controlan los puntos de contacto entre el usuario y el equipo.
Existen todo tipo de interfaces más o menos complejas, en nuestro caso la comunicación puede darse mediante voz o en su defecto, mediante texto, para transmitir las intenciones del usuario al agente conversacional.
Principalmente serán asistentes virtuales, pero también los servicios de mensajería instantánea podrían incluirse en esta sección.
Los principales competidores en el mercado son:

- **Google Assistant**: Es el asistente virtual creado por *Google* que implementa inteligencia artificial para muchas tareas del día a día y que existe en los teléfonos y altavoces inteligentes Google Home/Nest. Se puede interactuar con él tanto por medio de la voz como por texto, para hacer búsquedas por internet, programar eventos y alarmas, acceder a aplicaciones y otros servicios. Este asistente se puede implementar de forma experimental para proyectos no comerciales y de prueba, mediante el Google Assistant SDK[18] en diversos lenguajes, como *python* y *C++*.
- **Alexa**: Es el asistente virtual creado por Amazon que está instalado en los altavoces inteligentes de *Amazon*, como *Amazon Echo*. Consta de "skills", que son aplicaciones provistas por terceros que aportan funcionalidades adicionales. También se pueden crear rutinas para optimizar tu día a día.
- **Siri**: Creado por *Apple*, es de los primeros asistentes personales que salieron al mercado. Cuenta con su propia personalidad dependiendo del dispositivo que se encuentre, localizandose en *iOS, macOS, tvOS* y *watchOS*, todo productos de Apple.
- **Mycroft**[19]: Es el primer asistente virtual con IA con fuentes completamente abiertas, basado en linux.

[18] https://developers.google.com/assistant/sdk/guides/service/python
[19] https://mycroft.ai/

Herramientas de interpretación del lenguaje natural:

Nuestro lenguaje, ya sea escrito o hablado, constituye una forma muy compleja de datos, y por ello, pueden ser analizados y procesados de tal forma que puedan ser entendibles por máquinas. Debido a la complejidad y ambigüedad de la comunicación humana, se han presentado desafíos que están pudiéndose solventar gracias al desarrollo de la capacidad de cómputo y mejores modelos de redes neuronales.

Para afrontar estos desafíos, se han dividido las intenciones dependiendo de dos parámetros: La complejidad de la información y la complejidad de la carga de trabajo. Esto es debido a que no es lo mismo ordenes estrictas y bien definidas previamente, que una conversación fluida o encontrar información clave en una oración gracias a detectar patrones.

Expondremos varios ejemplos sobre como dicha herramienta tendría que actuar cuando le llega una oración:

- La frase "*Pon música*" presenta baja complejidad de información y de carga de trabajo, ya que solamente tiene que detectar el verbo "Poner" y la palabra "música", que ya se han establecido anteriormente en el agente utilizado y ejecutar el comando asociado a esta combinación. Así decimos que nos encontramos con una oración concreta y bien definida, que genera un árbol de decisión sencillo, obteniendo un modelo eficiente.
- La frase "*Quiero reservar una mesa para 4 personas para mañana a las 4*" tiene una complejidad añadida ya que hay que extraer mucha información de ella y de forma ordenada dentro de un contexto. Es necesario un sistema experto que realice el diagnóstico y extraiga la información precisa, en este caso, la intención de reserva, el número de asistentes, el día y la hora.

Para abarcar todas las posibilidades existen varios métodos, el primero de ellos el análisis de patrones. Este método permite cerrar unos supuestos previamente pensados y programados a través de una plantilla, pero su principal problema es la acotación de posibilidades. Es por eso que se recurre al Machine Learning, donde las decisiones son tomadas a través de un entrenamiento anterior. Gracias a esto, se puede escalar con cierta facilidad el número de oraciones procesadas.

Existen multitud de servicios de todo tipo que permiten abstraer el trabajo al programador y evitar ir a bajo nivel, gracias a la automatización de las optimizaciones.

Para esta sección analizaremos las herramientas de lenguaje natural mas utilizadas hasta la fecha:

DialogFlow[20]:

- Es la herramienta propuesta por *Google*, conocida anteriormente como *Api.ai*, en estos momento se muestra como una solución muy polivalente y al alcance de cualquier persona gracias a su dominio en sus modelos de redes neuronales especializadas en voz y lenguaje natural.
  Sus principales ventajas respecto a otros son la oferta de agentes predefinidos para todo tipo de negocio (en inglés), la posibilidad de ejecutar código directamente en la plataforma a través de Fullfilment, el soporte en múltiples lenguajes, la simplicidad en el manejo de la herramienta y la interpretación de emociones. Otra ventaja interesante es la integración directa con Google Assistant, *Google Home*, y otras plataformas de terceros como Telegram o *Skype*. Además se están centrando mucho en la sustitución de agentes telefónicos, con sus integraciones One-Click como *Avaya*, *SignalWire*, *VoxImplant*, *AudioCodes* y *DialogFlow Phone Gateway*. A todo esto hay que incluir su producto estrella, *Chatbase*, un programa de análisis que plasma toda la experiencia de la empresa con *Google analytics*.
  Cada proyecto de un agente, está vinculado a un proyecto en *Google Cloud Platform*, haciendo que se vincule directamente y sea necesario utilizar webhooks para ser utilizado por otros proveedores de cloud.
  Respecto al precio, *DialogFlow* cuenta con una capa gratuita 'ilimitada'[21], que en realidad tiene su límite en el número de peticiones por minutos. Según se escalen las peticiones al tamaño propio de una mediana empresa, el precio aumenta en su versión Enterprise desde 0.002 USD por solicitud de texto en su versión Essentials hasta 0.004 USD por solicitud en su versión Plus.
  En todos los casos, la versión gratuita es mas que suficiente para aplicaciones experimentales y para flujos de usuarios propios de pequeñas empresas. Y aunque la versión de pago es cara, nos dá la fiabilidad del progreso de *Google* ofreciéndonos funcionalidades exclusivas de su plataforma.

[20] https://dialogflow.com/
[21] https://cloud.google.com/dialogflow/quotas

Wit.ai[22]:

La alternativa actual de *Facebook*, startup fundada en 2013 que fue adquirida por Facebook en 2015.
Aunque no tenga soporte de calidad como sus competidores ni llegue al rendimiento esperado, su principal ventaja competitiva es que es completamente gratuita en cualquier caso.
Su rendimiento es mucho más lento y ofrece peores resultados, con opciones mucho mas limitadas, teniendo que acudir a otras herramientas externas para complementar su poder limitado.

IBM Watson Assistant[23]:

La apuesta de *IBM* para la interpretación del lenguaje natural lanzada en 2016, tiene la capacidad de ejecutarse en la nube o en servidores on-premise[24] y cuenta con soporte hasta en 13 lenguajes.
Con esta herramienta se logran obtener resultados con un rango de precisión mucho mayor que sus competidores, gracias a la inexistencia de la obligación de respuesta, que permite no responder en caso de que el chatbot no tenga clara la respuesta.
Cómo gran desventaja es que no cuenta con una capacidad de aprendizaje automática, dificultando su entrenamiento, ni tampoco acepta peticiones a través de mensajes de audio, requiriendo de una herramienta de transformación voz a texto adicional.
Su versión gratuita alcanza hasta los 10.000 mensajes de texto al mes, con una capacidad máxima de 100 nodos de diálogo y 80 entradas disponibles.
La capa de pago más económica tiene un precio de 0.0025$ por mensaje de texto, con las demás características ilimitadas.

Aunque cuenta con la mejor detección de intenciones del usuario y un precio levemente inferior, la falta de comodidades para el desarrollador le hace recurrir a frameworks para abstraer el entrenamiento.

[22] https://wit.ai/
[23] https://www.ibm.com/cloud/watson-assistant/
[24] on-premise: Posibilidad de ejecución en servidores propios de la empresa que lo contrata

Amazon Lex[25]:

La empresa que domina la nube en estos momento también tiene una oferta atractiva en cuanto al uso del lenguaje natural, que le dá vida a *Amazon Alexa*, el asistente virtual de la empresa.
Permite analizar todo tipo de información tanto en formato texto como audio.
Esta herramienta destaca por pertenecer al variado ecosistema de *Amazon Web Services*, que permite una integración directa con muchos de sus servicios, como por ejemplo, las Lambda Functions, que nos permitirán crear rápidamente bots gracias a la numerosa documentación y elementos predefinidos.
El principal problema que nos encontraremos en Amazon Lex, es la falta de *Contextos,* es decir, variables que permiten conocer el contenido de la conversación para personalizar según avance. Para implementar esto se requeriría otra capa adicional de personalización no incluida en este servicio. Otro de estos inconvenientes es su uso único en ingles, que te limita a mercados anglosajones

Durante el primer año, puedes utilizar de forma gratuita esta aplicación, con un límite de 10.000 peticiones de texto y 5.000 por voz al mes.
Posteriormente, sus precios son de 0.00075$ por petición de texto y de 0.004$ por petición de voz. Es decir, 0.75$ por cada 1000 peticiones de texto y 4$ por cada 1000 peticiones de voz.
Esto deja a *Amazon Lex* como una alternativa competitiva, aunque algo limitada, ya que se reserva al mercado anglosajón debido al idioma.

OpenAI API[26]

Quiero hacer una mención de honor a un producto lanzado el 11 de Junio de 2020 que puede revolucionar muchos sectores gracias a sus avances en el ámbito del lenguaje natural. *OpenAI* es una organización sin ánimo de lucro apoyada por Sam Altman y Elon Musk, y otros grandes empresarios de internet, preocupados por la deriva de la IA.
Esta API abierta y de pago por uso, da superpoderes a muchas aplicaciones conocidas, con la filosofía de "text-in, text-out" que brinda respuesta a los retos que se le plantea, según aparece en su web[27].

[25] https://aws.amazon.com/es/lex/
[26] https://openai.com/
[27] https://openai.com/blog/openai-api/

Proveedores de Servicios en la Nube:

Una gran revolución empresarial surgió a raíz de los avances en el campo de la computación en la nube, proveyendo a empresas de internet una infraestructura ajena a ellos, donde no tendrían que hacerse cargo de la compra, montaje, utilización, mantenimiento ni seguridad de los servidores como se había hecho antes de la aparición de servicios en la nube masivos. Este modelo, basado en el pago por uso o por tiempo de cómputo, permitió a muchas empresas de internet lanzar productos sin preocuparse en realizar grandes inversiones y lanzar productos que se podían ir escalando dentro de los propios servidores en la nube.
Este mercado mueve miles de millones y está impulsado por las grandes compañías tecnológicas de todo el mundo, entre las que destacan Google, Amazon, Microsoft, Alibaba e IBM, que entre ellas, cuentan con el 60% de toda la cuota de mercado[28] como se muestra en la figura 2.

Worldwide cloud infrastructure spending and annual growth
Canalys estimates, Q4 2019

Cloud service provider	Q4 2019 (US$ billion)	Q4 2019 market share	Q4 2018 (US$ billion)	Q4 2018 market share	Annual growth
AWS	9.8	32.4%	7.3	33.4%	33.2%
Microsoft Azure	5.3	17.6%	3.3	14.9%	62.3%
Google Cloud	1.8	6.0%	1.1	4.9%	67.6%
Alibaba Cloud	1.6	5.4%	1.0	4.4%	71.1%
Others	11.6	38.5%	9.3	42.4%	24.4%
Total	**30.2**	**100.0%**	**22.0**	**100.0%**	**37.2%**

canalys

Note: percentages may not add up to 100% due to rounding
Source: Canalys Cloud Channels Analysis, January 2019

Figura 2

Cada servicio en la nube tiene sus particularidades, con sus ventajas e inconvenientes, es por eso, que antes de elegir un servicio, es necesario determinar cuales son las necesidades del proyecto. Factores como servicios exclusivos, cercanía de servidores, presupuesto, tratamiento de

[28] Cloud Market share https://www.canalys.com/static/press_release/2020/Canalys--Cloud-market-share-Q4-2019-and-full-year-2019.pdf

datos, cadenas de bloques o *blockchain*, bases de datos, tecnología permitida, etc.. pueden decantarnos a elegir uno en concreto.
En nuestro caso, la mayoría de las necesidades pueden ser solucionadas mediante servicios SaaS, como Interpretación del lenguaje natural y Funciones como servicio. Sin embargo, no nos podemos cerrar a soluciones aisladas, ya que un bot puede tener cientos de funcionalidades, cómo enviar recordatorios, por lo que nos interesarán opciones relacionadas con calendarios y bases de datos, o incluso pagos mediante voz, por lo que nos interesaria que tenga integración con pasarelas de pago y envío de correo mediante un servidor de correo electrónico.

Para este apartado, nos centraremos en las *Funciones como servicio* ofrecidas por distintas plataformas. Este concepto es utilizado para denominar a una arquitectura que provee una abstracción completa hasta nivel de ejecución, solamente teniéndose que preocupar del código escrito. Su arquitectura sería similar a la mostrada en la figura 3[29]:

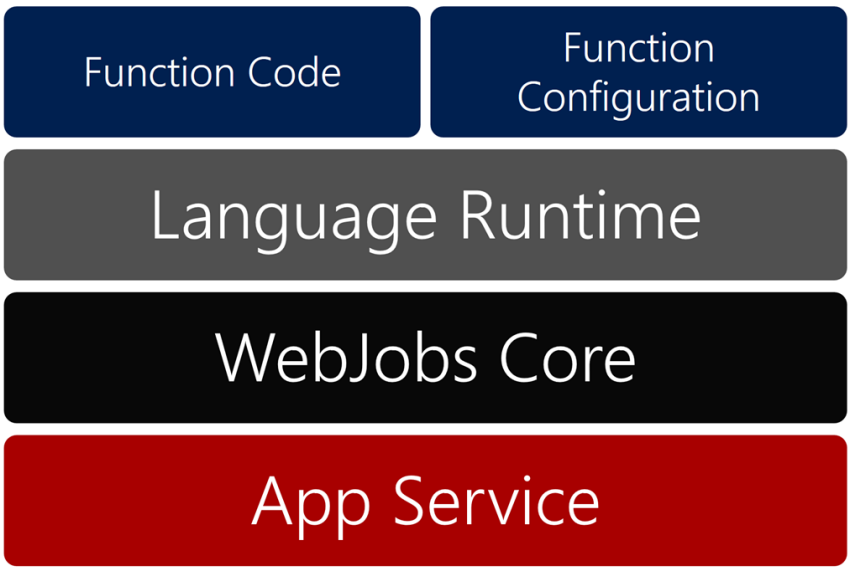


Figura 3: Arquitectura de las llamadas funciones sin servidor

[29] Arquitectura utilizada en Azure Functions de Microsoft
https://docs.microsoft.com/es-es/dotnet/architecture/serverless/serverless-architecture

Para un análisis de este mercado, nos centraremos en las tres empresas con mayor cuota de mercado, AWS, GCP y Azure, que explicaremos a continuación:

Amazon Web Services[30]:

Ha sido la primera plataforma que ofreció servicios de computación en nube pública, con lanzamiento en 2006. Surgió como apoyo al negocio de Amazon para gestionar la logística y operaciones de la compañía, que gracias al conocimiento generado, se creó una línea de negocio encargada de alquilar tiempo de cómputo a nuevas empresas de internet a un precio asequible, sin que estas tuvieran que comprar y montar sus propios servidores.
AWS cuenta con una madurez sin igual en sus soluciones, junto con un precio muy competitivo, y es por eso que es la preferida por muchas empresas y startups que se basan en productos digitales, cómo es el caso de Netflix y Uber. Por este hecho, AWS es el servicio en la nube que más dinero factura, y que mayores beneficios genera en toda la corporación Amazon.

Amazon fué el pionero en la incorporación de las Funciones como Servicio, a través de su producto *AWS Lambda*[31], que se introdujo al mercado en 2014 revolucionando el mercado con el concepto serverless abierto al público general. Cuenta con soporte en múltiples lenguajes, como node.js, Python, Java 8, C# (.NET core) y Go.
Este servicio cuenta con una capa gratuita de 1 millón de solicitudes al mes y 400000 GB/segundos de tiempo de cómputo al mes.
Posteriormente, su precio es de 0,20 USD por cada millón de solicitudes y de 0,000016 USD por cada GB/segundo[32].

[30] https://aws.amazon.com/
[31] https://aws.amazon.com/es/lambda/
[32] Este precio depende del volumen de memoria asignada a la función, más caro cuanta mas memoria esté asignada.

Microsoft Azure[33]:

Fué la segunda plataforma en aparecer después de AWS, en el año 2010. Microsoft ha podido encontrar un hueco en el mercado gracias a su especialización y sus servicios orientados a grandes empresas y gobiernos, facilitando la integración de nubes híbridas utilizando la nube de Azure y centros locales de empresas.

Sus servicios más diferenciales han sido la integración plena con otros servicios de *Microsoft*, como *Windows, Office, SQL Server, Visual Studio* y *Windows Server*, donde ya contaba con un gran mercado vigente. Además destaca en su apoyo a herramientas de Open Source, a las que dá soporte.

Las llamadas Azure Functions son la alternativa de Microsoft en lo que respecta a las Funciones como Servicio. Sus principales diferencias competitivas son la aceptación de otros lenguajes, como F# y Typescript, la posibilidad de desplegar código alojado en otras plataformas como Dropbox o GitHub.

Su versión gratuita y su versión de pago[34] son prácticamente iguales a nivel general que el precio de Amazon Lambda.

[33] https://azure.microsoft.com/
[34] https://azure.microsoft.com/es-es/pricing/details/functions/

Google Cloud Platform

El error de Google fué su tardía entrada en este mercado, cuya consecuencia es una menor cuota de mercado, que ha podido paliar gracias a ventajas en cuanto a precios en algunas de sus herramientas y servicios exclusivos de su plataforma, entre los que destaca el *Machine Learning*, *Data Analytics* y una gran integración con *Kubernetes*.
Su alternativa son las *Google Cloud Functions*[35], cuenta con los lenguajes Python, Go y node.js, menos que su competencia. Para compensar, tiene una integración sencilla con otros servicios exclusivos de *GCP*, como *Firestore*, *API Vision* y *Bigquery*.
Su versión gratuita cuenta con 2 millones de peticiones al mes, con un precio similar de 0,20 USD cada millón adicional.

También cuenta con un servicio llamado *Firebase*[36], que se considera de los pocos *BaaS* (Backend As A Service) del mercado, pudiendo implementar muchas funcionalidades propias de aplicaciones rápidamente, como inicios de sesión, visión artificial, machine learning, traducción de texto, activación de correo electrónico y otros servicios centrados en el desarrollo de negocio, para que solo tengas que centrarte en las funcionalidades específicas de tu negocio.
En resumen, *Google* está compitiendo ofreciendo servicios exclusivos y de alto valor para que sea sencillo integrarlos en todo tipo de negocios, centrándose en la usabilidad por parte del usuario de estas. Además está centrando sus esfuerzos en mejorar su plataforma ofreciendo herramientas poderosas para los *DevOps* basadas en Open Source.

[35] https://cloud.google.com/functions
[36] https://firebase.google.com/

Conclusiones:

Nos encontramos en un ecosistema robusto que evoluciona a pasos agigantados, apareciendo cada poco tiempo nuevas herramientas y nuevas funcionalidades apoyadas por las mayores empresas del mundo. Esto permite tener suficiente seguridad como para que elegir una solución no sea una decisión fundamental para el desarrollo de negocio, ya que el traspaso entre plataformas no es razón de peso.
Para nuestro caso acerca de un agente conversacional, se nos presentarían cada vez más posibilidades y de uso mas simple, haciendo que la implementación pueda ser gratuita para pequeñas y medianas empresas, ya que no llegarian a alcanzar el máximo de ejecuciones de la versión gratuita de ninguna de las plataformas.

Es por todo esto que la decisión puede tomarse en función de las integraciones y la facilidad de uso.
La plataforma que cuenta con estas funcionalidades más desarrolladas para la creación de agentes conversacionales es **Google**, en su ecosistema **Google Cloud Platform**.

Así que utilizaremos **Dialogflow** como herramienta de interpretación de lenguaje natural, que cuenta con una integración directa con **Google Assistant** y por tanto una capa integrada de **Speech-to-Text**. Además, cuenta con las **Inline Functions**, que nos permiten conectar con servicios como **Firebase** y **Google Calendar** utilizando muy pocas líneas de código.

3.- Herramientas utilizadas

Cómo se ha hablado anteriormente, nuestra aplicación estará dividida en tres partes fundamentales complementarias que atenderán a los requerimientos de negocio acordados, tal como se muestra en la figura 4:

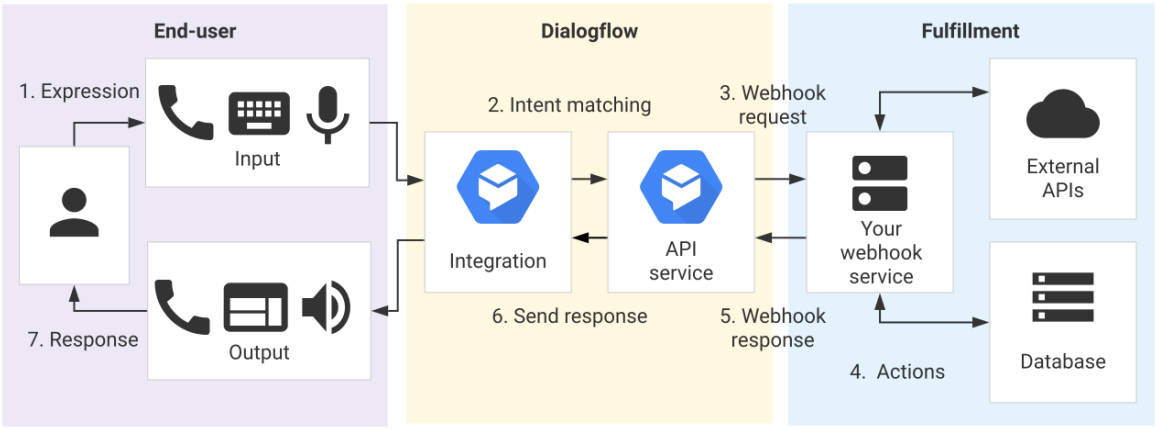


Figura 4: Comunicación entre humano y máquina

La primera parte será la interfaz de usuario, en este caso podrá recibir información a través de audio y de texto, siendo el primer caso procesada posteriormente para convertirlo a texto. Esta información es la que es enviada a DialogFlow a través del Google Assistant. También Google Assistant podrá responder a través de texto y audio.
La segunda parte estará formada por la herramienta DialogFlow, un procesador de lenguaje natural con el que se puede obtener el contexto de una frase recibida, detectando palabras y estructuras que se configuren dentro de ella. A partir de ahí, podremos ejecutar funciones incluidas dentro de DialogFlow (como pueden ser las 'Respuestas', el Fulfillment o respuestas a través de webhooks).
La tercera y última parte corresponde con las conexiones con elementos externos a DialogFlow, como pueden ser bases de datos, servidores o las cloud functions. Estas últimas las utilizaremos para problemas más complejos, ya que su ejecución es muy sencilla.

Como ya hemos hablado anteriormente de estas herramientas, se explicará las opciones que se pueden utilizar dentro de *DialogFlow*.

Para comenzar a utilizar esta herramienta es necesario profundizar en su funcionamiento y conocer como se ejecuta a través del asistente de Google. El flujo será tal que así:

1. El usuario escribe o dice una expresión a través del Google Assistant (u otra interfaz compatible).
2. La expresión convertida a texto en su caso es enviada a Dialogflow mediante un servicio de usuario en un mensaje de solicitud de intent de detección.
3. Dialogflow envía un mensaje de respuesta de intent de detección al servicio. Este mensaje contiene información sobre el intent coincidente, la acción, los parámetros y la respuesta definida para ese intent.
4. El servicio realiza acciones según sea necesario, como consultas a la base de datos o llamadas a la API externas.
5. El servicio envía una respuesta al usuario final por medio de *Google Assistant*.
6. El usuario final lee o escucha la respuesta.

Es necesario este simple flujo para entender un nuevo concepto de interfaz. Nos hemos acostumbrado a las tradicionales interfaces informáticas que requieren entradas predecibles y estructuradas para que puedan funcionar de forma correcta, enfrentarnos a una interfaz así, donde tenemos respuesta abierta plantea dificultades a la hora de diseñar las conversaciones.

DialogFlow[37]

Es una herramienta que nos permitirá tener entendimiento del lenguaje natural dentro de nuestra aplicación, a través de estas características que le dotan de superpoderes:

- Clasificación de las intenciones del usuario
- Extracción de las palabras claves del diálogo.

Gracias a esto, podemos tener una visión global del contexto de la conversación y actuar en consecuencia para tener respuestas preparadas para nuestro usuario.

Una vez esto conocido, se explicará cada uno de los apartados que podemos utilizar en DialogFlow. Los principales serán: Intents, Entities, Fulfillment, Integrations, Training, Validation & Analytics.

[37] Se está explicando la versión US (Global) de DialogFlow. Durante 2020 se han hecho varias actualizaciones, desbloqueando nuevas regiones como la asiática (AS), australiana (AU) o la europea (EU2). La US (Global) cuenta con todos los servicios y módulos, las demás están limitadas por regulaciones de la región.

Intents

Los "Intents", también llamados intentos en español, clasifican la intención del usuario dentro de una conversación. DialogFlow se encarga de hacer coincidir la expresión del usuario con el intent que mejor se adapta al contexto y a la frase. en la figura 5 podemos ver esquematizado un ejemplo.

Figura 5: Ejemplo de intención en DialogFlow

Dentro del Intent, los tres elementos básicos que tendremos que tener en cuenta para la creación de la conversación serán estos:

- **Frases de entrenamiento**: Son las frases que puede llegar a recibir nuestro agente, que tienen una estructura concreta que sirve para elegir una intención dentro de las existentes.
- **Parámetros**: Son las palabras que pueden variar dentro de una frase de entrenamiento para modificar la respuesta dentro de una intención.
- **Frases de respuesta**: Es la respuesta vinculada a una intención, cuya información puede ser influida por los parámetros enviados.

Con estas tres es posible crear una conversación estructurada que tenga sentido. Aunque para dominar mejor esta herramienta y poder crear conversaciones más profesionales es recomendable conocer cómo utilizar las demás:

- **Contextos**: Permite recordar parámetros creados en otros intentos para tener un flujo conversacional complejo, donde unas intenciones se desbloquean solamente después de hacer otras. Ver figura 6.

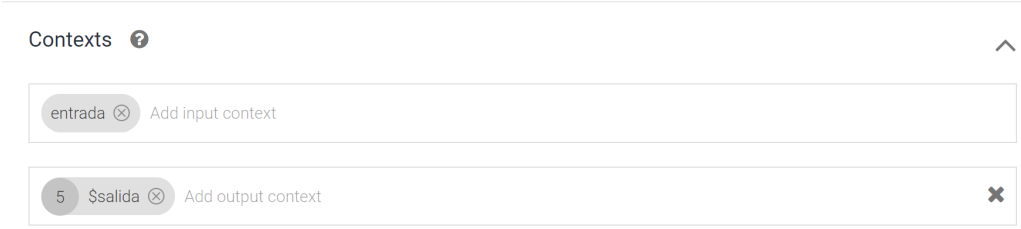


Figura 6: Interfaz del contexto

- **Eventos**: Es una forma de ejecutar un intento sin que exista una entrada por parte del usuario. Permite comenzar intentos a través de eventos externos, como puede ser iniciar sesión a través de Google Assistant, comenzar una conversación en Telegram o preguntar acerca del tiempo. También puedes personalizar dichos eventos para que ocurran cuando se de un momento concreto de la conversación. Ver figura 7.

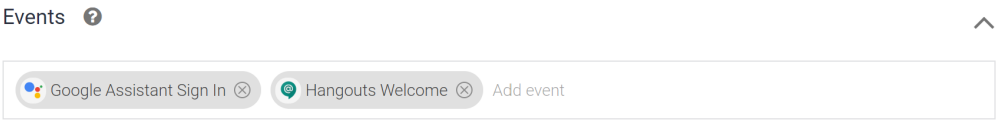


Figura 7: Interfaz del evento

-

- **Training phrases** (o frases de entrenamiento): Aquí escribiremos las posibles preguntas que hará el usuario. DialogFlow intentará clasificar algunas palabras como parámetros de forma automática, guardando fechas, número o nombres. Si no acierta, podemos hacerlo manualmente guardandolo como entidad personalizada. Aquí terminarán las frases que aceptemos que se encuentran en Validation. Ver figura 8.

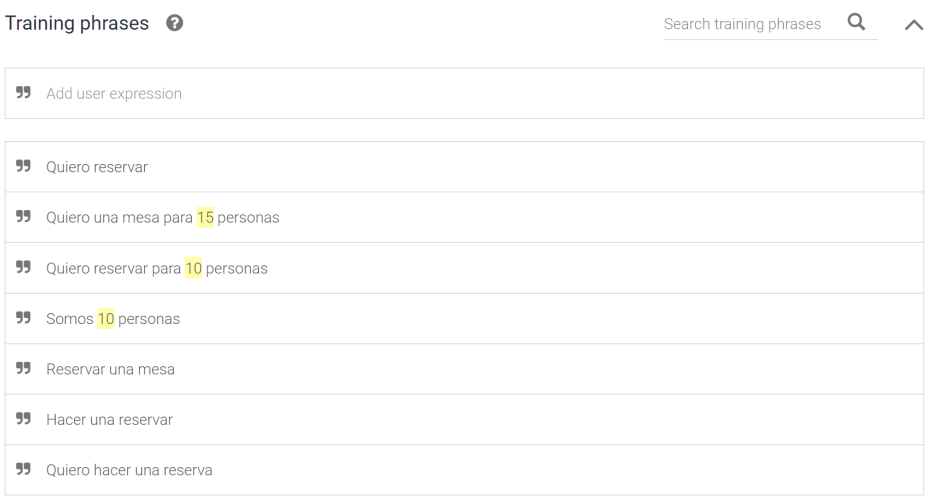


Figura 8: Interfaz de las frases de entrenamiento

- **Actions and Parameters**: Podremos configurar los parámetros existentes dentro del intento para que sean obligatorios, definiendo un "Prompts" que es una frase de respuesta cuando el agente detecta que falta algún parámetro obligatorio. Para ayudar al usuario, también podemos mostrar una lista de las opciones que existen. (Esto último solo se mostrará a través de la pantalla del asistente de Google). Ver figura 9.

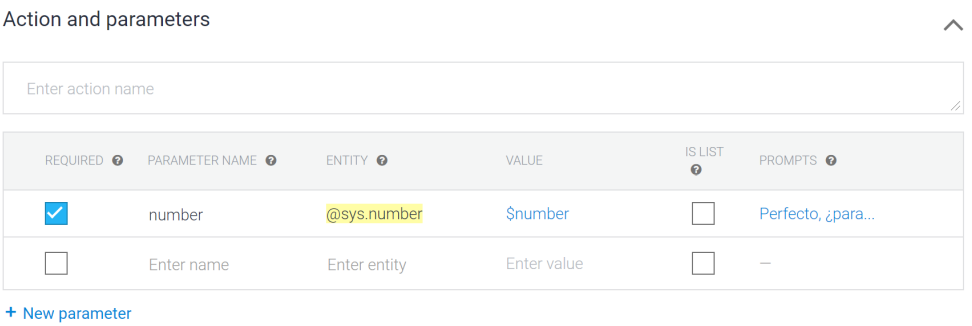


Figura 9: Interfaz de Acciones y Parámetros

- **Responses**: Definiremos la respuesta final cuando terminamos el intent. Se puede aleatorizar e incluir parámetros recogidos. Además, podremos configurar respuestas personalizadas para cada tipo de interfaz (Telegram, Slack, Viber, Messenger, Google Assistant). Ver figura 10.

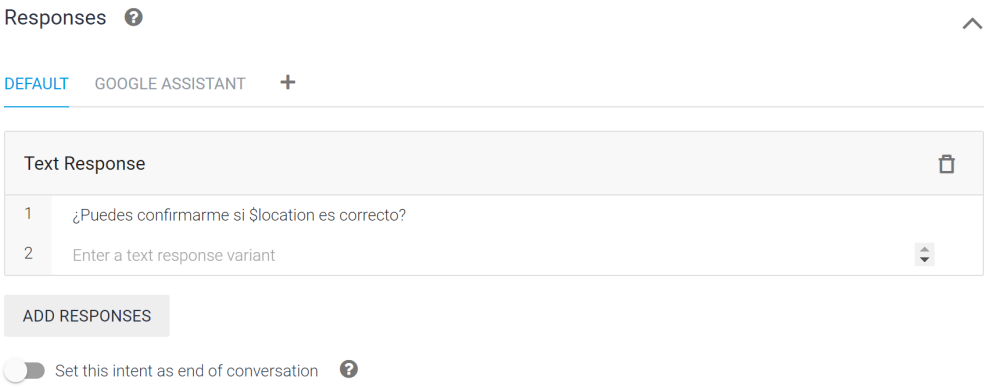


Figura 10: Interfaz de Respuestas

- **Fulfillment**: Nos permitirá sacar todo el provecho de DialogFlow, pudiendo programar la lógica que queramos dentro del Inline Editor. Si seleccionamos "Enable Webhooks" la configuración que tengamos dentro de respuestas no funcionará, y tendremos que incluirla dentro de la lógica de programación. Esta se podrá modificar dentro del Inline Editor que se nos proporciona dentro del menú de Fullfilment. También podremos ejecutar nuestro propio código en un servidor o Cloud Functions si lo preferimos.

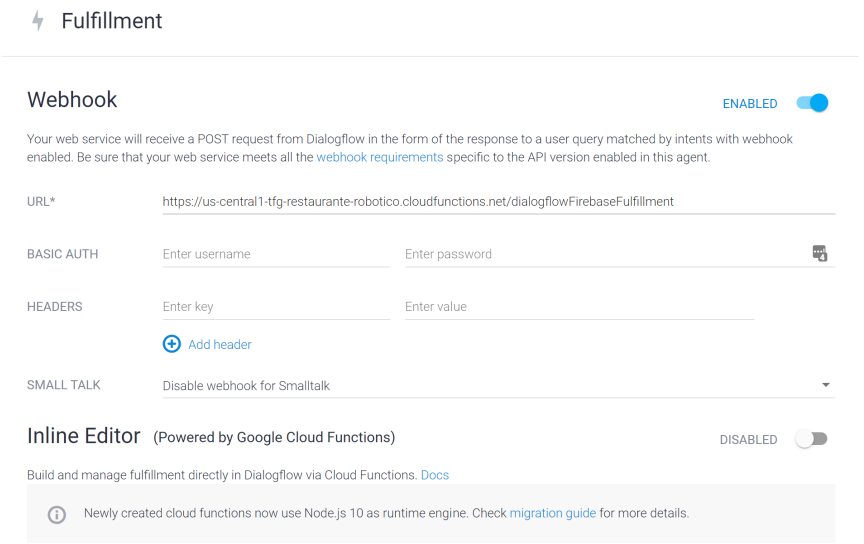


Figura 11: Interfaz de Fulfillment

Entities

Cada parámetro de los intents puede ser de un tipo, llamados tipos de entidad, que determinan fielmente cómo se extraen los datos de una expresión que hace el usuario final. Cómo se ve en DialogFlow, existen dos tipos de entidades, las predefinidas y las personalizadas. Ver figura 12

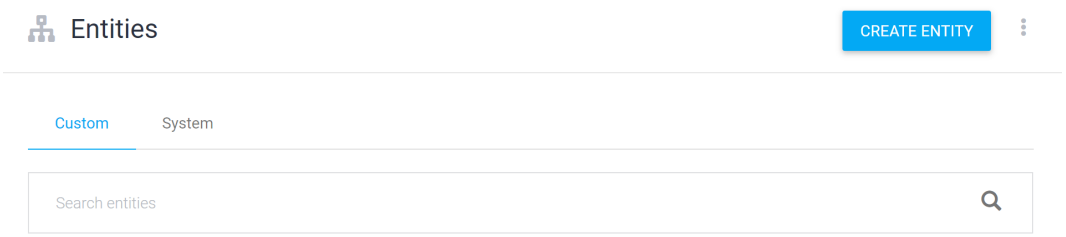


Figura 12: Interfaz de las entidades

Las **predefinidas** incluyen todo tipo de escenarios comunes en los que se pide información genérica. Estas pueden ser: fechas, email, teléfono, colores, horas, etc…
Según introduces frases de entrenamiento, el propio asistente las detecta automáticamente y las ordena según su criterio.

Las **personalizadas** sirven para todo tipo de ocasiones cuando la información sea más compleja o específica. Por ejemplo, se puede definir la entidad “animal doméstico” e incluir en valor de referencia todo tipo de animales domésticos como gatos, perros, loros, etc… Esto permite que el agente conversacional pueda vincular todo tipo de palabras con respuestas comunes. Además, podemos introducir sinónimos de cada una de esas palabras para adecuarlos a las diferencias dentro de una misma lengua.
Dentro de las personalizadas tenemos opciones adicionales utilizando la inteligencia artificial de la herramienta, pudiendo crear varios tipos de entidades como estas:

- **Entidad de asignación**: Esta es la más común, se definen sinónimos que son asignados a una palabra clave con la que podemos trabajar internamente. Ej: M como valor de referencia, pudiendo tener los sinónimos “medio, M, mediano, normal”.
- **Entidad de lista**: Sirve para enlistar características comunes, sin sinónimos. Ej: Para un tipo de colchón, los valores serían “viscoelástico, muelles, látex, bambú”

- **Entidad de expansión automática**: Similar en concepto a las listas, pero en este caso la IA puede incluir nuevos valores automáticamente si están relacionados con los existentes. Ej: Para una entidad de Frutas, los valores que se incluirían para poder incluir automáticamente más serían: Limón, naranja, pera, plátano, pomelo, etc… Es necesario incluir muchos ejemplos para que sea efectivo.
- **Entidad de coincidencia parcial**: Utilizando fuzzy logic, la IA permite clasificar una entidad cuando el orden de las palabras es distinto al normal o faltan palabras.

Fulfillment

Este apartado te permite utilizar integraciones para proporcionar respuestas más dinámicas mediante el concepto llamado entrega. Esta opción se habilita dentro de cada intent cuando queremos personalizar más la experiencia del usuario. Dentro de este apartado nos encontraremos dos posibilidades, utilizar webhooks o el propio Editor Inline vinculado al proyecto de Google a través de las Google Cloud Functions. La figura 13 muestra como es la interfaz tanto para el uso de webhooks como para utilizar el editor integrado.

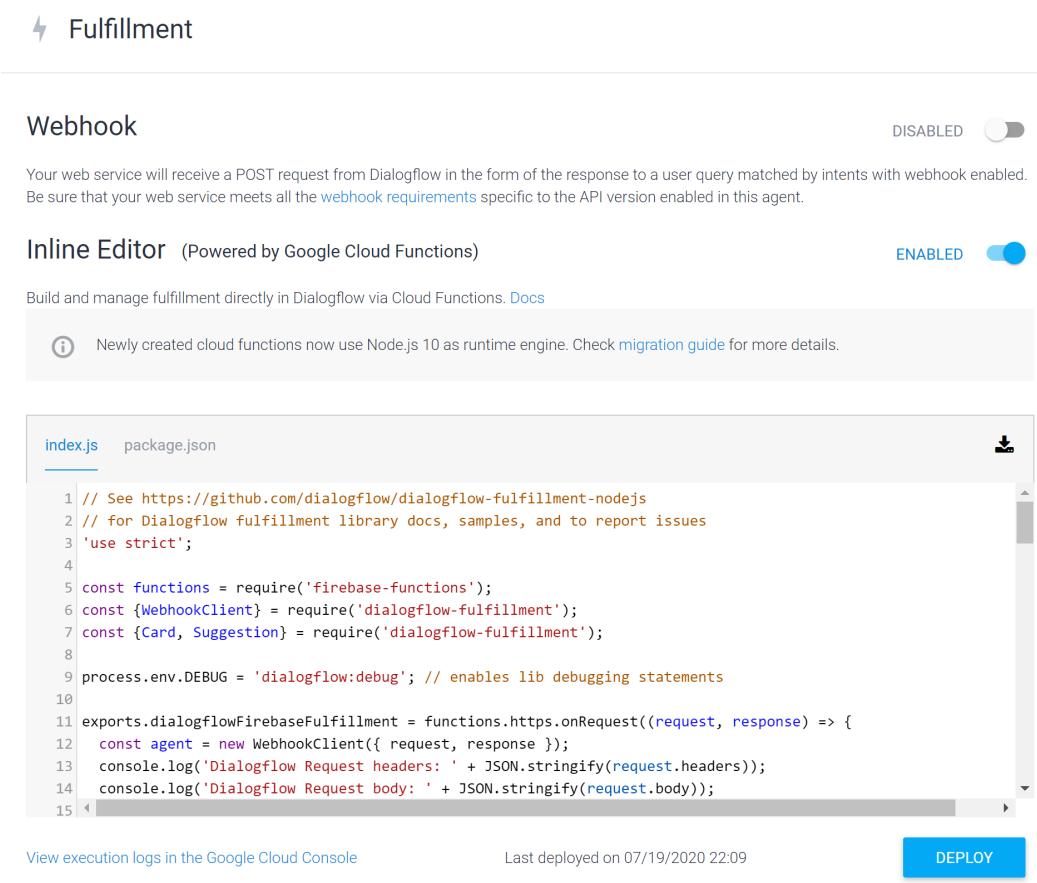


Figura 13: Interfaz de fulfillment completa.

La opción de webhooks[38] se puede utilizar para conectar con servicios externos, como puede ser el propio servidor, a través de solicitudes JSON o las Cloud Functions de Google. Esto permitirá mantener una lógica compleja dentro del propios sistemas, conectando con bases de datos para dar respuestas personalizadas o mantener un seguimiento de la ruta que el usuario tome.

La opción de Inline Editor es la más versátil, ya que permite personalizar respuestas dentro de un servidor de Node.js sin tener que preocuparnos de la infraestructura, solamente centrándonos en el código ya que utiliza la propia biblioteca de entregas de DialogFlow. Este editor está pensado para pruebas de entrega y prototipado rápido, ya que permite la ejecución en el momento. Cuando ya se hayan hecho todas las pruebas pertinentes, podemos descargar el código utilizado para llevar la aplicación a producción utilizando webhooks, alojándolo en un servidor propio o en funciones en la nube.

[38] Guia de Servicio de webhooks de DialogFlow

En el ejemplo del apartado 5, utilizaremos webhooks para poder trabajar en entorno local y no depender de las sucesivas subidas en la nube.

Actions On Google

Es una plataforma de desarrollo creada por Google que nos permitirá crear nuestra aplicación en *Google Assistant,* además de en otras interfaces tanto gráficas como por voz del ecosistema de Google, como pueden ser todos los altavoces inteligentes de Google.
En el modo que estamos utilizando actualmente para crear nuestra aplicación, nos permitirá personalizar la identidad visual y oral del asistente, conectar autenticaciones, configurar la invocación para el asistente, testear cualquier cambio, lanzarlo a producción y staging. Además, te proporciona un apartado de analítica sobre cómo está funcionando la aplicación en los terminales que la abren, permitiendo conocer la retención y uso de tu aplicación. Este apartado se puede compatibilizar con el apartado de analítica de Dialogflow y creando eventos personalizados dentro de Firebase.

Durante 2020, ha pasado por múltiples cambios que vale la pena mencionar, debido principalmente a la guerra de *Smart Home* que existe entre *Alexa* y *Google Home*.
A finales de 2020, podemos acceder a múltiples instancias de la plataforma dependiendo de nuestro objetivo y herramientas que utilicemos. Esto es debido a que quieren facilitar el uso por parte de los desarrolladores y no le suponga gran esfuerzo nutrir al nuevo ecosistema. La integración que se utiliza en este TFG utiliza una librería que conecta *Actions On Google* con *DialogFlow*, por lo que hace de *Actions On Google* un mero repositorio de la aplicación, como anteriormente se menciona.

Actualmente, si creamos un nuevo proyecto en Actions on Google podremos seleccionar los siguientes:

Smart Home

Este modo vendrá configurado con las librerías específicas de los distintos dispositivos inteligentes de Google. En la interfaz de Actions On Google, podremos conectar las acciones a través de webhooks.

Food Ordering

Este es el modo que más se aleja de los demás. Nos permitirá conectar con el backend de nuestro restaurante para gestionar pedidos a domicilio a través de Google Assistant.

Game

Nos permitirá crear experiencias conversacionales a las que jugar a través de Google Assistant y otros dispositivos inteligentes.

Custom

Nos permitirá seleccionar entre estos ejemplos:

- Blank Project
- The Jungle Escape
- Firestore
- Conversation Components
- Facts about Google
- Hello World
- Using Dialogflow.

Este último es como se está efectuando la aplicación que se mostrará más adelante.

Firebase

Firebase es una plataforma de Google usada para el desarrollo de aplicaciones web y móviles de una forma fácil y rápida sin renunciar a la calidad que eso podría implicar. Es lo que en la industria se llama Backend as a Service (BaaS).

Se compone de diversas herramientas básicas utilizadas en cualquier aplicación, todas bajo una misma plataforma para facilitar el uso.

Para entender cómo se utiliza esta herramienta y porque es tan popular, es necesario conocer las distintas fases de creación de una aplicación y cómo Firebase te ayuda en cada punto.

Se clasificarian en 4 fases diferenciadas, existiendo diversas herramientas en cada una de ellas. A continuación se explicarán las más importantes.

1. Desarrollo.
    - **Realtime Database:** Es una base de datos No SQL optimizada para la sincronización y almacenamiento de información en tiempo real, de fácil uso y que no requiere de configuración.
    - **Cloud Firestore**: Es una base de datos de fácil uso que almacena información en lo que llaman colecciones de documentos, un formato muy similar al JSON. Especializada en ejecutar consultas indexadas complejas y de escalado automático, llegando a soportar 1 millón de conexiones concurrentes y 10.000 escrituras por segundo. Esta es la base de datos que se utilizará para nuestra aplicación debido a sus capacidades.
    - **Storage**: Un sistema de almacenamiento en la nube donde guardar información más pesada de los usuarios, como imágenes o ficheros. Permite personalizar accesos gracias a sus reglas de seguridad, conocidas por su confiabilidad. Esta herramienta se utilizará para el almacenamiento de imágenes.
    - **Functions**: Ejecuta cualquier código escrito en funciones gracias a su librería disponible en varios lenguajes de programación. Permite una gran escalabilidad ya que asignará los recursos necesarios al momento de ejecución. Se utilizarán las Firebase Functions para albergar la lógica de negocio necesaria.
    - Otras herramientas de uso frecuente son **Machine Learning**, que permite acceder a soluciones preconfiguradas,

Authentication, para validar a nuestros usuarios de forma segura, además cuenta con un servicio de hosting

2. **Lanzamiento y monitorización**. Nos permite conocer el estado de nuestra aplicación y corregir errores de lanzamiento.
    - **Crashlytics**: Hace seguimiento de fallos y genera informes sobre que le ocurren a nuestros usuarios
    - **Performance**: Muestra métricas útiles para conocer el rendimiento de la aplicación.
    - Otras herramientas son **Test Lab**, para probar tu aplicación en dispositivos alojados en google, y **App Distribution**, un gestor para distribuir versiones preliminares.
3. **Interacción**: Presenta una serie de herramientas comunes con las que se puede interactuar con el usuario y mejorar los objetivos de negocio.
    - **A/B Testing**: Permite comparar distintas versiones de implementación de una funcionalidad para conocer qué funciona mejor.
    - **Admob**: Gestor de anuncios de Google con el que monetizar la aplicación.
4. **Analítica**: Google pone a disposición del desarrollador varias herramientas que se utilizan para conocer mejor al usuario y mejorar el marketing.
5. **Extensiones**: Son soluciones listas para su uso, como puede ser la redimensión de imágenes, envío de emails, sincronización con *Bigquery* entre muchas otras creadas por Google u otras empresas.

4.- Diseño de un sistema de interacción

Diseño de la experiencia conversacional[39]

Para poder desarrollar un agente conversacional que ayude a conseguir los objetivos de negocio marcados, es necesario darle importancia a la creación de una experiencia conversacional que permita hacerse entender con el usuario y mantenerlo entretenido.
Esto en muchas ocasiones es complicado, es por eso que antes de comenzar con el desarrollo de la aplicación hay que completar una serie de tareas de análisis y diseño que nos permitirán crear una relación con nuestro usuario y evitar que deje la conversación a medias.
La primera será la representación de la marca con la que el usuario está conversando.

Identidad de Marca

Normalmente, detrás de un agente conversacional existe una marca con unos valores definidos, que se dirige a su audiencia. Tal y como un banco no transmite su mensaje con un anuncio de una persona que malgasta su dinero, nosotros tampoco podemos hacer que el agente transmita una identidad que no sea propia de la marca. Es por eso, que tenemos que concretar las siguientes cosas:

1. **Nombre**: Restaurante robotizado
2. **Audiencia**: Estudiantes universitarios y parejas jóvenes de la ciudad
3. **Valores de la Marca**: Rapidez, innovación y celebración de reuniones.
4. **Tono de voz**: Joven, casual y amigable.

[39] Procedimiento obtenido del blog Medium sobre cómo Google diseñó su Google Assistant para el evento de Google I/O de 2018.
[https://medium.com/google-developers/how-we-designed-it-the-google-i-o-18-action-for-the-google-assistant-9370ffbaf9b0]

Acciones

Aquí identificamos las acciones que el negocio quiere realizar, localizando cual es la principal para construir la experiencia conversacional. Para ello se responderán a estas 3 preguntas:

- **Servicios**: ¿Cuáles son las principales actividades del negocio?
    - Dar comidas, enviar comida a domicilio y hacer eventos privados.
- **Ideas de acciones**: ¿Qué acciones podrán resolver los puntos de tensión que tienen nuestros usuarios?
    - Reservar a través de la voz, conocer el menú, poder hablar con el encargado.
- **Acción principal** ¿Cuál sería la más factible y la más urgente?
    - Reservar mesa.

User Persona

Necesitamos entender las características comunes de nuestros usuarios para conformar una audiencia, y así saber como crear una buena conversación, adaptandonos a su lenguaje y formas de hablar. Para ello, tenemos que tener claro estos puntos:

- **Perfil de usuario** (Nombre, edad, características principales). Aunque tengamos muchos tipos de cliente, para crear un cliente ideal lo definiremos lo más realista posible:
    - Una mujer de 22 años, que viene con sus amigas para un reencuentro después de 3 meses sin verse.
- **Motivaciones**. Que quiere lograr con nosotros.
    - Reunirse con sus amigas para pasar una buena velada y cenar.
- **Ansiedades**. Cuales son sus limitaciones y que le impide acudir a nuestro negocio
    - Reservar en horarios nocturnos. Tiene un presupuesto limitado. Es muy indecisa.
- **Contexto**: Cuál es su entorno cuando te descubre
    - En la universidad con amigos.
- **Factores temporales** ¿Cuando interactúa con el asistente?
    - En la cafetería
- **Otras suposiciones**: Otro tipo de consideraciones que pueden afectar a la experiencia del usuario.
    - Ya ha interactuado con su Siri

System Persona

Por otra parte, tenemos que dar una identidad a nuestro asistente a través del System Persona. Para ello, tenemos que definir varias características que le den contexto. Estas serán:

- **Nombre**: Robote
- **Descripción**: Es el asistente encargado de recoger las propinas que dan en el restaurante por el buen servicio. Cuando todo está oscuro, es quien atiende a los clientes rezagados que llaman.
- **Rol**: Recepcionista del Restaurante Robotizado.
- **Estilo**: Casual.
- **Personalidad**: Amable y chistoso.
- **Frase de prueba**: Mmmm… Me está entrando hambre ¿A ti también? ¿Por qué no echas un vistazo a nuestro menú?

Diálogos de prueba

Ya sabiendo como será la personalidad del asistente y como será la persona con la que nos comunicamos, tendremos que escribir unos diálogos de prueba. Con esto conseguiremos entender cómo se sentirá el usuario en un diálogo real, evitando escribir distracciones que pierdan la atención del objetivo buscado. Hay un dato que conviene recordar para no sobredimensionar el asistente con cientos de opciones. El 80% de los usuarios tomarán el 20% de las rutas típicas. Estas las consideraremos los casos de uso clave. Las peticiones del 20% restante de la audiencia las consideraremos desviaciones, en cuyo caso podremos registrarlas para incluirlas a nuestro sistema más adelante.

Primero, intentaremos fingir una conversación de forma oral con un único caso de uso, fingiendo con otra persona un juego de roles. Así entenderemos cuán natural puede ser la conversación que estamos planeando.

Diagrama de la conversación

Consideraremos los casos de uso más frecuentes en nuestro caso, creando un flujo conversacional que contempla las posibles respuestas que nuestros usuarios tengan.
Consejos para crear buenas conversaciones:

- No repetir preguntas donde se pida información que ya se ha pedido
- Basarse en el contexto anterior para respuestas posteriores
- Utilizar el experimento del mago de Hoz. Dos tipos:
    - Sencillo: Fingir una conversación con una personas que no conozca tu solución
    - Completo: Simular una conversación, pasando por un Text-To-Speech lo que quieras decirle al usuario. Esta forma permitirá tener un ambiente mucho más realista. Si no funciona bien el sonido, podremos utilizar el lenguaje de marcado SSML.

Durante la creación del diálogo tendremos que estar pendiente de las siguientes cosas:

- Detectar que la conversación sea natural.
- Detectar si los usuarios se encuentran confundidos.
- Conocer si el usuario se expresa de forma inesperada.
- Ver si el usuario muestra señales de frustración o impaciencia
- Observar quien ha hablado más.

Dentro de los Anexos, se podrá acceder al flujo de la conversación diseñada para la prueba.

Diseño del sistema conversacional

Para poder obtener los objetivos fijados en el diseño de la experiencia conversacional, se tiene que organizar correctamente la estructura del programa, a través del uso de las herramientas explicadas anteriormente y adaptándolas al uso que deseemos.
Dialogflow permite gestionar intents sencillos de forma fácil a través de su propia interfaz. Los intents más complejos se gestionarán a través de webhooks.
Firebase alojará la lógica de negocio mediante las Firebase Functions, almacenará las imágenes con su herramienta Storage y se utilizará la base de datos Cloud Firestore para guardar la información de pedidos, menú y usuarios.
Para facilitar la ampliación de información en DialogFlow, es necesario organizar bien la información que se presenta y las funciones utilizadas, por ello se realiza un diagrama de los distintos intents y funciones asociadas, que se verá en el punto 4.1.2.

Los intentos en Dialogflow se nombran por palabras separadas por puntos, siendo la primera una definición de la aplicación, la segunda la acción a realizar, y la tercera una especificación de dicha acción. Un ejemplo será:
restaurante.info.horario,
que muestra información sobre la apertura del restaurante y nos permitirá diferenciarlo de la función
restaurante.info.carta
que nos muestra información sobre los platos disponibles

Diagrama de flujo de la conversación

Esto es una representación de la conversación que se daría con el chatbot. Muestra las posibles entradas que el usuario puede dar, las respuestas del bot y las llamadas a fuentes externas para consultar y/o extraer la información necesaria. Este diagrama (Fig. 14) nos permitirá tener una visión completa de la conversación y nos permitirá gestionar las herramientas que queramos usar de forma modular (APIs, NPL, etc..).

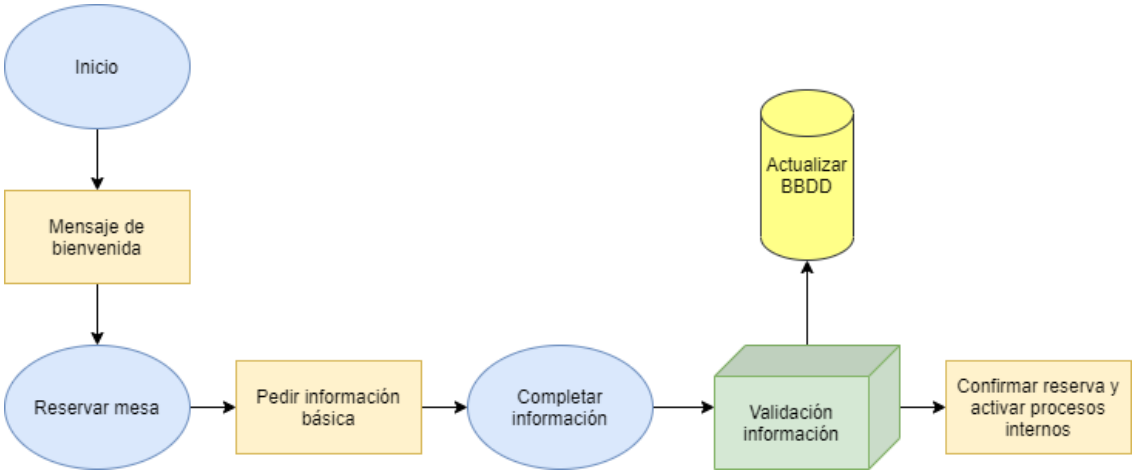


Figura 14: Flujo de ejemplo de una aplicación real

Este diagrama no debe incluir todas las posibilidades que haya en una conversación, ya que pueden ser numerosas e inesperadas, se tiene que adaptar a los objetivos de negocio que se persiguen solventar con la implementación del chatbot.

Para la integración continua en un negocio, tendremos que ir desarrollando cada vez diagramas más complejos cuando validemos los simples, centrándonos en la línea de negocio principal para comprobar la rentabilidad.

En este diagrama aparecen los intents que conllevan una acción específica relacionada con el negocio.

Además de estos, existirán dos intents especiales que se generan de forma predeterminada en DialogFlow:

Default Welcome Intent:

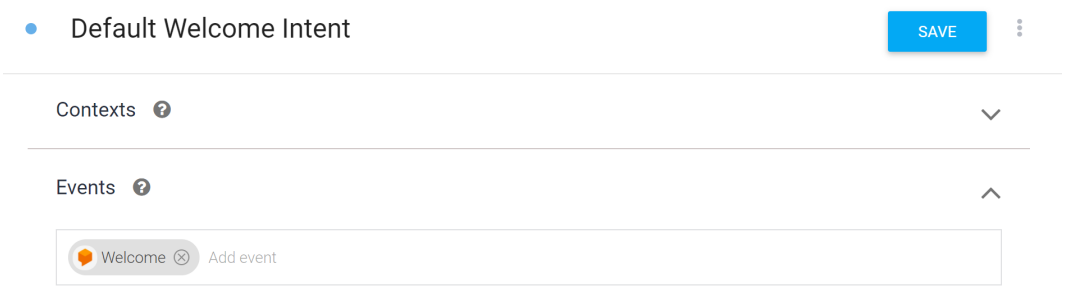


Figura 15: Evento de bienvenida por defecto

Es el primer intent con el que un usuario interactúa tras realizar la llamada al agente. Este intent es ejecutado debido a que tiene el evento "Welcome", permitiéndonos modificar el nombre.

Default Fallback Intent:

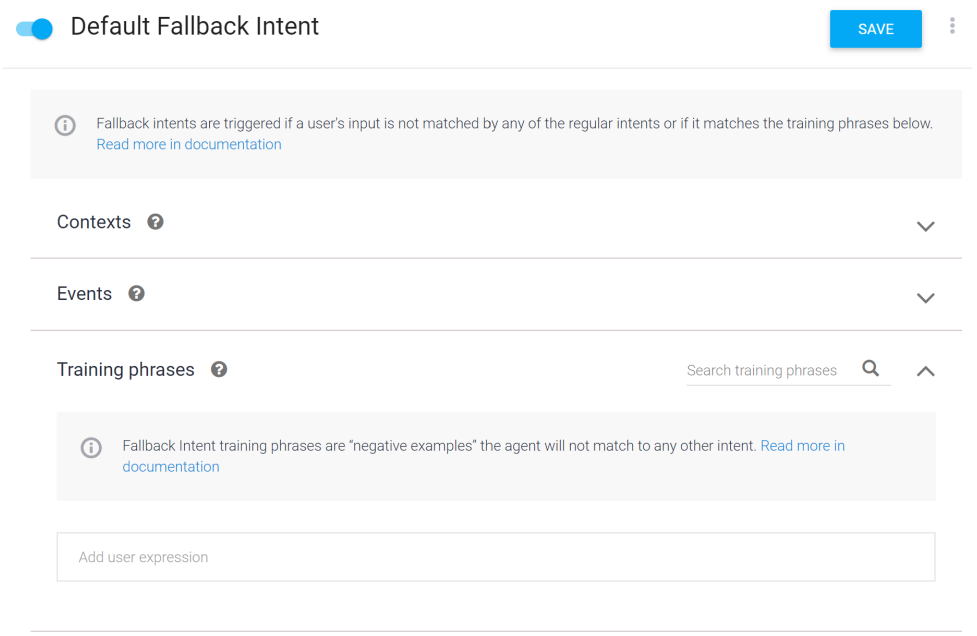


Figura 16: Intento de fallback predeterminado

Este intent se activa cuando el agente no reconoce la entrada que proporciona el usuario como invocación a algún intent. Además se pueden incluir ejemplos negativos, es decir, frases que podrían invocar a otras funciones, pero al introducirlas aquí lo evitan.

Debido a que el programa de Speech-to-Text de Google no puede reconocer todas las frases del usuario, este intent se vuelve de vital importancia para ayudar al usuario y evitar que se frustre.

Funciones utilizadas

- **Carta**: A través del *intent* llamado restaurante.info.carta, muestra los platos disponibles en el restaurante, pudiéndose mostrar por platos (primero, segundo y postre) o por categoría (ensalada, hamburguesa, pizza).
- **Horario**: A través del *intent* restaurante.info.horario, permite conocer el horario de apertura del restaurante.
- **Reserva en restaurante**: Esta función comienza a través del *intent* restaurante.reserva.crear, comenzando un flujo para reservar a una determinada hora en el restaurante.
- **Comida a domicilio**: Esta función comienza a través del *intent* restaurante.comida.domicilio, comenzando un flujo donde puedes seleccionar que quieres pedir.
- **Opciones**: Muestra que es posible hacer con el agente si el usuario se muestra perdido. Se realiza con el *intent* restaurante.opciones

A continuación en el apartado 5, se explicará la implementación de estas funciones y la explicación del código utilizado, así como los distintos servicios utilizados.

5.- Implementación y resultados de la solución

Preparación de un entorno de desarrollo ágil

Debido a que trabajaremos con tecnologías de Google, es necesario crear o tener una **cuenta de Gmail**, con la que podremos acceder a todos sus servicios y herramientas de forma gratuita. Para esta guía utilizaremos las herramientas de Actions on Google, DialogFlow y Firebase.

El primer paso será crear un nuevo proyecto en Actions On Google. Rellenamos la información y seleccionamos *Custom > Aplicación con DialogFlow*.

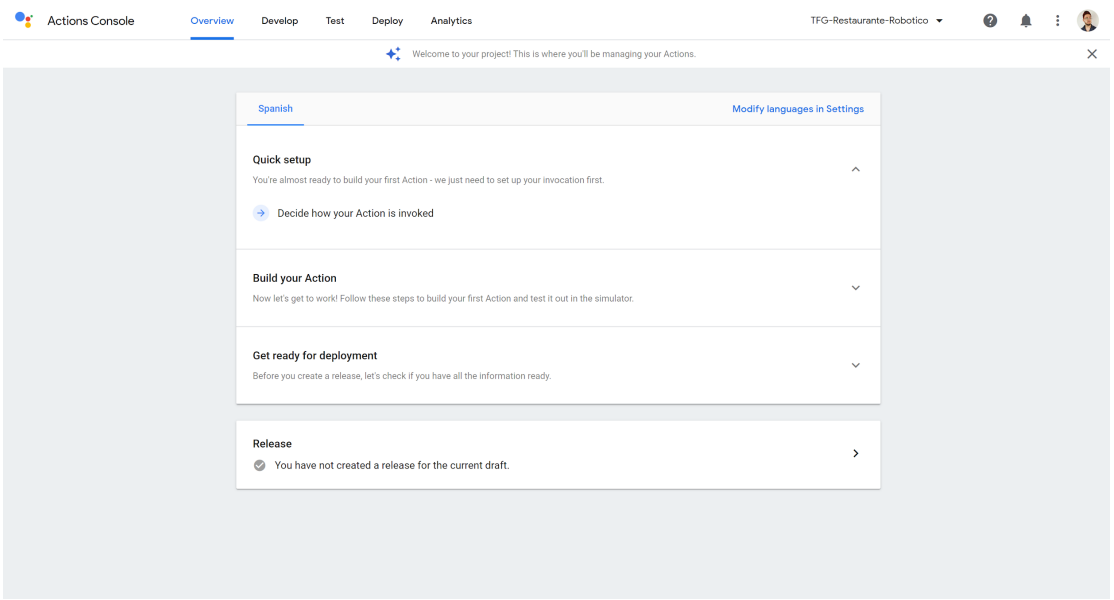


Figura 17: Configuración de Google Actions

Completamos el "Quick Setup" y entramos en "Build your Action" y seleccionaremos *Custom Intent*, tal y como vemos en la figura 17. De aquí nos llevará a crear un agente en DialogFlow. Estará todo integrado gracias a que están asociados a un proyecto de Google Cloud Platform.

Para evitar utilizar el "Inline Editor" de DialogFlow, que resulta muy deficiente y no tiene control de versiones ni ejecución rápida, vamos a

crear nuestro propio entorno de pruebas mediante las Cloud Functions de Firebase.
En mi caso, utilizaré un emulador de Firebase en local, conectando con un servicio externo el endpoint local con el requerido en el webhook de DialogFlow. (Esto es debido a que los webhooks de DialogFlow requieren de conexión HTTPS segura).
Ejecutando y configurando estos comandos en el repositorio local que queramos podremos cargar las bibliotecas requeridas.

- *npm install firebase-tools -g*
- *firebase init*
- *npm install actions-on-google*

Una vez esté instalado, podremos hacer nuestro primer *deployment* mediante el comando *firebase deploy*. Es necesario que haya una cuenta de facturación habilitada en el proyecto.

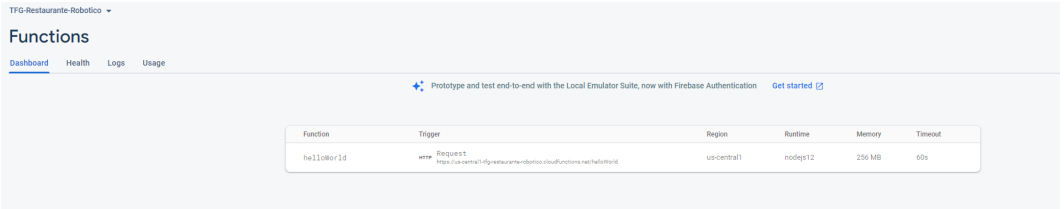


Figura 18: Panel de Firebase Functions

Se creará una función que realizará lo que hay dentro de ella cada vez que se ejecuta. Para que podamos conectarlo con dialogflow, necesitamos incluir varias líneas de código a mayores:

```
const functions = require('firebase-functions');
const admin = require('firebase-admin');

const {
    dialogflow,
    Image,
    BrowseCarousel,
    BasicCard,
    SimpleResponse,
    List,
    Button,
    Carousel,
    QuickReplies,
  } = require('actions-on-google');

const app = dialogflow();

exports.dialogflowFirebaseFulfillment =
functions.https.onRequest(app);
```

Código de inicialización

Con esto tendremos un *endpoint* al que conectar el *webhook* de DialogFlow y nos servirá para poner en producción nuestro proyecto.
Como siempre es necesario tener un entorno de pruebas donde ver rápidamente los cambios, simularemos las firebase functions en nuestro ordenador.
Para ello utilizaremos el comando *firebase emulators:start*.

Esto servirá cuando queramos poner en producción nuestro proyecto, pero para ir desarrollando y que podamos ir viendo los resultados, emularemos en local mediante el comando *firebase emulators:start*.

El endpoint que se creará será en local, que tendrá una estructura de este tipo:
http://localhost:5001/tfg-restaurante-robotico/us-central1/dialogflowFirebaseFulfillment).
Esta URL no podremos utilizarla con DialogFlow para las pruebas, por lo que tendremos que utilizar un servicio web llamado ngrok, que creará un túnel desde esa URL a una con HTTPS.
Será del estilo a:
https://a30b977cdcc3.ngrok.io/tfg-restaurante-robotico/us-central1/dialogflowFirebaseFulfillment

En la cuenta gratuita de ngrok, el subdominio cambiará cada vez que cierres el túnel y se necesitará cambiar la URL introducida en el webhook.

Se realiza una comprobación comprobando si se ejecuta el intent de bienvenida, finalizando la preparación del entorno.

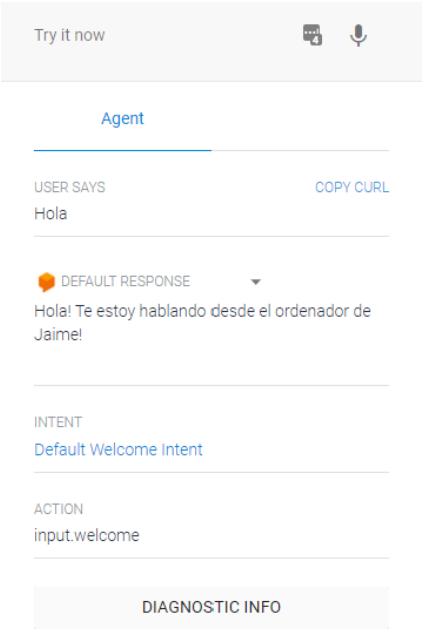


Figura 19: Entorno de pruebas integrado en DialogFlow

Configuración y ajustes del agente y entorno

Agente

Podremos acceder a la configuración haciendo click en la rueda dentada. Nos aparecerán varias pestañas que utilizaremos según las necesidades del negocio. Se definirán las más interesantes para el desarrollo:

- General: Podemos ver el proyecto de Google Cloud Platform asociado y configurar la generación de logs.
- Languages: Se pueden seleccionar varios lenguajes que utilizará el agente, incluso seleccionar variantes locales para mostrar más cercanía con el usuario.
- Speech: Permite variar la voz por defecto que viene en Dialogflow, ajustando la ganancia, el todo y la ganancia de volumen.

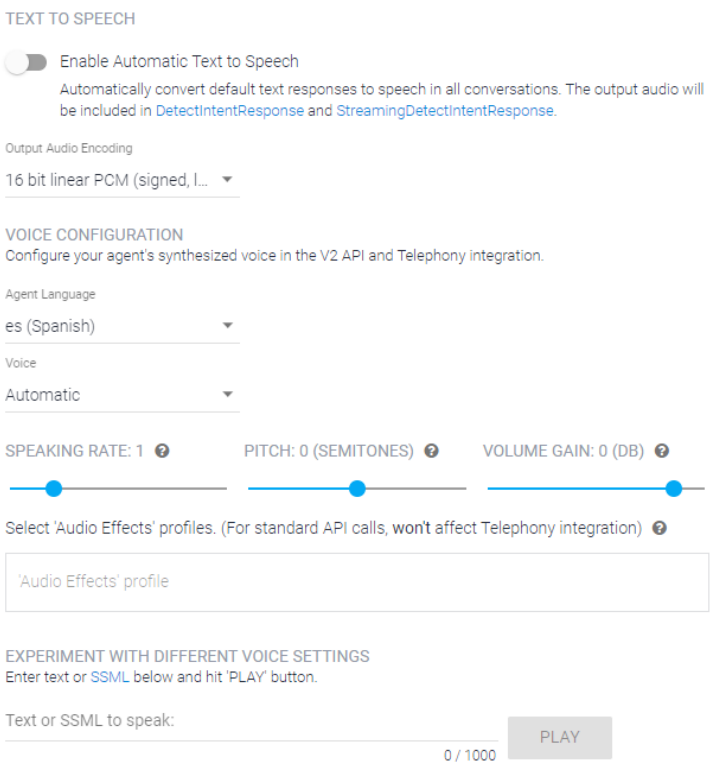


Figura 20: Configuración del tono de voz.

- Export and Import: Nos permite exportar nuestro agente a través de un archivo comprimido, para después ser importado en otra cuenta distinta. Muy útil para gestionar copias de seguridad y como traspaso de conocimiento entre desarrolladores, al poder crear intentos recursivos.

- Share: Vital para equipos de desarrollo, ya que permite que otras personas puedan modificar el agente e implementar mejoras. La persona que quiera acceder necesitará una cuenta de google y los permisos pertinentes.

Serverless Firebase Functions:

Para poder crear las funciones asociadas a los distintos intentos, se necesitan utilizar diversas librerías e inicializar el entorno que se ejecutará. En nuestro caso utilizaremos el siguiente código:

```
const functions = require('firebase-functions');
const admin = require('firebase-admin');
admin.initializeApp();

const {
    dialogflow,
    Image,
    BrowseCarousel,
    BasicCard,
    SimpleResponse,
    List,
    Button,
    Carousel,
    QuickReplies,
  } = require('actions-on-google');
const { convert } =
require('actions-on-google/dist/service/actionssdk');

const app = dialogflow();
const db = admin.firestore();
const platos_collection = db.collection('platos');
const contacto = db.collection('reservas');
const ocupacion = db.collection('ocupacion');

let pedido_array = {};
let now = new Date;

exports.dialogflowFirebaseFulfillment =
functions.https.onRequest(app);
```

Código de configuración de Firebase

Creación y manipulación de bases de datos

El fácil manejo de las bases de datos por parte de firebase y su integración en un servidor node.js, nos permite implementar esta solución para ofrecer un servicio más personalizado. Esta implementación se realizará a través de las *firebase functions*.

Declaraciones de cabecera

Para poder utilizar nuestra base de datos creada en firebase, necesitamos incluir las siguientes librerías:

```
const functions = require('firebase-functions');
const admin = require('firebase-admin');
admin.initializeApp();
const db = admin.firestore();
```

Visualización de la base de datos.

Para poder ver la estructura y los datos almacenados en la base de datos de firestore, necesitamos acudir a la web de firebase, en la sección de firestore

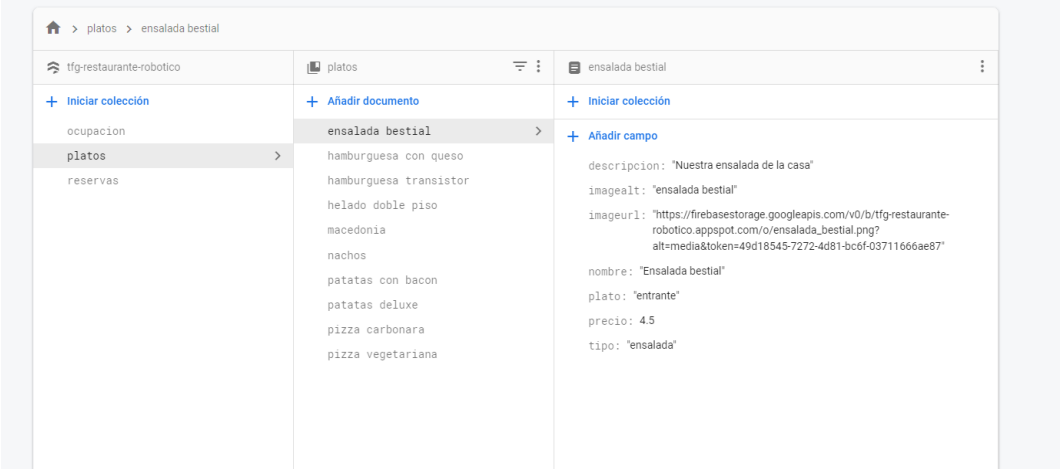


Figura 21: Muestra de la base de datos de Firestore

Escritura en la base de datos

Si queremos escribir en la base de datos, necesitamos conocer el nombre de la colección donde queramos escribir, el documento, el campo y el contenido que queramos escribir.

```
const res = await contacto.doc(ID).set({
  name: nombre,
  people: comensales,
  ID: ID,
})
```

En esta función, escribimos una respuesta con información que ha realizado el usuario y la almacenamos según su ID, que es con lo que identificamos al usuario en los siguientes intents que haga.

Modificación de un registro en la base de datos

Si lo que queremos es modificar o extender un registro en la base de datos, necesitamos utilizar otra función, ya que si no eliminaremos todo lo que no se defina. Un ejemplo de esta función sería:

```
const res2 = await contacto.doc(convID).set({
  year: year,
  month: month,
  day: day,
  hour: hour,
},{merge:true});
```

Que lo que hace es incluir la fecha de la reserva en la información de contacto una vez que se haya confirmado que existe sitio disponible.

Necesitamos crear un endpoint específico para utilizarlo en DialogFlow, así que para atender a los distintos intentos, incluiremos la siguiente línea de código, que es la única a la que atenderá DialogFlow cuando conectemos el webhook.
Para esto tendremos que incluir la siguiente línea de código:
exports.dialogflowFirebaseFulfillment = functions.https.onRequest(app);

Leer la base de datos

Para poder contrastar la información recibida con la existente en la base de datos, necesitamos leer la información correctamente, ya sea buscando según un registro en concreto, o dependiendo de unas condiciones específicas.

Para Obtener un objeto de un documento específico:

```
  const preciocomida = await
db.collection('platos').doc(option).get();
```

Nos devolverá un objetivo que podremos explorar con la función preciocomida.data().variable

Para obtener el resultado de una consulta a la base de datos:

```
    const snapshot = await platos_collection.where('plato', '==',
plato).get();
```

Nos devolverá un objeto con todos los documentos que cumplan dicha consulta.

Otras funciones de firebase

También podemos utilizar otras funciones más específicas que mejoran el rendimiento de algunas operaciones, como pueden ser sumar valores a una variable o añadir un nuevo valor a un array.

```
  const unionpedido = await contacto.doc(ID).update({
    pedido: admin.firestore.FieldValue.arrayUnion(option),
    cuenta: admin.firestore.FieldValue.increment(precio),
  });
```

Almacenamiento de imágenes y archivos en Firebase Storage

Las imágenes de los platos las alojaremos en la herramienta de Firebase llamada Storage, que almacenará el archivo en la nube.
Para poder utilizarlo entraremos en la web https://console.firebase.google.com/, una vez en el apartado Storage, podremos subir los archivos mediante el botón Subir Archivo, pudiendo configurar además las reglas de acceso a dichos archivos.

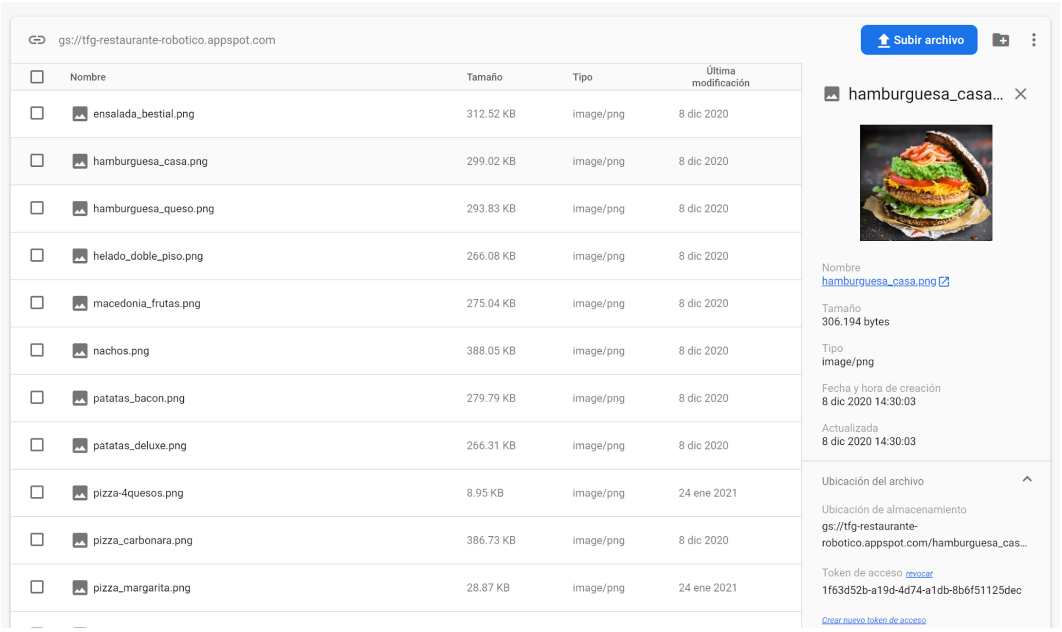


Figura 22: Uso de Firebase Storage para almacenar fotos de platos

Una vez incluida la imagen en la nube, podemos ver como nos muestra información acerca de esta imagen y nos vincula un token de acceso para que controlemos donde se utiliza nuestra imagen.

En nuestro caso utilizamos estas imágenes para mostrar los distintos platos del restaurante.

Creación de cada intento.

En este apartado se verán qué pasos se han realizado a la hora de crear los distintos intents que se han implementado en el trabajo final. En cada uno se explicarán las funciones y decisiones tomadas. El código completo se adjuntará en los anexos y nos centraremos en los puntos más importantes.

restaurante.comida.domicilio.inicio

Este es el intent con el que se comienza la acción de pedido a domicilio, ya que reacciona a la intención del usuario de pedir comida a domicilio, como las frases “Quiero pedir comida para llevar”, “Quiero que me traigas algo para cenar” y similares. Este intent establece un contexto para evitar que se mezclen intenciones del usuario y siga un flujo propio. Se pedirá el nombre, dando comienzo al siguiente intent relacionado “restaurante.comida.domicilio”.

restaurante.comida.domicilio.nombre

Un intent sencillo como contexto de entrada el definido en restaurante.comida.domicilio.inicio, para seguir con la rama del pedido a domicilio.
Cuando es introducido el dato correctamente, obtiene el nombre del usuario, junto con ID de la conversación que se usará como llave primaria por la que podremos unificar información de distintos intents.
Por último se establece un contexto que permitirá activar la petición sobre a qué domicilio enviarlo.

restaurante.comida.domicilio

Accesible solamente a través del contexto llamado “pedido_domicilio_1”, se una dirección al usuario para enviarla a nuestra base de datos mediante a través de las funciones serverless.
Una vez obtenidos los datos del usuario, se ejecuta la función asociada al intent donde se envía la información pedida junto un ID de la conversación para poder sobreescribir los datos que se requieran.

```
app.intent('restaurante.comida.domicilio', async (conv,params) =>
{

  const ID = conv.id;
  const calle = params.location['street-address'];
  const res = await contacto.doc(ID).update({
    estado: 'domicilio',
    ID: ID,
    calle: calle,
  })
})
```

Cuando se toma la dirección al usuario, se la repite para que nos confirme si es correcta. Esto puede ser mediante una respuesta afirmativa o negativa, dando introducción a dos intentos de seguimiento llamados “restaurante.comida.domicilio - yes” y “restaurante.comida.domicilio - no”.

En el caso afirmativo, se procederá a mostrar los diferentes platos que pueden ser pedidos.

En caso negativo, se incluirá un contexto único con el que el chatbot pueda volver a recoger la dirección y continuar normalmente el flujo.

restaurante.comida.pedido

Este intent nos permite mostrar al usuario todos los platos que se tienen disponibles en el restaurante, pudiendolo mostrar según tipo de plato y categoría. En el código a continuación se muestra la variable "todo" que se podrá utilizar en el futuro como input para mostrar solo un determinado tipo de platos.

```
app.intent('restaurante.comida.pedido', async (conv, params) => {
  //Predeterminado en todo, a futuro se podría limitar según
disponibilidad de platos.
  let todo = 'todo';
  conv.ask(`Elije que quieres que te llevemos:`);
  conv.ask(await getdata(todo));
})
```

GetData

En la función getdata() se hace un filtrado, en estos momentos inactivo, y obtiene un snapshot con dichas características, enviándolo a otra función que se encarga de mostrar por pantalla toda la información de los platos.

```
async function getdata(plato) {
  let carouselItems ={};
  if(plato === 'entrante' || plato === 'principal' || plato === 
'postre'){
    const snapshot = await platos_collection.where('plato', '==', 
plato).get();
    return(generaCarrousel(snapshot));
  }else{
    const snapshot = await platos_collection.get();
    return(generaCarrousel(snapshot));
  }
}
```

Para poder separar el tipo de plato de un menú, como pueden ser entrantes, principales y postres de otras peticiones por categoría, se realiza un condicional normalizando estas tres opciones. Las demás opciones se procesarán según la categoría que se haya definido en el plato.

Generar Carrusel:

Esta función devuelve un carrousel completo donde muestra cada plato existente con las condiciones que se han establecido.
Del snapshot se obtienen los datos del nombre del plato, imagen y tipo de plato para mostrarselas al usuario.

```
async function generaCarrousel(snapshot){
  let carouselItems ={};
  if (snapshot.empty) {
    console.log('No encuentro ninguno');
    return "No hay ninguno";
  }

  snapshot.forEach(doc=>{
    let itemdetails ={
      title:doc.data().nombre,
      description:doc.data().plato,
      image: new Image({
        url: doc.data().imageurl,
        alt: doc.data().imagealt,
        height: 600,
        width: 600,
      })
    };
   carouselItems[doc.id]=itemdetails;
   console.log(carouselItems[doc.id]);
  });

  return new Carousel({
    title:'Platos disponibles:',
    items: carouselItems,
  });
}
```

restaurante.info.carta

```
app.intent('restaurante.info.carta', async (conv,params) => {
  console.log('Hola');
  const respuesta = params.orden_plato;
  console.log(respuesta);
  conv.ask(`Genial, estos son los platos que tenemos`);

  //Permite que no aparezcan las tarjetas en las que no tengan
interfaz

  conv.ask(await getdata(respuesta));
  conv.ask('Hecho');
})
```

Existe un intento donde el usuario puede pedir la carta, pudiendo especificar si quiere ver entrantes, principales o postres, así como por categorías como pueden ser pizzas, hamburguesas, etc...

restaurante.info.horario

Un intent básico que proporciona el horario del restaurante cuando se le pregunte

restaurante.info.informacion

Proporciona información acerca del restaurante cuando el usuario no lo ha conocido previamente.
Puede volverse muy útil cuando el usuario no ha llegado a entender la propuesta de valor del restaurante.

restaurante.info.localización

Reacciona a cualquier pregunta acerca de la ubicación del restaurante, devolviendo la calle donde se encuentra.

restaurante.opciones

En caso de que el usuario denote duda o rechazo al chatbot mediante su interacción, se le mostrará qué acciones puede realizar.

restaurante.reserva.crear.inicio

Este intent captura la intención del usuario de realizar una reserva de mesa en el restaurante, estableciendo un contexto específico para recopilar la información necesaria en diversos intents.
Este contexto se llamará “reservamesadatos”.

restaurante.reserva.crear

Únicamente se podrá acceder a este intent si está establecido el contexto "reservamesadatos".
El chatbot pedirá información acerca del nombre al que se asignará la reserva y el número de personas que serán, para así comprobar ocupación, y, en el caso de ser un grupo más grande que las restricciones impuestas, se le recomendará pedir comida a domicilio.

```
app.intent('restaurante.reserva.crear', async (conv, params) =>{
    const nombre = params.person.name;
    const comensales = params.number;
    const ID = conv.id;

    if(comensales > 6)
    {
        conv.ask(`¡Lo lamentamos mucho ${nombre}! Debido a las
restricciones solamente se puede reservar para un máximo de 6
personas. \n Pero, ¿Sabías que puedes pedir a domicilio?
`);
//Aquí se abre la opción de pedir comida a domicilio.
    }
    else{
        conv.ask(`¡Gracias ${nombre}! ¿Puedes decirme para qué día y
que hora sería?`);
        //Lanzar intento de restaurante.reserva.fecha
        conv.contexts.set('reserva_fecha', 2);
        //Cuando se inicia una sesión con el chatbot, se guardan los
datos de identificación antes de proceder propiamente la reserva.
Además se guarda el ID de la conversación permitiendo guardar toda
la información en un solo slot.
        const res = await contacto.doc(ID).set({
          name: nombre,
          people: comensales,
          ID: ID,
        })
    }
})
```

restaurante.reserva.fecha

Este *intent* recopila la respuesta a la pregunta acerca de cuando quiere realizar una reserva y se accede a través del contexto llamado "reserva_fecha".
Mediante la fecha que el usuario proporciona, se extraen año, mes, día, hora y día de la semana por separado para contrastar con las restricciones existentes.
Se ha establecido que el horario de apertura sea únicamente de 9:00 a 17:00, y el lunes como día no laborable en el restaurante. La ocupación máxima del restaurante será de 20 personas en cada franja horaria de 1 hora.

```
app.intent('restaurante.reserva.fecha', async (conv,params) => {

  const fecha_bruto = params['date-time'].date_time;
  const fecha = new Date(fecha_bruto);
  console.log(fecha_bruto);
  var day = fecha.getDate();
  var weekday = fecha.getDay();
  var month = fecha.getMonth()+1;
  var year = fecha.getFullYear();
  var hour = fecha.getHours();
  convID = conv.id;
  let ID_collection;
  let slots_available;
  let slots_asked;

  if (isNaN(hour)) {
    conv.ask('No me has dicho a que hora, ¿Puedes repetirme para
cuando sería?');
    conv.contexts.set('reserva_fecha',1);

  }else{

    if (weekday === 1) {
      conv.ask('Lo siento, los lunes está cerrado. ¿Podrias
decirme otro día?');
      conv.contexts.set('reserva_fecha',1);
    }
    else if(!(hour >= 9 && hour <=17)){
      conv.ask('Lo siento, a esa hora estamos cerrados ¿Quieres
decirme otra hora?');
```

```
      conv.contexts.set('reserva_fecha',1);
    }
    else{

          const snapshot = await ocupacion.where('year',
'==',year).where('month', '==', month).where('day',
'==',day).where('hour', '==', hour).get();
          if(snapshot.empty){
            console.log('Documento vacío');
            // Error en esta carga de documentos
            const res = await ocupacion.add({
              year: year,
              month: month,
              day: day,
              hour: hour,
              slots: 20,
            })
          }
          console.log('Existe ya un documento con estas fechas');

          const idreference = await
db.collection('reservas').doc(convID).get();
          console.log(idreference.data().people);
          slots_asked = idreference.data().people;

          snapshot.forEach(doc => {
            console.log(doc.id, '=>', doc.data());
            ID_collection = doc.id;

            slots_available = doc.data().slots - slots_asked;
            slots_available = parseInt(slots_available);
            console.log(`Slots pedidos: ${slots_asked}`)
            console.log(`Huecos libres ${slots_available}`);
          });


          if (slots_available <0) {
            conv.ask('Estamos completos a la hora que nos has
comentado \n Dinos otro día por favor');
            conv.contexts.set('reserva_fecha',1);
          }
          else{
          const res = await
ocupacion.doc(ID_collection).update({slots: slots_available});
          const res2 = await contacto.doc(convID).set({
            year: year,
```

```
            month: month,
            day: day,
            hour: hour,
          },{merge:true});
          conv.close('Tenemos hueco a esa hora. Te enviaremos las
normas que hay que cumplir por el Covid-19 y el menú del día a
través de las notificaciones del asistente de Google el día de la
reserva ¡Muchas gracias por confiar en nosotros!');
          }
        }
      }
    })
```

Mediante las variables obtenidas de fragmentar la fecha, hacemos una consulta a nuestra base de datos no relacional llamada “ocupacion” si existe algún documento cuyas variables coincidan con las actuales.

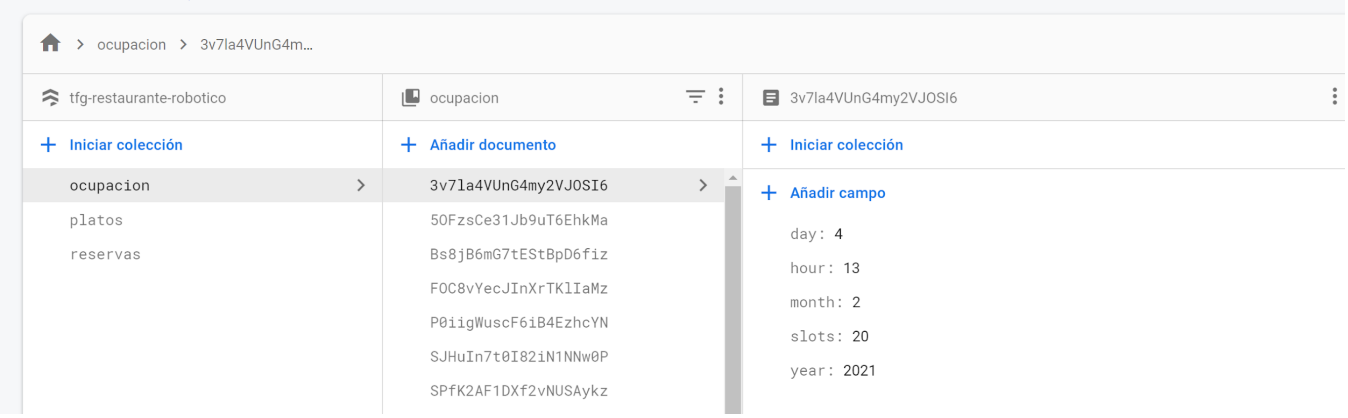


Figura 23: Base de datos y esquema para determinar ocupación

Si no existiera dicho documento, significa que nadie ha efectuado todavía una reserva en ese hueco, por lo que se crea un registro con esa fecha.

Posteriormente, con el documento ya creado, se obtiene el dato de huecos disponibles en dicho espacio de tiempo, que se ubica en el campo “slots”.

Si el número de espacios disponibles es menor que el de pedidos, se devuelve un mensaje al usuario sobre que no hay huecos disponibles y si quiere elegir otro día.

6.- Resultados

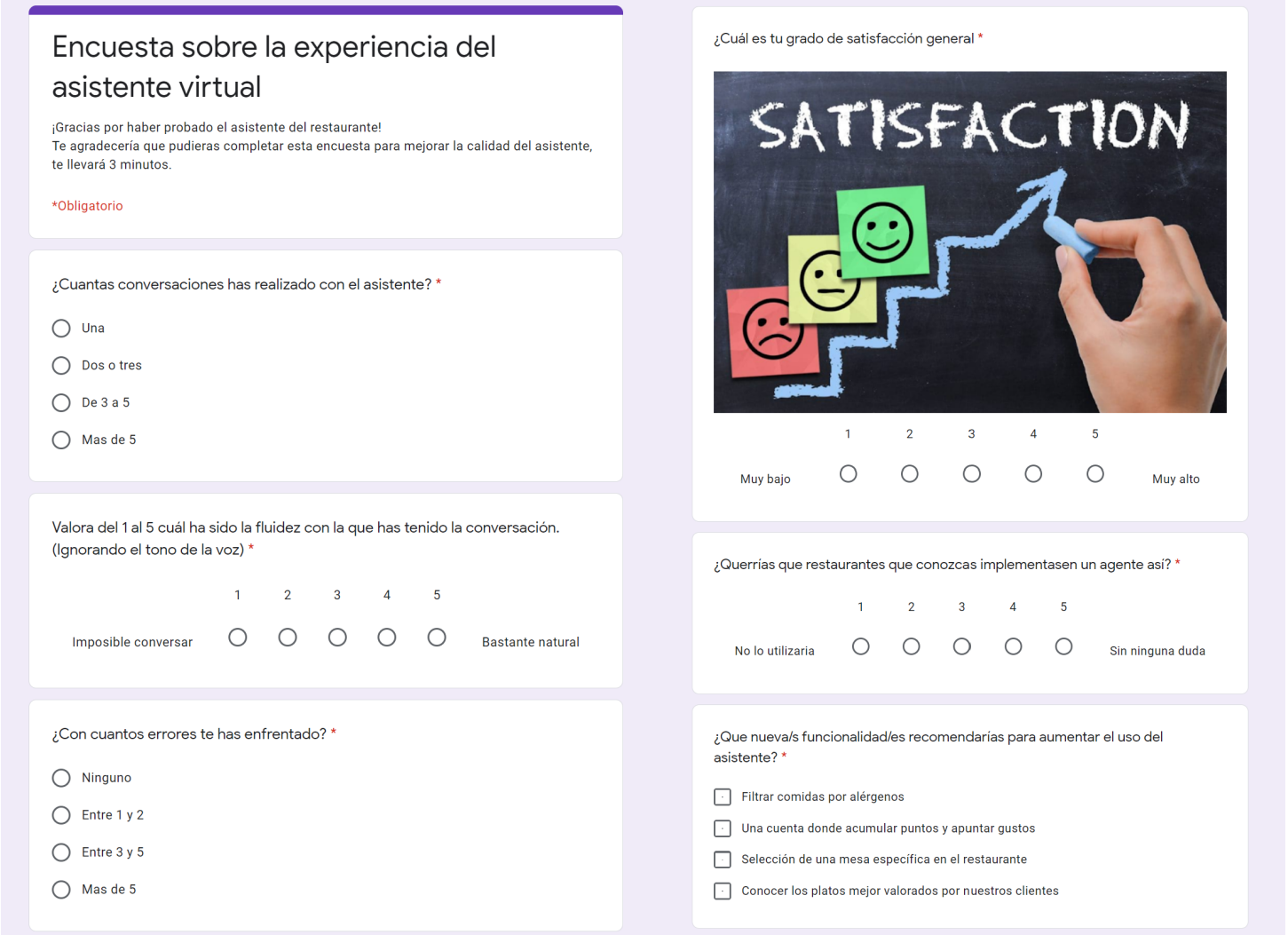
La forma más realista de poder analizar los resultados de la implementación de un agente conversacional y determinar su calidad en un ambiente real es por medio de un formulario creado para aquellas personas que lo han probado, cuantificando los resultados y pidiendo dar su opinión acerca de las funcionalidades percibidas.

Para su uso, solamente se les ha comunicado que es un asistente virtual para atender en cualquier momento a clientes en un restaurante, en medio de la actual pandemia de covid-19 y que es posible comunicarse de forma escrita y de forma hablada con el asistente.
No se les ha dado más indicaciones para ver de forma fidedigna la experiencia del usuario tratando de comunicarse con el asistente.
Se ha tratado de coger perfiles de distintas edades, para conocer las distintas dificultades afrontadas durante la conversación.

Preguntas de la encuesta

Las preguntas de la encuesta se han realizado pensando en conocer la experiencia que ha tenido el usuario con el asistente para saber si se necesita mejorar o corregir algún apartado, así como prospectar si puede ser rentable implementarlo en establecimientos debido a que aporta un valor diferencial al usuario y se vuelve una forma sencilla de procesar ese tipo de peticiones.
Las primeras preguntas se dirigen a evaluar la calidad del asistente, .

Debido a que el asistente se ha probado con personas en un ambiente ficticio, no ha sido posible conocer cuál sería la asiduidad con la que lo podrían estar utilizando si estuviera implementado en su restaurante favorito, y es por eso que se han incluido dos preguntas al final sobre su opinión acerca de su uso en el mundo real.



Encuesta sobre la experiencia del asistente virtual

¡Gracias por haber probado el asistente del restaurante!
Te agradecería que pudieras completar esta encuesta para mejorar la calidad del asistente, te llevará 3 minutos.

*Obligatorio

¿Cuantas conversaciones has realizado con el asistente? *

- Una
- Dos o tres
- De 3 a 5
- Mas de 5

Valora del 1 al 5 cuál ha sido la fluidez con la que has tenido la conversación. (Ignorando el tono de la voz) *

	1	2	3	4	5	
Imposible conversar	○	○	○	○	○	Bastante natural

¿Con cuantos errores te has enfrentado? *

- Ninguno
- Entre 1 y 2
- Entre 3 y 5
- Mas de 5

¿Cuál es tu grado de satisfacción general *

	1	2	3	4	5	
Muy bajo	○	○	○	○	○	Muy alto

¿Querrías que restaurantes que conozcas implementasen un agente así? *

	1	2	3	4	5	
No lo utilizaria	○	○	○	○	○	Sin ninguna duda

¿Que nueva/s funcionalidad/es recomendarías para aumentar el uso del asistente? *

- [ ] Filtrar comidas por alérgenos
- [ ] Una cuenta donde acumular puntos y apuntar gustos
- [ ] Selección de una mesa específica en el restaurante
- [ ] Conocer los platos mejor valorados por nuestros clientes

Figura 24: Contenido de la encuesta acerca de la experiencia del usuario

Análisis de las respuestas

Debido a que el número de respuestas a esta encuesta ha sido reducido, 14 respuestas, se considera una muestra demasiado pequeña para que pueda resultar de utilidad mediante método científico. De todos modos, cabe destacar que resulta de gran utilidad para conocer la tendencia general acerca de la interacción que han experimentado los usuarios y su posible futuro desarrollo para implementarlo en el mercado real.

Primera pregunta: Conversaciones con el asistente

Esta pregunta tiene como objetivo conocer la exploración que han realizado los usuarios del servicio y las funcionalidades que han experimentado.
Esta pregunta es la que más diferiria en su objetivo final, ya que tanto que en la encuesta se utiliza para conocer el proceso de exploración, en un ambiente real se utilizaria para medir el uso de la aplicación, calculando así diversas métricas de rendimiento como el número de interacciones diarias, horario de uso, *churn rate* (o tasa de cancelación).

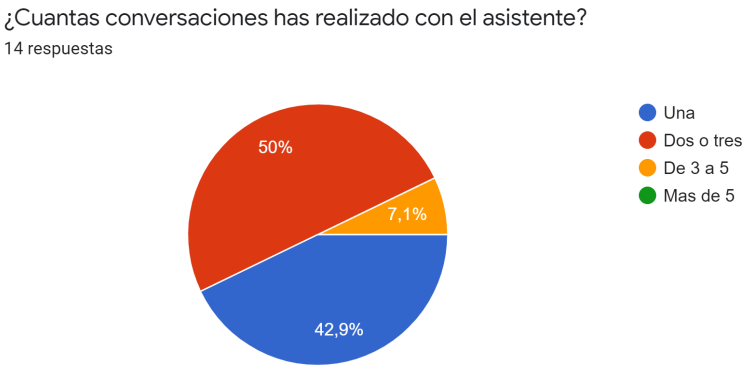


Figura 25: Número de conversaciones

Como era de esperar, la mayoría de personas han tenido menos de 3 interacciones con el agente. Esto es debido a que la conversación únicamente tiene fin en tres supuestos, que es cuando se termina de reservar una mesa, se termina de pedir comida a domicilio o te despides del asistente. Debido a que existen solo 3 formas de cerrarlo, la gran mayoría de personas han experimentado con alguno de los flujos conversacionales. No se ha tenido necesidad de mantener más de 5 conversaciones, debido a que es un agente conversacional circunstancial en fase de prueba. En un entorno real los resultados diferirian dependiendo del los días que hayan pasado desde el lanzamiento.

Segunda pregunta: Fluidez de la conversación

Para conocer cuál ha sido la sensación que ha tenido el usuario cuando mantenía una conversación con el agente conversacional, se ha tenido que puntuar del 1 al 5 la fluidez con la que han tenido la conversación.
Esto permite conocer si es necesario incluir nuevas expresiones que logren más cercanía con el usuario o pulir algunos apartados.

Figura 26: Resultados de la pregunta acerca de la fluidez del agente

La puntuación ha resultado satisfactoria, preferentemente debido a las interacciones cortas y debido al conocimiento de las funcionalidades por parte de los usuarios.
La forma de mejorar esto sería reduciendo la longitud de algunas frases y simplificando las interacciones para que se vuelva más real.

Tercera pregunta: Errores en el diseño

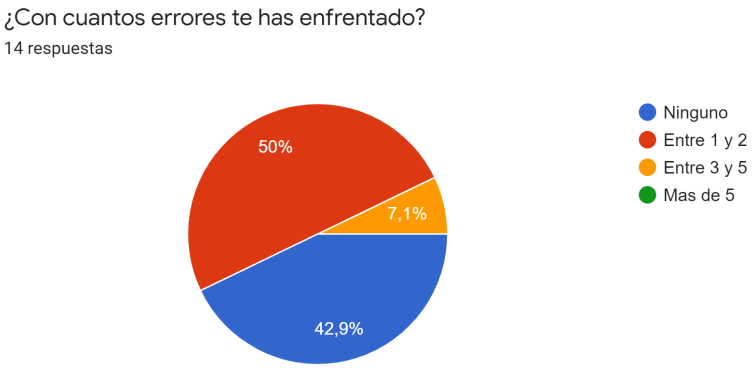


Figura 27: Resultado de la pregunta sobre errores encontrados

Se ha preguntado sobre qué errores en el concepto se han podido encontrar. Nos encontramos que ha habido una persona que se ha encontrado entre 3 y 5 fallos en sus conversaciones. Esto es debido a que fué la primera persona que probó el asistente, cuyos reportes ayudaron a mejorar el flujo. El resto de participantes encontraron 2 errores o menos, teniendo también en cuenta cómo cada persona clasifica que es un error y que no, debido a que el asistente quizás no responda bien a ciertas expresiones.

Cuarta pregunta: Grado de satisfacción general

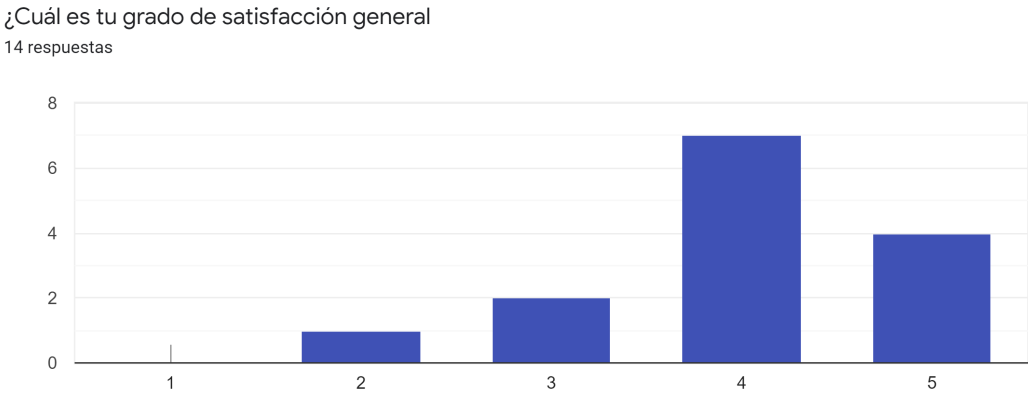


Figura 28: Resultados acerca del grado de satisfacción general

Cómo se aprecia en la figura 28, ha sido positivo para la mayoría de personas, excepto para una que fue la que experimentó más errores.
Según esta gráfica, es susceptible de ser mejorado buscando más puntuaciones de 5, ya que según ramas de la psicología del consumidor, son clientes satisfechos que recomendarian dicho servicio.

Quinta pregunta: Impacto en el mercado

Siguiendo las enseñanzas de emprendimiento y desarrollo ágil, se ha realizado una pregunta para conocer el posible impacto en el mercado y el uso que podrían darle clientes de restaurantes para así conocer si tendría sentido iterar el producto y lanzarlo al mercado.

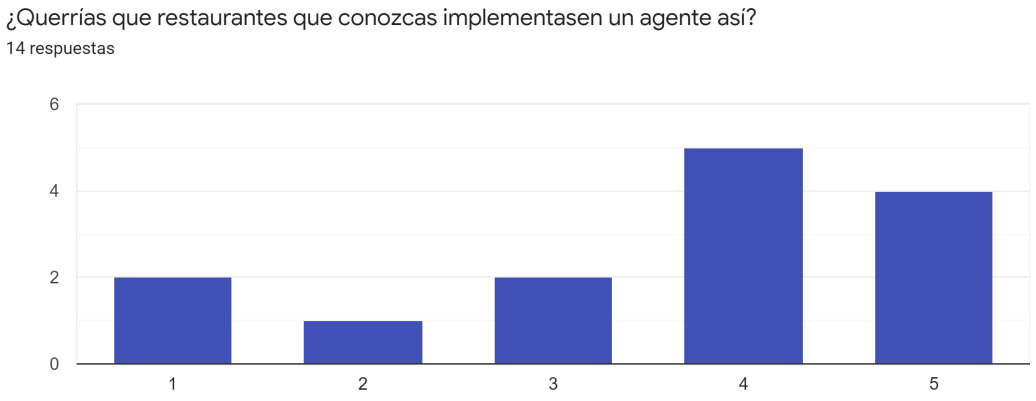


Figura 29: Resultados sobre la opinión acerca de la implementación a escala

En esta encuesta de la figura 29 hay opiniones mucho más dispersas, las negativas a la implementación pueden ser debido a la naturalidad con la que pedimos comida a domicilio a través de la pantalla del móvil o debido al proceso de completado de información actual. Las positivas pueden ser debido a la novedad del método.

La línea de mejora en este respecto, es utilizar la información albergada en la cuenta de google para crear un rellenado automático de información, agilizando el proceso.

Sexta pregunta: Líneas de mejora de funcionalidades

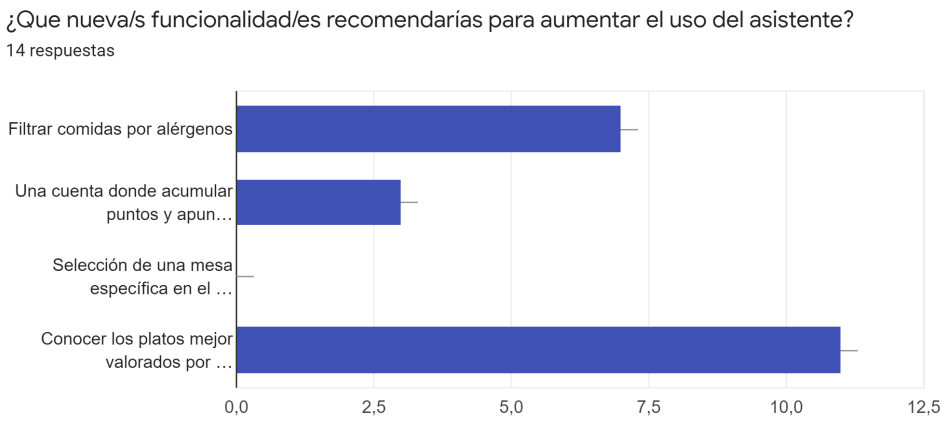


Figura 30: Resultados sobre nuevas funcionalidades

En esta pregunta se mostraban 4 opciones con selección múltiple, para que los participantes den su opinión sobre qué funcionalidad haría incentivar el uso de la aplicación.

Destaca la opción de conocer las valoraciones de los platos por parte de los usuarios, funcionalidad a la que todos los usuarios de aplicaciones de todo tipo estamos muy acostumbrados, como puede ser amazon, tripadvisor, uber, etc…

La segunda con más votación es el filtrado de comida por alergenos, vital para cualquier restaurante de calidad. En la versión actual del asistente, se ha dejado la posibilidad de incluir un parámetro en la petición del menú para enviar la palabra precisa desde DialogFlow y devolver los platos que sean aptos para cada tipo de alergia, como pueden ser celiacos, cacahuetes, soja, trigo, etc…

Sobre la cuenta para acumular puntos, se realizaría mediante la sincronización de la cuenta de google, creando una colección en firebase adicional donde guardar el perfil de dicho usuario y sus preferencias.

La selección de una mesa específica parece ser una funcionalidad no pedida por los usuarios en esta prueba, quizás debido al desconocimiento de la forma del establecimiento. Esta funcionalidad, requeriria trámites adicionales con el restaurante, así como modelar una aplicación en el *canvas interactivo*[40] de *Google Assistant*.

[40] El Interactive Canvas es un framework de Google Assistant que permite crear experiencias visuales dentro del asistente en múltiples dispositivos. https://developers.google.com/assistant/interactivecanvas

Resultado final

Según las encuestas y cambios realizados, este proyecto tiene la posibilidad de escalar tanto en número de intents, como de frases de entrenamiento por parte de Dialogflow, así como la creación de nuevas funcionalidades para los dueños del restaurante, como puede ser una interfaz para actualizar los platos disponibles, o para los usuarios como se ha desarrollado en la sexta pregunta.
Esto ayudaría a lanzar una primera versión en algunos negocios y así poder iterar hasta que alcance la utilización en un mínimo de negocios para que resulte rentable sin tener que cobrar en exceso.

Como resultado de la encuesta, vemos que el grado de satisfacción es lo suficientemente bueno como para confirmar que han cumplido con sus expectativas y que merece la pena iterar para aumentar el porcentaje de personas que querrían utilizar dicha aplicación.

7.- Estudio económico:

En este estudio económico se presentarán los costes de implementación en el estado actual del desarrollo, que es el que se ha realizado durante este trabajo de fin de grado. Todo lo planteado anteriormente ha servido para apoyar la viabilidad técnica, por lo que en este estudio se discutirá acerca de la generación como la del mantenimiento.
Cómo material adicional al estudio, también se desarrollará un plan de expansión para que sea considerado un proyecto con viabilidad económica basada en la economía de escala, permitiendo incorporar perfiles que se encarguen de mejorar la experiencia de uso, puedan crear funcionalidades más complejas y que permita escalar el proyecto a un modelo de Software como Servicio[41].

[41]Software como Servicio (o en inglés Software as a Service o PaaS) es un modelo de distribución de software que permite a usuarios conectarse a aplicaciones basadas en la nube a través de internet. Proporciona funcionalidades concretas al usuarios cercanas a sus necesidades de negocio. En este caso, se proporcionará al establecimiento un ecosistema completo con el que manejar toda la plataforma de gestión con el asistente sin necesidad de tener conocimientos de programación.

Estudio económico en la fase actual de desarrollo

Recursos empleados

Hay que tener en cuenta los recursos utilizados durante en transcurso del desarrollo, como puede ser el software, servicios en la nube y hardware, en este último solamente teniendo en cuenta la amortización de los recursos utilizados durante el periodo de tiempo utilizado.

- Software:
  - Sistema operativo: Windows 10
  - Visual Studio Code
- Hardware:
  - Ordenador portátil Surface Book 2
  - Ratón logitech MX Master 3
- Servicios en línea:
  - Dialogflow
  - Firebase
  - Ngrok
  - Notion
  - Canva
  - Suscripción a plataforma de cursos Platzi y Udemy.
- Material ofimático:
  - Libros de consulta
  - Documentación impresa

Costes directos

Los costes directos son los repercutidos directamente al desarrollo de la aplicación, estos son:

- Costes de personal
- Costes amortizables de programas y equipos
- Costes de materiales empleados y servicios en línea contratados.

Además de estos costes, al final del capítulo se incluirán los costes indirectos de forma estimada.

Costes de personal

Dado que el proyecto ha sido desarrollado por un ingeniero, desde el estudio del problema hasta la implementación y mantenimiento del problema, se utilizará de estimación relativa a un empleo de dicho rango.

El procedimiento habitual pasa por calcular el coste anual de un ingeniero y dividirlo respecto a las horas de trabajo dedicadas al proyecto, incluyendo el salario bruto, incentivos y la cotización a la Seguridad Social.

Las estimaciones utilizadas serán las siguientes:

- Salario bruto e incentivos: 31.000 € al año.
- Seguridad Social (35% del salario bruto): 10.850€ al año.

Así el coste total de un año es de **41.850€**.

Considerando una jornada de trabajo de **8 horas** y un calendario laboral de **243 días**[42] al año, el total de horas anuales de trabajo es de **1944 horas**.

Con esta configuración, el coste por hora de un ingeniero será aproximadamente de **21.60€/hora**. Este será el desglose de horas aproximadas que se han dedicado a cada apartado para realizar el proyecto

- Formación y documentación: 80 horas.
- Estudio del problema: 30 horas.
- Estudio de las herramientas: 20 horas.
- Desarrollo de la aplicación: 120 horas.
- Puesta a punto, implementación y mantenimiento: 20 horas
- Retroalimentación: 25 horas
- Elaboración de la documentación: 100 horas

Dando como resultado un total de **395 horas de dedicación**, que multiplicado con el coste calculado por hora de un ingeniero, obtenemos un coste de personal de **8.532€**.

[42] Se ha utilizado el siguiente caso práctico para establecer una jornada laboral típica. https://www.iberley.es/practicos/caso-practico-computo-anual-jornada-trabajo-no-especifique-convenio-colectivo-3231

Costes de amortización de equipos y programas

Para realizar estos cálculos, se necesita conocer la inversión total de equipos y programas y calcular la amortización lineal correspondiente al tiempo dedicado.
Para el ordenador, periféricos y sofware comprado, calcularemos una vida útil estimada de 4 años, debido a la rapidez del avance tecnológico. Esto significa que el factor de amortización de un año será un cuarto del valor, es decir, un 25%.

- Sistema operativo Windows 10 Pro: 145€, con un coste de 26.25€ al año.
- Ordenador portátil: 2600€, con un coste de 650€ al año.
- Ratón: 100€, con un coste de 25€ al año.

Como resultado, tenemos un coste anualizado de **701.25€** al año, que supondrá un coste por hora de **0.361€ por hora**.
Contabilizando el tiempo utilizado, la amortización final es de **142.5€**.

Costes derivados de otros materiales y servicios.

Estos costes se referirán a los consumibles de ofimática utilizados durante el transcurso del proyecto, como puede ser documentación impresa, así como suscripciones a plataformas de aprendizaje y cursos comprados.

Además se han incluido varios cursos de Udemy y Coursera utilizados para ampliar información.

El coste total se estima que rondaría los **120€**.

Servicios en línea contratados

Durante todo el proyecto, se han utilizado una amplia gama de herramientas y servicios en línea.
Dado que la mayoría se han utilizado o bien en el periodo de prueba gratuita o porque tienen una capa de uso gratuita extensa[43], el coste es de **0€**.

[43] Se ha calculado las solicitudes realizadas respecto al análisis de las herramientas de Dialogflow y Firebase explicadas en el capítulo 2, sin traspasar holgadamente la capa grautita.

Costes directos totales

Calculando los apartados anteriores, concluimos que la suma de gastos de personal, amortización y otros consumibles dará unos costes directos totales de **8794.5€**.

Costes indirectos

Son costes que no se han podido incluir en ninguno de los gastos directos planteados.

- Consumo electricidad: 30€
- Consumo de internet: 90€
- Costes de suscripción a plataforma de cursos: 300€

El total de costes indirectos supone un coste de **420€**

Costes totales

Calculando previamente tanto costes directos como indirectos a la realización de este trabajo de fin de grado, el coste total es de:

9214.5€

Plan de expansión mediante un modelo SaaS

Debido a que la ingeniería es una ciencia práctica y utilitarista, a continuación se mostrarán los costes que supondría desarrollar una aplicación real que se pueda implementar en restaurantes y sacar un beneficio económico.

La intención es poder desarrollar la aplicación para que pueda ser más recursiva y utilizada por varios restaurantes y así reducir costes fijos, como puede ser los costes de desarrollo, basándonos en una economía de escala.

Debido a que este agente conversacional se ha realizado para contrastar distintas hipótesis y mostrar la utilidad que una interfaz puede proporcionar, supondremos un escenario donde se precise la contratación de más personas especializadas para escalar la aplicación y cerrar contratos con establecimientos.

El tipo de perfil necesario para escalar la aplicación serían principalmente de ámbito tecnológico, centrándose en el desarrollo y la comercialización para llegar al máximo número de establecimientos en un tiempo relativamente corto, para así no recurrir a préstamos o financiación externa.

Perfiles de ámbito del desarrollo

Desarrollador/a Front-end:

Sería la persona encargada de mejorar la interfaz visual de la aplicación según los últimos estándares de Google Assistant, incorporando nuevos métodos como el *Interactive Canvas*, permitiendo desarrollar experiencias más gráficas en el asistente inteligente. También desarrollará la interfaz con la que las distintas encargadas de los restaurantes puedan incluir nueva información sobre su restaurante, como nuevos platos, días de vacaciones, ofertas, visualización de las reservas, etc..
Por último, pero no menos importante, también creará la parte visual e interactiva con tecnologías web de la página web de la aplicación, que se utilizará para la promoción de cara a establecimientos.

Según la plataforma de valoración de empresas *Indeed*, el salario de un perfil de un desarrollador front-end es de 26.000€/año[44].

Desarrollador back-end:

Será la persona encargada de incluir lógica de negocio más compleja en el servidor para atender a las necesidades de los establecimientos, así como optimizar las consultas para reducir el coste del servidor utilizando buenos estándares y tecnologías de servidor y base de datos que permitan un menor uso de recursos.
Según la plataforma indeed, el salario promedio de una desarrolladora back-end es de 32.000€/año.

Perfiles del ámbito de la comercialización y venta

Especialista en Marketing Digital

Para poder obtener una facturación adecuada en un periodo de tiempo corto, es necesario fomentar el uso de la aplicación entre los posibles clientes de un restaurante, ayudando a atraer a nuevas personas y mejorando la facturación del establecimiento. Además, se necesita hacer que la aplicación sea conocida entre los encargados de los restaurantes para que estén abiertos a contratar la aplicación.
Este perfil se encargará de crear contenidos con intención de búsqueda en la web, así como crear campañas de marketing dirigidas a dueños de negocios, por medio de social ads (Facebook y Youtube), anuncios de búsquedas (SEM de Google), SEO (Search Engine Optimization), con todos estos esfuerzos enfocados en dueños o gestores de establecimientos que buscan nuevas formas de hacer crecer su negocio.
Además, podrá ofrecer un servicio adicional a los establecimientos que así lo pidan para promocionar su negocio a través del marketing conversacional, aprovechando las funcionalidades actuales y futuras del asistente virtual.

Según el portal de empleo Glassdoor, este tipo de perfiles tienen un salario medio de 36.000€/año[45].

[44] Fuente: https://es.indeed.com/career/programador-front-end/salaries
[45] Fuente: https://www.glassdoor.es/Sueldos/digital-marketing-specialist-sueldo-SRCH_KO0,28.htm

Comercial de cierre de ventas:

Debido a que se está realizando un enfoque B2B (Business to Business) uno de los pasos más importantes en todo el flujo comercial es el cierre de la venta y su posterior mantenimiento, es por eso que se hace necesario un perfil comercial que pueda llegar a un acuerdo con el establecimiento después de la captación.
Deberá tener suficiente entendimiento en tecnología como para pedir los requerimientos necesarios por parte del establecimiento como por parte de la aplicación.
Este tipo de perfil suele contar con un salario base moderado y un salario por venta para motivar al perfil en la búsqueda de nuevos clientes.

Debido a que la captación de nuevos clientes se realizará online, se supondrá un salario de 24.000€/año.

Chatbot developer

Es el perfil que se encargará de crear la experiencia del asistente virtual satisfaciendo las distintas necesidades que puedan presentarse. Es un perfil que combina distintas especialidades de desarrollo y comercialización, donde necesita tener un conocimiento moderado de tecnologías web como ReactJS, tecnologías backend como firebase, conocimientos avanzados de marketing conversacional, así como el dominio de distintas herramientas de lenguaje natural.
Como es un perfil relativamente nuevo, no hay suficiente información como para conocer el salario promedio. Se ha hecho un ajuste del salario según el conocimiento de las distintas tecnologías y el resultado ha sido aproximadamente 31.000€/año.

Gastos fijos:

Los gastos fijos son todos los gastos que no dependen del volumen de ventas o servicios proporcionados, como en este caso sería el desarrollo y mantenimiento de la aplicación.
Para simplificar este ejercicio y debido a que los salarios corresponden con una gran parte de los gastos de una empresa digital, solamente se contarán los salarios y los impuestos correspondientes como gastos fijos.

En total, el gasto en salarios sería de:

- Salario bruto e incentivos: 148.000€/año
- Seguridad social (tomada como un 35% del salario bruto): 51800€/año

Gasto total aproximado: 200.000€/año.

Gastos variables:

Los gastos variables son los gastos directamente relacionados con el volumen de ventas de la empresa. En este caso, el gasto variable se debe principalmente al tiempo de uso de los servidores y otras herramientas o servicios relacionados.
Debido a que el caso de uso no requiere del uso intensivo de tiempo de cómputo, es posible optimizar dichas operaciones para que el uso por parte de cada establecimiento sea casi despreciable.

Sobreestimando un coste de tiempo de cómputo que puede utilizar cada establecimiento utilizando alguno de los servicios para el procesamiento en la nube, el coste anual no llegaría a superar los 50€/año por cada establecimiento dentro de la aplicación.

Gasto variable estimado por establecimiento: 50€/año.

Ingresos estimados

Para poder estimar el precio mínimo que se debería de cobrar a los establecimientos para proporcionar el servicio, calcularemos un punto clave en la estimación de ingresos y gastos.

Este es el punto de equilibrio, donde se igualan los ingresos a los costes fijos y variables asumidos hasta el momento.

El punto equilibrio separa el momento en el que una empresa comienza a tener ingresos, estableciendo un número mínimo de unidades vendidas.

Estas unidades pueden ser tanto una unidad de producto, como un servicio o una suscripción.

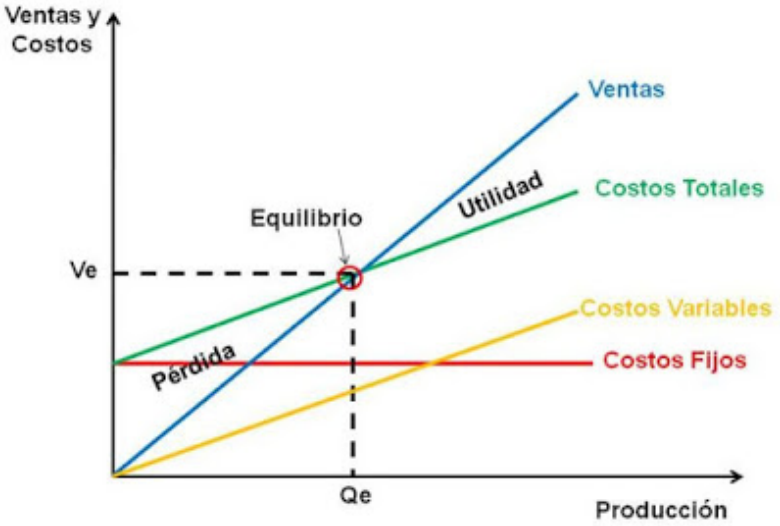


Figura 31: Gráfica del punto de equilibrio

Siendo las fórmulas utilizadas para calcular los costes, los ingresos y el punto de equilibrio según el número de establecimientos que utilicen el asistente, suponiendo únicamente un coste fijo anual.

- Costes totales: $C_F + C_V * n$
- Ingresos totales: $I_V * n$
- Punto de equilibrio: $C_F + C_V * n = I_V * n$

Siendo:

- C_V:Coste variables por establecimiento. En este caso sería de 50€/año por establecimiento
- C_F: Coste fijo del desarrollo. En nuestro caso es de 200.000€/año.

- I_VIngresos variables por establecimiento. Este sería el precio que necesitamos establecer para conocer el mínimo de restaurantes que deben de utilizar la aplicación.

Utilizando los costes variables y fijos calculados en el apartado anterior, el punto de equilibrio resulta en una ecuación con dos incógnitas.

- $C_F + C_V * n = I_V * n$
- $200.000 + 50 * n = I_V * n$

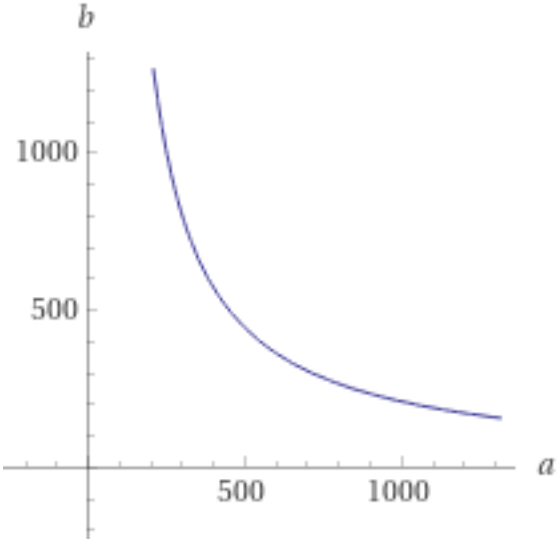


Figura 32: Gráfica sobre la relación entre Iv(a) y n(b)

Debido a esto, la variable más controlable inicialmente es el precio que se va a cobrar a los establecimientos por ofrecerles el servicio.
Por eso, se necesita estimar un precio de venta adecuado para que los establecimientos puedan aceptarlo y les permita aumentar sus ingresos.

Utilizando distintas fuentes, un restaurante de comida rápida promedio cuenta con una facturación de 600.000€/año[46].
De esa facturación, los gastos en proporción van dirigidos a:

- 29% a Materia prima: 174.000€/año
- 26% a Personal: 156.000€/año
- 25% a Gastos fijos generales, como alquiler o licencias: 150.000€/año
- Margen 20%: 120.000€/año

[46] Fuente: Hostelería de Madrid

Existen algunos gastos que no pueden ser reducibles en un restaurante de este tipo como pueden ser la materia prima, los gastos generales o una buena parte del personal.

Teniendo en cuenta el margen de un restaurante de comida rápida promedio de 120.000€/año y la posible reducción de dedicación en la plantilla a la contestación de llamadas debido a la gestión automatizada que realiza el asistente virtual, se ha supuesto un coste de implementación y mantenimiento anual de 1200€/año por establecimiento.
Esto supone un desembolso de únicamente el 1% del margen obtenido, teniendo como retorno una mayor captación y retención de clientes, y un ahorro proporcional en costes de personal.

Ingresos estimados por establecimiento:

1200€/año

Estableciendo que $I_V = 1200$€/año, el número de establecimientos que se necesita para comenzar a ser rentables sería de 168 restaurantes.

A partir del restaurante número 169 dentro de la aplicación, los beneficios por cada establecimiento serían de 1150€ al año por cada establecimiento adicional que se incorpore a la aplicación.

Número mínimo de establecimientos anuales en la aplicación para que alcanzar el punto de equilibrio entre ingresos y gastos:

168 establecimientos

8.- Conclusiones

Para poder analizar la calidad final del proyecto, los objetivos definidos al comienzo del proyecto se pueden dividir en varias partes:
La fase de exploración, la fase de desarrollo y el análisis económico.

El objetivo del estudio de las soluciones en el mercado actual ha permitido seleccionar las herramientas y servicios con mejor calidad-precio para las distintas funciones que pueden necesitarse en un ambiente real. Se ha concluido utilizar la **plataforma de *Google*** debido a su accesibilidad a la hora de desarrollar un agente con *DialogFlow* y su integración directa con *Google Assistant*. Además brinda la posibilidad de utilizar el asistente en combinación con otras herramientas de marketing y promoción de Google.

El objetivo de implementación y entrenamiento ha resultado en el uso Firebase para ejecutar la lógica de negocio del ejemplo del asistente para un restaurante. También se ha utilizado *Firestore*, que es una base de datos no relacional, permitiéndonos almacenar en una única colección las reservas de mesa y los pedidos a domicilio. *Firestorage* se ha utilizado como solución para almacenar las imágenes de los platos.
Se han desarrollado diferentes flujos de usuarios mediante funciones, que recogen los distintos intentos existentes en DialogFlow.

Uno de los objetivos que se tenía con la encuesta es conocer la posible utilización de un asistente virtual para tareas cotidianas como puede ser la reserva en un restaurante. Dicha recepción fue aceptada pero existieron algunos rechazos, a través de las entrevistas individuales como conclusión podemos determinar que esta solución se ha visto mejor recibida a personas de mayor edad, debido a que los más jóvenes están acostumbrados a otras plataformas de pedidos a domicilio existentes como *JustEat.* Se necesitaría una muestra más grande para poder verificar dicha demanda.

El objetivo acerca del estudio de la viabilidad económica de la implementación de un asistente virtual en un negocio común, se ha resuelto mediante economías de escala, pensando como solución la contratación de perfiles especialistas en diversos ámbitos para poder abstraer la aplicación y que la estructura pueda ser utilizada por varios restaurantes a la vez.

Como resultado a este análisis de viabilidad, en el ejemplo expuesto sobre su implementación en un restaurante de comida rápida, se ha llegado a la conclusión que para un equipo de 5 personas con puestos clave, sería necesario un ticket medio de **1200€/mes** por establecimiento y un mínimo de **168 establecimientos** dentro de nuestra aplicación.

Las líneas futuras de desarrollo, tal y como se ha anticipado en la definición de las tareas del estudio económico, serían la de abstraer la aplicación para que resulte factible crear una marca blanca para cada restaurante utilizando una estructura propia común. Paralelamente, también se desarrollarían nuevas funcionalidades para el usuario, como la posibilidad de crear una cuenta, gestionar los pagos mediante el asistente, posibilidad de filtrar la carta según más parámetros. Para el establecimiento se desarrollaría una interfaz con la que podrán gestionar todos los pedidos, reservas e incidencias, incluso integrar una solución que la conecte con sus sistema de gestión de clientes.
Por último, otra consecuente línea de negocio sería la de promoción y captación de nuevos clientes a través del marketing digital. En estos momentos, el marketing digital se ha tecnificado lo suficiente como para considerarse del ámbito del desarrollo, a través de campos de que mezclan marketing y tecnología como el *Martech*, que permite analizar datos obtenido del histórico para obtener patrones comunes que permitan reducir costes, predecir pedidos y atraer a nuevos clientes mediante campañas de pago.
En esta última línea de negocio, estarían implicados varios los perfiles del personal que se han presentado, centrándose en ciertas tareas como las siguientes:

- **Desarrollador/a front-end:** Creando landing page que permitan visualizar mejor la propuesta de valor y convertir a más clientes.
- **Desarrollador/a backend**: Creando la infraestructura de datos de clientes para poder incluir herramientas de inteligencia artificial y sacar conclusiones.
- **Especialista en marketing digital**: Estableciendo nuevos embudos de venta y efectuando las acciones requeridas para que los usuarios sigan utilizando la solución. Este perfil junto con el desarrollador de chatbots,serán los encargados de obtener la información requerida para mejorar la conversión a venta y fidelización de clientes para poder utilizarlas en modelos predictivos.

9.- Bibliografía

Modelos de distribución de software: SaaS, PaaS y IaaS:
https://azure.microsoft.com/es-es/overview/what-is-paas/

Inicios de los comandos de voz en aplicaciones prácticas:
https://www.manualslib.com/manual/18584/Bmw-745i.html?page=28

Teoría científica sobre el lenguaje natural de Noam Chomsky:
https://arxiv.org/pdf/1703.04417.pdf

Definición lenguaje natural por la UAM:
https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/

Diseño de la conversación de un asistente conversacional y guías de estilo:
https://designguidelines.withgoogle.com/conversation/conversation-design/welcome.html#welcome-how-to-use-this-site

Documentación en línea de Amazon Web Services acerca de todas sus herramientas en la nube:
https://docs.aws.amazon.com/index.html?nc2=h_ql_doc

Documentación en línea de Google Cloud Platform acerca de todas sus herramientas en la nube:
https://cloud.google.com/docs

Documentación en línea de Firebase:
https://firebase.google.com/docs

Documentación acerca de tecnologías web:
https://developer.mozilla.org/

Documentación en línea de DialogFlow:
https://cloud.google.com/dialogflow/docs

Documentación en línea de Actions On Google:
https://developers.google.com/assistant

Documentación en línea de Google Assistant:
https://developers.google.com/assistant

Corrección de errores:
https://www.stackoverflow.com

Develop DialogFlow chatbot locally:
https://medium.com/@antonyharfield/dialogflow-web-hooks-how-to-develop-locally-and-deploy-to-cloud-functions-48839919e998

Documentación en línea de Azure Cloud:
https://azure.microsoft.com/es-es/services/cognitive-services/speech-to-text/#features

Características de Mycroft, alternative del procesamiento de lenguaje natural:
https://mycroft.ai/

Documentación y características de wit.ai, procesamiento del lenguaje natural de Facebook:
https://wit.ai/docs

Características de IBM Watson:
https://www.ibm.com/cloud/watson-assistant/

Estado de los agentes conversacionales. Informe de Accenture
https://www.accenture.com/_acnmedia/pdf-77/accenture-research-conversational-ai-platforms.pdf

Chatbot de Domino's Pizza
https://www.chatbotguide.org/dominospizza-bot