Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Bullbot: Asistente Virtual para la Gestión de Servicios de la Biblioteca de la ULL

Bullbot: Virtual Assistant for Management of ULL's library services

Yeray Expósito García

La Laguna, 10 de junio de 2019

D. **María Elena Sánchez Nielsen**, con N.I.F. 42.848.599-J profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Bullbot: Asistente Virtual para la Gestión de Servicios de la Biblioteca de la ULL"

ha sido realizada bajo su dirección por D. **Yeray Expósito García**,

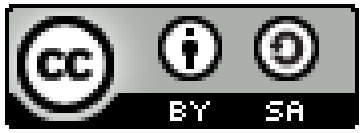
con N.I.F. 43.381.717-Z.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 10 de junio de 2019

Agradecimientos

Ante todo agradecer a mi familia por apoyarme y escucharme.
A mis compañeros en los que tengo confianza plena y con
los que he podido contar en todo momento. Y a mi tutora
María Elena por compartir su conocimiento conmigo
a lo largo del presente proyecto.

Licencia



Resumen

El objetivo de este trabajo ha sido elaborar un chatbot que sea capaz de gestionar los servicios de la Biblioteca de la ULL. El asistente actúa como una interfaz que facilita a los usuarios el acceso a la información de la web oficial de una manera cómoda y natural.

La idea de desarrollo surge como una forma innovadora, en la que se aprovecha la creciente corriente actual por el desarrollo y aplicación de la inteligencia artificial, para implementar un sistema de ayuda que deberá integrarse en un entorno tan exigente como es el de la biblioteca universitaria.

El sistema estará capacitado para ofrecer un servicio eficiente las 24 horas del día, espacio de tiempo en el que debe ser capaz de solventar las dudas del colectivo que desea sacar el máximo partido a los servicios de la biblioteca, que abarca desde alumnos y profesores hasta usuarios externos. Aunque se considera que será especialmente útil para el alumnado de nuevo ingreso, el cual acaba de entrar en contacto con el ámbito universitario y que por tanto requerirá de todo el apoyo posible para aprender a sacar partido de los recursos que tiene a su alcance, y la biblioteca es uno de ellos.

El proyecto en cuestión se desarrollará haciendo un uso intensivo de la tecnología de inteligencia artificial de IBM, campo en el que goza de años de experiencia. Con la que se espera alcanzar un resultado capaz de cumplir todas las expectativas.

Palabras clave: chatbot, asistente virtual, gestión de servicios, Biblioteca de la ULL, sistema de ayuda, bot conversacional, inteligencia artificial, tecnología de IBM, Watson Assistant, usuarios de la ULL.

Abstract

The objective of this work has been to develop a chatbot that is capable of managing the ULL Library services. The assistant acts as an interface that facilitates user's access to official website information in a comfortable and natural way.

The development idea emerges as an innovative way, which takes evolution advantage in the development and application of artificial intelligence, to implement a help system that must be integrated into an exigent environment such as the university library.

The system will be able to offer an efficient service 24 hours a day, timeframe in which it must be able to solve the doubts of people who wish to take full advantage of library services. Although we believe it will be especially useful for new students, who begin to get in touch with the university environment and who need help to learn how to take advantage of the resources at their disposal, and the library is one.

The project will be developed using IBM's artificial intelligence technology, a field in which he has a lot of experience. With this technology it is expected to achieve a result that meets the objectives.

Keywords: chatbot, virtual assistant, service management, ULL library, help system, conversational bot, artificial intelligence, IBM technology, Watson Assistant, ULL users.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

La tecnología tiene como fin satisfacer las necesidades del ser humano y mejorar su calidad de vida. En este aspecto la inteligencia artificial es un campo que se encuentra en pleno crecimiento, y que, a pasos agigantados, se ha introducido en nuestras vidas. Ordenadores, dispositivos móviles, electrodomésticos, chatbots integrados en la web, altavoces inteligentes, domótica o asistentes telefónicos son algunos ejemplos de usos de la inteligencia artificial con los que nos habremos topado más de una vez, y será algo con lo que debemos convivir, pues se trata de una tecnología con una amplia proyección, que unida al creciente desarrollo del internet de las cosas parece indicarnos que nos aguarda un futuro propio del cine de ciencia ficción.

1.1 Antecedentes

Los inicios de esta tecnología datan, nada más y nada menos, que desde la antigua Grecia teniendo como protagonistas a Aristóteles, que propuso el primer conjunto de reglas que describen una parte del funcionamiento de la mente humana, y a Ctesibio de Alejandría, de cuyas manos nació la primera máquina autocontrolada.

Más adelante, Alan Turing diseñó la denominada Máquina de Turing, capaz de llevar a cabo cualquier computo que hubiese sido definido formalmente, y propuso su famosa Prueba de Turing, un test destinado a averiguar si una máquina presenta o no un comportamiento inteligente.

A partir de 1956, durante la conferencia de Darthmounth, surge por primera vez el término de "inteligencia artificial" y es cuando se toma conciencia del potencial de este campo. Durante las próximas décadas no se cumplieron las expectativas, lo que llevó a que mermarán los fondos para investigación, y por consiguiente, se ralentizará el desarrollo de esta tecnología.

Con el nacimiento de la era digital surge la necesidad de mejorar la capacidad de procesamiento y análisis de la ingente cantidad de información que generan los usuarios, lo que llevó a que las empresas tecnológicas aumentaran sus inversiones. Como consecuencia de este auge, en 1997 aparecerá Deep Blue de la mano de IBM, famosa por superar al campeón mundial de ajedrez. Tras este hito, IBM continuo invirtiendo en este campo y en 2011 lanzó a Watson, el cual se ha convertido en uno de los sistemas inteligentes más famosos gracias a su capacidad para el procesamiento de lenguaje natural, el razonamiento y el aprendizaje automático. Además de IBM, existen otras empresas que realizan progresos en este sector como son Google, Sony o Microsoft.

1.2 Contexto del asistente

Figura 1.1: Logotipo de la Biblioteca de la ULL

Para la realización del presente proyecto se ha decidido desarrollar un chatbot para la web de la Biblioteca de la ULL. El principal motivo por el que se ha seleccionado dicho entorno es porque hoy en día existen pocos asistentes virtuales que actúen en un ámbito universitario. Dicha escasez se agudiza aún más si nos centramos, como sucede en este proyecto, en la gestión de los servicios de una biblioteca, un contexto que he encontrado de especial interés ya que de esta forma se puede garantizar que los usuarios de la biblioteca tendrán a su disposición un servicio de atención disponible las 24 horas del día, lo que es de especial importancia sobre todo para alumnos de nuevo ingreso que desconocen su funcionamiento. Otro aspecto a considerar, es que la cantidad de servicios que proporciona la biblioteca y que por tanto deberá dominar el asistente, se ajusta al tiempo del que se dispone para la realización del trabajo, de esta forma se puede garantizar que el asistente no se limitará, tan solo, a responder cuestiones triviales sino que se podrá profundizar en lo relativo a cada servicio.

1.3 Objetivos

El proyecto en cuestión, tal y como se ha planteado, se basa en la implementación de un chatbot que deberá desenvolverse en la web de la Biblioteca de la ULL, donde se encargará de la gestión de los servicio. Por ello, el objetivo principal del proyecto se basa en la creación de un bot conversacional capaz de satisfacer los siguientes apartados:

- Dada la importancia de la institución, es fundamental que sea capaz de garantizar la atención de los usuarios las 24 horas del día.
- Debe ser capaz de ofrecer un servicio completo que abarque todos los ámbitos de la biblioteca.
- Tendrá que adaptarse al usuario y disponer de algún sistema que le permita mantener un registro de aquellas cuestiones a las que no ha sido capaz de dar respuesta, con el fin de que, posteriormente, puedan ser incluidas por el programador.
- A su vez, se procurará que el servicio prestado sea accesible, de manera cómoda y natural, por el mayor número de usuarios posible. Con ese fin, se deberán incorporar servicios que permitan el reconocimiento de voz y la comunicación en una lengua extranjera.

Adicionalmente, se elaborará un prototipo web que tiene como finalidad proporcionar una imagen global del resultado final, además de permitir la conexión del asistente con otros servicios que mejoran la interacción con el usuario.

Capítulo 2
Estado del arte

En el capítulo anterior se ha introducido el tema principal del proyecto que se va a desarrollar y se ha concretado el ámbito en el que se tendrá que desenvolver el chatbot. A continuación, procederé a analizar la situación actual de los asistentes virtuales, hablaré sobre algunas de las plataformas y tecnologías más importantes para su implementación y expondré los últimos proyectos realizados en este ámbito.

2.1 Estado actual de los asistentes.

La inteligencia artificial es un campo que se encuentra en plena efervescencia, dentro del cual ha destacado su aplicación para la creación de asistentes virtuales. Los sistemas operativos para ordenadores, de la mano de Microsoft, fueron los primeros en disponer de este tipo de herramientas inteligentes. Con el tiempo también se desarrollaron asistentes para otros sistemas operativos como Mac o Linux, a la vez que ha ido ganado relevancia, especialmente, en el segmento móvil, ya que se tratan de dispositivos que presentan un uso elevado por parte de los usuarios lo que favorece la automatización de tareas, ya sea a través del reconocimiento de órdenes de voz o escritas. Llevando a la aparición de asistentes como Siri en Apple, Cortana en Microsoft o Google Assistant en Android. Más adelante surgirían los primeros dispositivos diseñados específicamente para albergar un asistente virtual, como es el caso de Amazon Echo y el asistente Alexa, Apple Homepod junto a Siri y Google Home con Cortana.

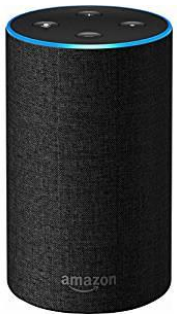


Figura 2.1: Amazon Echo

Figura 2.2: Apple Homepod

Figura 2.3: Google Home

Además de en dispositivos electrónicos, también se emplean asistentes virtuales, en forma de chatbots, en páginas web, donde a través de su capacidad para el procesamiento del lenguaje natural son capaces de entender al usuario con el fin de proporcionarle ayuda, como sucede en el caso de la página web de empresas como Iberia o Renfe. En este aspecto la inteligencia artificial aplicada a través de los bots conversacionales busca cambiar la forma en la que interactuamos con la tecnología y nos comunicamos por internet, procurando que sea de una manera natural. Sin embargo, el ser humano debe adaptarse a los cambios ya que a pesar de estas innovaciones muchas personas siguen optando por escribir sus búsquedas mediante teclado, en vez de emplear la interfaz de voz, más natural, de la que disponen muchos asistentes.

Algunos ejemplos de empresas españolas dedicadas al mundo de los chatbots son, en primer lugar, **Inbenta** [21], empresa creada en 2005, especializada en la búsqueda semántica, el procesamiento del lenguaje natural, en más de 20 idiomas, y la inteligencia artificial. Para ello dispone de un motor de búsqueda semántico que permite al bot entender al usuario incluso cuando plantea cuestiones desestructuradas, ambiguas o con errores ortográficos. Lo que indica que esta tecnología es capaz de tener en cuenta la relación existente entre las palabras para mejorar la comprensión del mensaje del usuario. Además incorpora en sus chatbots Machine Learning, permitiéndoles proporcionar respuestas no solo usando el significado de las palabras sino también el aprendizaje.

En segundo lugar, encontramos a **Commons** [22], esta empresa dispone de una plataforma para la creación de chatbots, denominada Commons Bot Platform en la que se busca facilitar el diseño de bots basados en el contexto. Para lo que dispone de un motor de razonamiento único. Su tecnología se caracteriza por disponer de dos niveles fácilmente distinguibles, el primero se ocupa del comportamiento del bot, de memorizar y manejar el contexto de la conversación. El segundo indica lo que tiene que hacer, adaptándose a los requerimientos del usuario.

Y en tercer lugar, tenemos a **Caravelo** [23][24], una empresa con sede en Barcelona centrada en la implementación de bots conversacionales para aerolíneas. Que ha desarrollado un chatbot denominado **Nina** que permite realizar las clásicas operaciones de reserva, modificación o cancelación de un vuelo, además de muchas otras, ofreciendo así un servicio integral a los clientes, resolviendo sus dudas, proporcionando información y ofreciendo los productos y servicios de la aerolínea. Esta tecnología está disponible, principalmente, para plataformas como Facebook Messenger o WhatsApp.

2.2 Técnicas para el procesamiento de la información

A simple vista, el funcionamiento de los chatbots puede resultar bastante trivial, ya que se limitan a recibir información del usuario, ya sea de forma escrita u oral, la procesan y responden. Sin embargo, bajo esa tarea de reconocimiento de la información se esconden una serie de técnicas que son las que constituyen la esencia de esta tecnología, permitiendo, cada vez mejor, que la comunicación entre el humano y el bot sea posible. A continuación, procederé a exponer algunas de esas técnicas.

2.2.1 Machine learning: Deep learning

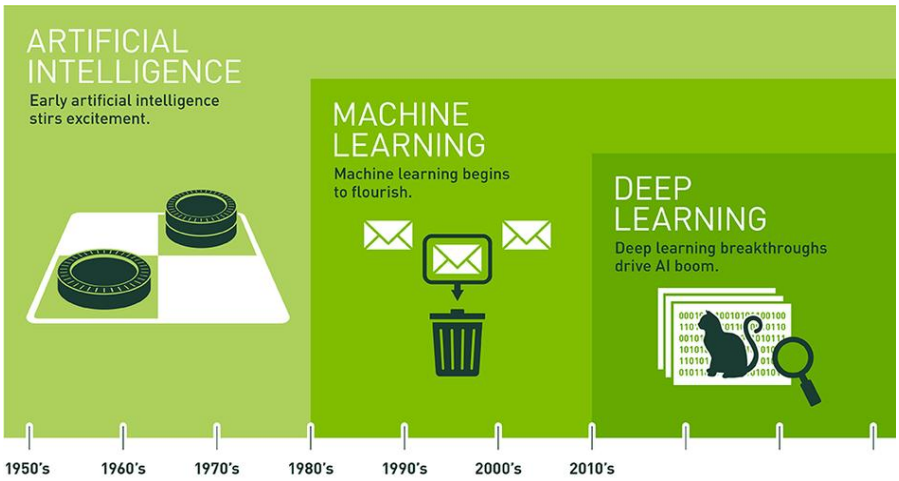


Figura 2.4: Comparación de conceptos

Las técnicas de **Machine learning** [32] son aquellas que permiten identificar los patrones que se producen dentro de grandes conjuntos de datos, de manera que intentan llevar a cabo predicciones en base a lo que hayan aprendido. En este aspecto se pueden distinguir diferentes tipos en función de cómo se identifiquen dichos patrones.

En primer lugar se encuentra el **aprendizaje supervisado**, en el cual debemos indicar la relación que existe entre ciertos datos y su resultado con el fin de entrenarlo. Creándose de esta forma una función que es capaz de predecir el valor correspondiente a una entrada teniendo en cuenta los ejemplos de los que ha aprendido. El algoritmo deberá ser capaz de generalizar, a partir de los datos de los que dispone, una respuesta para aquellas situaciones nuevas.

En segundo lugar, se encuentra el **aprendizaje no supervisado**, el cual no dispone de una salida esperada asociada a los ejemplos disponibles, por lo que debe ser capaz de identificar patrones o características comunes en dichos ejemplos con el fin de determinar aquellos casos que posean propiedades similares.

Además de los mencionados, existen otros dos tipos, el **aprendizaje semisupervisado**, que es una combinación de los dos algoritmos anteriores, y el **aprendizaje por refuerzo** [33], el cual se basa en un sistema de retroalimentación, en el que sus entradas son las consecuencias de sus respuestas en el exterior, de manera que el sistema aprende a base de ensayo y error, pues para cada una de sus acciones recibe una recompensa o castigo que usa para completar su conocimiento sobre el entorno hasta encontrar la mejor recompensa posible, lo que indica que el sistema ha identificado los pasos necesario para alcanzar el resultado correcto .

Por otro lado, se encuentran los algoritmos **deep learning** [34], que es uno de los más empleados para la creación de asistentes virtuales. Este sistema, centrado en el aprendizaje autónomo, emplea redes neuronales para detectar características ocultas de los datos que permitan definir un resultado que almacena en una estructura de neuronas. Esto supone un acercamiento hacia la forma de pensar del ser humano, pues se intenta emular el funcionamiento de nuestro sistema nervioso.

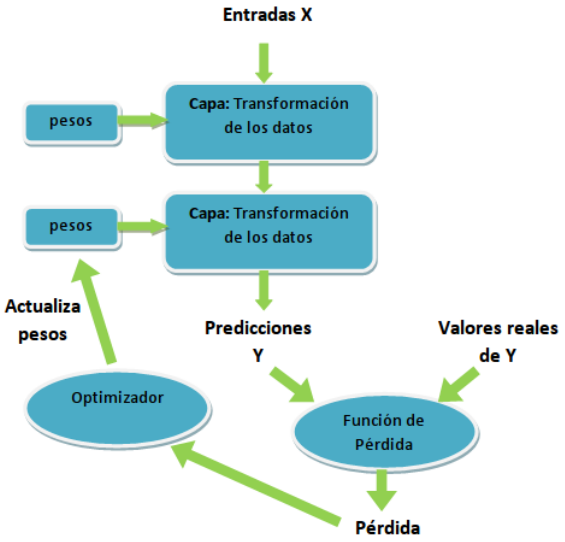


Figura 2.5: Funcionamiento de una red neuronal.

Dicha estructura neuronal se organiza en capas o niveles jerárquicos, en las que cada neurona dispone de una conexión con cada una de las presentes en la capa siguiente. Dicha estructura puede estar formada por cientos de capas, cada una de las cuales está dotada de unos pesos que afectan a la trasformación que sufren los datos en cada una de ellas a medida que se avanza. Estas capas son capaces de aprender automáticamente mediante el entrenamiento con datos. Para ello se requiere de una **función de pérdida** que se encarga de calcular la distancia que existe entre la predicción de la red neuronal y el resultado esperado, de manera que el valor de retorno de dicha función se emplea para ajustar los pesos de cada capa con el fin de mejorar el resultado ofrecido por la red neuronal. De esta labor se encarga el optimizador, a través de la implementación del método de **propagación hacia atrás**, el cual recibe el valor de error y lo propaga desde la capa de salida hacia las neuronas de la capa oculta (zona intermedia), de manera que cada neurona recibe una fracción de la señal de error proporcional a su contribución en el resultado ofrecido, que usa para mejorar su rendimiento y de esta forma aprender.

2.2.2 Natural Language Processing (NLP)

El **procesamiento del lenguaje natural (NLP)** [35] es una rama de la inteligencia artificial que ayuda a las máquinas a entender, interpretar y manipular el lenguaje natural, así como determinar la acción adecuada y responder en el mismo idioma del usuario. Para que esto sea posible, se lleva a cabo, mediante algoritmos, la conversión de los datos no estructurados, es decir el lenguaje natural, en datos estructurados que pueden ser usados por las máquinas. Algunas de las tareas básicas del NLP son la clasificación de textos por temática, el etiquetado de partes de un discurso, la detección del lenguaje, la identificación de relaciones semánticas, traducciones automáticas, chatbots, etc.

Figura 2.6: Esquema de tecnologías NLP, NLU y NLG.

Como se puede observar, NLP abarca todas las técnicas necesarias para operar con el lenguaje natural, dentro de las cuales se pueden distinguir dos grupos o subramas.

En primer lugar, se encuentra la rama de **compresión del lenguaje natural (NLU)**, la cual se encarga de averiguar el significado e intención que encierra un mensaje. Funciona a través de algoritmos de aprendizaje profundo, análisis estadístico del orden y frecuencia de las palabras y una gran cantidad de datos de capacitación. El uso de NLU en los chatbots permite mejorar la experiencia de usuario, ya que la conversación es más fluida, sin tener la necesidad de usar respuestas preestablecidas como sucede en el caso del resto de tecnologías NLP.

Y por otro lado, se encuentra la **generación del lenguaje natural (NLG)** que se basa en recrear de manera automática el lenguaje natural a partir de los datos estructurados con los que trabaja el computador, con el fin de que el ser humano pueda entenderlos. Su uso es fundamental en el ámbito de los chatbots para permitir la comunicación tanto por texto como por voz. A día de hoy existen dos formas diferentes de llevar a cabo este proceso, por un lado, se encuentran los sistemas basados en reglas, y por otro, los que emplean inteligencia artificial.

Los sistemas **basados en reglas** [36], buscan codificar una serie de normas que permitan traducir los datos, es decir, se basa en buscar todos los posibles casos de un contexto y elaborar una regla para cada una de esas situaciones, el principal inconveniente de estos sistemas es que disponen de un contenido limitado lo que afecta a la calidad del servicio. Además, presentan serias dificultades para afrontar escenarios que pueden requerir la codificación de miles de reglas. En este aspecto los sistemas **basados en inteligencia artificial** tienen ventaja, pues presentan una elevada escalabilidad gracias a que son capaces de establecer la estructura de la narrativa en función de los datos y del aprendizaje recibido en el proceso de entrenamiento. Además el hecho de que puedan aprender de los usuarios les permite mantener actualizada dicha narrativa y su contenido, lo que implica un menor mantenimiento.

2.3 Plataformas existentes para el desarrollo de chatbots.

Las principales empresas, en el ámbito mundial, se han percatado del impacto que los chatbots tienen a través de la red, y se han aventurado a ofrecer soluciones para la creación de asistentes virtuales, lo que ha permitido que los desarrolladores tengan a su disposición diversas infraestructuras para la elaboración de sus bots. Algunas de las empresas que proporcionan este tipo de plataformas de desarrollo son las que a continuación se exponen.

Por un lado se encuentra **IBM**, la multinacional estadounidense ofrece sus servicios para la elaboración de asistentes a través de la plataforma **IBM Cloud**. Donde es posible encontrar la herramienta Watson Assistant, que se centra en la conversación, permitiendo crear entidades, intenciones y establecer contextos. Además, proporciona otros servicios que complementan al chatbot, como es el caso de Speech to Text para el reconocimiento de voz, Text to Speech para convertir texto en audio, Language Translator para dotar al bot de la capacidad de entender varios idiomas, Tone analyze para detectar emociones, Natural Language Understanding, Machine Learning y un largo etcétera.

Figura 2.7: Logotipo de IBM Cloud

En segundo lugar, encontramos a **Microsoft**, que a través de su plataforma de cloud **Azure** ofrece su tecnología para la creación de bots, donde podemos encontrar la herramienta Azure Bot Service [25], un servicio específico para la creación de chatbots al que podemos integrar diferentes funcionalidades de inteligencia artificial a través de Azure Cognitive Service [26], como es el reconocimiento de voz e

Figura 2.8: Logotipo de Microsoft Azure

imágenes, búsqueda web, etc. También dispone de la tecnología **LUIS** (Language Understanding Intelligence Service**)** [27], un conjunto de servicios que giran en torno al procesamiento del lenguaje natural y que pueden integrarse fácilmente con los asistentes creados a través de Azure Bot Service.

Por otro lado, tenemos a **Google**, la empresa, por medio de su Cloud, ofrece la API Natural Language [28]. Esta herramienta, a través de modelos de aprendizaje profundo, es capaz de analizar y alcanzar el significado de textos, así como extraer la información de interés. Además es capaz de trabajar con diferentes idiomas. Por otro lado, dispone de la herramienta **Dialogflow** [29], que se caracteriza por poseer una gran capacidad para procesar lenguaje natural, también incorpora aprendizaje automático y productos como Speech to Text. Por último, mencionar a **Chatbase**, un servicio que permite a los desarrolladores analizar y optimizar sus agentes virtuales.

Figura 2.9: Logotipo de Google Cloud

En cuanto a **Amazón**, también ha aportado su grano de arena y ha desarrollado la tecnología **Lex** [30]. Se trata de un servicio que permite crear interfaces conversacionales operativas tanto por voz como por texto, con funcionalidades de aprendizaje profundo y comprensión del lenguaje natural. Por medio de dicho servicio se ponen a disposición de los desarrolladores las tecnologías de aprendizaje empleadas en Alexa.

Figura 2.10: Logotipo de Amazon Lex

Y por último, analizaré a **Facebook**. La empresa en cuestión dispone de **Wit.ai** [31], una herramienta centrada en el procesamiento del lenguaje natural que tiene como objetivo facilitar a los desarrolladores la creación de chatbots, aplicaciones móviles, automatización del hogar y dispositivos "wearables" y robots capaces de entender a los humanos. También hay que tener en cuenta que la empresa es dueña de algunas de las principales plataformas sobre las que pueden operar los asistentes, como es **Facebook Messenger** y **WhatsApp**.

Figura 2.11: Logotipo de Wit.ai

2.4 Proyectos relacionados con asistentes virtuales.

2.4.1 A nivel internacional.

En lo relativo al ámbito internacional tenemos los asistentes lanzados por las principales compañías. Como es el caso de **Jarvis** [1], desarrollado por Facebook en 2016, el asistente en cuestión utiliza varias técnicas de inteligencia artificial, como el procesamiento de lenguaje natural, reconocimiento de voz, reconocimiento de rostro y aprendizaje de refuerzo. También cabe indicarse que el sistema está escrito en Python, PHP y Objetive-C.

En el caso de Apple tenemos a **Siri** [2], el cual fue adquirido por la empresa en 2010. El asistente ha sido desarrollado para facilitar todo tipo de tareas en los dispositivos Apple, desde realizar llamadas y enviar mensajes cuando se está ocupado, hasta tareas cotidianas como poner la alarma añadir recordatorios, escuchar música, etc. También ha sido implementado para ser usado en casas inteligentes, pues es capaz de controlar los sistemas domóticos y dispositivos inteligentes solo haciendo uso de la voz. Dispone de un sistema de aprendizaje automático y es capaz de emplear 21 idiomas.

En cuanto a Amazon, no se ha quedado atrás y ha lanzado a **Alexa** [3]. El asistente en cuestión está incorporado en un altavoz que ha sido diseñado por la empresa, explícitamente, con ese fin, al

que han denominado Amazon Echo. Aunque también es posible encontrarlo en hardware de otros desarrolladores. Alexa es capaz de cumplir las clásicas funciones de un asistente de voz, como decir la hora, poner alarmas, reproducir música, etc. Además Amazón se ha aliado con marcas como Uber, Domino's, Just Eat o Kayak permitiendo a los usuarios acceder a estos servicios a través del asistente. Alexa también dispone de otros usos destinados al hogar inteligente y al control de dispositivos, siempre y cuando sean compatibles.

En tercer lugar, tenemos a **Google Assistant** [4] el asistente de Google. Este asistente está disponible tanto para Android como para iPhone y puede ser empleado en el hogar a través de altavoces que disponen del servicio ya integrado, al igual que sucede con Alexa. También está disponible en dispositivos "wareables" como, por ejemplo, sus smartwatch, y en televisores y dispositivos con Android TV.

Por último, analizaré a **Cortana** [5], el asistente de Microsoft. Dicho asistente está disponible para Windows 10 y dispositivos móviles con sistema operativo Window 10 Mobile, Window Phone, IOS o Android. También se puede encontrar integrado en el altavoz inteligente Invoke, en las consolas Xbox One y en Microsoft Band. Cortana está diseñado para ayudar a los usuarios en tareas sencillas como responder preguntas, ayudarles a gestionar su agenda, realizar recordatorios personalizados, hacer búsquedas o ejecutar otras aplicaciones desde el asistente.

2.4.2 A nivel nacional.

Algunos ejemplos de chatbots desarrollados por empresas españolas son, en primer lugar, **Faster.city** [6], un asistente cuyo objetivo es ayudar a los usuarios a desplazarse de manera rápida y cómoda por la ciudad, evitando obras, atascos o los problemas causados por factores meteorológicos.

En segundo lugar, **Billy** [7], un chatbot especializado en seguros, disponible para Whatsapp, Messenger y Telegram. Que se encarga de ayudar a los usuarios a encontrar las mejores ofertas de seguros para sus vehículos.

Y en tercer lugar, tenemos a **CorreYvuela** [8], un chatbot que tiene como objetivo facilitar la búsqueda y compra de vuelos. Actualmente, se encuentra disponible a través de WhatsApp, SMS, Telegram y Facebook.

Capítulo 3
Metodología

Con el fin de planificar el proyecto lo primero que se llevó a cabo fue la definición de los objetivos que se pretendían cumplir, una vez hecho se establecieron y ubicaron temporalmente las tareas que serían necesarias realizar para alcanzar las metas. De manera que a medida que se iban cumpliendo dichas tareas se llevaban a cabo, periódicamente, reuniones con la tutora del proyecto para mostrarle los avances, resolver dudas y que verificara si el progreso era el adecuado. La idea principal es que en cada revisión el proyecto debe mostrar alguna evolución respecto a la comprobación previa, lo que se ajusta al modelo de planificación **Scrum**. Las tareas especificadas pueden ser agrupadas en tres bloques, los cuales se explicarán de manera detalla en los siguientes apartados. Antes de continuar, indicar que las fechas de inicio y fin que se especifican en las tablas de tareas representan el tiempo exacto que tomó realizar cada una de ellas, ya que prácticamente todas se lograron concluir en un menor lapso de tiempo del especificado en el anteproyecto, lo que permitió incorporar nuevas tareas al proyecto, con el fin de hacerlo más completo.

3.1 Estudio previo y planteamiento

Tarea	Inicio	Fin
Estudio y análisis de las plataformas y tecnologías aplicables al proyecto.	26/02/2019	17/03/2019
Examinar el contexto de uso en el que las empresas actuales emplean asistentes virtuales.	18/03/2019	24/03/2019
Selección del entorno en el que se desenvolverá el asistente y planteamiento de las cuestiones a las que debería ser capaz de dar respuesta.	25/03/2019	31/03/2019
Desarrollar un diagrama de flujo para analizar las posibles interacciones, intenciones y respuestas.	01/04/2019	07/04/2019

Tabla 3.1: Tareas de la fase de estudio previo y planteamiento

El bloque de **estudio previo y planteamiento** está constituido por las cuatro tareas que se pueden apreciar en la tabla superior. En la primera se pretende examinar las diferentes soluciones que las empresas actuales ofrecen a los desarrolladores para la implementación de chatbots y seleccionar la herramienta que mejor se ajuste a nuestras necesidades. En la segunda tarea se han analizado diferentes empresas que disponen de asistentes virtuales en sus sitios web, con el fin de

contemplar que tipo de servicios ofrecen, cómo se los proporcionan al usuario y en qué contexto se desenvuelven. En tercer lugar, se procedió a definir el entorno en el que se debería desenvolver el asistente, para lo que se tuvo en cuenta diferentes factores, entre ellos el tipo de cuestiones a las que debería ser capaz de responder. Por último, se llevó a cabo el diseño de un diagrama de flujo, a través del cual se estudiaron los posibles flujos por los que podrían derivar las conversaciones con los usuarios, así como las intenciones y respuestas de éstos.

3.2 Implementación

Tarea	Inicio	Fin
Desarrollar en la plataforma seleccionada la primera versión del asistente.	08/04/2019	30/04/2019
Incorporar contextos, entidades y mejorar el entrenamiento del asistente para perfeccionar la experiencia de usuario.	01/05/2019	03/05/2019
Integrar el asistente con una base de datos.	04/05/2019	07/05/2019
Crear un prototipo web de la página de la ULL con el chatbot incorporado para ofrecer una imagen del resultado final.	08/05/2019	13/05/2019
Permitir que el asistente opere en una lengua extranjera (inglés).	14/05/2019	18/05/2019
Incorporar reconocimiento de voz.	19/05/2019	24/05/2019

Tabla 3.2: Tareas de la fase de implementación

Una vez seleccionada la plataforma que se va a utilizar, definido el contexto del asistente y planteado su funcionamiento, es hora de proceder con su **implementación**. Esta segunda fase consta de 6 tareas, cuya ubicación en el tiempo se puede apreciar en la tabla anterior. En la primera se procederá a elaborar la versión inicial del asistente en la plataforma seleccionada, utilizando como guía el diagrama de flujo elaborado en la fase anterior. En la segunda tarea se incorporarán al diseño del chatbot contextos, con los que se mantendrá cierta información entre cada iteración del asistente, se añadirán entidades y se aumentará el entrenamiento. Una vez creada la estructura básica del asistente, en la tercera tarea se conectará a una base de datos en la que podrá almacenar información relativa a los usuarios. A continuación, en la cuarta tarea, se desarrollará un prototipo en el que se integrará el asistente en una simulación de la web de la Biblioteca de la ULL, con el fin de mostrar un ejemplo del posible resultado final del proyecto. Aprovechando el prototipo web que se ha elaborado se procederá a introducir en el backend algunos servicios que complementan el funcionamiento del asistente, como es el uso de una segunda lengua o el reconocimiento de voz, tal y como se menciona en la quinta y sexta tarea.

3.3 Evaluación del funcionamiento

Tarea	Fecha de evaluación
Evaluación del asistente.	10/06/2019

Tabla 3.3: Tareas de la fase de evaluación del funcionamiento

Por último, en la tercera fase, se llevará a cabo la **evaluación del asistente**, en ella la tutora del proyecto someterá el chatbot a una serie de pruebas con el fin de verificar su correcto funcionamiento.

Capítulo 4
Diseño y Desarrollo

A lo largo de este capítulo expondré al detalle las diferentes implementaciones que se han llevado a cabo en el proyecto, así como las herramientas empleadas y el diseño del diagrama de flujo en el que se representa el funcionamiento del asistente. Pero primero veamos cual ha sido la plataforma seleccionada para llevar a cabo la implementación.

4.1 Selección de la herramienta de desarrollo

Figura 4.2: Herramienta de Google para diseñar chatbots

Figura 4.1: Herramienta de IBM para diseñar chatbots

A la hora de seleccionar la herramienta de desarrollo se han examinado, principalmente, dos tecnologías. La herramienta de Google denominada **Dialogflow** y **Watson Assistant** de IBM. Ambas herramientas ofrecen funcionalidades muy similares, sin embargo, la plataforma de IBM se muestra más completa en algunos aspectos.

Entre ellos se encuentra el hecho de que permite integrar código HTML en sus respuestas, dando la opción de añadir, además de texto plano, hipervínculos, imágenes, videos, etc. Por lo que permite elaborar respuestas de mayor calidad, lo que es de especial importancia pues se quiere implementar un servicio que debe interactuar con el usuario. Mientras que Dialogflow, al menos de forma nativa, no ofrece tal diversidad.

Por otro lado, Watson proporciona una interfaz bastante intuitiva y sencilla de emplear en la que se abstrae casi todos los aspectos del diseño del asistente, de hecho tan solo se deberá recurrir a manipular archivos JSON para acciones puntuales que se mencionaran más adelante.

También cabe indicarse que la herramienta de IBM dispone de un chat, para probar la configuración del asistente, incorporado en el entorno de desarrollo, aspecto que también está presente en Dialogflow, con la diferencia de que Watson da la opción de entrenar el asistente en tiempo real en medio de una prueba, además de incluir una segunda opción que permite al diseñador observar el recorrido que lleva a cabo el asistente, en el proceso de reconocimiento de un mensaje, a través del árbol de nodos que define su funcionamiento, lo que facilita, en gran medida, la detección y corrección de errores.

4.2 Tecnologías utilizadas

En este aspecto, la mayor parte de las herramientas empleadas forman parte de los servicios que están presentes en la plataforma de IBM Cloud.

4.2.1 IBM Watson Assistant

Watson Assistant [9] es la herramienta de IBM para la creación de chatbots. La tecnología en cuestión dispone de una interfaz gráfica intuitiva que simplifica muchas de las facetas del diseño del asistente, entre ellas la creación de entidades e intentos, con las que se detectan conceptos clave y la intención del usuario, respectivamente. También permite crear variables de contexto en las que almacenar información entre cada iteración del asistente. Para determinar la respuesta que se le debe dar al usuario se emplea una estructura en forma de árbol de nodos en la que en cada nodo se usa como condicionales las entidades e intentos que se hayan creado. La herramienta dispone de un catálogo con datos de entrenamiento creados por IBM para diferentes situaciones, que se podrán incorporar al proyecto para agilizar la implementación. Por último, indicar que admite hasta 13 idiomas y ofrece integración directa con puntos finales como Slack o Facebook Messenger.

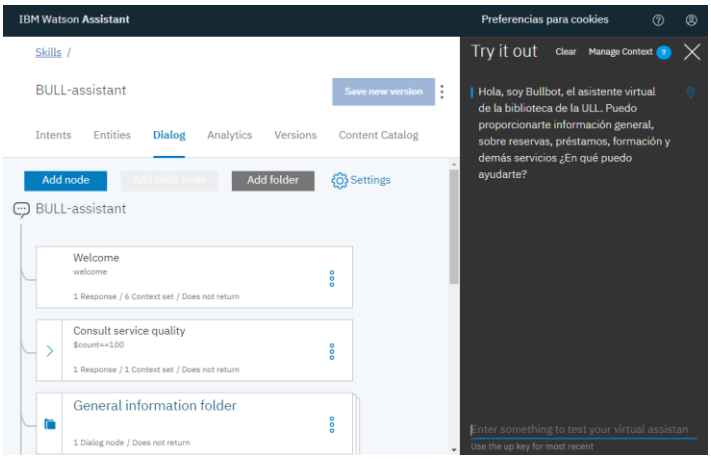


Figura 4.3: Interfaz de Watson Assistant

4.2.2 IBM Cloud Functions

Se trata de una plataforma que permite programar, en diferentes lenguajes, funciones como servicio, para así desarrollar código ligero que se ejecuta bajo demanda (Faas). Lo que significa que los desarrolladores podrán ejecutar código en un entorno escalable sin tener que preocuparse por la gestión de la infraestructura sobre la que se ejecuta dicha función. Por consiguiente se acelera y simplifica el desarrollo de aplicaciones [10].

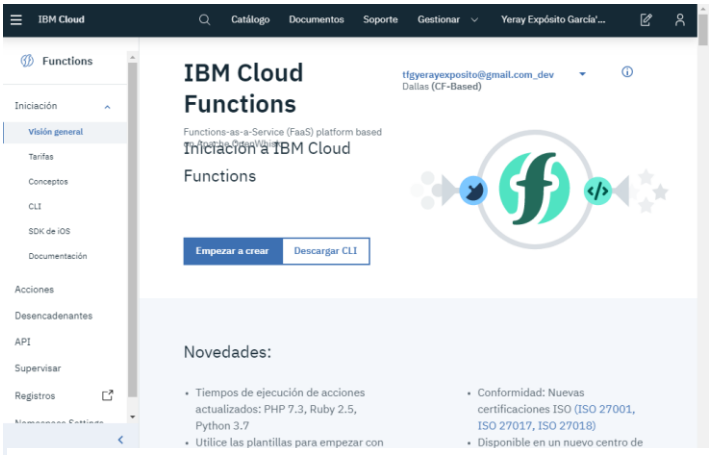


Figura 4.4: Interfaz de IBM Cloud Functions

4.2.3 IBM Db2 on Cloud

Es una **base de datos SQL** [11] en la nube completamente gestionada y compatible con Oracle PL/SQL. Creada para ofrecer un elevado rendimiento, pues es capaz de garantizar un funcionamiento del 99.99%, mitigando con ello las consecuencias de las paradas no planificadas y garantizando la integridad de la base de datos, además de permitir seleccionar un centro de datos offsite para la migración tras errores. También permite el despliegue y el escale bajo demanda.

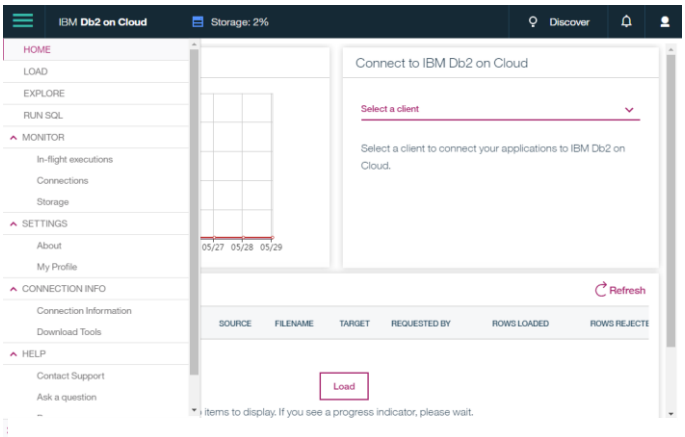


Figura 4.5: Consola de Db2 on Cloud

4.2.4 IBM Watson Language Translator

Language Translator [12] se trata de una herramienta de IBM que permite llevar a cabo traducciones de texto y que domina hasta 20 idiomas. Por defecto, todos los pares de idiomas utilizan tecnología de aprendizaje profundo para mejorar la velocidad y la precisión de la traducción. Además, el servicio ofrece múltiples modelos de traducción, que pueden ser personalizados según la terminología y lenguaje que se deseen emplear.

4.2.5 IBM Watson Speech to text

El servicio **Speech to Text** [13] proporciona una API para añadir funcionalidades de transcripción de voz a las aplicaciones. Para ello emplea técnicas que combinan la información sobre la estructura del lenguaje con la composición de la señal de audio. Permite transcribir automáticamente el audio de hasta 7 idiomas diferentes, incluso desde archivos de calidad inferior y en diversos formatos. También cabe destacar el hecho de que se trata de un motor altamente personalizable pues podemos configurarlo para que detecte, en tiempo real, palabras clave como nombres o si existen diferentes oradores.

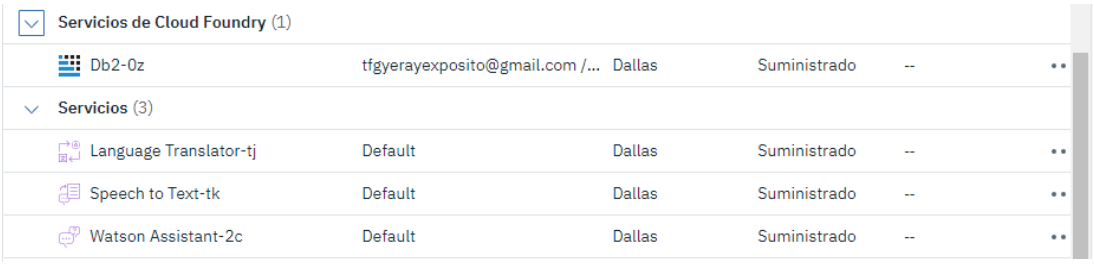


Figura 4.6: Lista de proyectos creados en IBM Cloud

4.2.6 Node.js

Node.js [14] es un entorno JavaScript del lado del servidor que utiliza un modelo asíncrono y dirigido por eventos. El punto fuerte de esta herramienta es su escalabilidad ya que dispone de la consistencia suficiente como para poder gestionar un número elevado de conexiones simultáneas con el servidor. Esto es posible gracias a su funcionamiento asíncrono, el cual permite que todas las tareas que se desarrollan en el servidor se hagan de forma paralela, evitando de esta forma pérdidas de tiempo por el bloqueo del flujo de trabajo.

También, cabe indicarse que en el proyecto se usará **express**, un framework de node.js inspirado en Sinatra (framework de ruby) que proporciona un conjunto sólido de características para aplicaciones web y móviles. Además, dispone de una gran variedad de middleware que pueden ser localizados en npm.

4.3 Análisis del flujo conversacional

Antes de afrontar la implementación del asistente, primero se debe llevar a cabo el planteamiento de las cuestiones a las que, teniendo en cuenta el contexto en el que se desenvolverá, debe ser capaz de dar respuesta. Una vez definidas esas cuestiones se llevará a cabo el diagrama de flujo.

4.3.1 Cuestiones principales

Tal y como se indicó en la introducción, el asistente se integrará en la página web de la Biblioteca de la ULL, por lo que para analizar las cuestiones que los usuarios pueden plantear habrá que examinar detenidamente los diferentes apartados del sitio web, atendiendo a la información que se desarrolla en cada uno de ellos. Al hacerlo, se han podido extraer los datos que se muestran a continuación.

Tema principal	Temas de preguntas
Información general	Ubicación de las bibliotecas, horario, datos de contacto e información sobre cómo realizar trabajos y citar.
Formación	Cursos para estudiantes de grado y posgrado, formación a la carta e inscripción en dichos cursos.
Servicios	Catálogo, Blog, escáneres, servicio de envío de artículos, servicio de atención online y biblioteca digital (PuntoQ, RIULL, revistas y libros electrónicos, tesis, patrimonio bibliográfico lacunense y prensa canaria digitalizada).
Reserva de salas	Reserva de carrels en cualquier facultad o reserva de salas especiales.
Préstamos	Préstamo de ordenadores y pendrives, uso de almacenamiento online, préstamo de libros domiciliario, intercampus e interbibliotecario.

Tabla 4.1: Cuestiones principales que debe saber tratar el asistente

En función de la información encontrada en la web se han clasificado las posibles cuestiones en cinco grupos, cada uno asociado con un tema concreto, información general, formación, servicios, reserva de salas y préstamos. De esta forma, en la siguiente fase se podrá plantear un esquema que se adapte a la información disponible, procurando que sea altamente escalable, lo que es de especial importancia de cara a futuras actualizaciones en el contenido que maneje el asistente. Una vez realizado el análisis, ya se está en condiciones de afrontar la creación del diagrama de flujo.

4.3.2 Diseño del diagrama de flujo

El desarrollo de un diagrama de flujo tiene como fin representar, de manera gráfica, la información recabada en el apartado anterior, a la vez que se indican las posibles interacciones, intenciones y respuestas que puede plantear el usuario, generando de esta manera un esquema, en forma de árbol, en el que se pueden apreciar los diferentes flujos conversacionales.

A la hora de llevar a cabo el diseño, se ha decidido representar cada uno de los cinco temas principales extraídos en la tabla anterior mediante ramas independientes, esto permite mantener un

esquema altamente escalable, que favorece en todo momento la incorporación de nuevos datos al diseño sin necesidad de modificar la estructura planteada. Dentro de las ramas principales, se han creado, a su vez, subramas en las que se desarrollan otros temas asociados al principal.

A continuación, se procederá a introducir los elementos básicos que han sido empleados para elaborar el diseño, para luego explicar detalladamente la estructura y funcionamiento de cada una de las cinco ramas principales que componen el árbol (ver apéndice A sección 4).

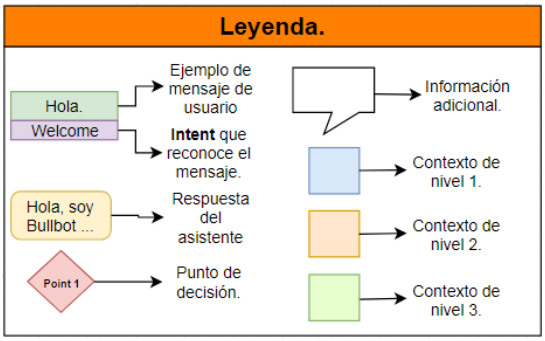


Figura 4.7: Leyenda del diagrama de flujo

En la imagen superior se puede apreciar la leyenda del diagrama, en la que se resume la simbología empleada. Los elementos más utilizados son las figuras que representan el mensaje del usuario y las de respuesta del asistente, las cuales se van alternando a lo largo del diseño. Los puntos de decisión representan situaciones en las que el flujo conversacional se ramificará en función de la respuesta que dé el usuario a la cuestión plantea por el asistente. Mientras, que las áreas de contexto indicarán zonas de árbol en la que se requerirá que el asistente almacene cierta información con el fin de mejorar la experiencia de usuario.

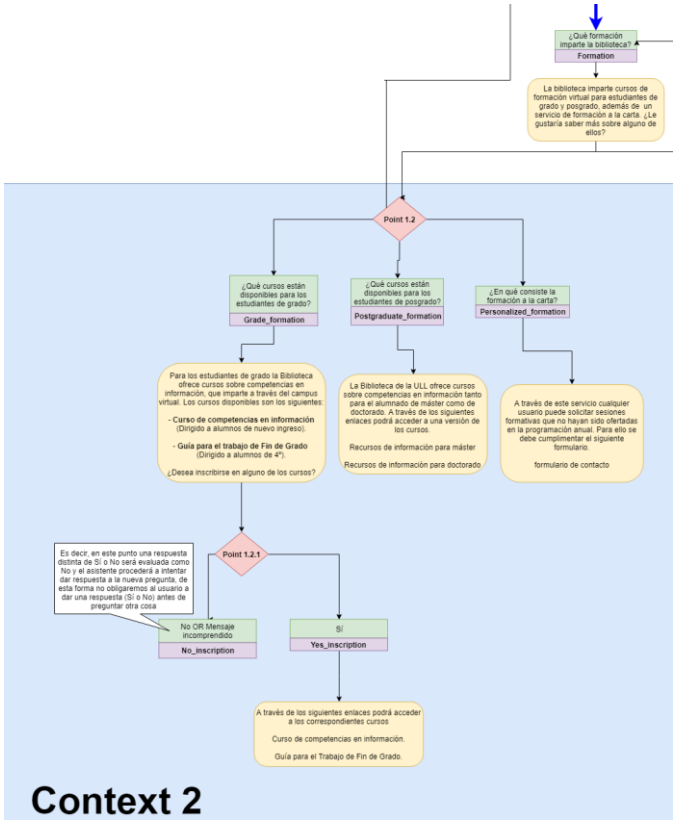


Figura 4.8: Rama de formación

La primera rama que se procederá a analizar es la relativa a la **formación,** como se ha indicado en la tabla 4.1, esta sección del árbol abarcar todos aquellos conceptos relacionados con los cursos que ofrece la biblioteca tanto para alumnos de grado como de posgrado e información sobre el sistema de formación a la carta, además de proveer de los enlaces de acceso correspondientes para darse de alta en los mencionados cursos. Como se puede apreciar en la imagen existe un contexto (área azul) que abarca la totalidad de la rama, con lo que se pretende indicar que en dichas regiones el asistente deberá almacenar cierta información con el fin de mejorar la experiencia de usuario, lo que permite que el cliente que haya accedido, por ejemplo, a la información sobre los cursos para alumnos de grado planteando la cuestión "¿Qué cursos están disponibles para los estudiantes de grado?" pueda acceder a otro tipo de información relacionada con la formación que ofrece la biblioteca sin necesidad de formular la totalidad de la pregunta, ya que el asistente habrá registrado que se ha accedido al contexto, por lo que en esta situación bastaría con que el usuario planteara "¿Y para alumnos de posgrado?" para acceder a los datos correspondientes a esa formación.

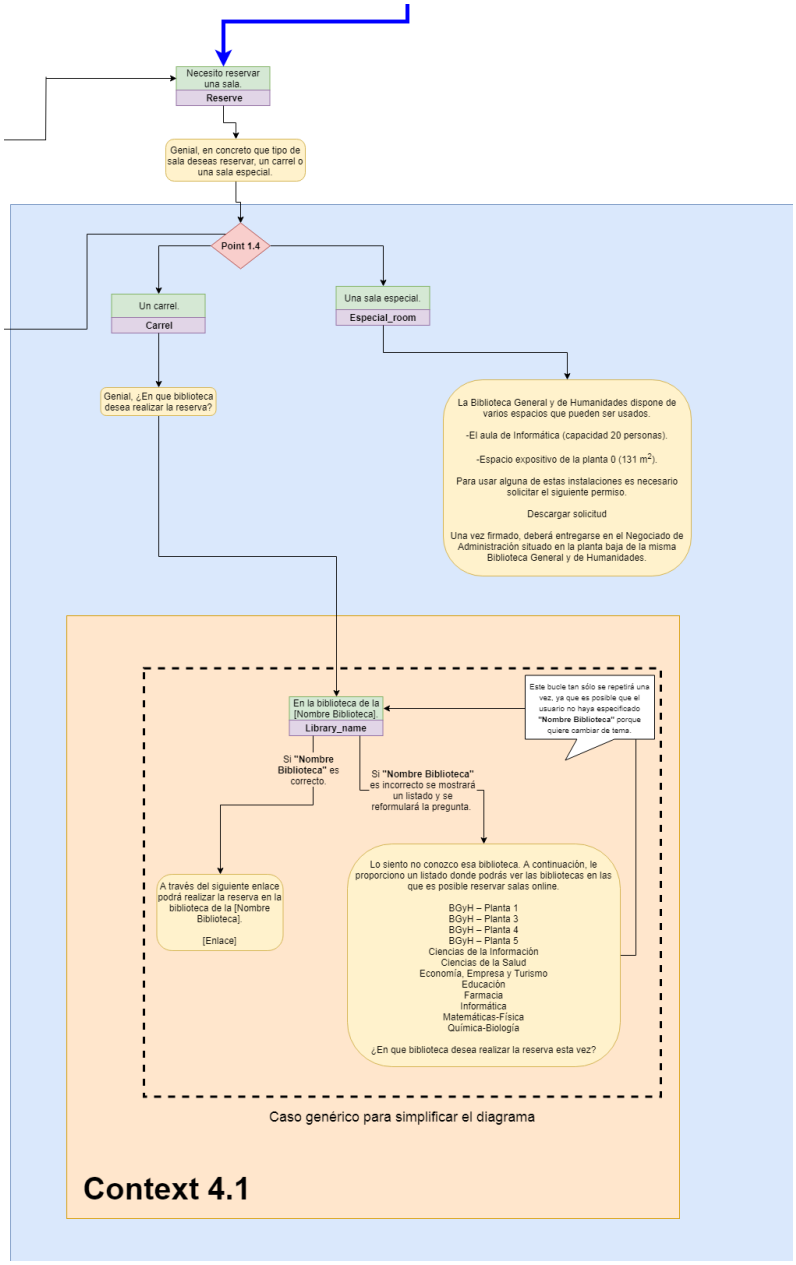


Figura 4.9: Rama para la reserva de salas.

La nueva rama que se analizará es la que se encarga de gestionar el acceso al **servicio de reserva de salas**. En este caso se puede observar que existe el contexto principal de la rama, en el que se distingue entre reserva de carrels y salas especiales, y un subcontexto que se centra en la reserva de dichos carrels en las distintas facultades. Este caso es un poco más complejo que el anterior ya que dispone de contextos anidados, pero la esencia del funcionamiento es la misma, en este caso el subcontexto permite agilizar el acceso a las reservas de carrels, ya que una vez se accede a él, por ejemplo con la sentencia "Deseo reservar un carrel en la facultad de Educación", la próxima vez que se desee reservar en otra facultad bastará con plantear "Y en la de Farmacia". Como se puede apreciar la dinámica es la misma que en la rama anterior. Este efecto en los contextos se logra obligando al asistente a buscar la respuesta a una pregunta primero en el contexto más próximo en el que se encuentra inmerso, en el ejemplo se trataría del subcontexto de área naranja, si no logra encontrar la respuesta, se desplazaría al inicio del contexto padre (área azul) donde repetiría la búsqueda. Este proceso de escalada en la jerarquía de contextos se repite hasta que se encuentra la respuesta o hasta que se alcanza el contexto general (que une todas las ramas) momento en el que si aún no se ha encontrado la respuesta se informará al usuario de la imposibilidad de ayudarlo.

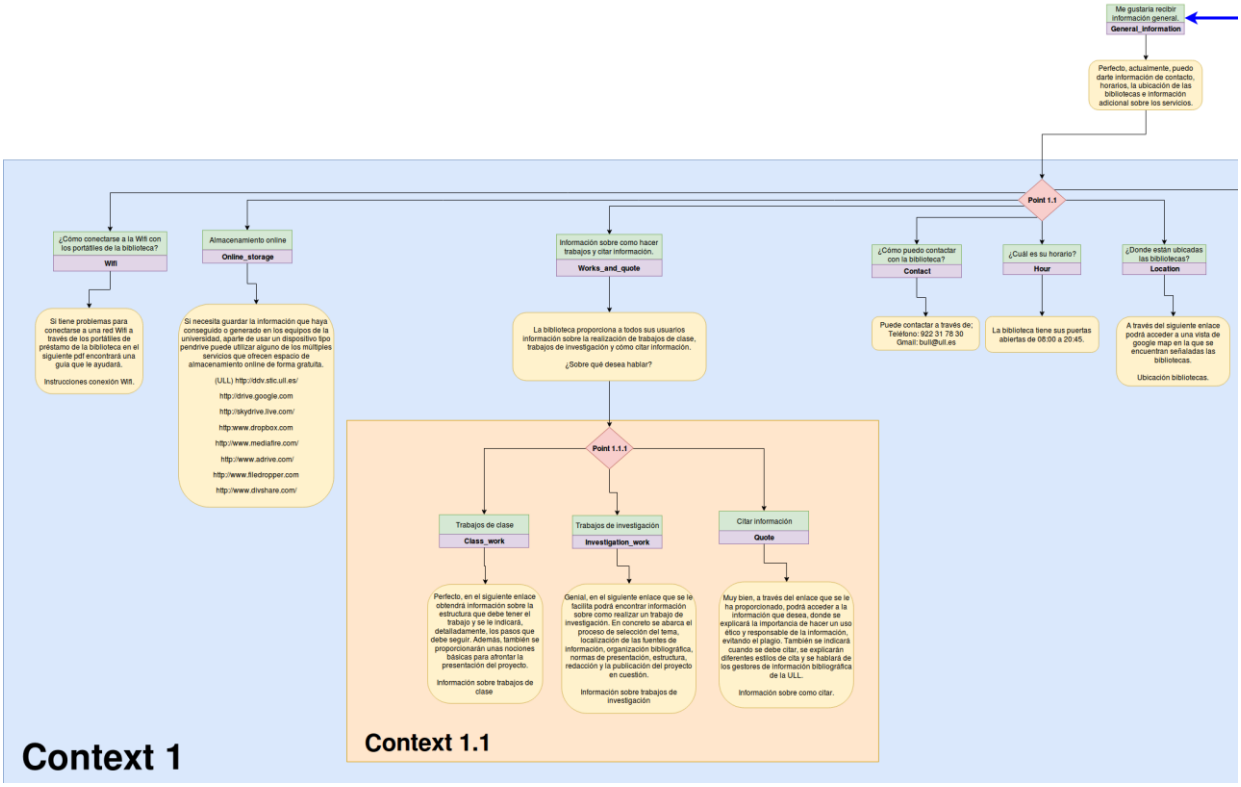


Figura 4.10: Rama de información general.

Esta tercera rama es la de **información general**, en ella se recogen datos como la localización de las bibliotecas en los diferentes campus de la universidad, datos de contacto, horario de apertura e información sobre servicios secundarios que la biblioteca ofrece a través de su web, como una serie de recomendaciones para la realización de trabajos tanto de clase como de investigación e información sobre cómo y cuándo realizar citas en un proyecto. Como se puede apreciar en la imagen, existe un contexto principal (área azul) que abarca la totalidad de la rama y un subcontexto (área naranja) que cubre la sección destinada a informar sobre la realización de proyectos. Los contextos en cuestión representan lo mismo que en los otros casos expuestos, por lo que en este aspecto no posee ninguna novedad. Tan solo indicar, que los diferentes flujos conversacionales que se pueden apreciar en el diseño en general de todas las ramas, incluida la presente, lo que muestran son todos los caminos y respuestas que es capaz de adoptar el asistente, sin embargo, en muchos casos éste puede tomar atajos para acceder directamente a ciertas secciones de las ramas,

permitiendo que la respuesta ofrecida sea lo más precisa posible. Por ejemplo, en la rama de información general, al inicio, aparece una respuesta del asistente en la que se aporta información sobre el contenido de dicha rama, para luego acceder a ella, en este caso, el asistente debe ser capaz de distinguir entre aquellas situaciones en las que se deba mostrar la información sobre la rama y cuándo simplemente deberá adentrarse y acceder directamente a la información solicitada. Para lograrlo, se ha decidido exponer al inicio de la rama cierta información sobre su contenido, a modo de índice, de manera que si se localiza la información deseada pueda ser accedida de forma inmediata. Esto se ha llevado a cabo también en el resto de ramas del diseño y se aplicado por igual a las subramas existentes.

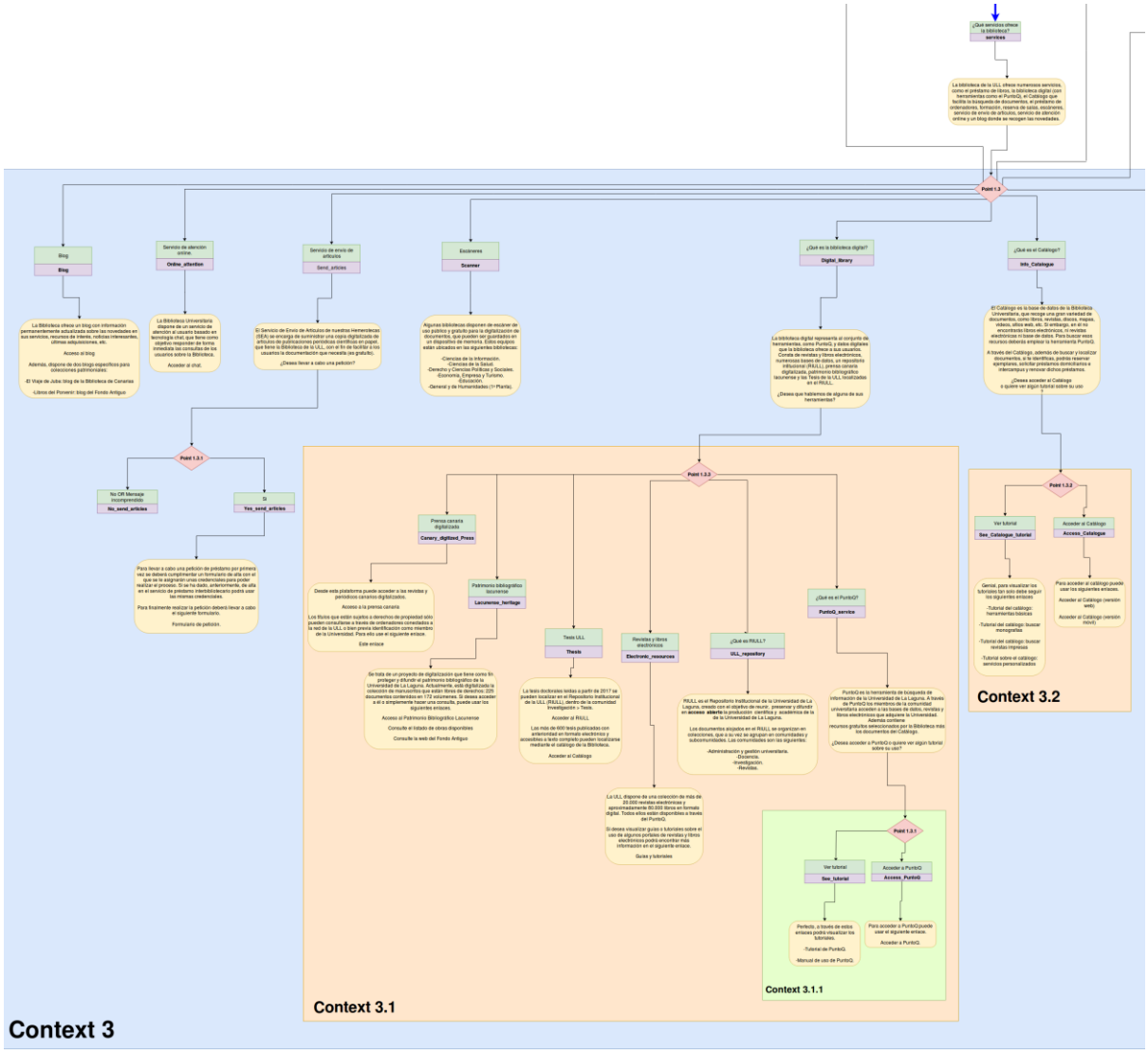


Figura 4.11: Rama de servicios

La rama que se analizará a continuación es la que recoge todos los **servicios** que proporciona la biblioteca que no han sido abarcados en alguna de las ramas principales, por lo que en ella se encontrará información sobre el catálogo, la biblioteca digital, escáneres, el servicio de envío de artículos, atención online y el blog de noticias. En lo que respecta a los contextos, esta rama es algo más compleja que las anteriores pues dispone de tres niveles de contexto, por un lado, se encuentra el principal que ocupa la totalidad de la rama y contiene la información de los diferentes servicios. En su interior, se pueden distinguir dos subcontextos, uno asociado con la biblioteca digital, donde se desarrollan temas relacionados con el PuntoQ, RIULL, revistas y libros electrónicos, tesis, patrimonio bibliográfico lacunense y prensa canaria digitalizada. Y otro que permite gestionar el

acceso al servicio del catálogo y a sus tutoriales, con el fin de que el usuario que haya accedido ha dicho servicio pueda hacer uso del tutorial, y viceversa, sin tener que especificar más información de la necesaria. A su vez, existe un tercer nivel de contexto (color verde) en el apartado de PuntoQ de la biblioteca digital, que cumple el mismo objetivo que el subcontexto del catálogo, pero aplicado a la herramienta PuntoQ.

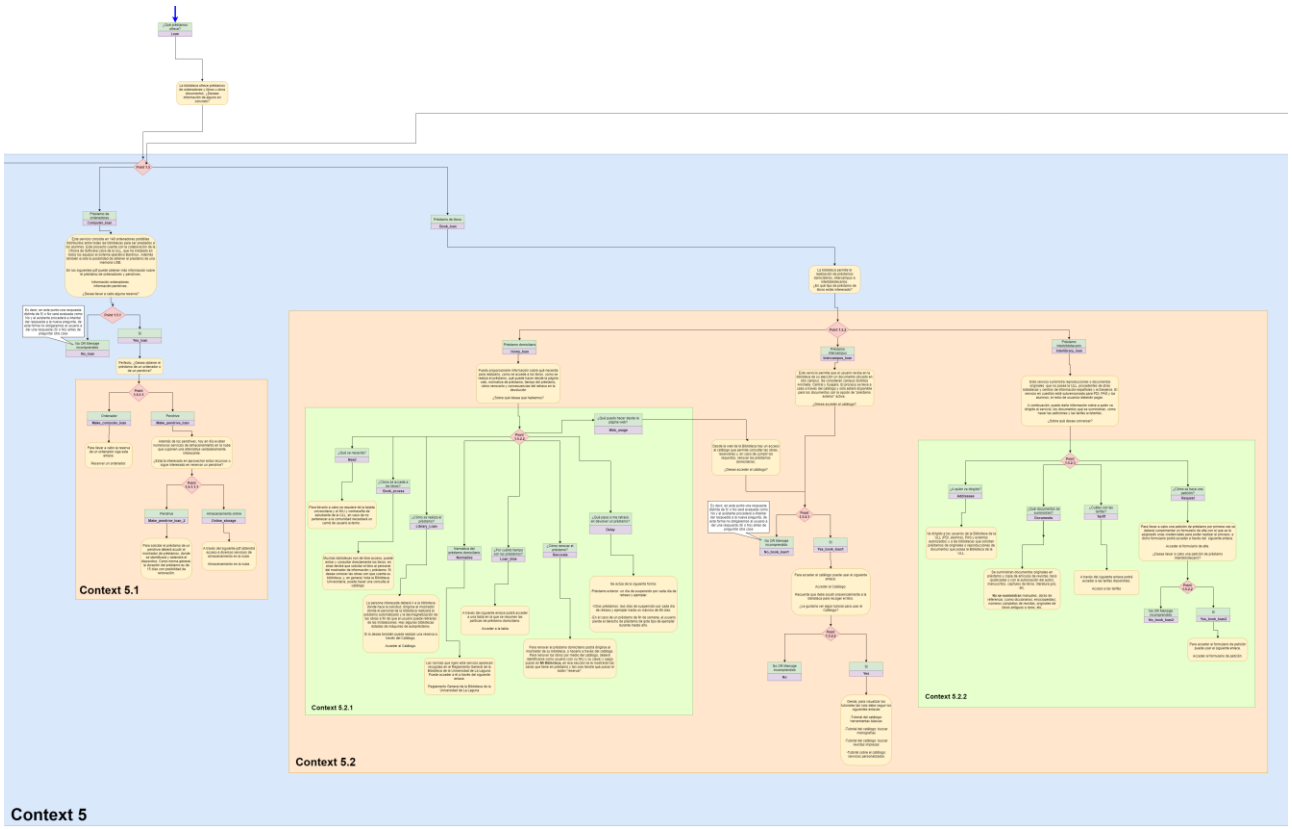


Figura 4.12: Rama para la gestión de préstamos.

A continuación, se examinará la rama en la que se gestiona el **servicio de préstamos** de la biblioteca. A través de ella se permite a los usuarios acceder a información sobre el préstamo de ordenadores y pendrives, así como el préstamo de libros domiciliario, intercampus e interbibliotecario. Como se puede apreciar, esta rama también dispone de tres niveles de contexto. El principal abarca las dos subramas elementales que permiten clasificar la información relativa, por un lado, a ordenadores y pendrives y, por otro, a libros. Cada una de esas dos ramas contiene un subcontexto, que en el primer caso facilita la realización de préstamos de ordenadores o pendrives, mientras que en el segundo caso hace lo propio con los diferentes tipos de suministros de libros. Finalmente, en la subrama de libros se distinguen dos contextos de tercer nivel (color verde), el primero abarca las cuestiones relacionadas con el préstamo domiciliario, mientras que el segundo se ocupa de las relativas al ámbito interbibliotecario.

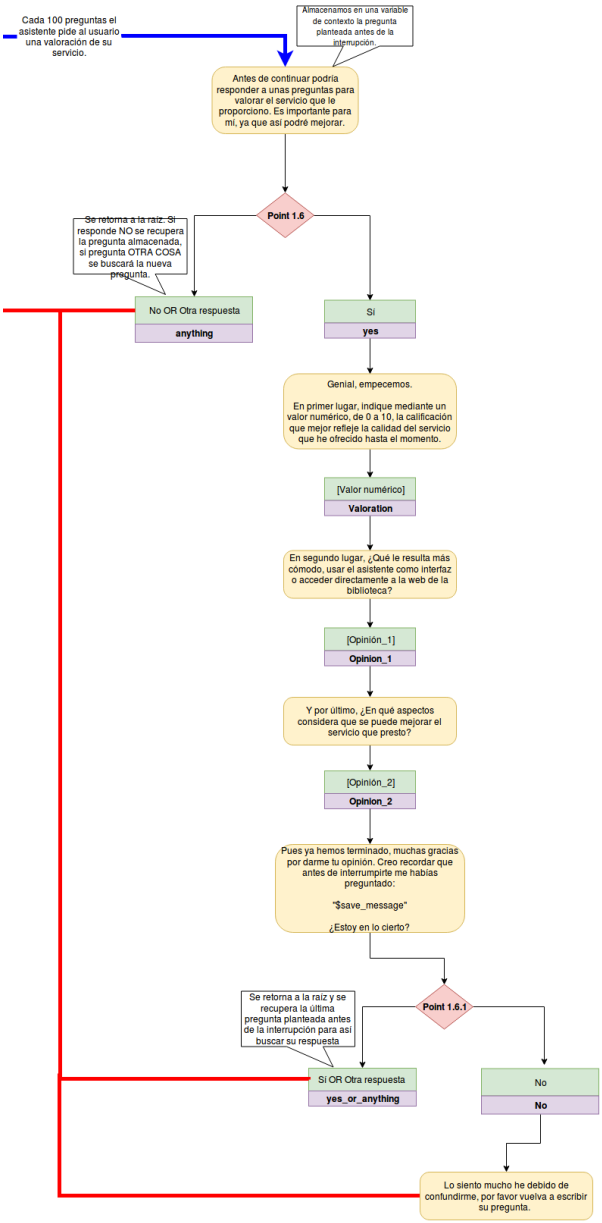


Figura 4.13: Valoración del servicio

Por último, se ha incluido en el diseño una rama adicional que no está relacionada con los servicios de la biblioteca, pues su objetivo es plantear una serie de cuestiones con el fin de que el usuario **valore el servicio** prestado por el asistente. En primer lugar, el usuario valorará el funcionamiento del servicio con una calificación de 0 a 10, a continuación, se le preguntará si le resulta más cómodo usar como interfaz el asistente o acceder directamente a la web, y por último, se le pedirá introducir una opinión sobre qué aspectos se deberían mejorar. Esa información, posteriormente, será almacenada en una base de datos

4.4 Implementación del asistente

A continuación, explicaré los pasos que se han seguido para crear el asistente a través de la herramienta de IBM Watson Assistant. Además de las características del prototipo web implementado.

4.4.1 Uso de IBM Watson Assistant

Watson Assistant permite la creación de los diferentes flujos de dialogo a través de nodos que cuelgan de una raíz, dando lugar a una estructura en árbol, idéntica a la que presenta el diagrama de flujo, de ahí la utilidad del mismo.

Para determinar el flujo a través del árbol de nodos y seleccionar la respuesta que se da en cada momento se emplean los **intents** y las **entities**. Los **intents** (precedidos por "#") permiten identificar el objetivo o el propósito de la entrada ofrecida por el usuario, mientras que las **entities** (precedidas por "@") representan información clave en el mensaje del usuario que es necesaria para determinar la finalidad. Estos conceptos son los que serán empleados en los nodos del árbol como condicionantes, para así determinar el modo de actuación del asistente, el cual puede ser proporcionar una respuesta al usuario, saltar a otro nodo o realizar una conexión con otro servicio (ver apéndice A sección 5). A continuación, podemos ver un ejemplo de la estructura interna de un nodo.

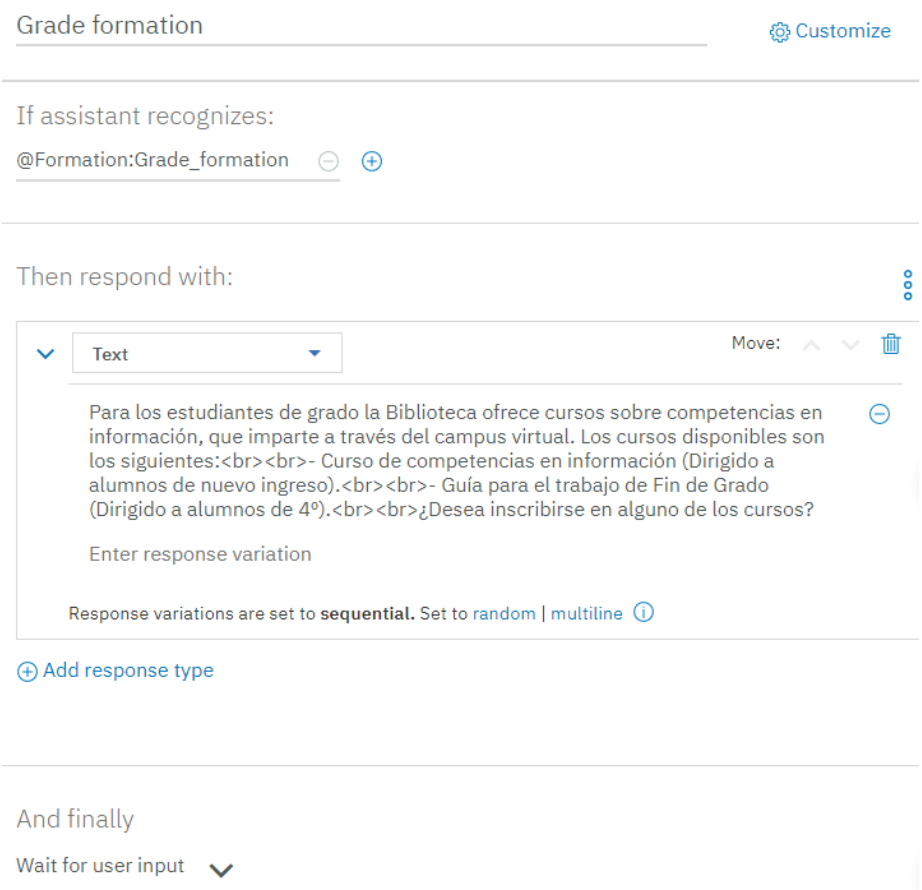


Figura 4.14: Estructura interna de un nodo

Como se puede apreciar en la imagen anterior los nodos disponen de una estructura bien definida. En la que primero se muestra el nombre del nodo, a continuación la condición que se debe cumplir para que el nodo acepte el mensaje, en tercer lugar, se especifica la respuesta que se debe proporcionar al usuario y finalmente se indica la acción que llevará a cabo el asistente después de enviar la respuesta.

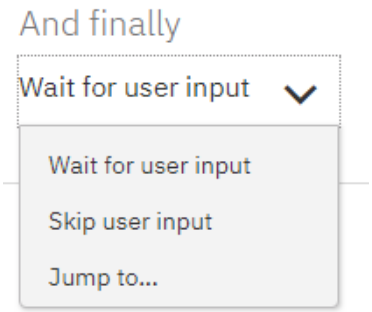


Figura 4.15: Acciones de los nodos

Esa acción puede ser, simplemente, una espera por una entrada de usuario (primera opción), impedir que la entrada se propague a sus nodos hijos (segunda opción) o saltar a otro nodo (tercera opción).

Por último, indicar que los nodos también disponen de una opción denominada "respuesta múltiple" que si se activa permitirá añadir en un mismo nodo varias respuestas, de manera que cada una irá asociada a una condición que en última instancia determinará cuál será el mensaje que se dará al usuario. La siguiente imagen ilustra lo explicado.

Figura 4.16: Nodo multirespuesta

En lo que respecta al árbol de nodos, a continuación, se muestra una imagen de la estructura general adoptada para llevar a cabo la implementación, que se usará para explicar como la herramienta procesa los mensajes del usuario [17].

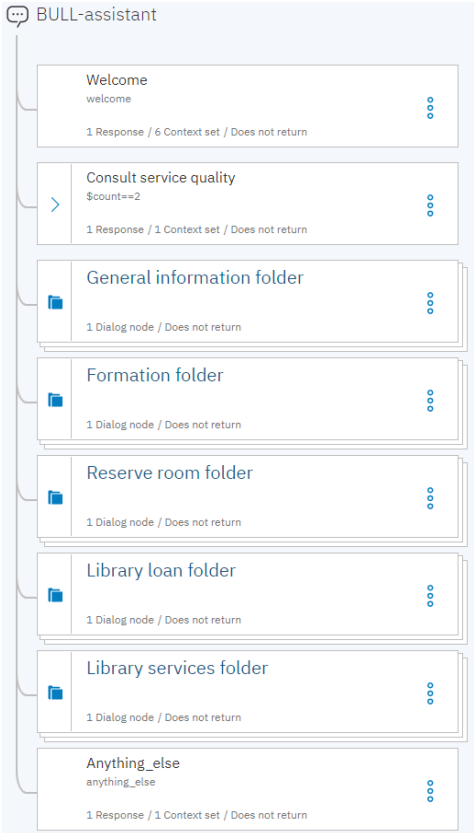


Figura 4.17: Árbol de nodos

Por defecto, cuando se crea un proyecto nuevo, al árbol se le añade un primer nodo, al que se ha denominado “Welcome”, con el que el asistente, al inicio, saludará al usuario, y un segundo nodo “Anything_else” destinado a recoger todas las cuestiones a las que el asistente no es capaz de dar respuesta.

En este caso, guiándonos por el diagrama de flujo, se ha creado una carpeta por cada rama principal, como se puede apreciar en la imagen anterior, con el fin de organizar dentro de ellas los nodos que describirán, para cada tema, la actuación del asistente. El orden en el que se coloquen los nodos o las carpetas en el árbol es de suma importancia, ya que la herramienta los recorrerá de manera secuencial comprobando las condiciones de los nodos, hasta que encuentra el adecuado, entonces el asistente actuará según se indique en ese nodo. De cara a la siguiente pregunta que plantee el usuario pueden presentarse diferentes escenarios.

- Si el nodo en cuestión se ha configurado para que proporcione una respuesta y no dispone de hijos, la próxima vez que el usuario plantee una pregunta se comenzará a buscar desde la raíz del árbol.
- Si el nodo se ha configurado para que proporcione una respuesta y posee hijos, la próxima vez que se realice una pregunta se comenzará a buscar, de manera secuencial, desde el primero de esos nodos hijos. Aunque en estas situaciones también es posible configurar el nodo para que actúe como en el caso anterior.
- Si en el nodo se ha establecido un salto hasta otro nodo, la próxima vez que el usuario plantee la cuestión se comenzará a buscar en el nodo de destino del salto, siempre y cuando, ese nodo haya aceptado el mensaje.

Como se puede apreciar Watson Assistant da bastante libertad al diseñador a la hora de crear los diferentes flujos conversacionales. A continuación, se desglosará el contenido de una de las carpetas del proyecto con el fin de mostrar un ejemplo del diseño realizado.

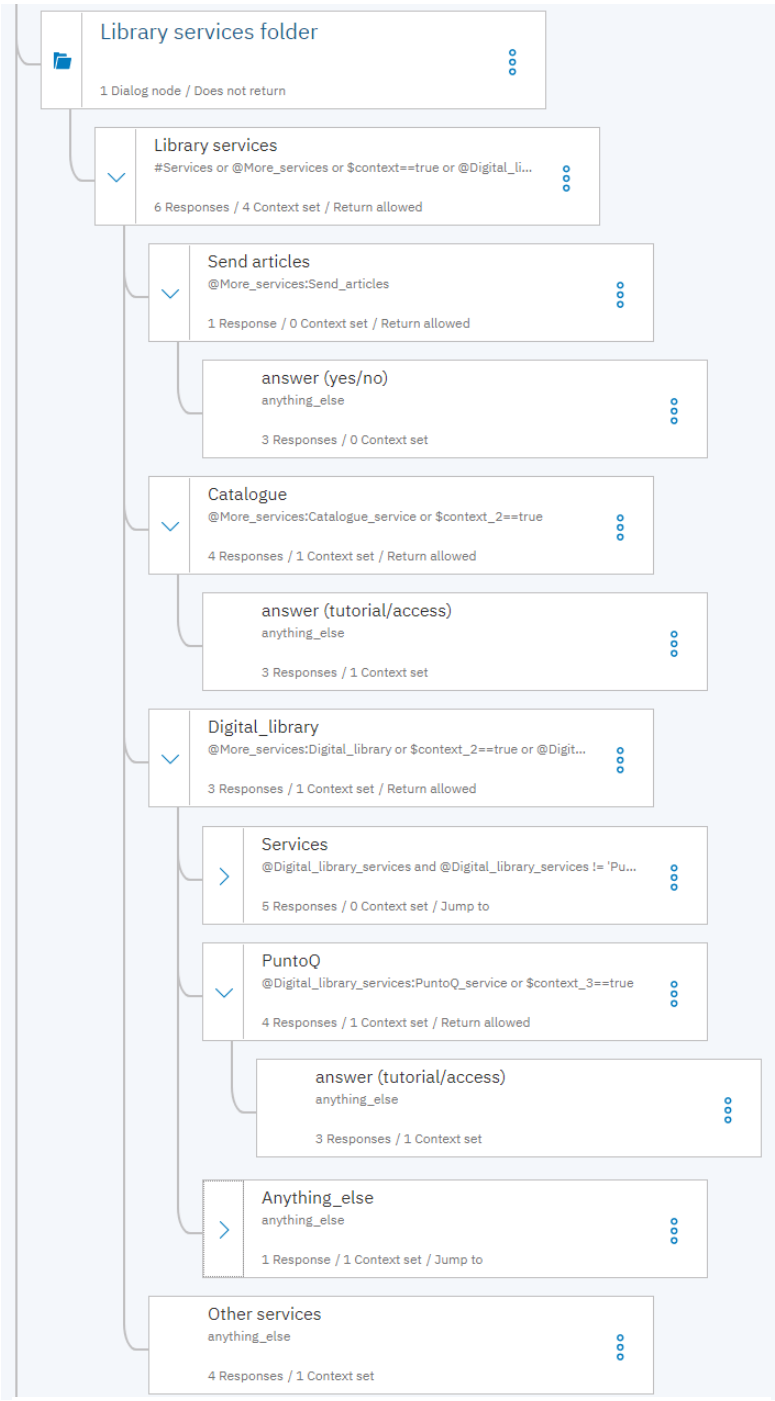


Figura 4.18: Library services folder

En la imagen superior se puede apreciar el contenido de la carpeta “Library services folder”, la cual tan solo dispone de un nodo, llamado “Library services”, que posee cuatro nodos hijos. Los tres primeros “Send articles”, “Catalogue” y “Digital library” aportan información relacionada con el envío de artículos, el catálogo y la biblioteca digital respectivamente. Desde “Send articles” y “Catalogue” se plantea una pregunta al usuario, de ahí que dispongan de un nodo hijo denominado “Answer” destinado a manejar la respuesta que se ofrezca. En cuanto a “Digital library”, dispone, a su vez de tres nodos hijos, “Services” que se trata de un nodo de tipo respuesta múltiple que aporta información sobre todos los servicios de la biblioteca digital, salvo PuntoQ, para el que se ha

destinado un nodo específico, pues deberá gestionar la respuesta que el usuario proporcione sobre si desea acceder a tutoriales sobre PuntoQ o acceder al servicio, desde ese mismo nodo, si el usuario responde algo no relacionado con la pregunta, se saltará a "Digital library", donde se inicia su contexto padre. El último hijo de "Digital library" denominado "Anything else", se encarga de recoger todas aquellas cuestiones no relacionadas con la biblioteca digital (contexto actual) y las envía al nodo "Library services" (inicio de su contexto padre), donde se buscará la respuesta. Por último, indicar que en el nodo final de "Library services", denominado "Other services", el cual posee una estructura de respuesta múltiple, se han agrupado el resto de servicios que proporciona la biblioteca, además, en el caso de que se plantee una pregunta que no esté relacionada con la rama de servicios, se encargará de gestionarla, enviando dicha pregunta al contexto general del árbol para que se busque su respuesta en otras ramas (ver apéndice A sección 5).

En lo que respecta a los contextos que se definieron en el diagrama de flujo, para representarlos en Watson Assistant se utilizarán variables de contexto, las cuales se representan con el símbolo "$". Estas variables permiten almacenar información entre cada iteración del asistente. En este caso se han usado tres, $context para el contexto principal de la rama, $context_2 para los contextos internos y $context_3 para el contexto que está relacionado con PuntoQ, que se localiza dentro de uno de los contextos definidos mediante $context_2. Estas variables inicialmente poseen como valor "False", de manera que cuando el asistente accede al nodo donde se inicia el contexto adquieren el valor "True", permitiendo que cada vez que se plantee una pregunta primero se busque la respuesta en ese ámbito, si no se encuentra, la variable de ese contexto se pondrá a "False" y se saltará al nodo inicial del contexto padre donde se repetirá el proceso.

Los pasos que se han seguido hasta ahora se han llevado a cabo a través de la interfaz que provee la herramienta, sin embargo, también es posible realizar estas tareas modificando manualmente el archivo JSON asociado a cada nodo. Más adelante se verá que algunas operaciones solo se pueden llevar a cabo accediendo y modificando directamente dicho archivo.

4.4.2 Desarrollo del prototipo web

El objetivo de desarrollar este prototipo web es el de ofrecer una imagen general de cómo podría ser el resultado final de integrar el asistente en la página web de la Biblioteca de la ULL.

Para crear la API en cuestión se emplearán, principalmente, 2 paquetes de npm, por un lado **Watson SDK API para Node.js** [15], con el que se podrá acceder a los servicios que proporciona IBM Cloud. Y **express**, con el que se creará un servidor HTTP que se encargará de atender las peticiones.

Una vez instalados los paquetes y creada la carpeta para alojar el proyecto, se procede a crear el servidor, para ello se crea el archivo "nodejs_program.js" donde se importarán las librerías antes instaladas para poder llevar a cabo la configuración. El servidor en cuestión escucha el puerto 3000 y será capaz de manejar peticiones de tipo POST, que contendrán los mensajes del usuario que el asistente deberá procesar.

Pero antes, se deberá establecer la conexión desde el **backend** a la instancia del proyecto de Watson en IBM Cloud, donde se ubica el asistente. Para ello se requiere de las credenciales del proyecto, las cuales son accesibles desde el panel de control de la cuenta creada en IBM, en el apartado de servicios. También se requerirá del ID del área de trabajo, visible desde el "Skill" creado en Watson Assistant.

Una vez preparada la conexión con el asistente, se procede a desarrollar, dentro de la función que se encarga de captar las peticiones POST, el código para enviar el mensaje del usuario al chatbot. El usuario envía su pregunta a través de un objeto JavaScript, por lo que primero se debe extraer esa información, una vez hecho, a través del método **message** se envía al asistente el mensaje, junto a otros datos como el ID del espacio de trabajo y el contexto actual.

```
var AssistantV1 = require('watson-developer-cloud/assistant/v1');

//Conexión al servicio Watson Assistant.
var assistant = new AssistantV1({
  username: 'Poner username',
  password: 'Poner password',
  url: 'Poner url',
  version: 'Poner versión'
});

//Función para enviar el mensaje a Watson.
//text --> contiene el mensaje del usuario
//context --> almacena el contexto actual
assistant.message({
      workspace_id: 'Poner espacio de trabajo',
      input: { text },
      context,
   }, function(err, response) {
         //err --> indica si se produce algún error
         //response --> contendrá la respuesta del asistente
         //Dentro de esta función se respondería a la petición
         //del usuario con el mensaje del asistente.
      }
);
```

Figura 4.19: Ejemplo de uso del método “message”

Una vez se reciba la respuesta del chatbot a la pregunta del usuario, se le reenviará como respuesta a su petición POST y, a continuación, se le mostrará dicha información a través del chat (ver apéndice A sección 1).

En lo relativo al **frontend**, el objetivo es crear una simulación de la página web de la Biblioteca de la ULL en la que se introducirá un chat que mostrará las cuestiones planteadas por el usuario en el lateral derecho y en un color determinado, mientras que las respuestas del chatbot se mostrarán en el lateral opuesto.

Para el diseño web se empleó **HTML**, **JavaScript** y **CSS**, en este caso no se ha empleado ninguna herramienta para facilitar el diseño, como podría ser Bootstrap, ya que la estructura de la página web que se pretende crear no es demasiado compleja. A continuación, se muestra una imagen del diseño final.

Figura 4.20: Prototipo web

En lo que respecta al diseño, el menú que se aprecia en la parte superior es una captura de pantalla de la web oficial de la biblioteca, lo que sí se ha implementado es el chat y los elementos que lo rodean. Para llevarlo a cabo, primero en el archivo HTML se ha creado una sección " … " que se ha centrado en el navegador y que representa el contenedor que contendrá los diferentes elementos de la web. Entre los que se encuentra el chat, también representado por una sección " … ", en la que se irán insertando los mensajes tanto del usuario como del asistente desde el código JavaScript. Por otro lado, se encuentra el campo "input", para que el usuario inserte el mensaje, y los botones para cambiar de idioma o iniciar la grabación de audio. En cuanto al código JavaScript se han elaborado tres funciones, a la primera se le pasa como parámetros el mensaje y el nombre del emisor, con lo que se elabora una plantilla del mensaje. La segunda función diseñada, está destinada a insertar la plantilla anterior en el chat. Y la tercera se encargará de hacer la petición POST con el mensaje al servidor, para lo que se usará la función **fetch**, una vez se obtenga la respuesta se empleará la información recibida para crear el mensaje que se insertará en el chat, usando para ello las dos funciones mencionadas anteriormente. Esta tercera función se invoca cada vez que el usuario presiona la tecla "enter" en el área de input, siempre y cuando se haya escrito un mensaje en él. Básicamente con estas tres funciones se ha podido implementar el chat. En lo que respecta al CSS, su principal tarea es permitir que se distinga, visualmente, cuáles son los mensajes del usuario y los del chatbot (ver apéndice A sección 2).

4.5 Integraciones

A continuación, describiré los diferentes servicios que se han incorporado junto al asistente con el fin de ofrecer al usuario un atendimiento más completo y de mayor calidad.

4.5.1 Base de datos

En primer lugar, se ha incorporado al funcionamiento del asistente una **base de datos**, en la que el chatbot pueda almacenar información relativa a los usuarios, como por ejemplo la valoración que éstos ofrecen sobre el funcionamiento del asistente o recoger todas aquellas cuestiones a las que el asistente no es capaz de responder, con el fin de facilitar la mejora del servicio.

En este caso, la herramienta seleccionada para crear la base de datos es **IBM Db2 on Cloud**, una base de datos SQL en la que se crearán dos tablas, una para las valoraciones, que tendrá como campos la fecha de la valoración, una calificación y dos campos para almacenar las respuestas a las dos preguntas que se plantean. En cuanto a la segunda tabla, tan solo dispondrá, como atributos, de la fecha y la pregunta no reconocida por el asistente. Indicar que en ambos casos la clave primaria es la fecha.

```
CREATE TABLE CLIENT_OPINION (
    Date TIMESTAMP NOT NULL,
    Note int NOT NULL,
    Service_question varchar(255) NOT NULL,
    Opinion varchar(255) NOT NULL,
    PRIMARY KEY (Date)
);
```

Figura 4.21: Script tabla 1

```
CREATE TABLE INFORMATION (
    Date TIMESTAMP NOT NULL,
    Question varchar(255) NOT NULL,
    PRIMARY KEY (Date)
);
```

Figura 4.22: Script tabla 2

Watson Assistant no permite integrar directamente en su funcionamiento la base de datos ya creada, así que se deberá emplear una tecnología que actúe como "puente" y conecte los dos servicios. Para solventar este problema, se empleará la herramienta de IBM denominada **Cloud Functions**. En ella se creará una acción, que sí podrá ser llamada desde cualquiera de los nodos del asistente, en la que se establecerá la conexión con la base de datos. Antes de continuar, indicar que una acción equivale a las clásicas funciones con la diferencia de que deben recibir y retornar un objeto JSON.

Para implementar la acción, la herramienta ofrece, como opción, diferentes lenguajes de programación, en este caso se seleccionará Python 3.7. Una de las ventajas que ofrece Cloud Functions es que ya dispone de las librerías de IBM instaladas, por lo que tan solo se deberá incluir en el código la que sea pertinente, que en este caso es **ibm_db** [19], la cual dispone de los métodos necesarios para establecer la conexión con la base de datos y realizar consultas (ver apéndice A sección 3).

Una vez concluida la acción, ya tan solo falta invocarla desde los nodos adecuados [16]. Para ello se deberá acceder al editor del archivo JSON del nodo en cuestión, en ese editor por defecto se puede ver el campo "output" donde se describe la respuesta que proporciona el nodo y, si en dicho nodo se ha modificado o creado alguna variable de contexto, también se mostrará el campo "context". En este caso, para llamar a la acción habrá que añadir, manualmente, el campo "actions", el cual habrá que indicar una serie de datos para que la conexión sea posible.

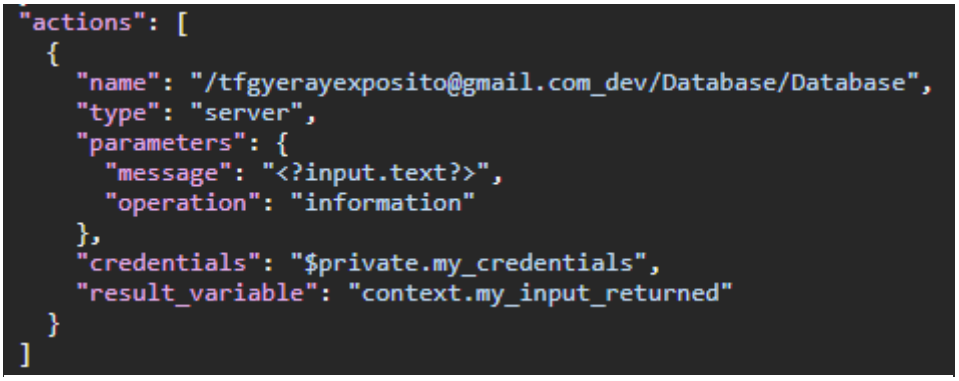


Figura 4.23: Campo "actions" del archivo JSON

En el campo "name" se ha indicado el nombre y ubicación de la acción mediante la siguiente sintaxis.

/<namespace>/[<package-name>]/<action name>

En "type" se señala el tipo de llamada que se desea hacer, que en este caso puede ser server o cloud_functions, pues ambas representan lo mismo. En "parameters" se indicará los campos y los valores que tendrá el objeto JSON que recibirá la acción como parámetro, en este caso "message" contendrá la entrada del usuario y "operation" indicará la operación que se realizará en la acción. En "credentials" se indican las credenciales de Cloud Functions, que en este caso se han almacenado en una variable de contexto. Por último, en el campo "result_variable" se introducirá el nombre que se usará para hacer referencia al objeto JSON que el servicio externo devuelve.

Una vez concluido este proceso en los nodos desde los que se desea hacer la llamada, ya se tendrá integrada la base de datos en el asistente, gracias a Cloud Functions.

4.5.2 Traductor de lenguaje

Con el fin de facilitar el acceso a la información del asistente a un mayor público, se ha habilitado una opción que permite que la comunicación con el chatbot pueda llevarse a cabo en inglés. Para ello se ha usado la herramienta de **IBM Watson Language Translator**.

Primero se ha creado un proyecto en dicha herramienta, desde el panel de control de la cuenta creada en IBM. Dado que anteriormente ya se ha instalado el SDK de Watson, tan solo se deberá incluir en el backend la librería que permite la comunicación con este servicio, para lo que se requiere de las credenciales de acceso. Una vez adquiridas desde la cuenta solo habrá que añadirlas a la función de conexión para poder acceder al proyecto creado.

En principio, en el frontend se han activado dos botones con los que el usuario podrá indicar el lenguaje en el que se llevará a cabo la conversación. El idioma seleccionado será enviado junto con la petición POST cada vez que el usuario realice una pregunta, con el fin de saber, en el backend, si es necesario traducir el mensaje antes de enviarlo al asistente, de la misma forma que también se debería traducir la respuesta del chatbot, de español a inglés, antes de ser mostrada al usuario.

Para añadir ese funcionamiento en el backend, habrá que modificar el contenido de la función que capta las peticiones POST. Primero se extraerá el valor del lenguaje y, a continuación, mediante sentencias "if else" se determinará si será necesario realizar las traducciones o no. La función que se emplea para realizar las traducciones posee la siguiente estructura.

```
const LanguageTranslatorV3 = require('ibm-watson/language-translator/v3');

//Conexión al servicio Language Translator.
const languageTranslator = new LanguageTranslatorV3({
  username: 'Poner username',
  password: 'Poner password',
  url: 'Poner url',
  version: 'Poner versión'
});

//text contiene la pregunta del usuario en inglés.
languageTranslator.translate( {text: text, source: 'en', target: 'es',} )
   .then(body => {
      //La traducción está almacenada en "body". Así que en esta sección
      //se debería llevar a cabo el envío del mensaje en español al
      //asistente.
   })
   .catch(err => {
      console.log(err);
   });
```

Figura 4.24: Ejemplo de traducción.

Como se puede apreciar, al método **translate** se le pasa, como parámetro, un objeto que contiene el texto, el lenguaje de origen y el de destino.

4.5.3 Reconocimiento de voz

Al incorporar el reconocimiento de voz lo que se pretende es dotar al chatbot de un recurso que permita al usuario comunicarse de manera natural con el asistente. Para llevar a cabo la implementación se ha empleado la herramienta de **IBM Watson Speech to Text**.

El proceso para incorporar el servicio es muy parecido al descrito en el apartado anterior. Primero se creará el proyecto en la herramienta de IBM. A continuación, se incluirá en el backend la librería necesaria para comunicarnos con nuestro proyecto, para lo que se requerirá de sus credenciales. Una vez obtenidas se podrá establecer la conexión.

En lo referido al frontend hay que llevar a cabo una serie de modificaciones, ya que el objetivo es que el propio navegador pida permiso al usuario para acceder al micrófono de su equipo, si éste se lo concede, el usuario podrá presionar un botón, que se ha habilitado en la web, para iniciar la grabación, la cual concluirá cuando presione ese mismo botón por segunda vez [18]. Momento en el que los datos de audio que se han almacenado se convertirán en un blob de datos de tipo **ogg**, que serán enviados al servidor, como un objeto **formdata**, a través de la función **fetch**.

```
var SpeechToTextV1 = require('ibm-watson/speech-to-text/v1');
var fs = require('fs');

//Conexión al servicio Speech to Text.
var speechToText = new SpeechToTextV1({
  username: 'Poner username',
  password: 'Poner password',
  url: 'Poner url'
});

//Params contiene los datos para el reconocimieto
//content_type --> Formato del archivo
//model --> Idioma del audio
var params = {
   content_type: 'audio/ogg',
   objectMode: true,
   model: 'es-ES_NarrowbandModel'
};

// Crear stream
var recognizeStream = speechToText.recognizeUsingWebSocket(params);

// Enviar datos de audio
fs.createReadStream(__dirname + '/public/uploads/binario.ogg').pipe(recognizeStream);

recognizeStream.on('data', function(event) {
   //El resultado del reconocimiento se almacena en
   //event, por lo que desde esta función se enviaría
   //el texto reconocido al usuario
});
```

Figura 4.25: Ejemplo de código para el reconocimiento de voz

Con el fin de que el audio se almacene en el servidor se ha usado el paquete de npm denominado multer [20]. Una vez instalado se ha añadido su librería al backend y se ha procedido a especificar la dirección en la que se almacenará el audio y el nombre que tendrá el archivo. A continuación, se añadirá a la función que capta las peticiones POST el middleware de multer, con el que se indicará que todos los archivos que se envíen con un campo “name” determinado serán cargados al servidor por multer.

Una vez se dispone del audio en el backend, tan solo se deberá enviar al servicio de reconocimiento de voz, el cual devolverá el texto reconocido. Dicho texto se enviará al frontend para que el usuario pueda visualizar en el chat la pregunta que ha formulado oralmente y a continuación se realiza, automáticamente, una nueva petición al backend con ese mismo texto, que se enviará, esta vez, al asistente para que lo procese. Tras ello se retornará la respuesta final al usuario.

4.6 Esquema de conexiones

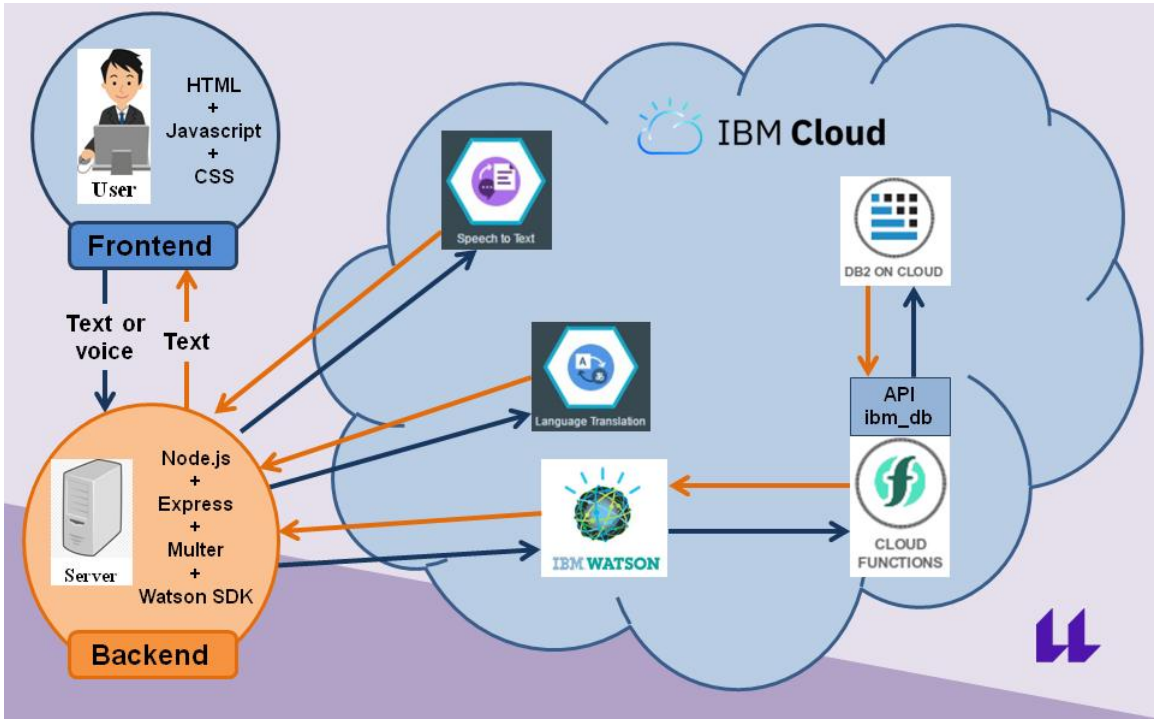


Figura 4.26: Esquema de conexiones.

En la imagen superior se puede apreciar un resumen de las conexiones entre los servicios y las herramientas empleadas a lo largo del proyecto. Para crear el frontend se ha usado HTML, JavaScript y CSS, mientras que en el backend se ha empleado node.js y express. De manera que el usuario desde la web envía una petición al backend que puede contener un mensaje de texto o datos de audio. El backend, que dispone del SDK de Watson para acceder a sus servicio, primero comprueba el tipo de información enviada desde el frontend, si se trata de un mensaje de texto en español podrá enviarlo de inmediato al servicio de IBM Watson Assistant donde el asistente procesará el mensaje, en la mayoría de los casos el chatbot se limita únicamente a enviar la respuesta adecuada, sin embargo, si la respuesta del usuario lo requiere, podría necesitar acceder primero a la base de datos, para ello se llama a la acción creada en Cloud Functions pasándole los datos requeridos, la cual mediante la API ibm_db es capaz de conectarse a dicha base de datos y así realizar la operación deseada. Si el mensaje de texto recibido por el backend estuviese en inglés, antes de enviarlo al asistente se traduciría a español mediante el servicio de IBM Language Translator, al igual que también sería necesario traducir a ingles la respuesta que el asistente da en español, antes de enviársela al usuario.

Por último, si el backend recibe datos de audio, primero serán almacenados en el servidor gracias al middleware multer, a continuación, la información del archivo de audio se enviará al servicio de IBM Speech to text, que llevará a cabo el reconocimiento de voz, y retornará el texto reconocido, el cual será enviado primero al frontend, para que se visualice la pregunta, y luego al asistente para que la procese, y retorne la respuesta definitiva.

4.7 Análisis de resultados

Una vez concluido el proyecto, se procedió a realizar la evaluación del funcionamiento del asistente. Para ello, se ha puesto en manos de diferentes usuarios con el fin de valorar su comportamiento y detectar aquellos aspectos en los que se deba mejorar. De este proceso se ha extraído la siguiente información.

El chatbot ha sido capaz de dar respuesta a la mayor parte de las cuestiones planteadas. Demostrando estar capacitado para manejar situaciones en las que los usuarios emplean palabras sinónimas y, en algunos casos, también preguntas que poseen la misma intención pero diferente estructura. Esto, en gran medida, ha sido posible gracias a la elevada cantidad de ejemplos y sinónimos que se han incorporado en los **intents** y **entities**, respetivamente, creados en el proyecto.

También, se ha podido observar que es capaz de guiar, satisfactoriamente, a través de su contenido a usuarios inexpertos sobre los servicios que proporciona la biblioteca y que por tanto no dominan los conceptos específicos asociados a cada uno de esos servicios. Este aspecto es bastante importante, ya que al tratarse de un servicio de ayuda su uso está centrado en los alumnos de nuevo ingreso. Esta característica ha sido posible, gracias a la estructura en árbol del diseño, en la que cada rama desarrolla un tema concreto y sus ramificaciones subtemas asociados al principal, lo que permite que el asistente se desplace por las ramas concretando cada vez más hasta alcanzar la respuesta adecuada, por lo que aunque el usuario plantee una cuestión en la que no se reconoce algún concepto, el asistente será capaz de responderle concretando la información todo lo que le es posible, en función de los datos reconocidos. A partir de ese punto, el usuario, apoyándose en lo que le indica el asistente, será capaz de completar su pregunta para así obtener la respuesta definitiva.

Los principales errores se han localizado en sentencias que poseen estructuras muy parecidas pero intenciones diferentes, ya que en algunos de estos casos el asistente reconoce dichas sentencias como iguales, cuando en realidad no lo son, lo que lleva a que se proporcione al usuario una respuesta no relacionada con su pregunta. Con el fin de solucionarlo, se podría llevar a cabo un estudio más detallado de los **intents** creados, con el fin de que no exista solapamiento entre las intenciones que son capaces de reconocer.

En general, los usuarios se mostraron satisfechos con el funcionamiento del asistente, pues les permitía obtener la respuesta a sus preguntas de manera más rápida que accediendo directamente a la web oficial de la Biblioteca de la ULL.

Capítulo 5
Conclusiones y líneas futuras

5.1 Conclusiones

Se ha diseñado un chatbot adaptado para que sea capaz de desenvolverse ante cualquier cuestión relacionada con la información presente en la web de la Biblioteca de la ULL, y que es capaz de prestar un servicio de ayuda continuo las 24 horas del día, lo que resulta de gran utilidad para una institución tan importante como ésta, que además es usada por multitud de personas. Estamos ante un servicio que permite que los usuarios saquen el máximo partido a los recursos, además de proporcionarles un sistema para la resolución de dudas en el que no existe el tiempo de espera, gracias a su capacidad de respuesta inmediata. También, se ha procurado que esté al alcance del mayor público posible de la manera más cómoda y natural, a través de opciones como el funcionamiento en inglés y el reconocimiento de voz.

Por otro lado, el asistente desarrollado ha sido capaz de dar respuesta a muchas de las cuestiones planteadas por los usuarios durante la fase de evaluación, donde se pudo comprobar que en general es capaz de proporcionar un servicio eficiente, posicionándose como una prometedora herramienta para completar los servicios de ayuda disponibles en la web de la Biblioteca de la ULL. Y que desde luego, en el futuro, podría evolucionar y adaptarse a otros contextos a nivel universitario.

5.2 Líneas futuras

El proyecto en cuestión dispone de algunas vías adicionales por las que continuar el desarrollo. En primer lugar, sería interesante ampliar progresivamente el campo de acción del asistente, por ejemplo diseñándolo para que sea capaz de responder también cuestiones relacionadas con los servicios de la Fundación General de la ULL, o incluso que sea capaz de abarcar la totalidad de la web de la ULL, lo que desde luego supondría un verdadero desafío. Por otro lado, también se podría desarrollar para que esté disponible en dispositivos móviles e integrarlo en plataformas como Facebook o Whatsapp para hacer más cómodo el acceso al servicio, ya que hoy en día muchos usuarios disponen de alguna de las aplicaciones mencionadas en su dispositivo móvil.

En la actualidad el chatbot actúa como una interfaz que permite a los usuarios acceder a la información y a los servicios de la biblioteca de manera más cómoda y eficiente, sin embargo, se podrían extender sus funcionalidades para que él mismo sea capaz de ofrecer y gestionar su propio servicio, como podría ser un servicio para el préstamo de apuntes entre alumnos de la ULL.

Por último, se podrían integrar en el asistente otras tecnologías como IBM Text to Speech, con el fin de seguir progresando en la búsqueda de un servicio cada vez más adaptado a la forma de comunicación humana.

Capítulo 6
Summary and Conclusions

6.1 Summary

This project has aimed to develop a chatbot that is capable of managing the ULL's library services.

The system will be able to offer an efficient service 24 hours a day, timeframe in which it must be able to solve the doubts of people who wish to take full advantage of library services. Although we believe it will be especially useful for new students, who begin to get in touch with the university environment and who need help to learn how to take advantage of the resources at their disposal, and the library is one.

Before implementing the project, the assistant's work environment was first analyzed to pose the questions that users could ask. Then a flow chart was made, where the assistant's performance is represented.

The project has been developed with intensive use of IBM`s artificial intelligence technology. In particular, the main tool has been Watson Assistant with which the chatbot has been designed, using intents, entities, context variables and a tree of nodes.

Also, a web prototype has been created that simulates the ULL's library website, where I have integrated the chabot. The objective is to offer a global image of the possible final result.

Next, I created a database with the IBM's Db2 on Cloud tool. And I have integrated it into Watson Assistant using the IBM Cloud Functions tool. This way the assistant can store information about the users.

In addition, a translation service has been incorporated into the backend of web prototype to allow the chatbot to communicate in English. Also, speech recognition was added using the IBM's Speech to Text tool.

6.2 Conclusions

It has designed a chatbot which is able to answer any questions related to the information on the ULL library website, and that it is able to provide a continuous help service 24 hours a day, which it is very useful for an institution as important as this one, which is also used by many people. It is a service that allows users to maximize resources and it provides them with a more efficient service

than other help systems, due to their capacity for immediate response. In addition, it has been tried to make it available to all people in the most comfortable and natural way, through options such as functioning in English and voice recognition. In summary, it is a tool that represents an advance in the help systems and that could be applied in other contexts at the university level.

Capítulo 7
Presupuesto

En este capítulo se realizará una estimación del presupuesto que se requeriría para llevar a cabo el proyecto en cuestión.

Fase 1: Análisis y Diseño del proyecto.			
Tarea	**Precio €/horas**	**Número horas**	**Precio por tarea**
Estudio y análisis de las plataformas y tecnologías aplicables al proyecto.	8 €	70	560 €
Examinar el contexto de uso en el que las empresas actuales emplean asistentes virtuales.	8 €	35	280 €
Planteamiento de las cuestiones a las que debería ser capaz de dar respuesta el chatbot.	10 €	35	350 €
Desarrollar un diagrama de flujo para analizar las posibles interacciones, intenciones y respuestas.	10 €	35	350 €
	Presupuesto de la fase 1 = 1540 €		

Tabla 7.1: Presupuesto de la fase de análisis y diseño del proyecto

Fase 2: Implementación			
Tarea	**Precio €/horas**	**Número horas**	**Precio por tarea**
Desarrollar en la plataforma seleccionada la primera versión del asistente.	15 €	144	2160 €
Incorporar contextos, entidades y mejorar el entrenamiento del asistente para perfeccionar la experiencia de usuario.	15 €	18	270 €
Integrar el asistente con una base de datos.	10 €	24	240 €
Crear un prototipo web de la página de la ULL con el chatbot incorporado para ofrecer una imagen del resultado final.	10 €	55	550 €

Permitir que el asistente opere en una lengua extranjera (inglés).	10 €	15	150 €
Incorporar reconocimiento de voz.	10 €	36	360 €
	Presupuesto de la fase 2 = 3730 €		

Tabla 7.2: Presupuesto de la fase de implementación

Presupuesto total del proyecto = Presupuesto fase 1 + Presupuesto fase 2 = **5270 €**

Capítulo 8
Apéndice A.

8.1 Sección 1: Código del backend

Para acceder al código del backend en GitHub presione aquí.

```
var AssistantV1 = require('watson-developer-cloud/assistant/v1');
const express = require('express');
const bodyParser = require('body-parser');
require('dotenv').config();
const LanguageTranslatorV3 = require('ibm-watson/language-translator/v3');
var ffmpeg = require('ffmpeg');
var multer = require('multer');
var SpeechToTextV1 = require('ibm-watson/speech-to-text/v1');
var fs = require('fs');

//----------------------------GLOBAL VARIABLES------------------------------

var special_node=['node_12_1557869479309','node_6_1557866654590'];
var check="empty";


//-----------------------------FUNCTIONS----------------------------------

function sendmessage(assistant,languageTranslator,text,context,res,language) {

   assistant.message({
      workspace_id: 'ae173244-8cbb-4030-99b8-e309883e6ec0',
      input: { text },
      context,
   },  function(err, response) {
         if (err) {
            console.error(err);
            res.status(500).json(err);
         }
         else {
            if(language=='en') {
               languageTranslator.translate( {text: response.output.text, source: 'es', target: 'en',} )
                  .then(body => {

                     //console.log(JSON.stringify(response.context.system.dialog_stack[0].dialog_node, null, 2));
                     for(var i=0;i<special_node.length;i++) {
                        if(response.context.system.dialog_stack[0].dialog_node==special_node[i]) {
                           check=special_node[i];
                        }
                     }

                     response.output.text=(body.translations[0].translation).replace(/https: \/\//g,"https://");

                     //console.log(JSON.stringify(response, null, 2));
                     res.json(response);
                     //console.log(JSON.stringify(response.output.text, null, 2));
                     //console.log('\n');
                  })
                  .catch(err => {
                     console.log(err);
                  });
            }
            else {
               //console.log(JSON.stringify(response.context.system.dialog_stack[0].dialog_node, null, 2));
               for(var i=0;i<special_node.length;i++) {
                  if(response.context.system.dialog_stack[0].dialog_node==special_node[i]) {
                     check=special_node[i];
                  }
               }

               console.log(JSON.stringify(response.output.text, null, 2));
               res.json(response);
            }
         }
      }
   );

}
```

```
//---------------------------SERVER CONFIGURE-------------------------------

const app = express();

app.use(bodyParser.json());
app.use(express.static('./public'));

var storage = multer.diskStorage({
    destination: function (req, file, cb) {
        cb(null, __dirname + '/public/uploads')
    },
    filename: function (req, file, cb) {
        cb(null, 'binario.ogg')
    }
});

var upload = multer({ storage: storage });
var type = upload.single('upl');

const port = 3000;

// Configurar cabeceras y cors
app.use((req, res, next) => {
        res.header('Access-Control-Allow-Origin', '*');
        res.header('Access-Control-Allow-Headers', 'Authorization, X-API-KEY, Origin, X-Requested-With, Content-Type, Accept, Access-Control-
Allow-Request-Method');
        res.header('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, DELETE');
        res.header('Allow', 'GET, POST, OPTIONS, PUT, DELETE');
        next();
});

//---------------------------SERVICE CONNECTION-----------------------------

//Conexión al servicio Watson Assistant.
var assistant = new AssistantV1({
  username: 'apikey',
  password: 'uFxQosNc9TM8SxbRV40dpG1ilaaKaKMtefgEnUcHKu1f',
  url: 'https://gateway.watsonplatform.net/assistant/api',
  version: '2019-02-01'
});

//Conexión al servicio Language Translator.
const languageTranslator = new LanguageTranslatorV3({
  username: 'apikey',
  password: 'PCLGZ2jL76r9fpU1MKRU3oRsPHgUTI4x17OxA_hJiOtZ',
  url: 'https://gateway.watsonplatform.net/language-translator/api',
  version: '2019-01-10'
});

//Conexión al servicio Speech to Text.
var speechToText = new SpeechToTextV1({
  username: 'apikey',
  password: 'GFZTpKIltk4v5ME6LMBENzAk6Vm_8R7vaf33jJlP1Nu1',
  url: 'https://stream.watsonplatform.net/speech-to-text/api'
});

//---------------------------------OPERATION--------------------------------

//Recepción de la petición desde el cliente.
app.post('/conversation/:language*?', type, function (req, res) {

   const { language } = req.params;

   if(language!='voice') {
      var { text, context={} } = req.body;
      console.log(language);

      if((check=='node_12_1557869479309'&&(text=='si'||text=='yes'))||(check=='node_6_1557866654590'&&text=='no')) {
         text=context.save_message;
         check="empty";
      }

      if(language=='en') {
         languageTranslator.translate( {text: text, source: 'en', target: 'es',} )
            .then(body => {
               sendmessage(assistant,languageTranslator,body.translations[0].translation,context,res,language)
               console.log(JSON.stringify(body.translations[0].translation, null, 2));
               //console.log('\n');
            })
            .catch(err => {
               console.log(err);
            });
      }
      else {
         sendmessage(assistant,languageTranslator,text,context,res,language)
      }
   }
   else {

      console.log(req.file);

      var params = {
         content_type: 'audio/ogg',
         objectMode: true,
         model: 'es-ES_NarrowbandModel'
      };

      // create the stream
      var recognizeStream = speechToText.recognizeUsingWebSocket(params);

      // pipe in some audio
      fs.createReadStream(__dirname + '/public/uploads/binario.ogg').pipe(recognizeStream);

      recognizeStream.on('data', function(event) {
         res.json(event.results[0].alternatives[0].transcript);
      });
   }

});

app.listen(port, () => console.log(`Running on port ${port}`));
```

8.2 Sección 2: Código del frontend

Para acceder al código del frontend en GitHub presione aquí.

8.2.1 Código HTML

```
<html>

<head>
  <title>BULLBOT</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <link rel="stylesheet" href="./style.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
</head>

<body>
   <div class="chat-column">
      <table id="language_table">
         <tr>
            <td class="language"><input id="Spanish" type="radio" name="Idioma" value="Español" checked=checked >Español</td>
            <td class="language"><input id="English" type="radio" name="Idioma" value="Inglés">Inglés</td>
            <td id="bullbot"><p>BULLBOT</p></td>
         </tr>
      </table>
      <div id="chat"></div>
      <table id="input_table">
         <tr>
            <td id="voice"><button id=record ><i class="fa fa-microphone" style="color:white"></i></buton></td>
            <td>
               <label for="textInput" class="inputOutline">
                  <input id="textInput" class="input responsive-column" autofocus placeholder="Say something" type="text">
               </label>
            </td>
         </tr>
      </table>
   </div>
   <script src="./script.js"></script>
</body>

</html>
```

8.2.2 Código JavaScript

```
const textInput = document.getElementById('textInput');
const chat = document.getElementById('chat');
let context = {};

const ChatTemplate = (message, from) => `
  <div class="${from}">
    <div class="message">
      <p>${message}</p>
    </div>
  </div>
  `;

const InsertTemplate = (template) => {
  const div = document.createElement('div');
  div.innerHTML = template;

  chat.appendChild(div);
  chat.scrollTop=chat.scrollHeight-30;
};
```

```
const SendMessage = async (text = '') => {

  var uri = 'http://localhost:3000/conversation/';

  var English=document.getElementById('English');

  if(English.checked) {
     uri=uri+'en';
  }
  else {
     uri=uri+'es';
  }

  const response = await (await fetch(uri, {
        method:'POST',
        headers: { 'Content-Type':'application/json'},
        body: JSON.stringify({
                text,
                context,
        }),
  })).json();

  context = response.context;

  const template = ChatTemplate(response.output.text, 'watson');

  InsertTemplate(template);
};

textInput.addEventListener('keydown', (event) => {
  if (event.keyCode === 13 && textInput.value) {

     const template = ChatTemplate(textInput.value, 'user');
     InsertTemplate(template);

     SendMessage(textInput.value);

     textInput.value = '';
  }
});

//----------------------------CRECORD AUDIO-------------------------

navigator.mediaDevices.getUserMedia({audio:true})
   .then(stream => {handlerFunction(stream)})

function handlerFunction(stream) {

   rec = new MediaRecorder(stream);

   rec.ondataavailable = e => {
      audioChunks.push(e.data);
      if (rec.state == "inactive") {
         let blob = new Blob(audioChunks,{type:'audio/ogg'});
         sendData(blob);
      }
   }
}
```

```
async function sendData(data) {

  var uri = 'http://localhost:3000/conversation/voice';

  var fd = new FormData();
  fd.append('upl', data, 'binario.ogg');

  const response = await (await fetch(uri, {
        method: 'POST',
        body: fd
  })).json();

  var template = ChatTemplate(response, 'user');
  InsertTemplate(template);

  SendMessage(response);
}

record.onclick = e => {

   if(record.style.backgroundColor == "green") {
      record.style.backgroundColor = "#57068C";
      rec.stop();
   }
   else {
      record.style.backgroundColor = "green";
      audioChunks = [];
      rec.start();
   }
}


SendMessage();
```

8.2.3 Código CSS

```
* {
    box-sizing: border-box;
}

/*Afecta a las tablas creadas para posicionar los botones y el input*/
#input_table, #language_table {
   width: 100%;
   text-align: center;
}

/*Afecta a la celda de la tabla que contiene el título BULLBOT*/
#bullbot {
   background-color: #5c068c;
   color:white;
   font-weight: 800;
}
```

```css
/*Afecta a las celdas de la tabla que poseen los idiomas*/
.language {
   width: 100px;
   background-color: #b76fff;
}

/*Afecta a la celda de la tabla donde se ubica el boton de audio*/
#voice {
   width: 20px
}

/*Afecta al botón para grabar audio*/
#record {
    background-color: #57068C;
    border-radius: 50%;
    border-color: white;
    width:35px;
    height:35px;
}

/*Afecta al contenido de la etiqueta <body>*/
body {
    background-image: url("./imagenes/background.png");
    font-family: Helvetica, Arial, sans-serif;
    margin: 0;
    padding: 0;
}

/*Afecta al contenido de la etiqueta <html>*/
html {
    font-size: 16px;
}

/*Afecta al contenido de las etiquetas <div>*/
div {
    word-wrap: break-word;
    line-height: 1.25rem;
}

/*Afecta a la columna que contiene el Chat, el campo de input y los botones*/
.chat-column {
    height: 100%;
    padding: 11rem 0 0.625rem 0;
    margin: auto;
    text-align: left;
    max-width: 50rem;
    min-width: 9.375rem;
}

/*Afecta al Chat que esta dentro de chat-column*/
#chat {
    margin: 0.75rem;
    overflow-y: auto;
    overflow-x: hidden;
    height: 62vh;
}
```

```css
/*Afecta a los <div> de los mensajes tanto del usuario como del asistente*/
.message {
    opacity: 0;
    opacity: 1;
    margin-top: 0.9375rem;
}

/*---------------------------MENSAJES DE USUARIO-------------------------------*/

/*Afecta al <div> que contiene el mensaje del usuario*/
.user {
    text-align: right;
}

/*Afecta, en concreto, al <div> que contiene mensaje del usuario*/
.user .message {
    border-radius: 1.25rem;
    border-bottom-right-radius: 0;
    text-align: left;
    display: inline-block;
    margin-left: 2.5rem;
    min-width: 2.5rem;
    position: relative;
    font-size: 1rem;
    color: #fff;
    letter-spacing: 0.015rem;
    line-height: 1.3125rem;
    background: #00B4A0;
}

/*Afecta al <p> que posee el mensaje del usuario*/
.user .message p {
    margin: 0.3125rem;
    padding: 0 0.9375rem;

}

/*---------------------------MENSAJES DEL ASISTENTE-------------------------------*

/*Afecta al <div> que contiene el mensaje del asistente*/
.watson .message {
    position: relative;
    border-radius: 1.5625rem;
    font-size: 1rem;
    color: #323232;
    letter-spacing: 0.015rem;
    line-height: 1.3125rem;
}

/*Afecta al <p> que posee el mensaje del asistente*/
.watson p {
    margin: 0.3125rem;
    padding: 0 1.25rem;
    border-radius: 8px;
    background-color:rgba(255,255,255,0.95);
    text-align: center;
}

/*Distintivo violeta al inicio del mensaje del asistente*/
.watson p:before {
    content: "";
    background: #9855D4;
    border-radius: 5px;
    position: absolute;
    z-index: 2;
    left: 0.4375rem;
    width: 0.3125rem;
    height: 1.3125rem;
    line-height: 1.3125rem;
}
```

```css
/*----------------------------SECCIÓN INPUT-------------------------------*/

/*Afecta a la sección de input en la que el usuario escribe sus preguntas*/
#textInput {
    border: none;
    border-radius: 15px;
    outline: none;
    background: #E2D2EF;
    font-size: 1rem;
    color: #323232;
    letter-spacing: 0.015rem;
    line-height: 1.3125rem;
    height: 2.5rem;
    width: 100%;
    padding-left: 0.125rem;
    margin-bottom: -0.125rem;
    overflow: auto;
}
```

8.3 Sección 3: Código de la acción de Cloud Functions

Para acceder al código de la acción en GitHub presione aquí.

```python
import sys
import ibm_db

def main(dict):

    conn=ibm_db.connect("DATABASE=BLUDB;HOSTNAME=dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net;

    if dict["operation"]=="information":
        #Preparando la sentencia SQL
        sql="INSERT INTO INFORMATION VALUES (CURRENT_TIMESTAMP,?);"
        stmt = ibm_db.prepare(conn, sql)

        #Vincular explícitamente los parámetros
        ibm_db.bind_param(stmt, 1, dict["message"])

        #Ejecución de la sentencia.
        ibm_db.execute(stmt)

    elif dict["operation"]=="client_opinion":
        #Preparando la sentencia SQL
        sql="INSERT INTO CLIENT_OPINION VALUES (CURRENT_TIMESTAMP,?,?,?);"
        stmt = ibm_db.prepare(conn, sql)

        #Vincular explícitamente los parámetros
        ibm_db.bind_param(stmt, 1, dict["valoration"])
        ibm_db.bind_param(stmt, 2, dict["opinion_1"])
        ibm_db.bind_param(stmt, 3, dict["opinion_2"])

        #Ejecución de la sentencia.
        ibm_db.execute(stmt)


    return { 'message': dict["message"]}
```

8.4 Sección 4: Diagrama de flujo.

Debido a su gran tamaño, se ha optado por añadir un enlace a la imagen del diseño almacenada en Google Drive para que se pueda apreciar mejor. Para acceder a ella presione aquí.

8.5 Sección 5: IMB Watson Assistant.

8.5.1 Intentos y entidades usadas

Intentos	Ejemplo de intención que acepta
#Access	Quiero acceder a PuntoQ
#Addressee	¿A quién va dirigido?
#Book_access	¿Cómo se accede a los libros?
#Formation	¿Qué formación imparte la biblioteca?
#General_information	Necesito información general
#Get_documents	¿Qué documentos se suministran?
#Library_Loan	¿Qué artículos presta la biblioteca?
#Loan	Llevar a cabo un préstamo
#Delay	¿Qué pasa si me retraso en devolver un préstamo?
#Need	¿Qué se necesita para usar el servicio?
#No	No
#Reserve_room	Necesito reservar una sala
#Services	¿Qué servicios proporciona la biblioteca?
#Tutorial	Acceder al tutorial
#Works_and_quote	Necesito datos sobre cómo hacer trabajos de clase
#Yes	Sí
#Location	¿Dónde está localizada la biblioteca?
#Know	Me gustaría saber a qué hora abre la biblioteca

Entidades	Valores
@Digital_library_services	Thesis, PuntoQ_service, ULL_repository, Electronic_resources, Lacunense_heritage y Canary_digiyized_Press.
@Formation	Personalized_formation, Grade_formation y Postgraduate_formation.
@General_information_entity	hour, location, Online_storage, contact y Wifi
@Home_loan_questions	Loan_time, Web_usage, Normative, Renovate y Delay.
@Interlibrery_loan_questions	Tariff y Request
@Main_topics	Library_Loan y Reserve_room
@More_services	Online_attention, Catalogue_service, Digital_library, Scanner, Blog y Send_articles.
@Reserve	Matemáticas-Física, Informática, Educación, BGyH-Planta_1, BGyH-Planta_3, BGyH-Planta_4, BGyH-Planta_5, Ciencias de la Información, Ciencias de la Salud, Farmacia, Química-Biología y Economía, Empresa y Turismo.
@Type_book_loan	Home_book_loan, Intercampus_book_loan, Interlibrary_book_loan
@Type_loan	Pendrive_loan, Book_loan y Computer_loan
@Type_room	Carrel y Especial_room
@Works_and_quote	Investigation_work, Quote, Class_work
@sys-number	Cualquier valor numérico

8.5.2 Árbol de nodos

Debido a su gran tamaño, se ha optado por añadir un enlace a una imagen en la que se puede contemplar la totalidad del árbol de nodos diseñado en Watson Assistant. Para acceder a ella presione aquí.

Bibliografía

Para realizar la presente bibliografía se ha seguido la norma UNE 50-104-94.

1. *Cómo funciona Jarvis, el sistema creado por Mark Zuckerberg para "controlar" su casa* [en línea]. Recuperado de https://www.infobae.com/tendencias/2016/12/20/como-funciona-jarvis-el-sistema-creado-por-mark-zuckerberg-para-controlar-su-casa/ (accedido el 25/05/2019).
2. Apple. [en línea]. Recuperado de https://www.apple.com/es/siri/ (accedido el 25/05/2019)
3. Fonbellida, L. *Así funciona Alexa, la inteligencia artificial de Amazon* [en línea]. Recuperado de http://luisfombellida.com/asi-funciona-alexa-la-inteligencia-artificial-de-amazon/ (accedido el 25/05/2019)
4. Google. [en línea]. Recuperado de https://assistant.google.com/intl/es_es/ (accedido el 25/05/2019)
5. Microsoft. [en línea]. Recuperado de https://www.microsoft.com/es-es/windows/cortana (accedido el 25/05/2019)
6. Polo, J. D. *Faster city, un asistente conversacional que recomienda medios de transporte y rutas* [en línea]. Recuperado de https://wwwhatsnew.com/2017/08/02/faster-city-un-asistente-conversacional-que-recomienda-medios-de-transporte-y-rutas/ (accedido el 26/05/2019)
7. *Billy seguros* [en línea]. Recuperado de https://billyseguros.com/ (accedido el 26/05/2019)
8. Juste, M. *CorreYvuela, billetes de avión por WhatsApp* [en línea]. Recuperado de http://www.expansion.com/economia-digital/companias/2017/06/20/594916edca4741045d8b45ab.html (accedido el 26/05/2019)
9. IBM. *Watson Assistant* [en línea]. Recuperado de https://cloud.ibm.com/docs/services/assistant?topic=assistant-getting-started&locale=es (accedido el 28/05/2019)
10. IBM. *IBM Cloud Functions* [en línea]. Recuperado de https://cloud.ibm.com/docs/openwhisk?topic=cloud-functions-getting-started&locale=es (accedido el 28/05/2019)
11. IBM. *Db2 on Cloud* [en línea]. Recuperado de https://cloud.ibm.com/docs/services/Db2onCloud?topic=Db2onCloud-getting_started_db2oncloud (accedido el 28/05/2019)
12. IBM. *Language Translator* [en línea]. Recuperado de https://cloud.ibm.com/apidocs/language-translator (accedido el 28/05/2019)
13. IBM. *Speech to Text* [en línea]. Recuperado de https://cloud.ibm.com/docs/services/speech-to-text?topic=speech-to-text-gettingStarted (accedido el 28/05/2019)

14. Node.js Fundation. *Sobre la documentación* [en línea]. Recuperado de https://nodejs.org/es/docs/ (accedido el 28/05/2019)

15. IBM. *Watson APIs Node.js SDK* [en línea]. Recuperado de https://github.com/watson-developer-cloud/node-sdk (accedido el 19/05/2019)

16. IBM. *Cómo realizar llamadas mediante programación desde un nodo de diálogo BETA* [en línea]. Recuperado de https://cloud.ibm.com/docs/services/assistant?topic=assistant-dialog-actions#dialog-actions (accedido el 30/05/2019)

17. IBM. *Visión general del diálogo* [en línea]. Recuperado de https://cloud.ibm.com/docs/services/assistant?topic=assistant-dialog-overview#dialog-overview (accedido el 29/05/2019)

18. Gottfried, J. *Javascript Tutorial: Record Audio and Encode it to mp3* [en línea]. Recuperado de https://medium.com/jeremy-gottfrieds-tech-blog/javascript-tutorial-record-audio-and-encode-it-to-mp3-2eedcd466e78 (accedido el 19/05/2019)

19. IBM. *Desarrollo de aplicaciones en Python con ibm_db* [en línea].Recuperado de https://www.ibm.com/support/knowledgecenter/es/SSEPGG_10.5.0/com.ibm.swg.im.dbclient.python.doc/doc/c0054699.html (accedido el 02/05/2019)

20. *Multer* [en línea]. Recuperado de https://www.npmjs.com/package/multer (accedido el 30/05/2019)

21. *Inbenta: inteligencia artificial para atender clientes* [en línea]. Recuperado de https://destinonegocio.com/co/casos-de-exito-co/inbenta-inteligencia-artificial-para-atender-clientes/ (accedido el 04/06/2019)

22. Commons. [en línea]. Recuperado de http://www.commons.fm/ (accedido el 04/06/2019)

23. Planeta Chatbot. *"Una herramienta perfecta para ofrecer un servicio mucho más personalizado"* [en línea]. Recuperado de https://planetachatbot.com/entrevista-inaki-uriz-caravelo-chatbot-e3157268b9af (accedido el 04/06/2019)

24. TICbeat. *Iñaki Úriz (Caravelo): "En cinco años, la mitad de las aerolíneas tendrán chatbots"* [en línea]. Recuperado de https://www.ticbeat.com/entrevistas/inaki-uriz-caravelo-en-cinco-anos-la-mitad-de-las-aerolineas-tendran-chatbots/ (accedido el 04/06/2019)

25. Microsoft. *Azure Bot Service* [en línea]. Recuperado de https://azure.microsoft.com/es-es/services/bot-service/ (accedido el 04/06/2019)

26. Microsoft. *¿Qué es Azure Cognitive Services?* [en línea]. Recuperado de https://docs.microsoft.com/es-es/azure/cognitive-services/welcome#language-apis (accedido el 04/06/2019)

27. Microsoft. *Language Understanding* [en línea]. Recuperado de https://azure.microsoft.com/es-es/services/cognitive-services/language-understanding-intelligent-service/ (accedido el 04/06/2019)

28. Google. *Natural Language de Cloud* [en línea]. Recuperado de https://cloud.google.com/natural-language/ (accedido el 04/06/2019)

29. Google. *Dialogflow* [en línea]. Recuperado de https://dialogflow.com/ (accedido el 04/06/2019)

30. Amazon. *Amazon Lex* [en línea]. Recuperado de https://aws.amazon.com/es/lex/ (accedido el 04/06/2019)

31. Facebook. *wit.ai* [en línea]. Recuperado de https://wit.ai/ (accedido el 04/06/2019)

32. García, R. *Inteligencia artificial ¿qué, cómo, cuándo y dónde?* [en línea]. Recuperado de https://planetachatbot.com/inteligencia-artificial-qu%C3%A9-c%C3%B3mo-cu%C3%A1ndo-y-d%C3%B3nde-a0d27ddc0f3d (accedido el 05/06/2019)

33. Gómez, G. *[1/2] Introducción al Aprendizaje por Refuerzo* [en línea]. Recuperado de https://planetachatbot.com/introducci%C3%B3n-al-aprendizaje-por-refuerzo-f910d669d077 (accedido el 05/06/2019)

34. López Briega, R. E. *Introducción al Deep Learning* [en línea]. Recuperado de https://relopezbriega.github.io/blog/2017/06/13/introduccion-al-deep-learning/ (accedido el 06/06/2019)

35. *NLU y NLP: ¿qué son y cómo funcionan?* [en línea]. Recuperado de https://www.heynowbots.com/post/nlu-y-nlp-qu%C3%A9-son-y-c%C3%B3mo-funcionan (accedido el 07/06/2019)

36. Llorente, D. *Tecnologías de Generación de Lenguaje Natural (NLG): Inteligencia Artificial Versus Sistema basado en Reglas* [en línea]. Recuperado de https://planetachatbot.com/tecnologias-de-generacion-de-lenguaje-natural-inteligencia-artificial-sistema-basado-reglas-e537bb796468 (accedido el 07/06/2019)