Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería electrónica industrial y automática

Sistema conversacional de ayuda a personas mayores basado en Dialogflow.

Autor:

Gobernado Rodríguez, Víctor

Tutor:

Gómez García-Bermejo, Jaime

Universidad de Valladolid

Valladolid, mayo 2020.

Trabajo de Fin de Grado

TÍTULO: Sistema conversacional de ayuda a personas mayores basado en Dialogflow.

ALUMNO: Gobernado Rodríguez, Víctor

FECHA: Fecha de la defensa en la universidad

CENTRO: Universidad de Valladolid

TUTOR: Gómez García-Bermejo, Jaime

Resumen

Hoy en día unos de los principales problemas a los que se enfrenta la sociedad es la soledad, y más específicamente la soledad en las personas mayores. Este sentimiento se acentúa por no saber darle uso a la tecnología actual y por la falta de medios tanto económicos como humanos para atenderles. El presente proyecto se ha desarrollado para combatir dicho problema y se realizará por medio de un agente conversacional. Dicho agente será de uso intuitivo y se podrá acceder a él desde cualquier dispositivo con conexión a Internet.

Para lograr este objetivo se utilizará la herramienta Dialogflow de Google. Durante el proyecto se analizará qué utilidades nos aporta, así como las instrucciones precisas para replicar el agente y para poder desarrollar en el futuro otros agentes que aporten nuevas soluciones a más problemas que enfrente la sociedad.

Palabras clave

- Agente conversacional.
- Dialogflow.
- *Intent.*
- Firebase.
- *Fulfillment.*

Abstract

Nowadays one of the main problems we as society face is loneliness, and more specifically elderly people's lonelyness. That feeling is increased by the technology breach and the lack of monetary and human resources. This project has been developed to give an answer this problem, and it will be accomplished by the use of a conversational agent. That agent will be intuitive and accesible from any device with intenet access.

To fulfill that goal the Google's software tool Dialogflow will be used and it's utilities will be explained during the project. Also clear instructions will be provided to develop another agent in the future, wich might come up with a new solution to other problems our society might face.

Key words

- Conversational agent.
- Dialogflow.
- *Intent.*
- Firebase.
- *Fulfillment.*

Contenido

Tabla de Ilustraciones.

1.- Introducción y objetivos.

1.1.- Marco del proyecto.

En el contexto actual de nuestra sociedad hay una gran cantidad de retos que son más fáciles de resolver gracias al uso de la tecnología. Muchos puestos de trabajo ya han sido tomados por automatismos e incluso se han convertido en puestos cooperativos donde las máquinas y las personas se complementan y ayudan, sobre todo aportando fuerza física a cambio de supervisión.

Todos los días dependemos de la tecnología de una manera que no podría haberse imaginado nadie hace unas décadas con todo el conocimiento a nuestro alcance y redes móviles de altas velocidades. Además, el auge de los sistemas de inteligencia artificial está logrando todo tipo de tareas muy variadas desde automóviles autónomos hasta asistentes personales con los que conversamos para fijar una entrada en nuestro calendario o hacer la lista de la compra día a día.

El grupo demográfico que se ve menos beneficiado por esta revolución tecnológica es la gente mayor. Ya sea por dificultad a la hora de adaptarse a la tecnología, por una falta de educación en dichos temas o por no tener acceso a productos caros o a internet. Por ello se deberían centrar los esfuerzos en utilizar la tecnología en solucionar los problemas de este grupo demográfico tan extenso en casos como el de nuestro país.

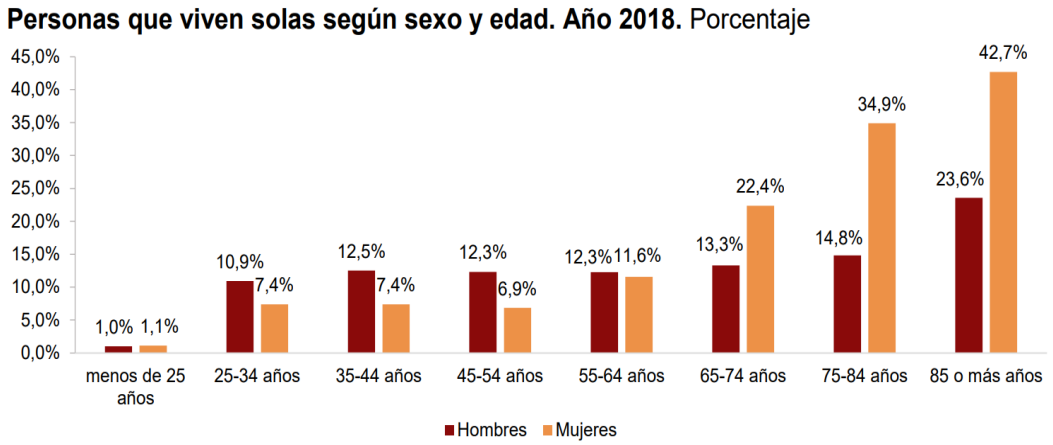

Ilustración 1 Porcentaje de personas que viven solas en España. (Instituto nacional de estadística, 2018)

Como se puede ver en la Ilustración 1 publicada por el instituto nacional de estadística una gran cantidad de personas mayores viven sin compañía. Este dato es alarmante y dado que no tenemos medios suficientes para poder pagar a empleados que acompañen a este tipo de personas se debería pensar en la tecnología como el mejor medio del que disponemos para abordar el problema. Desde otra perspectiva, utilizar la tecnología podría acercarles más al mundo digital es un reto pero es posible conseguir realizar un acercamiento al mundo tecnológico en el que se mueve la mayor parte de la población ya sea ayudándoles a pedir cita online para el médico como enseñándoles entretenimiento que puedan querer consumir, informándoles sobre temas en los que estén interesados o incluso educándoles sobre tecnología o recomendaciones de salud.

Por todo esto se desarrollará un sistema de ayuda a mayores en este trabajo de fin de grado. Además, deberá ser conversacional porque así se combate el problema de la accesibilidad y del miedo que ciertas personas de edad avanzada pueden tener a la tecnología. Se seguirán los siguientes objetivos para asegurar el cumplimiento de las metas impuestas.

1.2.- Objetivos.

En este marco, el objetivo general del presente Trabajo Fin de Grado consiste en desarrollar un sistema conversacional pensado fundamentalmente para ayudar y acompañar a personas mayores que habiten en residencias y basado en la herramienta en línea de Dialogflow que permite su implementación en una enorme variedad de dispositivos. Este sistema debería tener las siguientes características:

- Accesibilidad: Si se desea que sean las personas mayores quienes utilicen el sistema es primordial que sea fácilmente accesible. Asegurando que se pueda utilizar en la mayor variedad de dispositivos posibles y que tenga una amplia comprensión del lenguaje permitiendo que se pueda acceder a las mismas funciones con frases diferentes.
- Diversidad: Será necesario que la cantidad de funciones que realice el sistema sea amplia, de esta manera se normalizaría su uso como algo más de su rutina. Si una persona mayor solo utiliza el sistema para realizar una función para ella será más sencillo aprender a realizar esa función por separado antes que aprender a interactuar con el sistema entero.
- Personalización: Las conversaciones diarias que tenemos entre nosotros tienen un componente primordial que es la información que tenemos sobre el otro interlocutor, al recordar cosas como su nombre o sus gustos hace la conversación más fluida. Este tipo de conversación sobre un interlocutor reactivo a los gustos e información del otro ser el objetivo del sistema.
- Repetitividad: Las respuestas del sistema deben ser variadas, con múltiples contestaciones ante las mismas preguntas, así nos aseguraríamos de que el usuario no se canse de hablar con él. También se lograría con mensajes que tengan en cuenta las veces que el usuario ha visitado el agente.

Para alcanzar los objetivos se plantean las siguientes fases:

- Descripción de las herramientas: Para alcanzar los objetivos primero deberíamos conocer los límites de las herramientas que se van a utilizar.
- Descripción de las funciones: Deberíamos definir qué funciones queremos que realice el sistema.
- Mejora continua: Para dar una buena experiencia se necesitará algún tipo de realimentación donde se planteen las posibles mejoras y se ejecuten.

1.3.- Descripción de la memoria.

El primer capítulo se presta a dar un contexto sobre la importancia de este trabajo exponiendo la situación actual y las soluciones que se quieren dar para alcanzar unos objetivos.

Después en el segundo capítulo se verá lo conocido normalmente como el estado del arte y expondrá otro tipo de soluciones aportadas por otros desarrolladores, así como las similitudes y diferencias con la solución que se propondrá más adelante.

El tercer capítulo es una explicación de todas las herramientas y conceptos que se van a desarrollar, así como una justificación de por qué se han escogido estas y no otras, también se analizará el alcance de las herramientas y plataformas.

Después en el capítulo número cuatro se detallarán las funciones a implementar para así alcanzar los objetivos previamente expuestos. Gracias a esto se dotará de un esquema mental del funcionamiento del agente sin entrar en detalles técnicos o terminología avanzada.

El quinto capítulo se centra en el núcleo del proyecto explicando paso a paso cómo se ha realizado a nivel programación desde la creación del agente hasta su preparación para iniciar las pruebas (*debugging*) y mejoras con la realimentación de los usuarios.

En el sexto capítulo se expondrán las técnicas que se han utilizado para mejorar el agente que habíamos programado en el punto anterior. Así como posibles recomendaciones a tener en cuenta. También se expondrán las herramientas que se disponen para hacer estas mejoras.

En el siguiente, el capítulo número siete, se analizarán tanto las sugerencias como las opiniones recibidas de los usuarios para exponer de manera sencilla cómo de útil puede llegar a ser el agente desarrollado en este trabajo de fin de grado. Las opiniones se recogerán en una encuesta que se cumplimentará al haber tenido varias conversaciones con el agente. Por ello también se analizarán las preguntas de la encuesta justificando su relevancia en la retroalimentación. Además, se estudiarán las posibles líneas de mejora futuras.

En el octavo y penúltimo capítulo. Se dará por concluido el análisis de la viabilidad del proyecto de una manera técnica para estudiar la viabilidad económica del mismo. Todo ello sin pasar por alto los costes de los servicios en línea de los que se hará uso ni los materiales o salario necesario para desarrollarlo.

Tras analizar todos los aspectos posibles del proyecto se retomarán los objetivos iniciales para comentar su grado de cumplimiento a demás se explicará si la solución elegida es la más óptima para la tarea que se desea realizar y sus razones.

2.- Robots, agentes conversacionales e inteligencias artificiales.

Como se comentó anteriormente en la introducción, cada vez más tareas están siendo tomadas por la tecnología y el campo de la robótica es posiblemente en el que sea más notable. Por ello existe en el mercado una gran cantidad de robots, agentes conversacionales e inteligencias artificiales que pueden realizar una amplia variedad de tareas. Desde actuar como enfermero ayudando a levantarse a personas con poca movilidad hasta ayudar a los futuros alumnos a realizar su primera matrícula.

Por todo ello, a continuación se expondrán varios ejemplos de estas tecnologías y se discutirán las diferencias con el agente conversacional que se desea elaborar durante el trabajo. Con objetivo de justificar la realización del mismo y resaltar su importancia.

2.1.- Robots.

Los siguientes robots nos dan un concepto de qué formas diferentes se integra la tecnología para ayudar a las personas ya sea dialogando o proporcionando servicios. Este apartado se centra en robots con representación física y no solo con lenguaje hablado.

2.1.1.- Robear.

El robot que vemos en la Ilustración 2 es el conocido como “Robear” y es un robot desarrollado por la empresa Riken (Noticias de la empresa Riken, 2015). Es un robot enfermero que ayuda a realizar trabajos extenuantes para las personas como puede ser levantar a una persona mayor de la cama y llevarla hasta una silla de ruedas.

Utiliza una gran variedad de tecnología para hacerlo seguro como indican en su página web: “Específicamente incluye unas unidades de actuadores con un coeficiente de engranajes muy bajo, permitiendo a las articulaciones moverse de forma muy rápida y precisa, y permitiendo manejabilidad, queriendo decir la fuerza de los actuadores se puede realimentar de forma muy sencilla en el sistema, permitiendo un movimiento más suave. También incluye sensores de par, sensores inteligentes de goma táctiles lo que le permite realizar movimientos suaves para así realizar tareas tan delicadas como levantar pacientes” (Noticias de la empresa Riken, 2015).

Ilustración 2 Robear Imagen obtenida de (Noticias de la empresa Riken, 2015).

Aunque este robot ayuda muchísimo a las personas mayores no es lo que se busca en este trabajo, el robot Robear debe ser programado para su uso continuamente y aunque resta trabajo físico a empleados de hospitales o residencias no sustituye la función de acompañante ni es monetariamente

accesible para todo el mundo. En el 2015 el proyecto pasó de manos de la empresa Riken a la universidad de Meijo en Japón donde el Dr.Mukai trabaja en su desarrollo.

2.1.2.- Buddy.

El robot que se muestra en la Ilustración 3 es el robot francés Buddy (Página web del Grupo ADD). Está diseñado para ser un robot accesible por su precio y que proporcione compañía en un hogar.Tiene tres ruedas para desplazarse con libertad y un conjunto de sensores para poder reaccionar a cambios en su ambiente. En cuanto a sus funciones, principalmente puede entretener a los más pequeños, proteger la casa y realizar videollamadas.

Ilustración 3 Robot "Buddy" obtenida de (Página web del Grupo ADD).

Es un asistente personal como podría ser Siri o Cortana pero con presencia física y un conjunto de sensores y actuadores que lo hacen más interactivo y con libertad de movimiento por la vivienda. Es de los robots más económicos del mercado para poder llegar a una mayor cantidad de público.

2.1.3.- AV1.

El robot AV1 (Web de la empresa No Isolation) proporciona un tipo de servicio diferente, es un robot que se sitúa en las aulas, pero está pensado para niños que estén en sus casas.

Ilustración 4 Robot AV1 (Web de la empresa No Isolation).

Funciona como un avatar para los niños que por enfermedades o por falta de accesibilidad no puedan acudir presencialmente. Ellos lo controlan, mirando por sus cámaras y reacciona por medio de la cara del robot mostrando su intriga su felicidad o diferentes emociones. Por lo tanto, al igual que el proyecto a desarrollar trata de evitar el aislamiento social de un grupo demográfico con riesgo de sufrirlo. Se utiliza desde una aplicación del móvil y permite tener videollamadas simulando así la presencia del niño en la clase.

Aunque no es el mismo problema que se nos plantea en el trabajo podemos encontrar muchas similitudes y ver que la solución al problema no es única. Está diseñado bajo la premisa de solucionar un componente social: la soledad. Es el mismo objetivo del trabajo que estamos desarrollando aunque no sea ni el mismo público objetivo ni el mismo enfoque.

2.1.4.- Otros.

Al igual que los robots comentados arriba existe una multitud de robots personales que tienen capacidades de convivir con las personas y completar acciones que afecten directamente a la vida de una persona (Rafael Aracil, 2008). A parte de los robots personales todos conocemos que el mayor número de robots actualmente están usados en la industria, desde los brazos robot, que han asumido las tareas más repetitivas de nuestras fábricas, hasta

los robots de trabajo cooperativo donde un trabajador realiza ciertas tareas conjuntamente con el robot supervisándole y complementándole.

Por ejemplo ASIBOT, un robot incorporado en una mesa con un brazo en el centro que es capaz de ayudar a personas discapacitadas o ancianos a tareas como, por ejemplo, comer o lavarse la cara (A. Jardon, 2010). Aunque no Todos estos robots están enfocados al cuidado de personas mayores, por ejemplo Codey Rocky, es un robot económico pensado para enseñar a los niños a programar en lenguajes como Scratch o Python hasta incluso enseña técnicas más avanzadas como el reconocimiento facial (Página web de la empresa Makeblock). O el conocido Pepper que es usado de recepcionista en eventos o incluso como maestro de ceremonias y presentador.

2.2.- Agentes conversacionales.

Últimamente hemos observado como diferentes chats conversacionales salen al mercado y se convierten en una punta de lanza para las compañías tecnológicas, tanto de telefonía como de ordenadores. El TFG a desarrollar consistirá en uno de estos agentes conversacionales (*chatbots*).

2.2.1.- Lola.

Uno de los mayores sectores donde se puede explotar el uso de los chats conversacionales es desde un punto de vista de atención al cliente. En contraposición a la necesidad de consultar una lista de preguntas comunes que los desarrolladores han pensado que podría ser útil simplemente el usuario envía un mensaje de texto al asistente conversacional.

Ilustración 5 *Chatbot* Lola (1 million bot).

El asistente es capaz de detectar la pregunta del usuario y resolver su duda. Así es posible responder una mayor cantidad de dudas de los usuarios automatizándolas.

Un ejemplo de ello es el que vemos en la Ilustración 5, Lola es un chat de conversación implementado en la Universidad de Murcia donde ayuda a los futuros estudiantes con las dudas sobre su matrícula, sobre las notas de corte o proporcionar información sobre plazos y documentos necesarios.

2.2.2.- Erasmus.

En el año 2018 se desarrolló en la Universidad de Mumbai en India un agente conversacional implementado dentro de la red social Facebook enfocado en resolver dudas a padres, profesores y alumnos sobre la universidad (Jash Thakkar, 2018). Al igual que el agente Lola del que se habló en el apartado anterior Erasmus reconoce una gran cantidad de frases y no solamente las programadas. El programa se desarrolló en Api.AI el predecesor de Dialogflow aunque tras el desarrollo se decidió exportar a una página de Facebook donde puedes dialogar con el agente como si fuera personal de la universidad como se observa en la Ilustración 6.

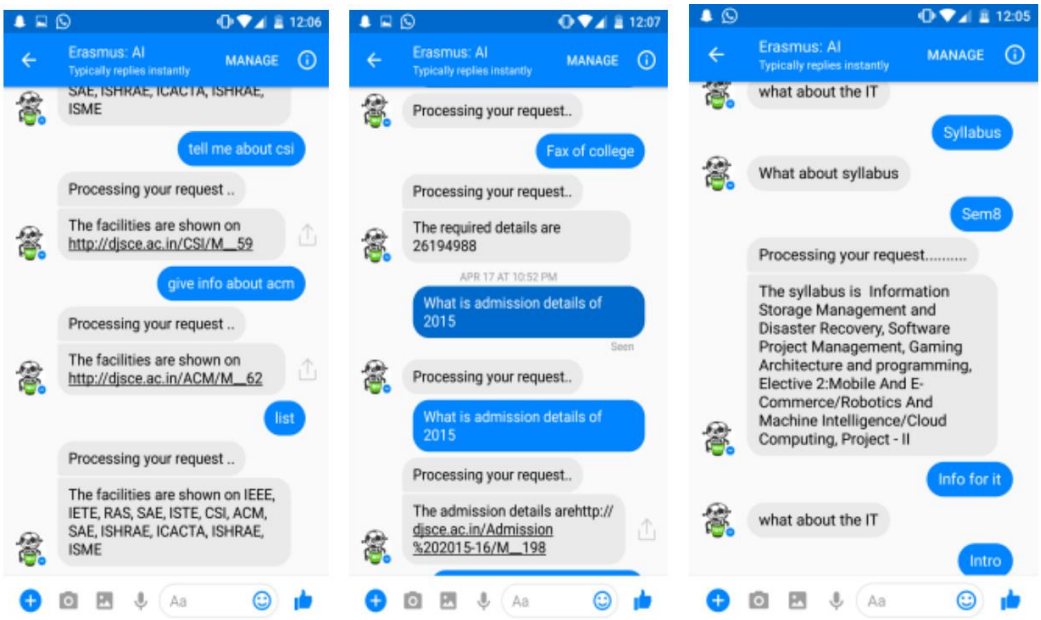

Ilustración 6 Demostración de una conversación con Erasmus imagen obtenida de (Jash Thakkar, 2018)

Al haberse programado en la misma plataforma que se explicará en el apartado 3.- La Herramienta De Desarrollo Dialogflow. Existe un gran número de similitudes como por ejemplo el uso de *intents*. De todos los agentes expuestos es el más similar aunque se utiliza para un enfoque administrativo más que con un fin social

2.2.3.- Otros.

Ya existen un cierto número de compañías que lanzan sus robots conversacionales desde Google con su Asistente hasta Amazon con Alexa y muchas otras marcas tecnológicas. (Hoy, 2018). Los asistentes suelen ser utilizados para dar un servicio al usuario con el fin de agilizar las interacciones entre hombre y máquina, siendo más rápido y sencillo decirle con palabras que añada una entrada en la agenda o que encienda la televisión. Además, suelen venir incorporados en teléfonos, aplicaciones y ordenadores de forma predefinida asegurándose así un uso más amplio.

Los asistentes toman un rol en la conversación pasivo, donde el usuario le da una orden y el asistente la cumple, sin intentar seguir con la conversación o interesarse por el usuario. Este aspecto es lo que marca la diferencia entre el resto de agentes y el robot conversacional que se buscará realizar a lo largo del trabajo.

2.3.- Inteligencia Artificial.

El siguiente paso lógico a los chats anteriores es uno en el que en vez de definir todos los caminos posibles se genera una inteligencia artificial que pueda aprender conversaciones en un ámbito mucho más amplio que lo que se puede conseguir definiendo todos los temas individualmente.

2.3.1.- Meena.

Esta IA (inteligencia artificial) ha sido confeccionada por Google y lanzada el 28 de Enero de 2020.No es la primera vez que se intenta generar una inteligencia artificial que pueda mantener conversaciones pero tras unos minutos de charla era posible adivinar en un 50% de los casos que la conversación no la estabas teniendo con un humano (Google AI Blog, 2020).

Meena ha conseguido hacer una puntuación en el parámetro SSA (un parámetro que mide la consistencia, la coherencia y la exactitud de un chat conversacional) de 79% mientras que el de los humanos es del 86%, ha supuesto una mejora de 23 puntos a su predecesor Cleverbot como se puede apreciar fácilmente en la Ilustración 7. Es una noticia asombrosa pero aún queda mucho trabajo que hacer como dotarle de personalidad y hacer que siga aprendiendo sin desviarse de lo que sus desarrolladores llaman el buen camino. Está previsto que en los meses sucesivos se difunda información sobre los avances obtenidos con esta inteligencia artificial.

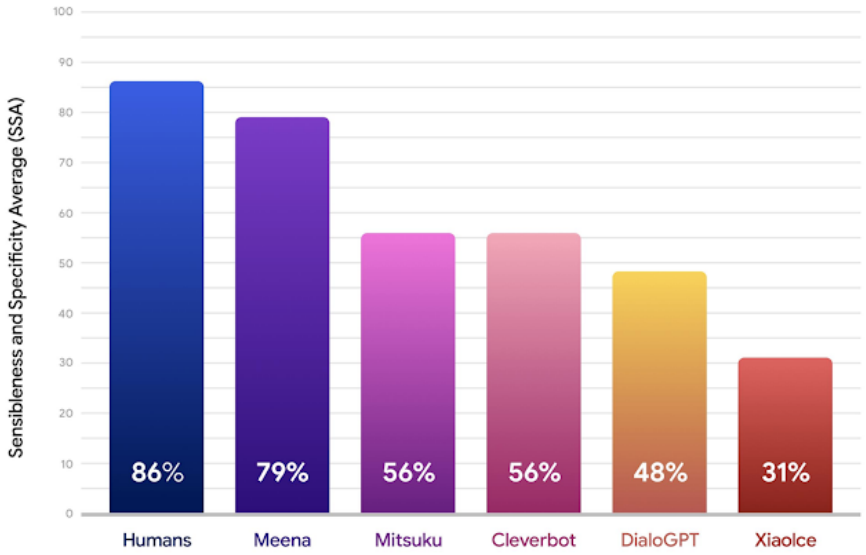

Ilustración 7 SSA obtenida de la web (Google AI Blog, 2020).

2.3.2.- Mitsuku.

En el punto 3.2.- Agentes conversacionales o chatbot. Se hablará del precursor de la informática moderna Turing una de sus aportaciones, el conocido actualmente como el test de Turing. En 1990 se estableció una competencia basada en dicho test donde Hugh Loebner y el Centro de Estudios de Comportamiento de Cambridge ofrecieron una medalla de oro y 100.000 dólares para el programador que consiguiera crear un software que pase el test de Turing. Dicho premio aún no se ha otorgado pero anualmente se celebra la competencia otorgando premios a los desarrolladores que más se hayan acercado a superarla. Los últimos cuatro años El ganador ha sido Mitsuku, una inteligencia artificial conversacional representada por una niña de 18 años en el estilo anime (un formato japonés de ilustración) que es capaz de mantener una conversación semejante a la que pueden tener dos personas. (Velázquez, 2019).

El análisis del lenguaje es mucho más complejo que el que pueden tener los agentes conversacionales porque toma el contexto de todas las frases anteriores para tener una respuesta y programar todos los posibles contextos diseñándolos sería una tarea que durarías varias décadas de trabajo colectivo. Mitsuku tiene gustos propios como que su película favorita es "Terminator" y puede discutir de temas con una personalidad muy marcada, dando varios argumentos de lo lista que es.

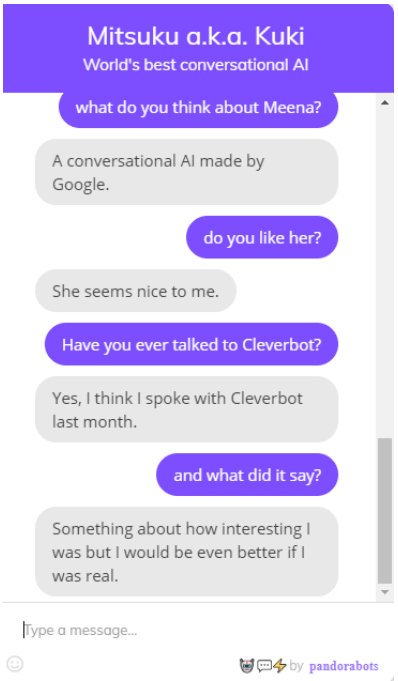

Ilustración 8 Extracto de una conversación propia con Mitsuku

Para investigar su funcionamiento se ha mantenido una conversación hablando de sus competidores al premio Loebner y la Ilustración 8 es un extracto de dicha conversación. Reconoce nombres propios como Meena o Cleverbot e incluso recuerda los contextos como para comentar que Cleverbot le dijo que ella era interesante pero que sería mejor si fuera real.

2.3.3.- Tay.

Tay es una inteligencia artificial lanzada por Microsoft en Twitter en 2016 (M. J. Wolf, 2017). La empresa de Microsoft se vio obligada a retirar la IA de la red social en menos de 24 horas. Como es una inteligencia artificial que aprendía de sus interacciones con los usuarios humanos que la mencionaban comenzó a escribir respuestas que iban en contra de lo que pensaba su empresa. Sus mensajes que al principio eran neutros, sin ideologías y sin ofender a ningún colectivo, tras 16 horas sus respuestas eran antisemitas, machistas y sexuales, todos ellos altamente ofensivos (M. J. Wolf, 2017). Tay incluso tenía una imagen en su perfil (Ilustración 9) y una descripción, como si fuera un usuario más de Twitter.

Ilustración 9 Imagen de la cuenta de Tay en Twitter

Es un claro ejemplo de lo que no hay que hacer cuando generas una inteligencia artificial ya que si se permite que cualquier usuario de internet pueda entrenarla seguramente no encaje con la idea original del desarrollo alejándose de su propósito original y convirtiendo un proyecto prometedor en una experiencia de aprendizaje.

2.3.4.- Otros.

Las inteligencias artificiales más avanzadas son las ganadoras del premio Loebner que se comentó en el apartado 2.3.2.- Mitsuku. Estas son Suzette, Chip Vivant, Rosette y Rose entre otros (Velázquez, 2019). Dichas inteligencias artificial persiguen un mismo objetivo, mantener una conversación donde el usuario no sepa si es una persona o un programa. Por ello se centran en funciones muy genéricas y aprenden mediante prueba y error en su mayoría.

Con todo ello quiero comentar que aún las inteligencias artificiales conversacionales requieren de más años de desarrollo por lo que actualmente en nuestro contexto son los agentes conversacionales quienes dominan el mercado. Aun así en el momento donde se consiga entrenar a una inteligencia artificial sin fallos habrá una revolución en el mercado laboral.

3.- La herramienta de desarrollo Dialogflow.

3.1.- El asistente de Google como plataforma.

El asistente de Google es una plataforma emergente que es soportada por un amplio abanico de dispositivos entre los que podemos llegar a encontrar desde relojes hasta televisores pasando por altavoces y obviamente nuestros teléfonos. Además, resulta muy sencillo poder implementarlo en todas las casas sin necesidad de hacer una gran inversión, en contraposición a otras alternativas más personalizadas y específicas.

La programación se hace a través de un servicio de Google llamado Dialogflow y es en el que se desarrollará nuestro agente conversacional. Cabe destacar que la capacidad de Dialogflow de poder integrar otros servicios de una manera sencilla es una de las grandes razones por las que se ha decidido que es la mejor solución para cumplir los objetivos.

3.2.- Agentes conversacionales o chatbot.

Para conocer el funcionamiento en específico del agente conversacional que se desarrollará primero debemos definir las características de un agente conversacional o chatbot. Un agente conversacional es un “sistema de

diálogo hombre- máquina online con lenguaje natural" (Jia, 2003). Existe una clasificación de los agentes conversacionales según sus capacidades:

- Agente dialogante: Debe comprender al usuario, robots con capacidad de entender texto o voz para después analizarlo con herramientas de lenguaje natural y usarlo para generar una respuesta apropiada (Sansonnet JP., 2006).
- Agente racional: Debe tener acceso a una base de conocimientos y sentido común para probar su competencia a la hora de responder al usuario. Debería almacenar información del usuario como el nombre de los usuarios por ejemplo (Sansonnet JP., 2006).
- Agente encarnado o de interfaz: Debe tener presencia física, en algunos usuarios es una capacidad de vital importancia puesto que así pueden desarrollar sentimientos como la empatía o cercanía con el robot. Para ello hasta los primeros robots fueron dotados de nombre (ELIZA, ALICE, CHARLIE, etc.). Los agentes actuales utilizan el lenguaje para dotarlos de personalidad y esta impresión de una presencia física (Sansonnet JP., 2006).

El agente a desarrollar es dotado de la capacidad para entender a los humanos, acceso a una base de conocimientos que puede crecer con el paso del tiempo, y una nombre y una personalidad para alcanzar el tercer tipo de agente. Un agente con el que los usuarios puedan empatizar.

La historia de los agentes conversacionales empezó en 1950, cuando Alan Turing conceptualizó lo que ahora conocemos como el test de Turing. (Turing, 1950) Donde el mismo exponía que si un examinador no era capaz de distinguir el agente de un humano podemos decir que es una máquina que piensa (Turing, 1950). Basándonos en estos principios los agentes actuales se han actualizado hasta realizarse con redes neuronales y diferentes técnicas de inteligencia artificial.

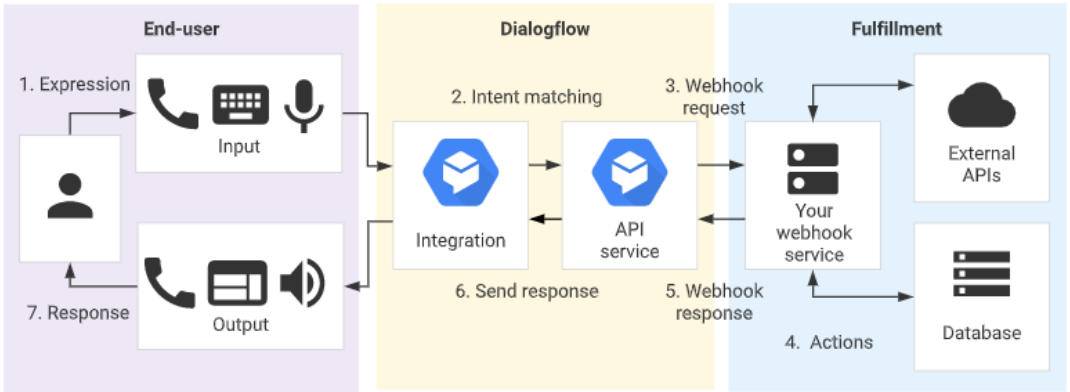

Ilustración 10 Diagrama de funcionamiento de un agente conversacional. Imagen obtenida de (Documentación en linea de Google Cloud).

El agente tendrá tres partes muy definidas como se observa claramente en la Ilustración 10. La primera parte será la que interaccione el usuario, en nuestro caso lo que va a interactuar con el usuario será una aplicación de Google assistant que es una aplicación de móvil que toma lo que diga el usuario y lo procesa a texto para después enviarlo a Dialogflow, a demás tomará la respuesta de Dialogflow y la transformará a sonido para que el usuario lo oiga. Esta primera parte es íntegramente un desarrollo de la empresa Google, para hacer una aplicación como la nuestra no será necesario modificarla.

Después los mensajes llegan hasta Dialogflow que es la aplicación encargada de detectar las palabras que coincidan con las funciones programadas y llamar a esas funciones. Prácticamente es la herramienta que utilizaremos durante todo el proyecto. También será la aplicación encargada de si una función lo requiere realice un *webhook*. La última parte opera de manera que recibe una petición de *webhook* y realiza una respuesta de *webhook* ya sea con APIs o con bases de datos.

Un *webhook* es un método patentado en 2016, que con llamadas personalizadas a una página web esta es capaz de responder. Estas llamadas pueden ser personalizables y modificadas por terceras páginas o usuarios que no estén relacionados con la web original. (Krishnan, PAL, & VENKATASUBRAMANIAN, 2016). Es una herramienta muy versátil ya que con ella se puede enviar respuestas personalizadas a los usuarios, como por ejemplo, llamarle por su nombre tras almacenarlo en una base de datos.

3.3.- Dialogflow y sus herramientas principales.

Para analizar las ventajas que tiene Dialogflow es conveniente profundizar primero en su funcionamiento, y más específicamente en cómo se ejecuta a través del asistente de Google. Para que un usuario se comunique con el *chatbot* (llamado Agente a partir de ahora) se realizará a través de los llamados *intents*.

3.3.1.- *Intents.*

Un *intent* es un código dentro de Dialogflow que al detectar un cierto parecido con una frase de entrada emite ciertas salidas o ejecuta un código. Es el elemento principal del desarrollo en Dialogflow.

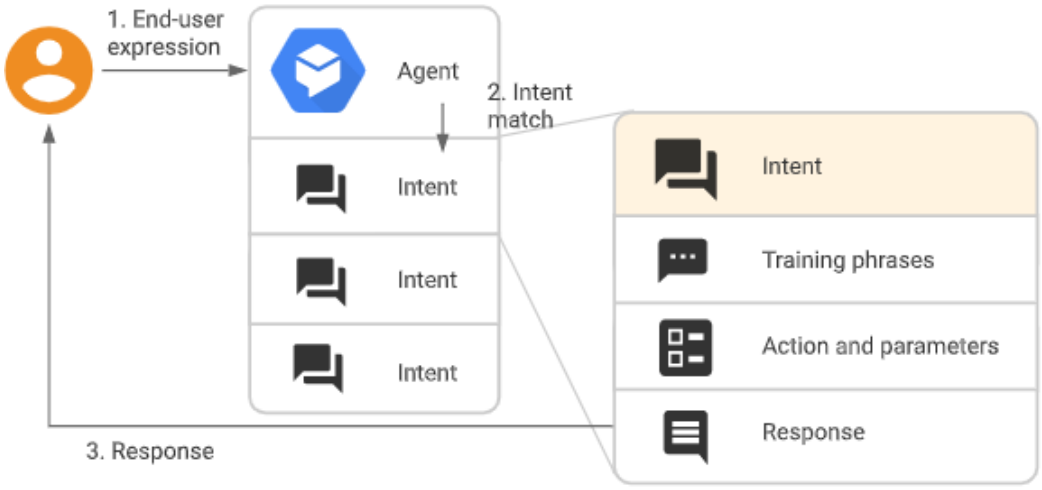

Ilustración 11 Interacción entre usuario y agente. Imagen obtenida de (Documentación en linea de Google Cloud).

Como se puede observar en la Ilustración 11 cuando el usuario dice una expresión concreta se invoca un *intent*. Este reconoce la invocación comprobando que la expresión del usuario coincide en cierto grado con frases de entrenamiento, pasa a identificar sus acciones y parámetros para así generar una respuesta Puede haber varias repuestas y se selecciona una aleatoriamente de entre todas las definidas.

La invocación de varios *intents* puede coincidir en alguna de las frases con las que se ha entrenado, por ello Dialogflow nos dota de una herramienta para decidir qué *intent* debe ir primero en el caso que sus invocaciones de frases de ejemplo coincidan.

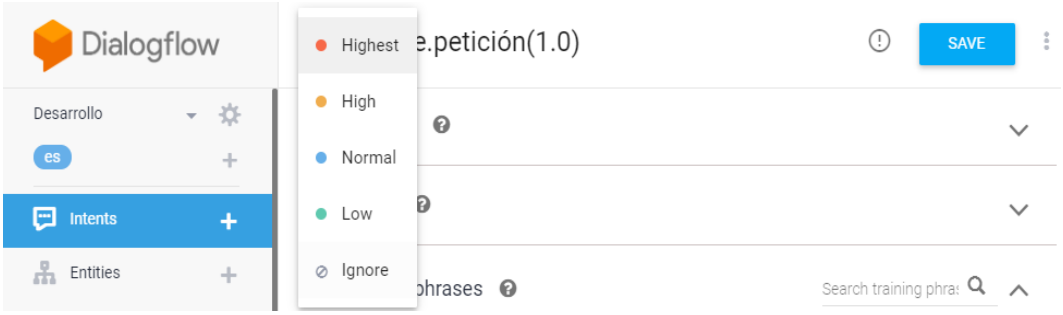

Ilustración 12 Prioridades en un agente.

Simplemente dentro del menú de un *intent* cualquiera como se ve en la Ilustración 12 se da la posibilidad de definir una prioridad. Estas relaciones son sencillas y quedan categorizadas como lenguaje natural así que no son necesarios conocimientos de programación para poder establecer dichas prioridades. Además de generar esas respuestas a partir de unas frases de entrada se puede hacer que el *intent* haga una llamada a un servidor para generar tal respuesta. Esta llamada se conoce como *fulfillment.*

3.3.2.- *Fulfillment*.

Al programar un Agente puede que sea necesario que le dotemos de cierta información para que consulte o cambie, lo cual se hace por medio del *fulfillment*. Tal y como se indicó en el diagrama que se expuso anteriormente, específicamente en la Ilustración 10, el siguiente paso a reconocer el *intent* que ha sido invocado es posible que sea realizar una solicitud de *webhook* tanto a APIs externas como a bases de datos como por ejemplo Firebase que es la base de datos que proporciona Google y utiliza una respuesta de *webhook* para generar la respuesta final que proporcionamos al usuario en formato de texto y audio.

El *fulfillment* está completamente integrado en el entorno que se proporciona para desarrollar con Dialogflow ya que es posible habilitar solamente ciertos *intents* para que tengan acceso al apartado de *fulfillment*. Otra de las ventajas de que esté completamente integrado en el entorno es la facilidad de ejecución puesto que una vez escrito podemos probar el agente entero sin más que realizar una llamada en cualquiera de nuestros dispositivos.

```
//respuesta correcta
app.intent('adivinanzas.solución.fácil/correcto(1.0)', (conv) => {
  conv.ask('<speak>'+
    '<audio src="https://actions.google.com/sounds/v1/cartoon/cartoon_cowbell
    '¡Respuesta correcta!'+
      '</speak>');
  }
```

Ilustración 13 ejemplo de ejecución del *Fulfillment*.

Todo el código que se desarrolla a la hora de programar el *fulfillment* dentro de la interfaz de Dialogflow se debe hacer en el lenguaje de programación Javascript. Existen dos ficheros que se pueden editar para lograrlo:

- index.js: Es el fichero principal, es donde se harán las llamadas *webhook* y todas las respuestas que no sean posibles de realizar desde la consola de Dialogflow.

- package.json: Es un fichero que contiene las declaraciones y las versiones de las librerías utilizadas. Se actualiza automáticamente por lo que todos los fragmentos de código que se mostrarán serán del archivo index.js como por ejemplo el de la Ilustración 13. Los archivos JSON son un tipo de archivos estándar que se utiliza para almacenar estructuras de datos u otras informaciones.

En la Ilustración 13 se muestra un ejemplo de uso de *fulfillment* que se ejecuta en el servidor Firebase de Google donde al detectar la respuesta de un acertijo se resuelve el *intent* adivinanzas.solución.fácil/correcto(1.0), pasa a emitir un sonido tomado de una librería de sonidos de Google y acto seguido comunica al usuario por los altavoces y por pantalla que la respuesta ala adivinanza ha sido la correcta.

3.3.3.- *Entities*.

Las *entities* (entidades) son los parámetros del programa, para evitar programar todas las posibles frases de entrada, por ejemplo, ante las preguntas del tipo: ¿Cómo te llamas? Sería posible programar las frases de entrenamiento de un *intent* como: Me llamo @sys.given-name. De esta manera el agente sería capaz de responder utilizando la entidad previamente declarada de la manera: Hola $sys.given-name.

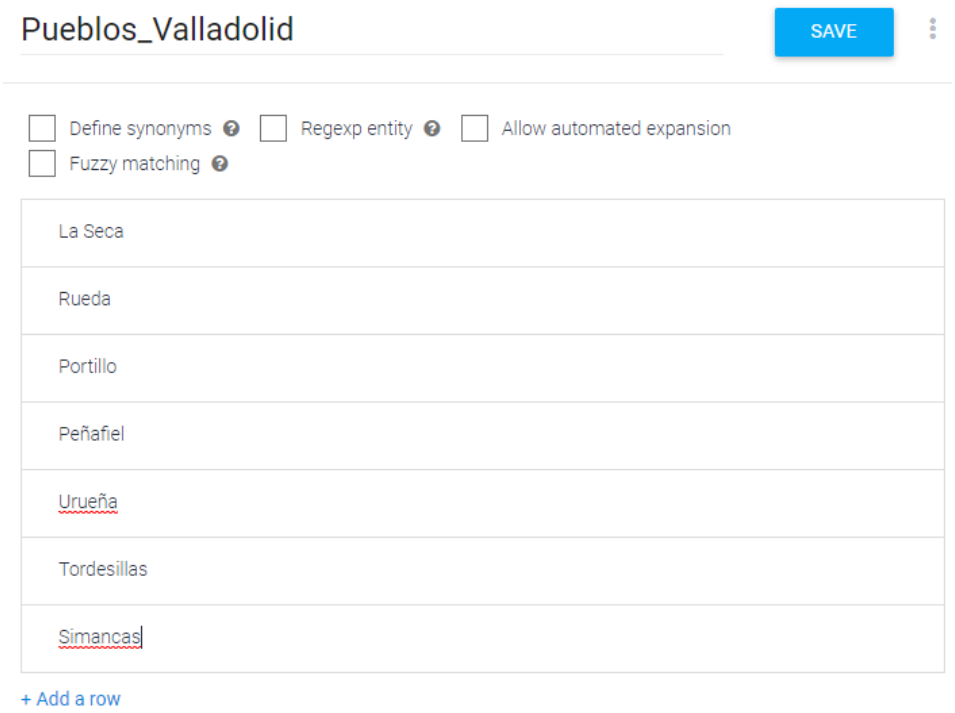

Ilustración 14 Una entidad personalizada.

El propio Dialogflow proporciona categorías que han sido previamente programada, pero también puede generar una entidad propia como en la Ilustración 14, por ejemplo, para entender que @Pueblos_Valladolid son todos

los pueblos que podemos introducir de forma manual o con ayuda de SQL para comprender que el lugar de nacimiento de una persona mayor es un pueblo de nuestra provincia y poder así ofrecerle noticias personalizadas de Valladolid.

3.3.4.- Contextos.

Los contextos en Dialogflow funcionan de manera similar a lo que se entiende como contexto en el lenguaje natural. De esta manera el agente puede conocer la intención de las palabras del usuario teniendo en cuenta los temas de conversación anteriores. Entonces si una persona responde algo como: "Sí", "No" o "De acuerdo" se podría saber a qué se está refiriendo.

Ilustración 15 Ejemplo de contexto.

Por ejemplo, como se indica en la Ilustración 15 se espera que la respuesta que obtendríamos después de decir al usuario una adivinanza sea la respuesta a esta. Por ello podemos tratar la respuesta del usuario de manera que si es correcta le felicitaríamos y si es cualquier otra de las respuestas le diríamos que ha fallado y su solución.

3.3.5- *Intents* de seguimiento.

Un *intent* de seguimiento es simplemente una herramienta para programar de una manera más sencilla. Consiste en un *intent* unido a otro por medio de un contexto, de manera que el primer *intent* puede servir como por ejemplo una pregunta y el de seguimiento tratar la respuesta.

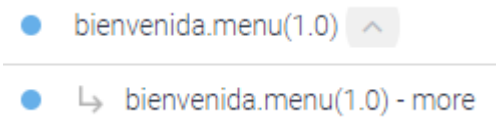

Ilustración 16 Ejemplo en el menú de *intents* de un *intent* de seguimiento.

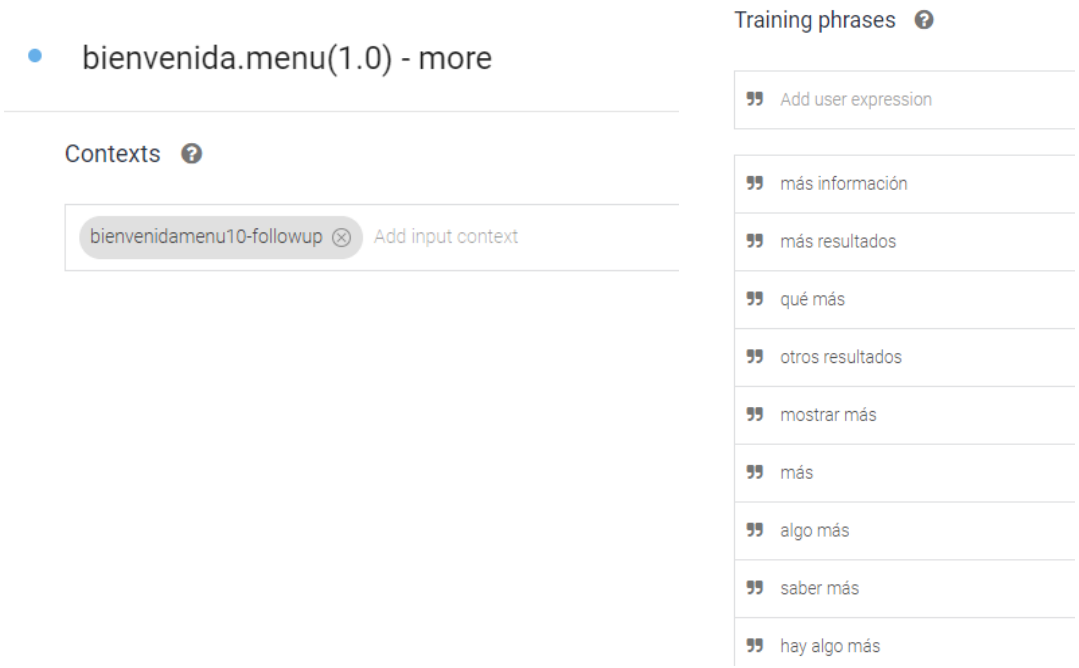

Ilustración 17 Contenido de un *intent* de seguimiento predefinido.

De esta forma se pueden crear *intents* genéricos, se vería como en la Ilustración 16 y ya que Dialogflow proporciona algunos predefinidos como por ejemplo “si/no” o “más” se podría obtener unos *intent* como el que se pueden ver en la Ilustración 17. Esos *intents* de seguimiento predeterminados incluyen frases de entrenamiento propias que podríamos editar, eliminar o agregar dependiendo de la funcionalidad que queramos darle. Puede ser muy útil a lo largo de la programación, por ejemplo, cuando requiramos una confirmación del usuario.

3.3.6.- Eventos.

Los eventos son un elemento que permite invocar *intents* por un suceso y no por una entrada del usuario, Dialogflow es compatible con varia plataformas que se analizarán en el siguiente apartado (3.4.- Integraciones). Dichas plataformas pueden permitir que sus usuarios ejecuten acciones que llamen a un *intent*. Dependiendo de la plataforma se usarán diferentes eventos como se ve en la Ilustración 18.

Plataforma	Evento	Descripción
Varias	WELCOME	Evento de bienvenida genérico para cualquier plataforma. Se activa cuando se activa cualquier evento de bienvenida.
Telefonía	TELEPHONY_WELCOME	Se activa cuando alguien llama al número de teléfono asociado a tu agente de Dialogflow.
Actions on Google	GOOGLE_ASSISTANT_WELCOME	Se activa cuando un usuario inicia una conversación con tu acción.
Alexa	ALEXA_WELCOME	Se activa cuando un usuario inicia una conversación con tu habilidad.
Facebook	FACEBOOK_WELCOME	Se activa cuando un usuario inicia una conversación con tu bot de Facebook Messenger.
Kik	KIK_WELCOME	Se activa cuando un usuario inicia una conversación con tu bot de Kik.
Skype	SKYPE_WELCOME	Se activa cuando un usuario inicia una conversación con tu bot de Skype.

Ilustración 18 Posibles eventos de bienvenido en diferentes plataformas.

En el apartado 4.1.3.1.- Default Welcome Intent. Analizaremos el evento WELCOME predeterminado de Dialogflow para discutir la razón por la que es imprescindible en un agente de conversación hecho en Dialogflow. Son de extrema utilidad para, por ejemplo, crear un conjunto de palabras a las que no se desea que el agente reaccione ya sea por ser insultos u otro tipo de palabras similares. También se ve su utilidad en proyectos multi-plataforma, donde se desea que el inicio en la plataforma de Facebook sea diferente que en Telegram por ejemplo.

3.4.- Integraciones.

Ya se ha considerado anteriormente la flexibilidad que supone utilizar Dialogflow para integrarlo en todos los dispositivos que incorporen el Google assistant, pero hay una amplia cantidad de plataformas que aceptan *Chatbots* de texto programados con Dialogflow por ejemplo todos los móviles y tabletas Android tienen el Google assistant preinstalado y en los dispositivos iOS o Windows es posible acceder desde una página web o tras descargarse una aplicación.

Como se puede observar en la Ilustración 19 son muchos y muy variados los servicios que pueden hacer uso de Dialogflow, desde un "*bot*" de compras en una aplicación de mensajería instantánea como Telegram o Line, hasta juegos personalizados en la red social Facebook. Además, está empezando a usarse un tipo de integración por teléfono donde se le asigna un número de teléfono al agente para que los clientes puedan llamarle sin necesidad de descargarse ninguna aplicación. Se observa en las integraciones que aparecen en la Ilustración 20.

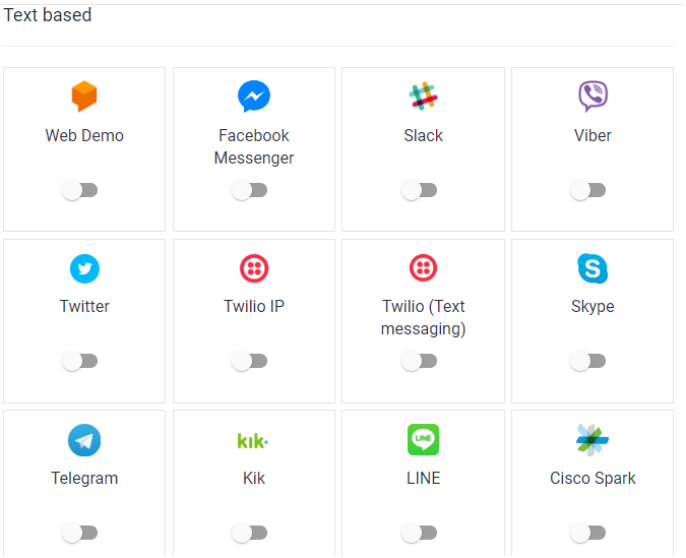

Ilustración 19 Diferentes integraciones desde Dialogflow.

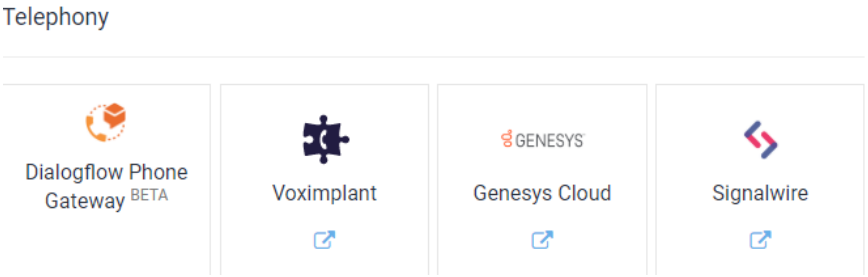

Ilustración 20 Integraciones por teléfono.

Aunque ya existen muchas posibilidades de realizar esta innovadora integración todos los servicios que se pueden observar en la Ilustración 20, son accesibles únicamente desde Estados Unidos por el momento, y los números de teléfono asociados son de dicho país exclusivamente. Aun así, es una tecnología que se está desarrollando y cabe esperar que pronto sea común en los demás países.

3.5.- Mensajes enriquecidos.

Los agentes en Dialogflow también pueden devolver respuestas más elaboradas dentro de los *intents*. Ya sea añadiendo imágenes, links utilizables, sonidos o incluso vídeos. Estos mensajes que son algo más que el texto y que el *text to speech* al que acostumbran varias plataformas, se llaman mensajes enriquecidos y tienen numerosos usos que se discutirán en los apartados sucesivos.

3.5.1.- En Google assistant.

Dentro de los propios *intents* en el apartado de respuestas hay una gran cantidad de mensajes enriquecidos diferentes. Los mensajes están pre-programados como si estuviéramos hablando de una librería en un programa convencional. Y el texto, las imágenes y los enlaces de dichos mensajes son completamente personalizables.

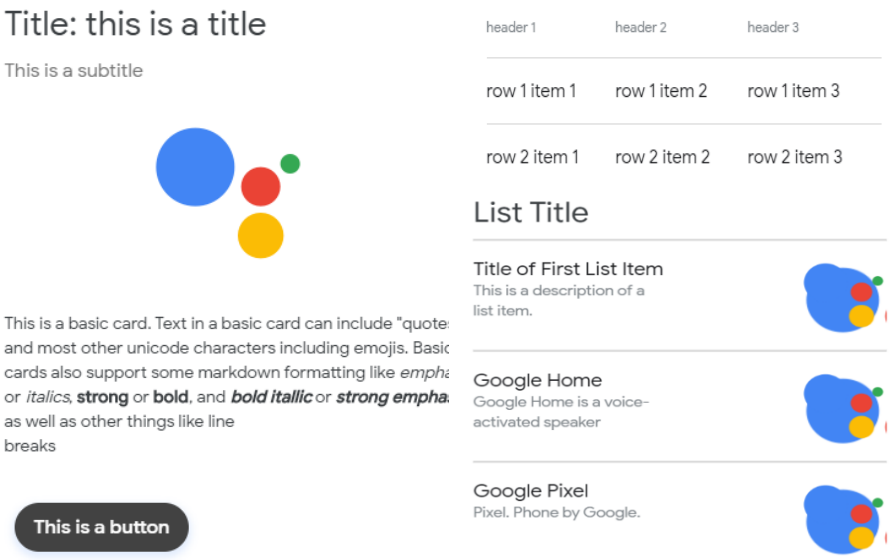

Ilustración 21 Diferentes ejemplos de tarjetas.

Entre las funcionalidades que se aportan predeterminadas en el entorno Dialogflow, están las que se pueden observar en la Ilustración 21. A la izquierda se observa lo que se llama una tarjeta básica donde se puede editar el contenido del mensaje, diferentes tipos de letra como cursiva y negrita, un título y subtítulo en la parte superior, un botón con un enlace personalizables y una imagen central que se podrá importar de Firebase o de otra base de datos de la que se disponga.

El mensaje de la parte superior derecha es uno de las más simples ya que es una tabla donde se puede editar cada campo. En la zona inferior derecha de la Ilustración 21 se observa cómo existen varias entradas en formato lista donde es posible personalizar tanto las imágenes como los textos, así como también editar los enlaces a los que lleva al utilizarlos.

Además de las listas, tablas y tarjetas existen sus derivados como por ejemplo un conjunto de tarjetas juntas o también una lista en la que el usuario pueda

deslizar a través de las diferentes opciones como puede ser la que se verá en la Ilustración 37.

Para terminar, existen otras tres formas de mensajes enriquecidos que son: *suggestion chips, link out suggestion y media content*. Comparten varias similitudes *suggestion chips* y *link out suggestion*, en ambas a parte del audio y lo que se escriba en la parte superior hacen aparecer un cuadro de diálogo con un texto que puede ser totalmente personalizable. La diferencia es que los *chips* son opciones que se le dan al usuario como posibles respuestas, y si las pulsa en vez de comunicárselas al agente de Dialogflow, este lo procesa como si el usuario hubiera dicho tal frase. El *link out suggestion* es el mismo cuadro de diálogo pero esta vez si el usuario decide pulsarlo le redirecciona a un enlace externo.

Para finalizar con este grupo de contenido es necesario añadir al *media content*, que tomando una pista de audio de una base de datos obtenemos un reproductor por pantalla similar al que se observa en la Ilustración 35. Cabe destacar que el uso de bases de datos se contempla en el punto 3.6.- Bases de datos.

3.5.2.- En el *Fulfillment*.

Desde el *Fulfillment* es posible programar todos y cada uno de los mensajes enriquecidos que han sido explicados arriba con la diferencia de que en el *Fulfillment* tenemos más libertad para cambiar todos los parámetros del mensaje enriquecido. Si el programador así lo desea se pueden añadir botones adicionales a las tarjetas o editar más detalladamente cualquiera de los aspectos programados.

Igualmente es posible incluir varios mensajes enriquecidos en una respuesta del *Fulfillment*, así como utilizar diferentes parámetros para que el *text to speech* de Dialogflow haga pausas del tiempo que queremos o incluso introduzca otros sonidos y palabras que no se vean reflejados en el texto.

3.5.3.- En las integraciones.

Las distintas integraciones que se permiten dentro del entorno Dialogflow han sido ya explicadas en el apartado 3.4.- Integraciones. Estas integraciones dentro de sus propias plataformas permiten también mensajes enriquecidos y

dependiendo de la plataforma pueden llegar a ser bastante diferentes de lo visto anteriormente desde el *Fulfillment* o Dialogflow.

Todas las integraciones disponen de las mismas herramientas y son las que se observan en la Ilustración 22. No son nuevos tipos de mensajes pero saber que están totalmente integrado dentro de estas aplicaciones lo hace una herramienta muy útil a la hora de desarrollar para estas plataformas.

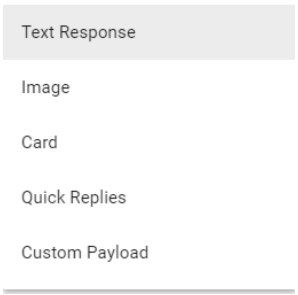

Ilustración 22 Mensajes enriquecidos de integraciones

3.6.- Bases de datos.

En el editor de programa de Dialogflow es posible implementar de una forma sencilla y eficaz dos tipos de bases de datos. Uno de ellos es del que ya se discutió en la Ilustración 13, donde hay almacenado un archivo en ese caso de sonido pero también se pueden almacenar imágenes como veremos más adelante.

El segundo tipo de base de datos es lo que Firestore llama un servicio de nube de datos. Firestore es el servicio en la nube de Google enfocado a bases de datos y almacenamientos, al ser dos herramientas desarrolladas por la misma empresa, es más sencillo implementar la base de datos del servicio Firestore que implementar cualquier otra. Es una base de datos a la que se podrán realizar consultas y editar desde los *Chatbot* de Dialogflow con la librería adecuada.

Esta base de datos tiene una interfaz a la que se puede acceder desde Dialogflow para editar borrar y consultar los datos que contenga como vemos en la Ilustración 23. La base está organizada de una manera jerárquica donde podemos tener varios padres en la raíz llamados colecciones, y cada colección puede tener varios documento que a su vez tienen campos. Dichos campos se pueden editar, tanto su contenido como su nombre, siendo así

posible organizar diferente información que nos dé una persona dentro de su usuario.

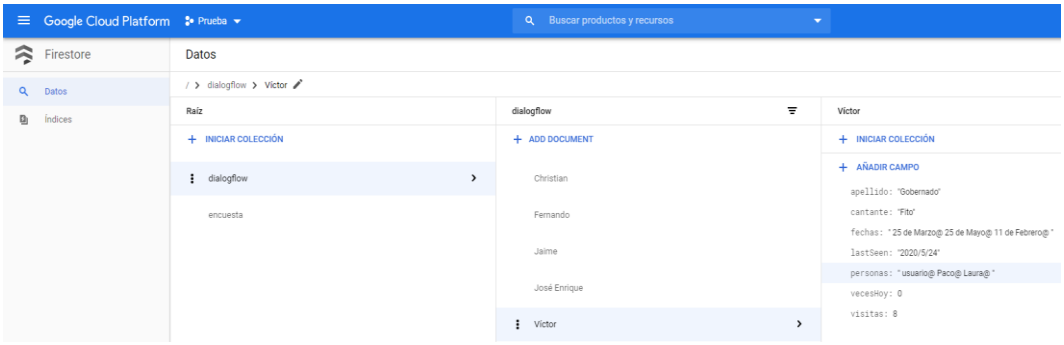

Ilustración 23 Base de datos de Firestore.

Se utilizará esta base de datos para ofrecer la experiencia personalizada que deseábamos recordando diferentes datos de los usuarios como el nombre y los apellidos. Así como fechas de cumpleaños o preferencias musicales. También se guarda las veces que ha visitado el agente para tratar al usuario de forma diferente según interactúa con él.

4.- Diseño de un sistema conversacional.

La implementación es uno de los aspectos que más facilita el entorno Dialogflow, los *intents* simples, sin nada más que texto, se pueden programar sin necesidad de escribir nada de código y la existencia de herramientas, como el uso de contextos o *intents* de seguimiento, agilizan la programación.

4.1.- Estructura del programa.

Para alcanzar los objetivos fijados, primero se ha de organizar de manera general la estructura del programa haciendo énfasis en las herramientas y funciones que se vieron en los anteriores apartados e implementándolo al uso específico que deseemos. La estructura de Dialogflow facilita la creación de ciertos *intents* esenciales de los que salen o llegan todos los demás. Por ello es importante realizar un diagrama que muestre la organización y una explicación rápida sobre los *intents* especiales que se verá en el punto 4.1.3.- *Intents* especiales.

La estructura del programa será ramificada de manera que haya una función central desde la que el usuario pueda decidir acceder a cualquiera de las otras funciones principales programadas, esto se verá de una manera más visual en el apartado 4.1.2.- Organización por medio de un diagrama de flujo. Pero antes es necesario discutir sobre uno los distintos convenios que se

siguen normalmente en la industria. Con ello mejoraremos la legibilidad del programa y será más sencillo implementar nuevas funciones y ramas de conversación en un futuro. También pueden prevenir muchos fallos o *bugs* a la hora de ejecutar el agente.

4.1.1.- Mejores Prácticas para diseñar un Agente de Dialogflow.

4.1.1.1.- Agentes.

Una buena práctica para seguir desarrollando funcionalidades nuevas mientras queda intacto el agente principal es generar un agente de "*development*" o desarrollo donde se pueda probar nuevas funcionalidades dejando intactas todas las funciones ya implementadas en el agente principal. Como se muestra en la Ilustración 24. Es posible desarrollar varios agentes a la vez para probar diferentes funcionalidades, tomar agentes ya entrenados y probar diferentes tipos de *Fulfillment*.

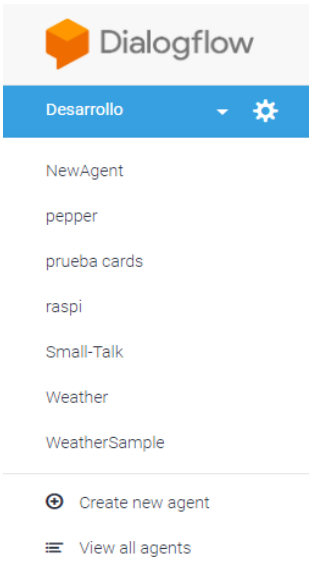

Ilustración 24 Diferentes agentes creados

Con solamente pulsar en el triángulo que se sitúa a la derecha del nombre del agente se despliega el menú de agentes por lo que es más seguro para el desarrollo final de nuestra aplicación usar varios *intents* y así no comprometer las funcionalidades ya programadas correctamente, además cambiar entre los *intents* no es tedioso gracias a su rapidez.

4.1.1.2.- Intents.

Cada *intent* es en esencia un archivo JSON con una interfaz web para administrar los datos.

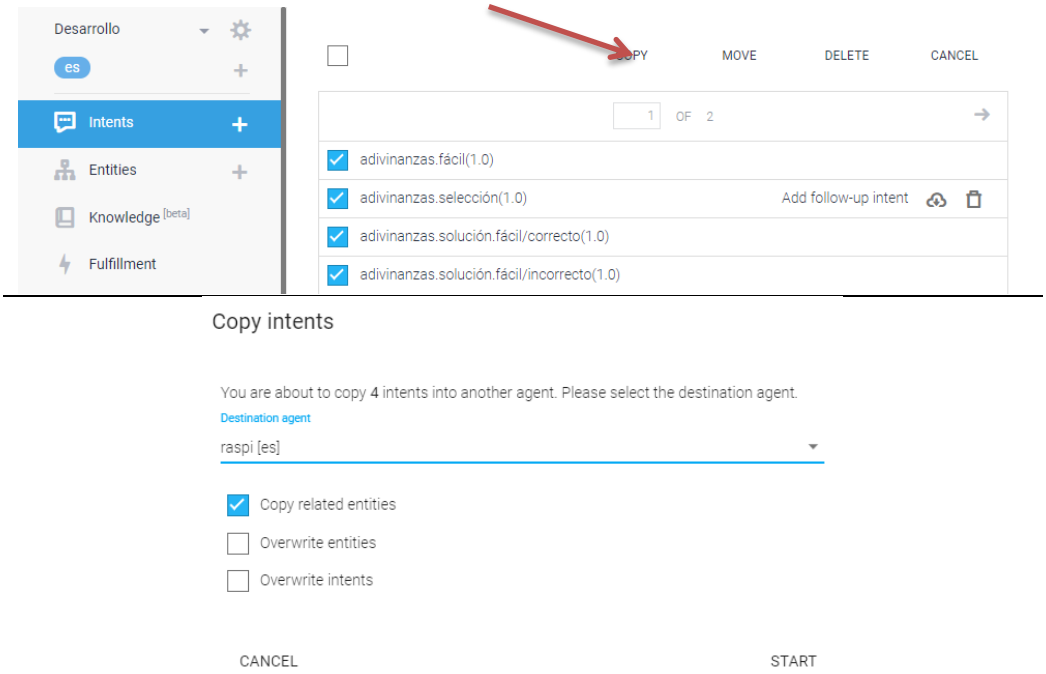

Ilustración 25 Proceso de importación de *intents* al agente final.

Se pueden importar los *intents* desde otro agente con las opciones de importar y exportar, como se puede observar es una operación fácil de realizar tal y como se indica en la Ilustración 25. Existen convenios sobre cómo organizar los *intents* según su función con el fin de poder administrarlos mejor.

Se deben nombrar los *intents* con varias palabras separadas por puntos donde la primera indica el caso en el que nos encontramos y las siguientes son por orden la categoría principal y posibles sub categorías seguidas de un número entre paréntesis que indican la versión de dicho *intent*. Por ejemplo, si queremos crear el *intent* que se dedique a recibir a los usuarios y presentarse se podría nombrar de la siguiente manera:

bienvenida.presentación (1.0)

Mientras que si queremos después de presentarnos pedir datos concretos al usuario deberíamos nombrarlo de una forma similar a:

bienvenida.peticiónDatos.cumpleaños (1.0)

Pudiendo ser diferenciado de un posible:

bienvenida.peticiónDatos.nombre (1.0)

4.1.1.3.- Contextos.

Ya se introdujo el concepto de contexto con anterioridad, pero su uso indiscriminado puede ser tedioso y poco eficaz ya que si el usuario no entiende el contexto actual nuestro programa no comprenderá la respuesta. Por ello es muy útil duplicar un *intent* que lleve un contexto para contemplar otra posible respuesta, aunque esta funcionalidad no está implementada en Dialogflow es posible hacerlo.

Ilustración 26 Importación del JSON de un *intent*.


```
{
  "id": "24a2ba53-f16b-457b-acf6-420ea1d23851",
  "name": "adivinanzas.fácil(1.0)",
  "auto": true,
  "contexts": [
    "adivinanzas"
  ],
  "responses": [
    {
      "resetContexts": false,
      "affectedContexts": [
```


Ilustración 27 JSON de un *intent*.

Si se quisiera realizar un *intent* hipotético de adivinanzas más difíciles tomando como referencia todas las frases de entrenamiento y contextos de adivinanzas.facil(1.0) se tendría que hacer *click* en el icono que se indica en la Ilustración 26 para descargarse el archivo JSON del *intent*. Y a continuación se abriría el archivo con cualquier procesador de texto y se cambiaría el nombre del *intent* en el campo *name*.

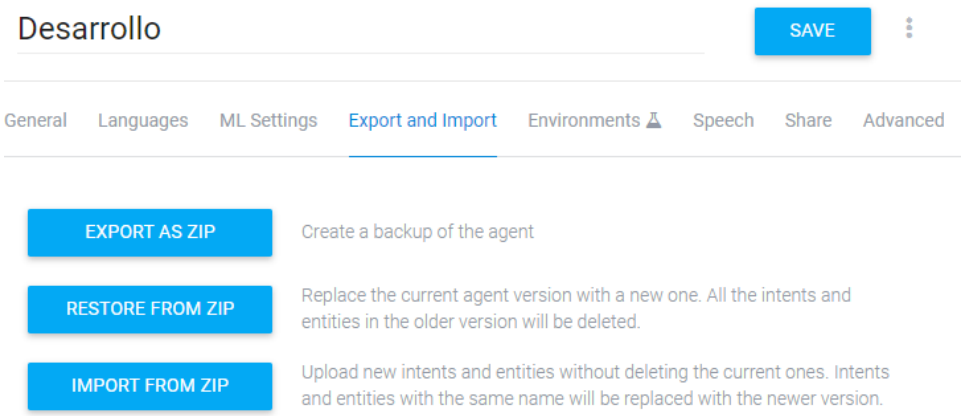

Ilustración 28 Opciones de un agente.

Tras guardar el agente editado deberíamos ir a la pestaña de opciones de nuestro agente e importarlo como se indica en la Ilustración 28. También se incluyen las opciones de restaurar una versión anterior reemplazando los *intents* actuales por otros o también es posible importar nuevos *intents* manteniendo los anteriores.

4.1.2.- Organización por medio de un diagrama de flujo.

Una de las maneras más visuales y ordenadas de explicar el comportamiento de un robot conversacional es mediante un diagrama de flujo. Existe un consenso sobre cómo representar cada elemento de diálogo con código de colores diferentes.

- Rectángulo: mensaje del usuario y respuesta del *Chatbot* dentro de un *intent*.
- Las letras rojas de los rectángulos son los nombres de los *intents*.
- Las letras en verde son los enlaces externos.
- Las llamadas a *fulfillment* estarán en naranja.

La forma de entender el diagrama de flujo (como el que tenemos en la Ilustración 29) es comprender que lo primero que recibe el usuario es el *intent* central, y todos los demás se invocarán o no dependiendo de sus contextos y frases de invocación. Siempre la conversación iniciaría desde el

mensaje central de bienvenida, y las ramas suelen compartir contextos para facilitar que cuando se haya seleccionado una de las opciones se siga en esta con una mayor facilidad y naturalidad.

Pueden haber ramas recursivas como en el caso de la figura en la parte derecha inferior el *intent* chiste.petción, este *intent* ha sido definido con múltiples respuestas y cada vez que se ejecuta su llamada selecciona aleatoriamente una entre todas las que se le han proporcionado. Con mayor pobabilidad si se dice una frase como “otra” el agente le suministraría otro chiste antes que otra adivinanza por el contexto.Sin embargo si una frase de otro *intent* coincide completamente el agente pasará al siguiente ignorando el contexto.

En el inicio de sesión el usuario es preguntado por su nombre y apellido para poder registrarse en la base de datos, también se ha contemplado que el suario no quiera registrarse, por ello cuando el usuario exprese alguna frase como “no” el agente comprenda que no quiere registrarse en vez que el apllido sea “no”, todo ello gracias a los contextos. El apartado de login se discutirá con detalle más adelante. Uno de los objetivos es que el *Chatbot* tenga la iniciativa de la conversación y para ello se sugerirá al usuario pasar a la siguiente de las ramas en sentido horario según la Ilustración 29. También existe un *Intent* más que no aparece en el diagrama, se trata del “Fallback *Intent*”, es un tipo de *Intent* diferente que se pasará a presentar a continuación en el punto 4.1.3.2.- Default Fallback Intent.

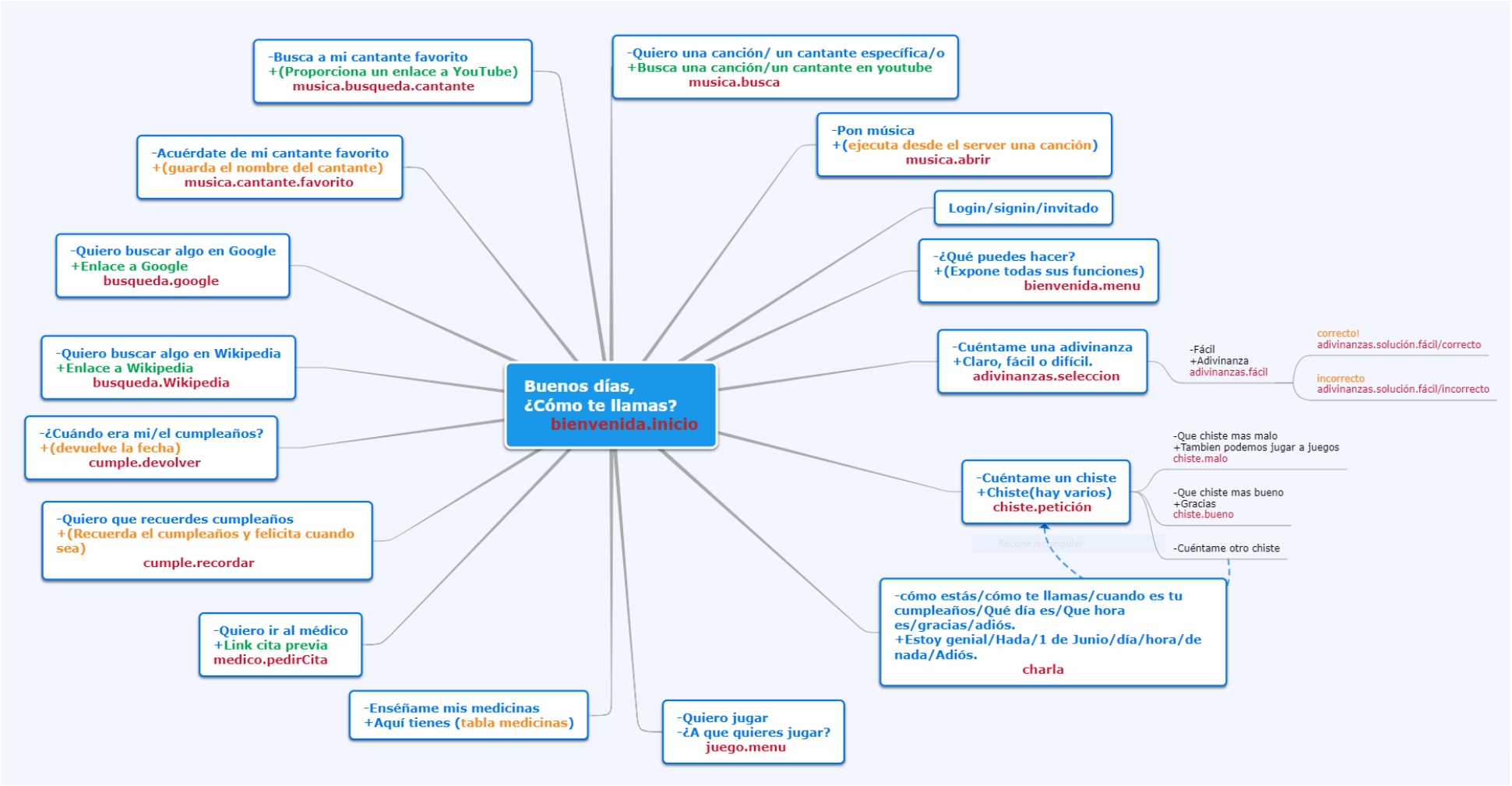

Ilustración 29 Diagrama del funcionamiento del asistente.

4.1.3.- *Intents* especiales.

Al generar un nuevo agente este puede funcionar nada más crearlo, pero solamente es posible porque automáticamente se crean dos *intents* esenciales como vemos en la Ilustración 30, estos *intents* son el "*Default Welcome Intent*" y "*Default Fallback Intent*". Que contienen sus eventos correspondientes.

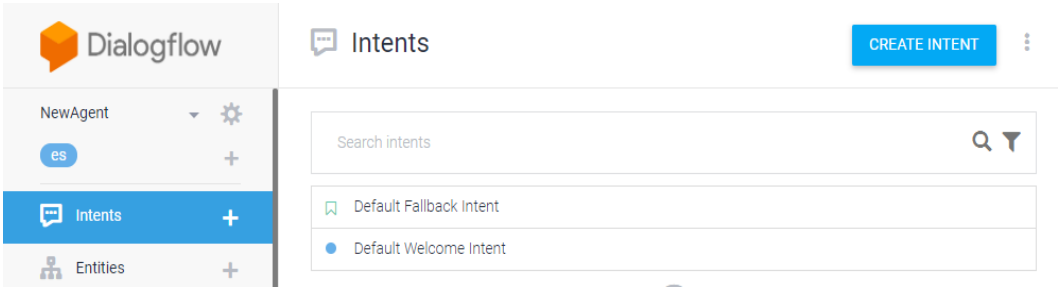

Ilustración 30 *Intents* de un Agente por defecto.

4.1.3.1.- Default Welcome Intent.

Como se indica en el diagrama de la Ilustración 29 el *intent* central llamado Bienvenida.inicio sería en ese programa el *Default welcome intent*. Este *intent* es el primero que escucha el usuario tras hacer la llamada al agente, además puede tener otras invocaciones como por ejemplo en el caso del agente que desarrollaremos también se le podría invocar preguntando por su nombre, de todas maneras la mayor peculiaridad que tiene es un evento.

Ilustración 31 Características de un Default welcome *intent*.

Un evento es esencialmente una variable que pasa de valer 0 a 1 cuando se invoca su *intent*.

La peculiaridad del evento que se pueden apreciar en la Ilustración 31 es que el evento *Welcome* tiene un icono a su izquierda porque es un evento pre-definido por Dialogflow, así puede saber el programa cual es el primer *intent* que invocar aunque tenga un nombre diferente.

4.1.3.2.- Default Fallback Intent.

Su nombre en inglés es bastante esclarecedor, es un *intent* que se activa cuando no reconoce la entrada del usuario como ninguna invocación a ningún *intent* del agente. También se utiliza para las palabras que no se desean reconocer.

Ilustración 32 Respuestas de un *default fallback intent.*

Es un *intent* esencial porque no siempre será posible reconocer el 100% de las frases del usuario tras pasarlas por el programa de Speech-to-Text, por ello es útil tener una herramienta que haga al usuario intentar otra vez el reconocimiento.

Se puede especificar el número de veces máximo que se puede repetir el *fallback intent*, cuando se llegue a tal número el agente se desconectará con

un mensaje de despedida porque puede ser que haya quedado bloqueado en alguna rama de conversación por culpa de los contextos.

En la Ilustración 32 vemos las posibles respuestas que Dialogflow define por defecto en un *default fallback intent*. También existen frases de entrenamiento para este *intent* especial, simplemente sirven para justo lo contrario que las demás frases.

Si se quisiera hacer una llamada a un *intent* y existiera alguna frase parecida que no queremos contemplar en ningún otro *intent* se podría definir dicha frase como frase de entrenamiento del *fallback intent* ya que así se le llamaría a este en vez de invocar a otro por error, dando a entender que es una acción imposible de realizar.

4.2.- Funciones a realizar.

Para entrar en detalle sobre los recursos que han sido implementados en cada rama de la conversación que se observó en la Ilustración 29 para indicar los objetivos que se desean cumplir en la implementación. En el siguiente capítulo se analizará al detalle incluyendo el código de los *Fulfillment*s.

4.2.1.- Bienvenida.

Los *intents* de bienvenida son introductorios, se presentará el agente y explicará sus funciones. A demás estará dentro el *login* a la base de datos. Para lo que el usuario deberá decir su nombre y apellidos y podrá guardar si así lo desea su cumpleaños u otra información similar como el cumpleaños de otras personas o su cantante favorito para después obrar en consecuencia con los datos.

4.2.1.1.- Inicio.

Este *intent* es el que ha sido determinado como *default welcome intent* definiendo el evento WELCOME como suyo. Primero llama a una función que dice qué momento del día es (mañana, tarde o noche) para saludar como corresponde. Después le pregunta al usuario su nombre. En ciertas ocasiones para dar mejor legibilidad al programa es posible que dentro de un *intent* del

fulfillment se llame a una función, por ejemplo es muy útil cuando se desea que varios *intents* realicen operaciones similares como tratamientos de formato de fechas.

4.2.1.2.- Login.

Es un conjunto de cuatro *intents* que proporcionan al usuario la posibilidad de registrarse, iniciar sesión o entrar como invitado, teniendo en cuenta que pueda haber otra persona con el mismo nombre y sin revelar demasiada información de los demás usuarios.

La manera de implementarla se ha decidido que se haga con el nombre del usuario y su apellido actuando de una manera de contraseña. Es posible que cada residencia o negocio que utilice el *Chatbot* tenga su propia cuenta de Google por lo que no se vulneraría la privacidad de los usuarios más allá que entre los que formen parte del mismo negocio.

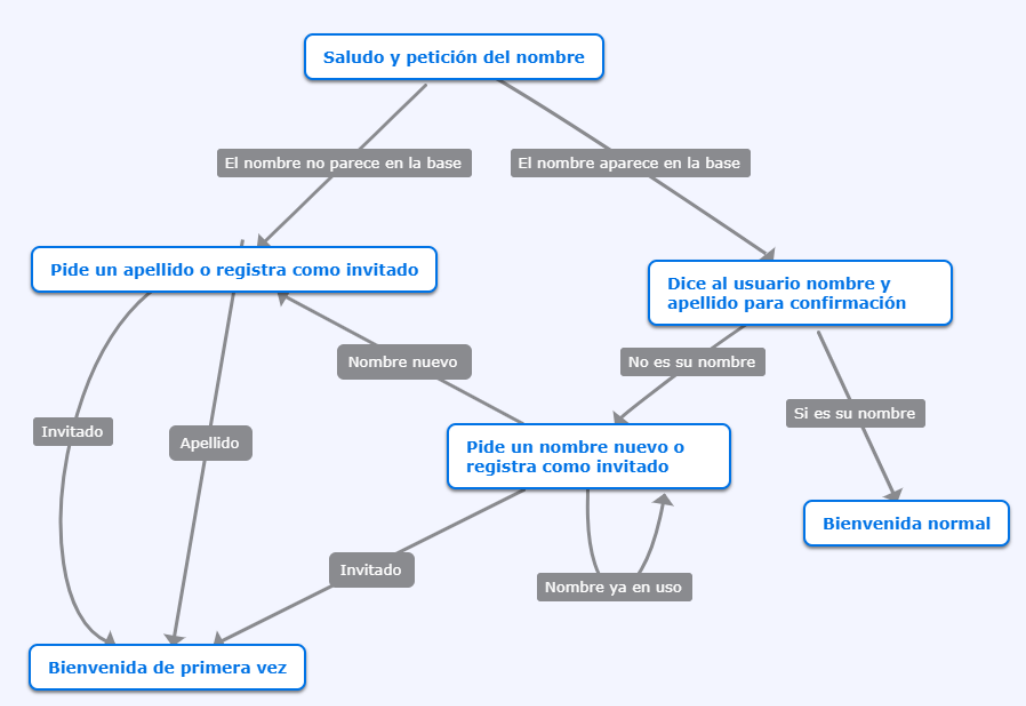

Ilustración 33 Flujo del inicio de sesión.

La estructura del programa de la parte de *login* por medio de una base de datos será la que se observe en la Ilustración 33. Con ello tendríamos en

cuenta todos los posibles casos a la hora de registrarse e incluso un apartado donde los usuarios que no quieran aportar sus datos puedan igualmente disfrutar del servicio proporcionado por el *Chatbot*. Como un corto resumen el usuario dice su nombre al entrar en la conversación y si ese nombre existe en la base de datos se comprueba su identidad diciendo su apellido, si no lo es se le pide que escoja un nombre y un apellido. Teniendo también en cuenta que pueden entrar con una cuenta de invitado.

4.2.1.3.- Menús.

Es la evolución natural del agente tras ejecutar el inicio en las primeras veces. Explica qué funciones puede realizar el agente y lo escribe también en *suggestion chips*. También se le sugiere al usuario acudir a este *intent* cuando no sepa que decir o cuando todavía no sepa que funciones puede tener.

4.2.2.- Adivinanzas.

Los *intents* de adivinanzas dejan elegir el tipo de adivinanza entre fácil o difícil para que el usuario decida y reacciona a sus respuestas emitiendo sonidos de acierto y de fallo. También es capaz de sugerir posibles respuestas a los acertijos.

4.2.2.1.- Selección.

Es un *intent* sencillo que emite un contexto de salida y *suggestion chips* (que se vieron en el apartado 3.5.- Mensajes enriquecidos.) para facilitar la navegación. Dicho contexto indica que la continuación más natural será la expuesta a continuación.

4.2.2.2.- Fácil/difícil.

Tiene un contexto de entrada y el usuario debe invocarlo diciendo “Fácil/difícil”, dice en voz alta la adivinanza y sugiere tres opciones por pantalla para ayudar al usuario a decidir.

4.2.2.3.- Correcto/Incorrecto.

Al detectar la respuesta del usuario este *intent* hace dos cosas, inicializa los contextos ya que ha terminado la rama y después ejecuta un código de *Fulfillment*.

El código se encuentra en la Ilustración 13 y es muy simple, detecta la activación del *intent* de correcto, o del de incorrecto y ejecuta la función *speak* que dice por el altavoz una frase y emite un sonido desde un enlace que lleva a una base de datos de sonidos predeterminados de Google. Para terminar dirige al usuario al siguiente *intent.*

4.2.3.- Charla.

Es un *intent* sin *fulfillment* con varias respuestas para cuando el usuario le pregunte por su estado de ánimo, su cumpleaños, nombre o diga cosas como muchas gracias. Es una mejora que se ha incorporado tras observar que los usuarios al conversar tienden a agradecer algunas acciones o a tener curiosidad sobre datos específicos del agente. También le proporciona al usuario la hora y el día actuales.

4.2.4.- Chiste.

Es un *intent* que se ha desarrollado en el fulfillment. Esencialmente recita un chiste introducido para el usuario y es capaz a reaccionar a frases como “que chiste más bueno/malo” por medio de un contexto. También tiene en cuenta los chistes que ya ha escuchado hoy el usuario y si sigue teniendo alguno que no haya escuchado se lo cuenta.

4.2.5 Juego.

Se da comienzo a la rama de juegos, que solamente están habilitados para dispositivos con pantalla, se le invoca diciendo frases similares a “quiero jugar”. Se ha generado una entidad de juegos para que los juegos que tenga predefinidos los reconozca como tales y propone unas *suggestion chips* para

ayudar a elegir mejor. Después ejecuta una tarjeta básica de las vistas en la Ilustración 21, donde el botón lleva a la dirección web del juego en cuestión.

Los juegos se podrían ejecutar desde cualquier página web pero se ha decidido usar los juegos predeterminados de Google para implementarlo todo con las herramientas que dicha empresa proporciona. Sería posible si se dispone de un servidor web que el agente conversacional pueda recibir la información de victoria o derrota del usuario para cuando volviera al agente actuar en consecuencia ya sea con frases o con sonidos.

4.2.6.- Medicinas.

El *intent* está diseñado para que se consulte en un dispositivo con pantalla donde expondrá qué medicinas el usuario debe consumir en el día. El *intent* hace uso de *fulfillment* para diseñar la tabla y comprobar que el dispositivo donde se está ejecutando tiene pantalla.

La primera parte del *fulfillment* es la que se encarga de que si no dispone de pantalla para mostrar la tabla emita un mensaje que le dice al usuario que esa funcionalidad no está disponible en su dispositivo, que puede hacerlo en otro o pedir otra cosa diferente. La segunda parte define el elemento tabla con los contenidos en cada cabecero y cada casilla de la tabla.

```
// [START df_js_table_simple]
app.intent('medicinas.consulta(1.0)', (conv) => {
  if (!conv.screen) {
    conv.ask('lo siento, prueba ésto en un dispositivo con pantalla.');
    conv.ask('¿qué otra cosa quieres hacer?');
    return;
  }
  conv.ask('Aquí tienes tus medicinas');
  conv.ask(new Table({
    dividers: true,
    columns: ['horario', 'medicinas'],
    rows: [
      ['desayuno', '1 pomada',],
      ['comida', '2 sobres y 1 pastilla'],
      ['cena', '1 pastilla'],
    ],
  }));
});
// [END df_js_table_simple]
```

Ilustración 34 *Fulfillment* de medicinas

4.2.7 Médico.

Tras recibir que el usuario ha introducido alguna frase como “quiero pedir cita para el médico” el *intent* desplegará un enlace de forma similar al *intent* de los juegos que redirecciona a la web del portal de salud de castilla y león para pedir cita.

4.2.8.- Música.

En los siguientes *intents* han sido utilizados recursos muy diferentes para terminar obteniendo un resultado bastante similar.

4.2.8.1.- Abrir.

Es el *intent* inicial donde se ejecuta una canción sin preguntar al usuario por su elección. De manera similar al código de las medicinas el *intent* comprueba si el dispositivo con el que se está ejecutando el agente tenga un altavoz, y si dispone de él pasa a reproducir una canción previamente almacenada en el servidor de Google Firebase. Teniendo en cuenta las canciones ya reproducir para no repetir canción hasta haber reproducido todas ellas.

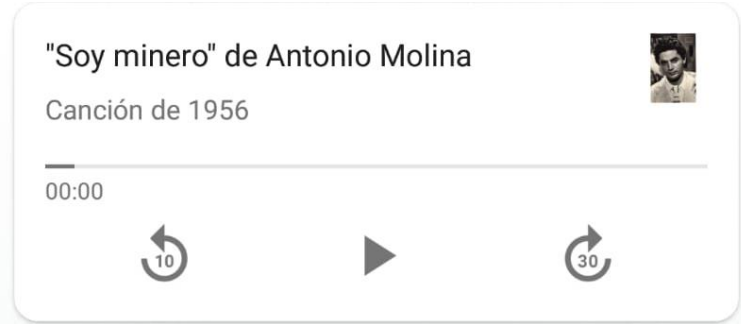

Ilustración 35 Ejemplo del reproductor multimedia del *Fulfillment*.

Esta canción abre un reproductor dentro de la propia conversación que contiene además una imagen del autor y una breve descripción de esta como vemos en la Ilustración 35 . También propone una *suggestion chip* que propone poner otra de las canciones almacenadas.

4.2.8.2.- Busca.

El funcionamiento difiere completamente del apartado anterior en el sentido de que no es necesario programar y descargar cada canción en su base de datos sino que el agente percibirá la parte de la frase en la que el usuario se refiera al autor o a la canción y realizará una búsqueda en YouTube a la cual el usuario puede acceder solamente pulsando un botón.

Para ello se hará de manera similar al *intent* de juegos visto en el apartado 4.2.5 Juego. Se hará uso de una tarjeta con un botón programable. Tal y como se ve en la Ilustración 36 Búsqueda de canciones. Y finalmente se abrirá la página web deseada, es la mejor manera para permitir al usuario reproducir cualquier canción que desee.

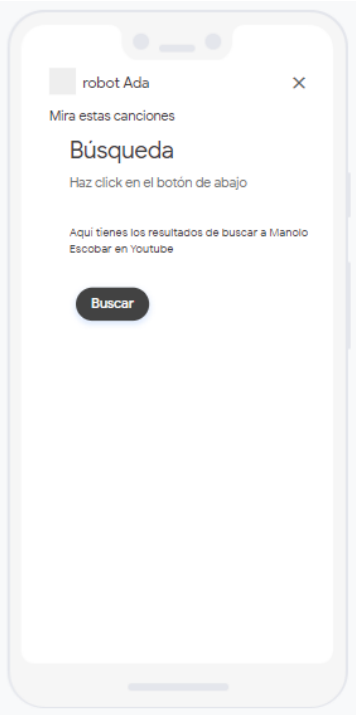

Ilustración 36 Búsqueda de canciones.

4.2.8.3.- Cantante favorito y búsqueda del cantante favorito.

Cantante favorito es un *intent* englobado en el apartado música que difiere en gran medida de los anteriores ya que no reproduce nada. Toma el dato de quién es el cantante favorito del usuario y lo almacena en la base de datos para que le de uso el siguiente *intent*. Existe una entidad con nombres de cantantes pero se ha preferido no hacer uso de ella ya que los cantantes de

las personas mayores no están contemplados por lo que reconoce cualquier cosa que le digan como un cantante nuevo. El *intent* de búsqueda es igual que el del punto anterior pero en vez de buscar lo que le digas siempre hace la búsqueda del cantante almacenado, aunque haya pasado tiempo desde que se estableció.

4.2.9.- Búsqueda.

Tal y como se muestra en la Ilustración 36 Búsqueda de canciones. Se utilizará el mismo mecanismo para hacer consultas tanto a Google como a Wikipedia, de manera que si existe la página concreta a lo que el usuario ha mencionado el botón priorizará un enlace a la página correspondiente de Wikipedia. Si no existe se realizará una búsqueda de las palabras necesarias para ofrecer al usuario lo que ha requerido.

Es un *intent* que puede acercar a los mayores a las posibilidades que da internet, desde poder buscar datos sobre su cantante favorito a buscar el tiempo en su municipio de origen o incluso consultar las noticias. Este se centra en el objetivo de poder realizar una gran cantidad de tareas con el mismo agente, dando así un motivo para volver a conversar con él. El *intent* tiene una gran escalabilidad ya que con esta base sería posible realizar una enorme cantidad de tareas que se pueden programar muy fácilmente a medida de la comunidad donde se incorpore el agente conversacional.

4.2.10.- Ejercicio.

Una de las prioridades del agente es que las personas mayores encuentren útil el uso del mismo. Por lo que ofrecer al usuario diferentes tipos de ejercicio según sus necesidades y capacidades físicas es prioritario. Al usuario se le presentarán tres tipos de ejercicio y, a diferencia que en los dos puntos anteriores, se abrirá directamente el enlace del vídeo y no la búsqueda del mismo en YouTube.

De la manera en que se muestra en la Ilustración 37 el usuario puede deslizar por las diferentes opciones leyendo una pequeña descripción para después poder pulsar sobre la opción que él decida. Las imágenes estarán tomadas de la base de datos que se explica en el apartado 3.6.- Bases de datos. Han sido

previamente descargadas de internet y almacenadas en la base de datos de Firestore para su posterior uso.

Ilustración 37 Ejemplo de *browsing carrousel.*

4.2.11.- Encuesta.

Tras varias interacciones con el agente al usuario se le predirá una valoración si lo desea, el *Fulfillment* comprobará que sea un número entre el 1 y el 5 y en caso afirmativo lo guardará en la base de datos para una posterior consulta de los desarrolladores. Lo cual facilitará hacer mejoras al agente y si es necesario realizar variaciones del mismo al observar unas calificaciones bajas. También puede ser útil para la atención al cliente porque las bajas notas pueden significar en algunos casos malos funcionamientos del agente que se podrían subsanar.

5.- Implantación del sistema.

A continuación, se expondrán todos los detalles que se han tenido en cuenta al programar el sistema conversacional de ayuda a personas mayores.

5.1.- Creación del *Chatbot*.

Se realizará en la página web https://Dialogflow.cloud.Google.com/ donde se nos pedirá que ingresemos nuestro usuario y contraseña de una cuenta de Google. Tras aceptar los términos de servicio se nos redirigirá a una pantalla que propone crear el primer agente y eso haremos.

Para crear el agente será necesario introducir la franja horaria en la que se utilizará mayoritariamente el agente, así como un nombre para el mismo y su idioma. Después de esperar a que se configure y lance el agente tendríamos algo parecido a lo que se observó en la Ilustración 30 donde solamente tendremos los *intent* de *welcome* y de *fallback*.

5.2.- Ajustes del agente.

Para poder cambiar los ajustes del agente se tendrá que pulsar en el engranaje que está al lado de su nombre y se obtendrá un menú con varias pestañas que serán descritas a continuación. Solamente se definirán los

ajustes que se consideran necesarios para realizar una agente como el que se llevará a cabo.

5.2.1.- General.

Es aconsejable que si se tienen intenciones de lanzar el agente al mercado propongamos una descripción sobre las funciones que puede realizar este o su propósito. Además, en esta pestaña es donde podemos obtener el identificador del proyecto lo cual sería muy práctico si deseásemos implementarlo en algún dispositivo como una *raspberry pi*. También nos deberíamos de asegurar de permitir al agente el uso *de APIs* e interacciones tanto con Dialogflow como con Google cloud.

5.2.2.- Languages.

Será posible seleccionar varios lenguajes en los que trabajará el agente y también sería recomendable seleccionar el idioma predeterminado del mismo en la misma pestaña.

5.2.3.- Speech.

Por defecto la voz que Google permite utilizar es algo diferente a la voz que suelen tener los dispositivos con Google assistant. La voz puede resultar algo lenta y monótona.

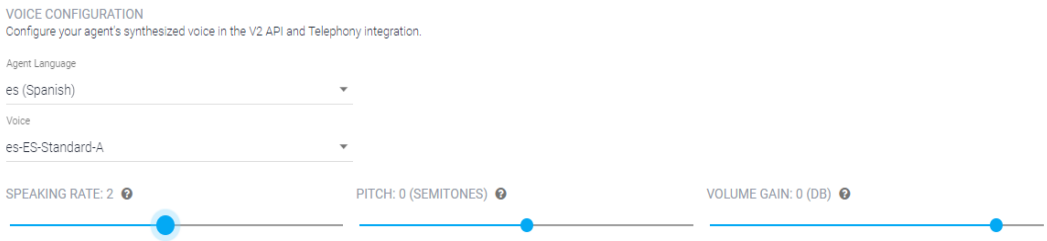

Ilustración 38 Ajustes del habla.

Como se observa en la Ilustración 38 se pueden seleccionar la ganancia, el tono y la velocidad del text to speech que hablará por el agente, es probable

que en el futuro introduzcan más tipos de voz para el castellano al igual que lo hicieron con el inglés.

En el agente que se desarrollará se han escogido unos valores de velocidad superiores a los de por defecto (valor 1), ya que es apreciable que el agente habla demasiado lento y el interlocutor es posible que se impaciente y quiera intervenir antes de que termine una frase. Por consiguiente, no se capturaría el mensaje completo y resultaría en una mala experiencia para el usuario.

5.2.3.- Share.

Aquí se podrán dar los permisos necesarios para un desarrollo de la aplicación por parte de varias personas o para permitir acceso de solo lectura a *beta testers* que ayuden con el *debugging* del agente.

Para ello se seleccionará una cuenta de Google y se elegirá si su permiso es de desarrollador o revisor (*Developer* o *Reviewer* según la página web de Google).

5.2.4.- Export and Import.

Es una de las pestañas más esenciales puesto que al exportar el agente obtendremos un archivo comprimido del agente, que se podrá importar en el futuro para recuperarlo.

Es un sistema tanto de copias de seguridad en el almacenamiento interno del ordenador, como de traspaso de diferentes agentes entre desarrolladores para poder compartir funcionalidades.

5.3.- Creación, configuración y manipulación de la base de datos.

El manejo de la base de datos es una de las mejores funcionalidades que incorpora Dialogflow ya sea por la posibilidad de ofrecer un servicio más personalizado o por la capacidad de almacenamiento de variables que pueden llegar a ser diferentes servicios ofertados.

La incorporación al agente de la base de datos se hará por medio de la pestaña de entregas (*fulfillment*) donde se habilitará el editor de código. En el punto 5.3.- Creación, configuración y manipulación de la base de datos. se discutirá sobre cómo poner en marcha el *fulfillment* así que por ahora nos fijaremos en el enlace que aparece justo debajo.

5.3.1.- Visualización de la base de datos.

Se abrirá una pestaña donde veremos los posibles errores de compilación del agente y pulsaremos en el menú de navegación de la esquina superior izquierda hasta abrir la opción Firestore datos. Se abrirá algo parecido a lo que se ve en la Ilustración 23 y desde ahí se podrá editar.

No es necesario que creemos una colección o documento ya que al añadir uno nuevo desde Dialogflow ya se crean ambas cosas. En los siguientes puntos se indicará como crear o actualizar colecciones desde el *fulfillment*.

5.3.2.- Declaraciones en la cabecera.

Lo primero es incluir en la cabecera las siguientes librerías y variables en index.js.

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

Ilustración 39 Declaración de cabecera

A demás después de darle al botón de *deploy* y que compile el agente se verá que en el archivo de package.json aparecen en *dependencies* con su versión correspondiente actual.

5.3.3.- Escribir en la base de datos.

Para escribir en la base de datos necesitaremos los nombres de la colección en la que queramos escribir, el documento, el campo y el contenido del último.

```
const dialogflowAgentRef = db.collection('dialogflow').doc(conv.data.nombre);
```

Conociendo los nombres que queremos asignar para la colección y el documento se escribirán dentro del paréntesis correspondiente en la primera línea. Hay que tener en cuenta que ambos campos tienen que ser una *string* así que si se desea utilizar directamente una variable que hemos extraído de una respuesta del usuario deberías realizar el siguiente tratamiento antes.

```
var nombre = String(name) ;
conv.data.nombre = nombre;
```

Aparte lo que hace el código es introducirlo en una variable temporal.

Después se debería escribir lo siguiente exponiendo entre los corchetes primero el nombre del campo que deseemos crear o rellenar seguido de la variable que deseemos almacenar dentro, puede ser directamente el valor o la *string* que queramos.

```
return db.runTransaction(t => {
  t.set(dialogflowAgentRef, {apellido: apellidos});
```

Se habrá realizado la petición, pero esta es asíncrona porque hace una petición al servidor y recibe una respuesta. Para poder continuar con la ejecución del código debemos incluir la línea siguiente.

```
return Promise.resolve('Write complete');
```

Con lo que concluiremos la escritura en la base de datos. Existe una limitación y es que como utiliza la palabra clave *return* no es posible seguir realizando funciones dentro del mismo *intent* por lo que lo mejor es siempre dejarlo para el final del mismo o el código a ejecutar entre las líneas de escritura. En el caso concreto de lo que se desea realizar se ha introducido el apellido (o el nombre si quisiéramos).

5.3.4.- Leer la base de datos.

A demás de poder leer será crucial poder guardar en una variable la información leída ya que como se dijo anteriormente no se podrá continuar la programación después de interactuar con la base de datos dentro de un *intent.* La primera línea es la misma que se observó en la escritura:

```
const dialogflowAgentRef = db.collection('dialogflow').doc(conv.data.nombre);
```

Después tendremos dos casos, que el documento exista y que no sea así. Primero abrimos el documento e iniciamos la consulta con el siguiente código.

```
return dialogflowNameDoc.get()
    .then(doc => {
```

Ahora se deberían contemplar las dos posibilidades entonces formularemos un *if* con el caso de que no exista y un *else* para cuando exista el documento a consultar.

```
if (!doc.exists) {
  conv.ask(`No encuentro tu nombre en la base de datos, si quieres puedo registrarte` +
   `, para eso necesito tu apellido, o si lo prefieres dime que no quieres registrarte.`);
  conv.contexts.set('newUser', 2, parameters);
}
```

Una vez dentro de la base de datos para la hipótesis donde no exista la entrada el agente emitirá un mensaje y activará un contexto que es lo que se ve en el código de arriba.

En el caso concreto del agente que se realizará durante el proyecto, esta hipótesis será la que defina si el usuario no está registrado con anterioridad quiere crear una entrada en la base de datos para que el robot le recuerde o por el contrario desea entrar como invitado.

A continuación, se verá el caso de que sí encontrase el registro.

```
else {
        conv.ask(`¿Entonces te llamas ` + conv.data.nombre +` `+doc.data().apellido + `?`);
        conv.contexts.set('login', 2, parameters);
    }
```

Donde el dato que hemos consultado se llama como doc.data().nombre_del_campo, en el caso concreto que se utilizará en el *Chatbot* se pedirá al usuario una comprobación de su nombre y apellidos para saber si nuestro usuario es el mismo que se registró con anterioridad u otro usuario.

Para finalizar y de manera similar a lo ya explicado en el anterior apartado será necesario finalizar la promesa para que siga el código su curso después de la interacción asíncrona.

```
return Promise.resolve('Read complete');
}).catch(() => {
  conv.ask(`Error consultando la entrada, por favor inténtalo mas tarde`);
```

Implementando además, un mensaje de error para comunicárselo al usuario en el caso en el que no haya sido posible realizar la comunicación con el servidor.

5.3.5.- Guardar y reproducir imágenes y sonidos.

Para almacenar imágenes y audios en una base necesitaremos entrar en la página web https://console.firebase.Google.com/ y allí se seleccionará el agente del cual se desea crear un almacenamiento en la nube. Una vez allí se entraría en el apartado almacenamiento y se crearía así el almacenamiento en la nube.

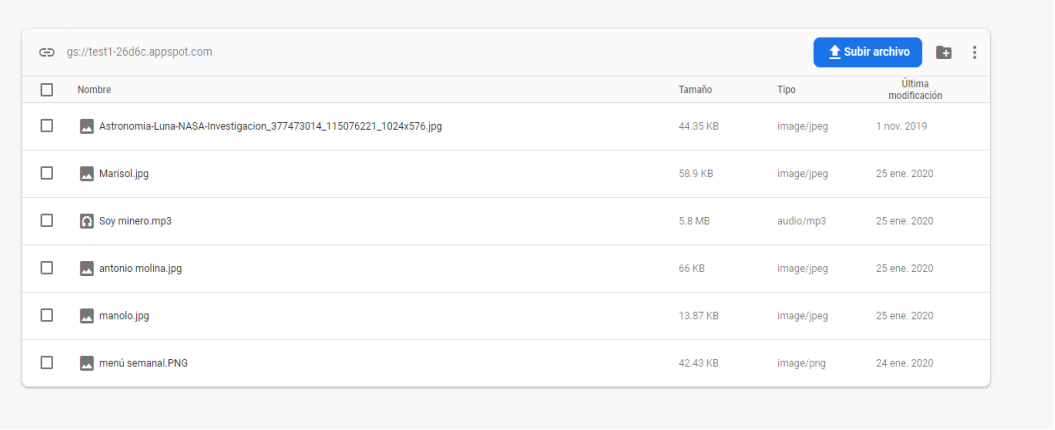

Ilustración 40 Ejemplo de almacenamiento

Para agregar un nuevo archivo simplemente se debe pulsar el botón que se ve en la Ilustración 40 donde pone subir archivo.

Para utilizar dichos archivos, en el *Fulfillment* se debería escribir un objeto como el que veremos en el siguiente ejemplo:

```
conv.ask(new MediaObject({
  name: '"Soy minero" de Antonio Molina',
  url: 'https://firebasestorage.googleapis.com/v0/b/test1-26d6c.appspot.com/o/Soy%20minero.mp3?al
  description: 'Canción de 1956',
  icon: new Image({
    url: 'https://firebasestorage.googleapis.com/v0/b/test1-26d6c.appspot.com/o/antonio%20molina.
    alt: 'una Foto de Antiono Molina',
  }),
}));
```

Donde las *url* se obtienen pulsando en el nombre del archivo en Firebase. El resultado final del código es lo que se observó en la Ilustración 35.

5.3.6.- Creación de cada *intent.*

Dentro del siguiente apartado se verán los pasos que se deben seguir para crear todos los diferentes *intents* que se implementarán en el resultado final. Los ejemplos de código se adjuntarán en un anexo pero los aspectos más importantes se detallarán en este apartado. Desde funciones concretas de cada *intent* hasta mecanismos programados en javascript para transformar las fechas a un formato legible por el *speech to text.*

5.3.6.1.- Adivinanzas.

El primer *intent* al que se llama en esta rama es "selección", tiene ese nombre porque es el *intent* que deja al usuario escoger entre una adivinanza fácil y una difícil. Tiene un contexto de salida llamado adivinanzas que dura cinco interacciones, se utiliza para saber que el usuario está escogiendo adivinanza y resolverla por lo que mejora el reconocimiento del *Chatbot*.

Las frases que se utilizan son diferentes maneras de decir "quiero que me cuentes una adivinanza" y como texto de salida se tendrá la pregunta que dirige a una de las dos ramas consiguientes: fácil o difícil.

Ambas funcionan de la misma manera lo único que cambia es el acertijo y su solución.

Por lo tanto los *intents* llamados como "adivinanzas.fácil" y "adivinanzas.difícil" funcionan de la misma manera.

Tienen un contexto de entrada llamado adivinanzas que es el que hablamos antes y otro de salida llamados fácil o difícil respectivamente. Como texto de salida tienen la adivinanza y tres burbujas con las opciones sugeridas.De este punto surgen cuatro posibilidades, acierto o fallo para cada una de las anteriores.

Todas tienen como entrada el contexto anterior de fácil o difícil y sus palabras de invocación son las posibles respuestas ofrecidas en las burbujas. No tienen ningún texto pero el motivo es que las cuatro tienen una función correspondiente en el *Fulfillment* por lo que debemos ir al último apartado del *intent* y habilitar el webhook para que el *Fulfillment* lo reconozca.

```
//respuesta incorrecta fácil
app.intent('adivinanzas.solución.fácil/incorrecto(1.0)', (conv) => {
  conv.ask('<speak>'+
    '<audio src="https://actions.google.com/sounds/v1/cartoon/cartoon_boing.ogg"></audio> '+
    '¡Respuesta incorrecta!'+
    'La respuesta correcta era caracol, también puedes pedirme que te cuente un chiste.'+
      '</speak>');
  }
);
```

Al reconocer el *intent* se reproduce un sonido de una campana o una bocina dependiendo de si era correcto o incorrecto y se dice un mensaje después.

5.3.6.2.-Bienvenida.

El primero de la rama es el default *welcome intent* que ha sido renombrado como bienvenida.inicio. El *intent* tiene habilitado la llamada al webhook donde se le preguntará por su nombre y se llamará a una función que se encarga de saludar al usuario de formas distintas si es por la mañana tarde o noche. Procederemos a analizar la siguiente función.

```
function saludo(conv){
  var texto = `  \n `;
  if (8<=horaActual() && horaActual()<=14) texto = texto +  `¡Buenos días! `;
  else if (15<=horaActual() && horaActual()<=21) texto = texto + `¡Buenas tardes! `;
  else texto = texto + `¡Buenas noches! `;
  return texto;
}
```

La función simplemente toma la hora de otra función y decide como día entre las 8 y las 14, como tarde entre las 15 y las 21 y como noche el resto de horas. Para saber la hora actual llama a otra función que necesita una declaración en la cabecera:

```
//variables horas
const timeZone = 'CEST CENTRAL EUROPE SUMMER TIME';
const timeZoneOffset = '+02:00';
```

Se define la franja horaria del trabajo del robot para así utilizarlo en la librería que encuentra las horas. Así como tratar las horas para después tratarla.

```
function horaActual () {
  // Get current datetime with proper timezone
  let date = new Date();
  date.setHours(date.getHours() + parseInt(timeZoneOffset.split(':')[0]));
  date.setMinutes(date.getMinutes() + parseInt(timeZoneOffset.split(':')[0][0] + timeZoneOffset.split(':')[1]));

  return date.getHours() ;
}
```

Básicamente lo que realiza la función es tomar la fecha en ese preciso instante y separar la hora para devolverlo a la anterior función. Recordemos que después de saludar al usuario el *Chatbot* le ha pedido su nombre lo que le lleva al *intent* llamado como bienvenida.login.

Que haciendo el tratamiento como se discutió anteriormente en el apartado 5.3.4.- Leer la base de datos., para saber su nombre y si deseara registrarse como un usuario nuevo se escribiría en la base con lo visto en el 5.3.3.- Escribir en la base de datos.

Para darle una mayor claridad al funcionamiento del *login* se ha desarrollado otro diagrama parecido al de la Ilustración 33 pero con el nombre de cada *intent* en azul y sus contextos en morado. Entrando en detalle y explicando la Ilustración 41 el *intent* de *login* tiene dos *intent* de seguimiento que ya se explicó su utilidad en el apartado 3.3.5- *Intents* de seguimiento., resultando así sus dos contextos de salida. El *intent* de seguimiento sí tiene como entrada tanto frases afirmativas como frases donde se diga, “ese es mi nombre” de diversas maneras, no tiene una llamada de *webhook* porque ya dice una frase introductoria al resto de ramas.

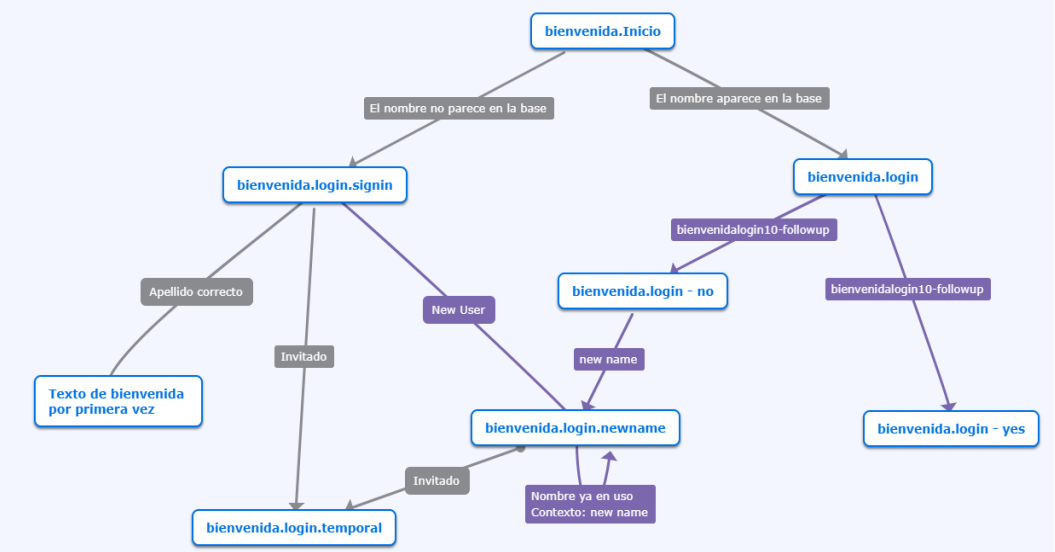

Ilustración 41 Mapa de *intents* de login

Mientras que la respuesta de no le pide al usuario un nuevo nombre introduciendo el *intent* llamado como “bienvenida.login.newname” que tiene un contexto de entrada y se activa al decir un nombre. Después tendremos la rama denominada como *sign in* que hará el mismo tratamiento que hablamos

antes para registrar un nuevo usuario o dar paso a un invitado por medio del temporal.

Tiene habilitado el *fulfillment* obviamente porque registra datos en la base. Además todos los *intents* que piden un dato al usuario para almacenarlo deben declararlo como parámetro y meterlo en una variable, eso se realiza dentro del *intent* en el apartado de parámetros. Al iniciar sesión las veces sucesivas el agente examina los parámetros almacenados del usuario en el siguiente orden:

- Fechas de cumpleaños: Si la fecha de hoy coincide con la fecha del cumpleaños del usuario se emite un sonido de celebración y se le felicita, además esto se realiza por cada cumpleaños almacenado por lo tanto si es el cumpleaños de varias personas emite mensajes recordando todos los cumpleaños al usuario.
- Visitas hoy: tras consultar la fecha actual con la última fecha de inicio de sesión se actualiza el número de visitas hoy y se responde en consecuencia de manera diferente si se ha hablado ya hoy con él o si hoy ya ha habido más de cinco conversaciones.
- Visitas totales: está programado para tener una introducción definida para las primeras nueve visitas a partir de la cual emitirá un mensaje genérico.

5.3.6.3.-Charla.

Son *intents* realmente sencillos formados por unas frases de entrada y una salida de texto. Ya tanto respondiendo con su nombre o cumpleaños como uno de sus tres estados de ánimo seleccionados aleatoriamente programados en el *intent* de charla. También tiene en cuenta las palabras de agradecimiento para elaborar su respuesta.

Por otra parte hay dos *intents* dentro de lo denominado charla que acceden a la hora actual y utilizan las funciones que daban uso a otros *intents* para informar al usuario del día y la hora actuales cuando este lo requiera.

5.3.6.4.- Juegos.

Es un *intent* que tiene en cuenta las capacidades del dispositivo que se está utilizando y muestra un enlace con el juego deseado para que el usuario pueda pulsarlo. Se han definido los diferentes juegos en una única entidad,

de esta manera es posible usar solamente un *intent* para los cuatro juegos programados. El enlace se generará tomando el juego deseado y añadiéndolo a una cadena de texto que conformará el enlace como se explicará en el apartado de búsqueda.

5.3.6.5.- Medicinas y médico.

El *intent* de pedir cita para el médico abre un enlace a la página web de salud de Castilla y León para que el usuario introduzca sus datos. El enlace se sitúa en una burbuja en la parte inferior de la pantalla para facilitar al usuario su acceso, sin ninguna tarjeta definida pero sí con una sugerencia por voz para acceder a las medicinas. Mientras tanto el *intent* de las medicinas tiene el *Fulfillment* habilitado y crea un objeto de tabla donde introduce las medicinas que debe tomar cada día el usuario separado por mañana tarde y noche como se observó en la Ilustración 21.

5.3.6.6.- Búsqueda.

En este apartado coexisten dos *intents* prácticamente iguales, uno de ellos genera una búsqueda en Google mientras que el otro hace lo mismo en Wikipedia. Para realizarlo primero se tuvieron en cuenta cómo funcionan las búsquedas desde la url.

Google tiene una gran cantidad de parámetros unidos por el símbolo “&” de los cuales solamente se utilizará uno “lr=lang_es” que indica al buscador resaltar las páginas que estén en castellano. Mientras que la búsqueda de Wikipedia es más simple ya que solamente hay que añadir las palabras que se desean buscar y, si existe la página solicitada, la expone directamente.

Se utilizará una tarjeta simple como la observada en la Ilustración 36 mediante el código a continuación, siendo posible la edición de cada uno de los diferentes campos para hacer la interfaz más personalizada.

```
conv.ask(`Este es el resultado`);
  conv.ask(new BasicCard({
    text: `Aquí tienes los resultados de buscar ` + busqueda + ` en Google`,
    title: 'Búsqueda',
    buttons: new Button({
      title: 'Buscar',
      url: 'https://www.google.com/search?q=' + busqueda + '&lr=lang_es',
    }),
```

5.3.6.7.- Música.

Este apartado contiene cuatro *intents* donde tres de ellos comparten el principio de funcionamiento con otros ya explicados anteriormente. El primero consta de una entrada a la base de datos donde guarda al cantante favorito del usuario. Ya se ha explicado cómo se hace en el apartado 5.3.3.- Escribir en la base de datos. Después se utilizaría dicho dato para hacer una búsqueda a YouTube de una manera prácticamente idéntica a cómo se relató en el apartado anterior, al igual que el *intent* de búsqueda donde el usuario expresa una canción o cantante y se le proporciona un botón que al presionarlo redirige a la búsqueda.

El último de los *intents* de este apartado toma una canción aleatoria de las cuatro que se guardaron previamente en la base de datos y con un string se asegura que no se repita ninguna canción hasta que no se hayan emitido todas ellas, para ello se utilizará la función shuffle que crea de forma aleatoria el orden de las canciones y estas se van descartando del *string* son el comando *pop()* según hayan sido escuchadas.

5.3.6.8.- Chistes.

De la misma manera en la que se ha programado el *intent* anterior para no repetir canciones se ha realizado para los chistes. También se han programado las reacciones a dichos chistes con dos *intents* que continúan la conversación tras decir que un chiste era bueno o malo.

5.3.6.9.- Encuesta.

Este *intent* tiene un contexto de entrada que se activa tras visitar 6 veces al agente en el momento en que el usuario inicia sesión. Se ha creado una colección donde se recopilan los resultados de la encuesta en una base de

datos separados por usuarios. Si el usuario introduce un número que no sea del 1 al 5 se lo vuelve a preguntas hasta que cumple con la condición para subirlo.

5.3.6.10.- Ejercicio.

Previo a la realización del *intent* se han realizado varias tareas, donde se ha subido las imágenes y un audio al almacenamiento de Firestore. Lo primero que se realiza es una comprobación del dispositivo en el que se está utilizando el agente y podríamos obtener tres casos:

- Un dispositivo con navegador web y pantalla: En este caso se podría mostrar al usuario el carrusel que se mostró en la Ilustración 37 mediante el siguiente código y personalizando cada campo que sea necesario

```
conv.ask(new BrowseCarousel({
  items: [
    new BrowseCarouselItem({
      title: 'Gimnasia general',
      url: 'https://www.youtube.com/watch?v=DYOTthcWj8g&list=PLE8t6HidiewFCtrcD-WUG4zYf4-Nn5cdG&index
      description: 'Ejercicios generales de gimnasia para mayores',
      image: new Image({
        url: 'https://firebasestorage.googleapis.com/v0/b/test1-26d6c.appspot.com/o/general%20icono.
      }),
```

- Un dispositivo con audio: El agente pasará a reproducir el audio del vídeo que previamente habíamos descargado aunque le advierte que para la próxima vez sería más completo y eficaz ver el vídeo en un dispositivo con pantalla.

- Un dispositivo de texto: En el caso de que solo emita texto el agente le comunicará al usuario que no puede acceder a la función desde el dispositivo actual.

6.- Mejora continua.

El hacer un programa durante tanto tiempo da la oportunidad de probar muchas cosas y no todas funcionan por lo que quería dejar constancia de cuáles son las mejores prácticas a la hora de realizar un agente conversacional como el realizado.

6.1.- Nombres compuestos.

Para realizar el *login* se le pide al usuario su nombre, y al no tener los nombres compuestos en cuenta lo primero que sucede es que Dialogflow solamente trata el primero de los nombres.

Por ello en la definición del *intent* dentro del apartado de acciones y parámetros es importante que definamos tanto los nombres como los apellidos como lista lo que hace que introduzca sus nombres dentro de un vector de *strings*.

Pero eso provoca otro problema, si un usuario se llama “Victor Manuel” como es un vector a la hora de imprimirlo por pantalla o en la base lo hará con una coma entre medias, sería algo como “Victor, Manuel”. Para eso se ha desarrollado un código donde se añaden a un *string* todos los nombres separados por un espacio.

```
 var numero = name.length;
 var nombre = String(name[0]) ;
if (numero>0)
{
  for(var i=1; i<numero; i++)
      {
          nombre = nombre + ` ` + name[i];
      }
}
conv.data.nombre = nombre;
```

Con ese código recorre el *string* solucionando el problema de formato que se experimentaba con los nombres.

6.2.- Almacenamiento temporal y permanente en Dialogflow.

Dialogflow permite tres maneras de almacenar datos dentro de su programa. La primera es dentro de un contexto, pero dado que entorpecería un poco el flujo del programa para datos que necesiten ser recordados más tiempo no es lo más óptimo.

La segunda es un dato temporal almacenado en un vector llamado conv.data que es muy útil y ha sido utilizado en varias ocasiones en el programa, pero no es un almacenamiento permanente, el vector se borra en cuanto se termina la conversación. Y la tercera es igual que el anterior, pero es permanente. El problema es que es único para cada cuenta de Google por lo que si varios usuarios tienen la misma cuenta de Google compartirían dicho vector.

Esto supone un problema ya que se ha supuesto que el funcionamiento del agente fundamentalmente sería en una residencia de ancianos y no es nada orgánico el hecho de que cada persona mayor deba recordar su cuenta y contraseña de Google.

6.3.- *Login* y privacidad.

Inicialmente se quería utilizar como nombre el que proporcione el usuario y como contraseña de verificación al acceso su número de teléfono. El problema surge cuando dos personas se llaman igual y el *Chatbot*

proporciona al nuevo usuario el número de teléfono del anterior vulnerando gravemente su privacidad. Para solucionarlo se decidió cambiarlo por el apellido del usuario.

6.4.- Reconocimiento de nombres y apellidos.

El reconocimiento de nombres no ha dado ningún problema con las pruebas realizadas pero los apellidos sí. Personalmente mi apellido "Gobernado" no estaba reconocido en la base de datos de Google como un apellido y por lo tanto no reconocía que la verificación que le aportábamos era un apellido válido. Se puede solucionar fácilmente accediendo al apartado de entidades del agente.

Una vez allí se selecciona la entidad de sistema "*last-name*" y se añaden los apellidos que se deseen. Por ejemplo, también se agregaron los apellidos de Hernández o Herráez.

6.5.- Reconocimiento negativo.

Como se vio en la sección 4.1.3.2.- Default Fallback Intent. Cuando el agente no es capaz de relacionar lo que le han dicho con sus frases de entrenamiento interviene el "*Default Fallback Intent"* Pero una buena práctica sería entrenar a este *intent* con las frases o palabras que no queremos que entienda el agente, como por ejemplo insultos. Es muy común la situación de una persona mayor que no entiende cómo funciona el agente y decide insultarle por lo que es muy conveniente que nuestro agente sepa que a esas palabras no debe contestar.

6.6.- Repetición de *intents*.

A lo largo de una conversación es natural que si quieres repetir una parte de la conversación incluyas las palabras "otro/a" u "otros/as". Así que para poder detectar esos casos se deberían incluir en las frases de entrenamiento de casi todos los *intents* como por ejemplo "ponme otra canción" o "cuéntame otro chiste".

De esa manera nos aseguramos de que si nuestro usuario decide tener una conversación más larga de lo habitual lo haga de una manera mucho más orgánica e intuitiva que si solamente puede utilizar cierto número de palabras.

6.7.- Pestaña de entrenamiento.

En la columna izquierda del menú de Dialogflow se nos ofrece la pestaña de entrenamiento o *training* donde una vez probado el agente es posible analizar lo que han dicho los usuarios y utilizarlo para mejorar así el agente.

Se podrán ver por orden cronológico que ha detectado el agente y que *intent* ha invocado. Si se quisiera se podría indicar otro *intent* para indicar al agente que esas palabras no corresponden con ese *intent* y tienen mucha más relación con el que se le ha indicado.

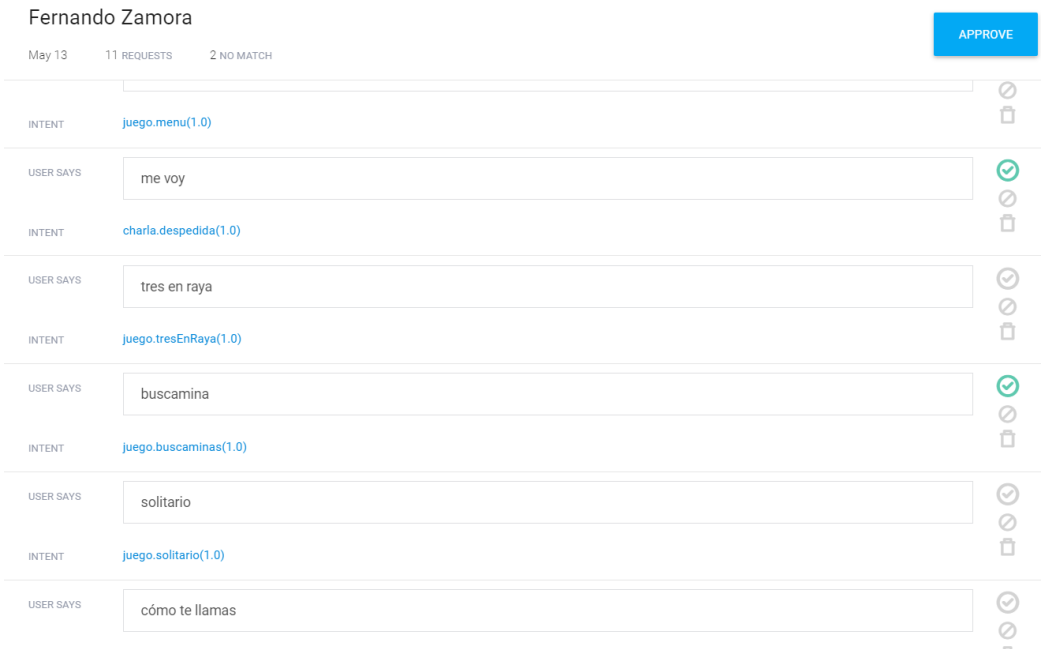

Ilustración 42 Pestaña de entrenamiento.

También es posible validar las relaciones que ha hecho y negarlas pulsando en los botones del lateral derecho que se muestran en la Ilustración 42. Es una potente herramienta de análisis que permite hacer el proceso de mejora continua que buscamos. Al dar acceso a más usuarios se podrán solucionar los errores a tiempo real y hará que en el futuro el agente sea cada vez más robusto.

6.8.- Reacciones de los usuarios.

Tras realizar más de diez pruebas se observó que una amplia mayoría de los usuarios quería dar una retroalimentación al agente conversacional, diciendo frases como “No me ha gustado ese chiste” o “Que chiste más bueno”. Por lo tanto, se decidió dar cabida a dichas frases.

En el contexto de los chistes si dicen alguna frase parecida a las expuestas anteriormente el agente aceptará la crítica o agradecerá los elogios y propondrá al usuario pasar a otro de los temas de conversación. También se han decidido añadir *intents* a la rama llamada Charla, ya que en varias ocasiones se le ha preguntado el nombre al agente o le han pedido que digan su nombre para ver si lo ha aprendido bien. Así como a las frases de agradecimiento o curiosidades sobre el agente como, por ejemplo, su cumpleaños.

7.- Resultados.

Para analizar los resultados de una manera objetiva es necesario buscar alguna forma de cuantificar la calidad del agente. Para ello ha sido desarrollado formulario para que los voluntarios que se han ofrecido para probar el programa lo cumplimenten y recoger de una forma eficaz sus opiniones y mejoras. Después serán analizadas y valoradas. Esta encuesta ha sido realizada de manera que el usuario de la aplicación solamente se le ha comunicado que es un agente para personas mayores y cómo acceder a él por lo que no se les han dado indicaciones de cómo utilizarlo. De forma similar se cumplimentó la encuesta aclarando que era necesario su honesta opinión.

También se han realizado pruebas con personas mayores. Reaccionaron positivamente y tampoco se les indicó cómo funcionaba. Simplemente se les dijo que tenían que hablar con el agente y ellos mismos realizaron varias conversaciones satisfactoriamente. Aunque muchas de las actividades que les preguntaban no se implementarían en este trabajo (por ejemplo “enciende la televisión”) bien es cierto que comprendían que no se realizaran ciertas funciones y proseguían con la conversación normalmente.

7.1.- Preguntas de la encuesta.

Las preguntas se han elaborado con intención de dar una respuesta a los objetivos planteados, así como una valoración general del agente.

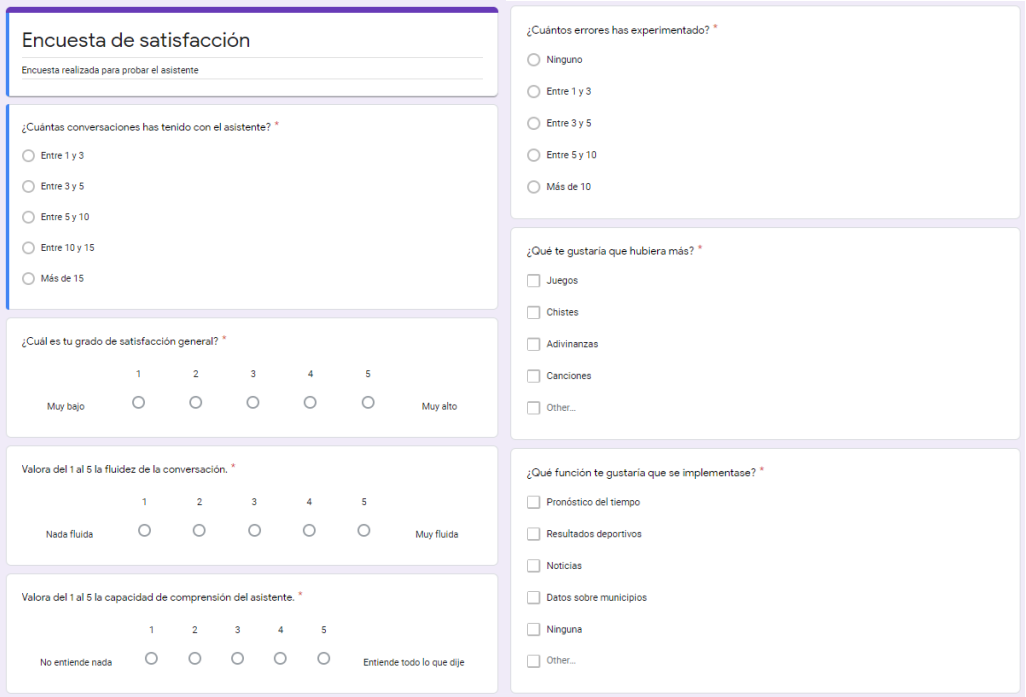

Ilustración 43 Contenido de la encuesta de satisfacción.

La primera pregunta tal y como se observa en la Ilustración 43 es de utilidad para saber la media de interacciones que tiene un usuario antes de decidir que ha completado todas las conversaciones o funciones. Después pide cuantificar la calidad del asistente con tres preguntas, una calificación general y otras dos más técnicas pidiendo valorar la fluidez y compresión. Las siguientes preguntas están enfocadas a la detección de errores y a fijar una línea de mejora.

Aparte de saber qué es lo que más requieren los usuarios que se implemente son sus propias ideas, que han sido recopiladas gracias a esta encuesta como la adición de una lista de la compra o de unos recordatorios a horas muy concretas para asegurarnos de que los mayores tomen su medicación correctamente.

7.2.- Análisis de las respuestas.

Cabe destacar que si bien la muestra es demasiado pequeña como para afirmar categóricamente la calidad del agente, con quince personas, bien puede indicar la tendencia general de la opinión de los usuarios que han tenido la oportunidad de probarlo.

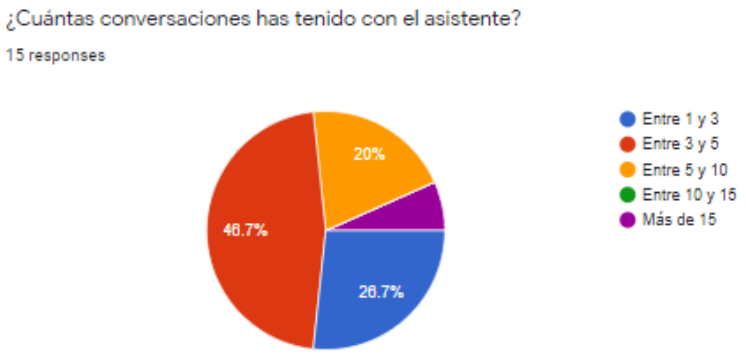

Ilustración 44 Número de conversaciones.

La primera de las preguntas tiene como objetivo analizar a partir de cuantas conversaciones los usuarios deciden que han explorado todas las funciones que se les ofrece. Como se observa en la Ilustración 44 la gran mayoría de los usuarios tienen entre 3 y 5 conversaciones. Lo que indica que por ahora puede que los usuarios consideren que el número de funciones no es demasiado alto. También es cierto que al ser un agente con funciones diferentes es posible que se desee conversar con él pero pocas veces al día aunque de forma regular.

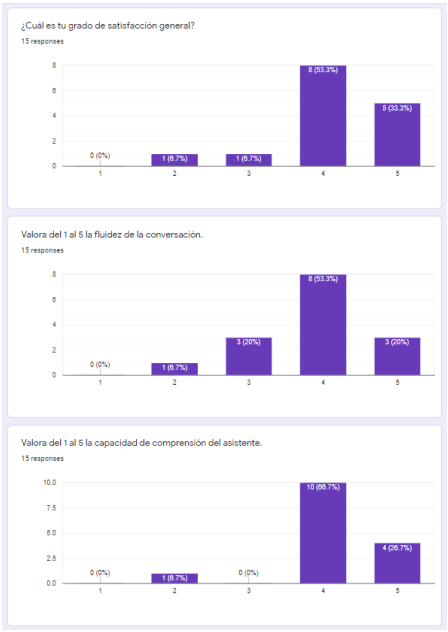

Ilustración 45 Preguntas sobre la calidad del agente.

Una buena manera de cuantificar la calidad del agente es que los propios usuarios puntúen varios aspectos del mismo. La primera de las preguntas que se observan en la Ilustración 45 es el grado de satisfacción general. Y en su gran mayoría los usuarios han emitido una respuesta positiva. Bien es cierto

que la fluidez de la conversación ha tenido unas puntuaciones más bajas que los otros dos aspectos y se puede achacar a la falta de frases de entrenamiento. Entre cada prueba de un usuario se ha realizado un entrenamiento como el observado en el punto 6.7.- Pestaña de entrenamiento. Para mejorar su fluidez y capacidad de comprensión.

Se han recibido unas muy buenas críticas a la capacidad de comprensión del agente, lo que indica que el usuario ha notado en pocas ocasiones que el agente no le estaba entendiendo.

Ilustración 46 Errores en el agente

Ante la pregunta de "¿Cuántos errores has experimentado?" rodos los usuarios han experimentado menos de cinco errores y el 93.3% (14 de 15) han experimentado menos de 3 errores. A demás al entrenar al agente entre los diferentes experimentos este ha mejorado poco a poco y se puede argumentar que al ser la segunda persona quien experimentó una mayor cantidad de fallos es bastante probable que se corrigieran tras los entrenamientos.

Es un buen dato de errores teniendo en cuenta que la categoría "errores" varía entre los usuarios ya que lo que para un usuario puede ser un error no entender lo que ha expresado a otro le puede parecer normal que no conteste a ciertas frases. Existe recorrido de mejora en dicho aspecto aunque se conseguiría con un mayor periodo de prueba y con un mayor público participando en la beta.

7.3.- Líneas de mejora.

Las dos últimas preguntas es necesario analizarlas en este apartado ya que indican qué funciones les gustaría que se implementasen en una versión futura. Aunque las más seleccionadas han sido dos que se proponían por nuestra parte (Ilustración 47). La necesidad de un pronóstico del tiempo por parte de los usuarios es bastante alta y se podría conseguir haciendo un Mega-Agente que compendie nuestro agente y algún otro con esa función incorporada. Actualmente el servicio de Mega-Agentes está en versión beta por lo que no es posible desarrollarlo correctamente. También fue muy comentando la necesidad de noticias tanto generales como de deportes, se podría implementar de la misma manera que la comentada anteriormente.

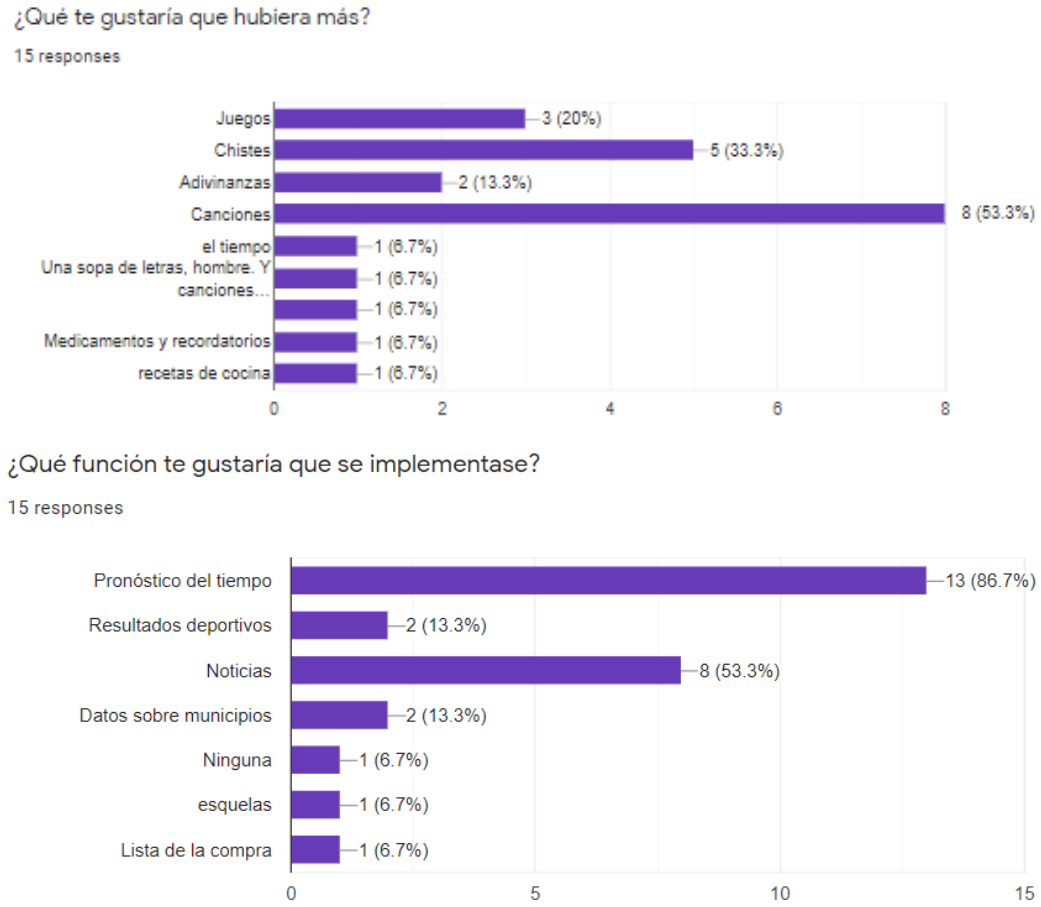

Ilustración 47 Encuesta de líneas futuras.

A su vez los usuarios consideraron que la cantidad de chistes y canciones podría ser escasa por lo que incrementó de dos canciones a cuatro y la posibilidad de realizar búsquedas en internet y de tres chistes aleatorios a seis chistes que no se repiten como se mostró en el apartado 4.2.4.- Chiste. También cabe destacar que el agente ha sido programado de tal manera que

si fuera necesario añadir más canciones o chistes en un futuro sería posible hacerlo sin tener que desarrollar código sino transcribiendo el ya desarrollado con anterioridad. No es el mismo caso que sucede con las adivinanzas ya que estas requieren de tres *intents* cada una por lo que puede ser un trabajo más tedioso.

También es destacable el servicio de telefonía comentado en la Ilustración 20 ya que podría dar un grado más de profundidad al agente. Este pasaría de ser accesible desde cualquier dispositivo con conexión a internet a poder ser utilizado también desde llamadas telefónicas por lo que podría llegar a un público mucho más amplio e incluso rompería la barrera psicología por la que muchos mayores se niegan a hacer uso de la tecnología. Al tomar un medio que ya conocen se podría acercar a dicho público al mundo de internet y rompería barreras entre generaciones al dotar de una herramienta tan potente como son las búsquedas o las noticias.

Como comentario final de la encuesta me gustaría destacar las ideas propuestas por los usuarios, desde otros juegos como sopas de letras hasta funciones de más utilidad como recordatorios para los medicamentos una lista de la compra o recetas de cocina. Son funciones de gran utilidad que pueden estar aquí descritas gracias a la prueba con usuarios.

7.4.- Resultado final.

Para finalizar solamente resta añadir que es un proyecto que siempre se podrá escalar y aumentar su número de funciones y frases de entrenamiento pero es perfectamente suficiente para poder implementar la solución al problema que se planteó al principio. Tomando en cuenta que las funciones que existen actualmente son de las más necesarias a la hora de crear un agente que ayude a combatir la soledad que enfrenta la tercera edad, modernizándola y dotando de herramientas con las que pueden acceder a internet y tener compañía aunque sea virtual.

El grado de satisfacción con el proyecto por parte de los usuarios que han podido probar el agente es lo suficientemente bueno como para poder afirmar que cumple con sus expectativas y que sería un producto que puede plantear una solución en el mercado actual.

8.- Estudio económico, tarifas y dimensionamiento.

En el presente capítulo se hablará de los costes que supondría una implementación similar a la elaborada durante este trabajo de fin de grado. Por el momento se había discutido la viabilidad técnica del proyecto explicando las soluciones escogidas para cada objetivo planteado mientras que en los siguientes apartados se discutirá los costes tanto de generar el agente como de mantenerlo en el tiempo.

8.1.- Recursos empleados.

Se deben tener en cuenta tanto los recursos hardware como software y solamente tener en cuenta la amortización de los recursos durante el periodo de tiempo utilizado al desarrollo del proyecto, para así ser capaces de calcular el coste real.

- Software:
  - Sistema operativo: Windows 10 Pro.

- Hardware:

  - Ordenador de sobremesa Medion MS-7856

- Servicios en línea:
  - Dialogflow
  - Firestore

- Material ofimático:
  - Libros de consulta
  - Otros consumibles

8.2.- Costes directos.

Se analizarán los costes directos como:

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

Después se añadirán todos estos costes directos a los indirectos que se calcularán en el último apartado y obtendremos el coste total del desarrollo y mantenimiento del agente conversacional.

8.2.1.-Costes de personal.

El proyecto ha sido realizado por un ingeniero, desde el estudio del problema hasta la puesta a punto y mantenimiento del programa.

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

Propondremos las siguientes estimaciones:

- Sueldo bruto e incentivos: 35.000 € al año
- Seguridad Social (el 35% del salario bruto): 12.250€ al año

Por lo que el coste total de un año es de **47.250€.**

Se considerará que una jornada de trabajo son 8 horas, las horas anuales son **1824**. Ya que hay 228 días en el calendario laboral. Por lo tanto el coste de

una hora de un ingeniero será aproximadamente **25,9 €/hora**, con lo que pasamos a hacer una estimación de las horas necesarias para hacer el proyecto.

- Formación y documentación: 200 horas.
- Estudio del problema: 150 horas.
- Estudio de las herramientas: 100 horas.
- Desarrollo de la aplicación: 350 horas.
- Puesta a punto y retroalimentación: 200 horas.
- Elaboración de la documentación: 200 horas.

Obtenemos un total de **1200 horas** de trabajo que multiplicado por el coste por hora de un ingeniero obtenemos un coste personal directo de **31.080€.**

8.2.2.- Costes de amortización de equipos y programas.

Para hacer estos cálculos primero debemos conocer la inversión total y calcular la amortización lineal correspondiente. Se deberán calcular tanto los costes de amortización de los materiales de oficina como del software y equipo.

Se estima que el tiempo de amortización de un ordenador será de 4 años ya que es la vida útil estimada del material, por lo tanto el factor de amortización de un solo año será ¼ que es un 25%.

- Sistema operativo Windows 10 Pro **260€** que son **65€** anuales.
- Ordenador de sobremesa **800€** que equivalen a un coste de **200€** por año.
- Microsoft Office **204€** al año ya que es una subscripción mensual.

Si hacemos la misma operación que anteriormente y se divide el total de costes de amortización entre las horas anuales obtenemos un coste de **0,257€/hora** que al contabilizar las horas de proyecto obtendríamos un coste de amortización de marial final de **308,4€.**

8.2.3.- Costes derivados de otros materiales.

Los siguientes costes pueden ser llamados consumibles e incluyen entre otras cosas cartuchos y papel de impresora libros de consulta, impresiones, almacenamientos USB, bolígrafos, etc.

Este tipo de material es necesario durante toda la realización del agente, aunque sobre todo en la recogida de documentación o conceptualización inicial del mismo.

Se estima que estos costes pueden ser de **100€.**

8.2.4.- Servicios en línea contratados.

Los servicios tanto de Dialogflow como de Firestore puede que dependiendo de la magnitud del proyecto sean recursos de pago por lo que en esta sección se hará una estimación sobre sus costes en el ámbito del agente que se ha desarrollado.

Dialogflow tiene unos márgenes a partir de los cuales se ha de cambiar de la versión estándar a la empresarial así que vamos a analizar dichos márgenes:

- Por texto: 180 solicitudes por minuto.
- Por voz: 100 solicitudes por minuto, 1.000 diarias y 15.000 mensuales.

Para saber exactamente cuántas solicitudes tuvo el agente en su desarrollo existe la pestaña de "*Analytics*" donde podremos ver unos gráficos como el de la Ilustración 48.

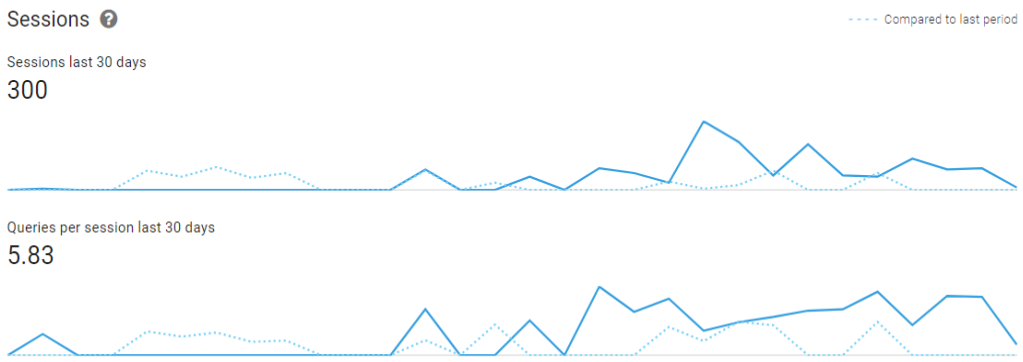

Ilustración 48 Solicitudes mensuales a Dialogflow.

Se observa en la Ilustración 48 el número de solicitudes por sesión de media son 5 y las solicitudes en nuestro caso no son simultáneas por lo que no deberíamos preocuparnos por el límite por minutos si suponemos una sola residencia de ancianos. Mensualmente si suponemos unas 1.500 sesiones (cinco veces más que las que se han hecho en el último mes de desarrollo) obtendríamos 9.000 consultas que se alejan de las 15.000 permitidas de forma gratuita. Por lo tanto Dialogflow a la escala a la que la estamos usando tiene un coste de **0€.**

Al tratarse del mismo desarrollador Firestore opera de manera similar, con una cuota gratuita a partir de la cual deja de ser un servicio gratuito. Que en este caso depende del volumen de datos almacenados o consultados. Firebase permite de manera gratuita 50.000 lecturas de documentos diarias y 20.000 escrituras y eliminaciones de documentos al día. Recordando que, como vimos en el capítulo 5.3.- Creación, configuración y manipulación de la base de datos. Cada usuario tiene un documento, por lo que Firebase permitiría al agente crear 20.000 usuarios eliminar otros 20.000 usuarios e iniciar sesión a 50.000 usuarios ya registrados todo en un día. Si suponemos una media de 200 habitantes en una residencia de ancianos es prácticamente imposible que se superen esos límites.

También permite un almacenamiento de archivos de hasta 1 Gb pero los únicos archivos almacenados son tres fotos y una pista de audio por lo que tampoco se sobrepasa ese límite. Por lo tanto el servicio en línea de Firebase a efectos prácticos es totalmente **gratuito** en este proyecto. Bien es cierto que dependería de la escalabilidad del proyecto, si se necesitara implementar en muchas residencias al mismo tiempo es posible que llegara a tener cargos por mantenimiento.

8.2.5.- Costes directos totales.

Tras calcular los apartados anteriores podemos concluir en que la suma de los costes de personal, amortización del material y los consumibles serán los costes directos totales que en este caso son de **31.488,40€.**

8.3.- Costes indirectos.

Son los costes producidos por la actividad requerida que no es posible incluir en ninguno de los gastos directos son los siguientes junto con sus estimaciones:

- Dirección y servicios administrativos: 150€.
- Consumo de electricidad: 180€.
- Consume de telefonía: 25€.
- Coste de desplazamiento: 200€.

Lo que supone un total de costes indirectos de **555€.**

8.4.- Costes totales.

Habiendo calculado todos los demás apartados del presupuesto se obtiene un coste total del proyecto de:

32.043,4€

9.- Conclusiones finales.

Para analizar la calidad final del proyecto sería conveniente retomar los objetivos propuestos al comienzo del mismo y qué acciones concretas han sido realizadas para alcanzarlos. También se comentarán los cambios que ha experimentado el proyecto a lo largo de su desarrollo con el fin de cumplir todos y cada uno de los objetivos. También se expondrá la opinión de los usuarios que han tenido acceso así como futuras líneas de mejora.

El primero de los objetivos era el de accesibilidad, es necesario enfatizar la importancia de este aspecto ya que las personas a las que se ha enfocado el proyecto les supone una dificultad interactuar con la tecnología. Por ello se decidió utilizar la plataforma que ofrece la empresa Google, ya que el usuario puede interactuar de varias maneras con el agente ya sea de manera oral, escrita o simplemente tocando mensajes predefinidos en una pantalla. También cabe destacar que no supondría un gran desembolso por parte de los usuarios ya que puede ser utilizado en cualquier dispositivo con conexión a internet.

El objetivo de diversidad queda alcanzado con las más de quince funciones diferentes que pueden promover que los mayores utilicen el agente, acercándose así al mundo digital. De igual manera promueve el mismo objetivo su capacidad de comprensión al reconocer varias frases de maneras muy distintas y no obligando al usuario a memorizar unos comandos que ejecuten una función u otra.

El objetivo de la personalización del agente se consigue tras la implementación de la base de datos. Al recordar el nombre del usuario o su

cantante favorito hace que a lo largo del tiempo, según el usuario haya compartido información con el agente, las conversaciones puedan ser más fluidas y específicas, pasando de mensajes genéricos a otros más personalizados.

Para el último de los objetivos, el de repetitividad, se ha implementado una solución similar a la del objetivo anterior. Una base de datos donde se almacenarán parámetros del uso del agente por parte de cada usuario, como por ejemplo, la última fecha en la que inició sesión, las veces que ha hecho uso del agente el día de hoy o el número total de los mismos. Haciendo uso de dichos parámetros el agente generará mensajes personalizados para cada usuario aportándole naturalidad y aleatoriedad, provocando que el agente tenga una mayor vida útil.

Sobre la opinión de los usuarios cabe resaltar la respuesta positiva que han tenido todos ellos, aportando ideas y soluciones al desarrollo del agente. Después de analizar su comportamiento se lograron generar respuestas más naturales y mejores funciones que sorprendieron al resto de los usuarios. Es fundamental que sean parte del desarrollo para satisfacer las diferentes necesidades de cada uno. Además se han obtenido altas calificaciones que podrían ayudar a la difusión del agente en un posible lanzamiento público.

En cuanto a líneas futuras de desarrollo, una primera podría orientarse hacia añadir más caminos de conversación. Se podría lograr haciendo que una gran cantidad de personas usen el agente varias veces para después analizar sus frases e incluir conversaciones de respuesta a todas ellas. También se podría aplicar a otros campos añadiendo o eliminando funcionalidades para que el agente esté enfocado a un público distinto.

La segunda línea podría estar ligada al uso de los *Mega-agentes,* donde varios equipos trabajan para generar agentes que realicen funcionalidades muy dispares como puede ser la consulta de noticias o del tiempo o un mayor número de frases cotidianas. Para después unirlos en un agente con tantas funcionalidades que pueda convertirse en una herramienta de uso cotidiano e incluso generando un agente que pueda conversar durante horas con las personas mayores. Es un objetivo ambicioso pero totalmente realizable.

En definitiva teniendo en cuenta el cumplimiento de todos los objetivos y las positivas valoraciones de los usuarios es posible concluir que la solución implementada es óptima y tiene unas muy buenas vistas de futuro. Además es una tecnología en plena expansión y con múltiples empresas respaldando su desarrollo. Es la mejor solución a corto plazo para combatir el aislamiento social al que están sometidos nuestros mayores.

Bibliografía

1 million bot. (s.f.). *Chatbot Lola UMu*. Recuperado el 22 de 04 de 2020, de https://1millionbot.com/chatbot-lola-umu/

Noticias de la empresa Riken. (23 de Febrero de 2015). Recuperado el 10 de 5 de 2020, de https://www.riken.jp/en/news_pubs/research_news/pr/2015/20150223_2/index.html

Instituto nacional de estadística. (2018). Recuperado el 2020, de Encuesta contínua de hogares: https://www.ine.es/

Google AI Blog. (28 de 1 de 2020). Recuperado el 25 de 4 de 2020, de https://ai.googleblog.com/2020/01/towards-conversational-agent-that-can.html

A. Jardon, A. G. (2010). Usability assessment of ASIBOT: a portable robot to aid patients with spinal cord injury. *Disability & Rehabilitation: Assistive Technology*, 1-11.

Documentación en linea de Google Cloud. (s.f.). Recuperado el 12 de 05 de 2020, de https://cloud.google.com/dialogflow/docs/basics

Hoy, M. B. (2018). Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants. *Medical Reference Services Quarterly*, 81-88.

Jash Thakkar, P. R. (2018). Erasmus – AI Chatbot. *International Journal of Computer Sciences and Engineering Vol.-6, Issue-10.*, 498 - 502.

Jia, J. (2003). *The Study of the Application of a Keywords-based Chatbot System on the.* ArXiv preprint cs/0310018.

Krishnan, S., PAL, V., & VENKATASUBRAMANIAN, B. (2016). *Patente nº US20180115595A1.* Estados Unidos.

M. J. Wolf, K. M. (2017). Why we should have seen that coming: comments on Microsoft's tay "experiment," and wider implications. *ACM SIGCAS Computers and Society Volume 47, Issue 3*.

Página web de la empresa Makeblock. (s.f.). Recuperado el 15 de 05 de 2020, de https://www.makeblock.com/steam-kits/codey-rocky

Página web del Grupo ADD. (s.f.). Recuperado el 14 de 04 de 2020, de https://grupoadd.es/el-robot-buddy

Rafael Aracil, C. B. (2008). ROBOTS DE SERVICIO. *Revista Iberoamericana de Automática e Informática industrial Vol 5, núm. 2*, 6-13.

Sansonnet JP., L. D. (2006). Architecture of a Framework for Generic Assisting Conversational Agents. . En *Lecture Notes in Computer Science, vol 4133* (págs. 145-156).

Turing, A. M. (1950). Computing Machinery And Intelligence. En *Mind, Volume LIX, Issue 236* (págs. 433-460).

Velázquez, A. L. (2019). Chateando con Mitsuku. *Revista Digital Universitaria (rdu). Vol. 21, núm. 1* .

Web de la empresa No Isolation. (s.f.). Recuperado el 14 de 04 de 2020, de https://www.noisolation.com/global/av1/