UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de un bot conversacional como soporte en el primer nivel de un servicio al cliente en una empresa de servicios de mantenimiento de robots de manipulación logística.

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Catalá Valenzuela, Jorge

Tutor/a: González Ladrón de Guevara, Fernando Raimundo

CURSO ACADÉMICO: 2022/2023

RESUMEN

Un chatbot es una aplicación que intenta simular la conversación con un usuario. Dotados de ciertos niveles de aprendizaje automáticos permiten atender las consultas habituales de los clientes en un *customer service* para ayudarles de una forma eficaz. Aquellas consultas que no pudieran ser atendidas en una primera instancia por el bot serían derivadas al técnico correspondiente. Que a su vez podría escalarlas a otro técnico con más experiencia dependiendo de la complejidad del caso. Se obtienen así ventajas en la prestación del servicio. Este trabajo pretende desarrollar un prototipo en el primer nivel de atención al cliente en el dominio de una empresa de servicios con la ayuda de las herramientas de TypeScript (Angular), Dialogflow y MySQL.

Palabras clave: Chatbot, Customer Service, JavaScript, TypeScript, Angular, Dialogflow.

RESUM

Un chatbot es una aplicació que intenta simular la conversació amb un usuari. Dotats de certs nivells d'aprenentatge automàtics permeten atendre les consultes habituals dels clients a un *customer service* per ajudar-les d'una forma eficaç. Aquelles consultes que no pogueren ser ateses a una primera instància pel bot serien derivades al tècnic corresponent. Que alhora podria escalar-les a altre tècnic amb més experiència depenent de la complexitat del cas. S'obtenen així avantatges a la prestació del servici. Aquest treball pretén desenvolupar un prototip a primer nivell d'atenció al client a un domini d'una empresa de servicis amb l'ajuda de les ferramentes de TypeScript (Angular), Dialogflow i MySQL.

Paraules clau: Chatbot, Customer Service, JavaScript, TypeScript, Angular, Dialogflow.

ABSTRACT

A chatbot is an application that tries to simulate a conversation with a user. Equipped with certain levels of automatic learning, it permits us to attend to the usual requests of customers in a customer service to help them in an efficient way. Those requests that could not be answered in the first instance by the bot would be referred to the corresponding technician. Which in turn could escalate them to another more experienced technician depending on the complexity of the case. Thus, advantages are obtained in the provision of the service. This work aims to develop a prototype in the first level of customer service in the domain of a service company with the help of the next tools, TypeScript (Angular), Dialogflow and MySQL.

Key words: Chatbot, Customer Service, JavaScript, TypeScript, Angular, Dialogflow.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

1.1 CONTEXTUALIZACIÓN

Actualmente las tecnologías enfocadas a la automatización están presentes en prácticamente todos los aspectos de la vida cotidiana, desde la forma en que compramos productos hasta la forma en que nos comunicamos. La inteligencia artificial y el aprendizaje automático son tecnologías clave que están impulsando esta automatización.

En el ámbito laboral, la automatización ha permitido a las empresas mejorar la eficiencia y la precisión en una variedad muy amplia de tareas, tanto internas como aquellas que precisan de contacto con los clientes. Los chatbots y los asistentes virtuales, por ejemplo, se están convirtiendo en una herramienta cada vez más popular para la automatización de procesos.

En el ámbito personal, la tecnología automatizada está cambiando la forma en que interactuamos con los dispositivos en nuestro día a día. "Los asistentes virtuales han supuesto un gran avance para todos en este siglo XXI. Han allanado el camino para nuevas tecnologías en las que podemos hacer preguntas a las máquinas e interactuar con los IVA (*Intelligent Virtual Agent*) como lo hacemos entre seres humanos. Esta nueva tecnología ha atraído a prácticamente todo el mundo de diferentes formas, como *smartphones, laptops* u ordenadores." [1]. Algunos de estos IVAs aparecerían en el año 2011 con la implementación de Siri [2]. Más tarde, en 2014, se presentaba a Alexa [3], por parte de Microsoft a Cortana [4] y en 2016 se anunciaría Google Home [5]. Se trata de asistentes virtuales con una mayor conectividad entre dispositivos del mismo ecosistema y con la premisa de hacer la vida "más sencilla".

Estos se basaban en la premisa de facilitar búsquedas en la red o ejecutar órdenes informáticas simples con comandos de voz. Algunos incluso consiguieron distinguir entre tonos de voz con el fin de lograr una configuración más personalizada y sumar una ligera capa de seguridad.

Todas estas tecnologías están cada vez más presentes en hogares y espacios laborales, transformando la forma en la que vivimos y, por supuesto, en la que trabajamos. Y se espera que continúen evolucionando.

1.2 OBJETIVOS

Este trabajo tiene como objetivo demostrar el gran avance de las tecnologías con enfoque en la automatización en las facetas de independencia/autosuficiencia, compatibilidad e implantación con el resto más populares en empresas/negocios. También pretende dar una visión sobre la utilidad de dichos sistemas de cara al usuario,

sirviendo como facilitadores para la resolución de problemas con una complejidad media/baja.

Para esto realizaremos el diseño de un bot conversacional (chatbot) centrado en el ámbito empresarial e implementarlo de forma eficaz en un entorno laboral simulado.

Con el fin de complacer estos requisitos marcaremos unos hitos para así poder satisfacer el objetivo del trabajo:

- Diseñar y desplegar un entorno laboral simulado como pueda ser un portal web de una empresa ficticia (*front-end*).
- En relación con el punto anterior, hacer una pequeña base de datos para albergar datos relacionados con la empresa ficticia como puedan ser productos en venta, sus precios, incidencias, etc.
- Implementar un analizador de lenguaje natural (chatbot) capaz de comprender las peticiones de los usuarios y que gracias a la interpretación del lenguaje e inteligencia artificial pueda ofrecer una respuesta/solución coherente y esté capacitado para redirigir dichas peticiones al servicio adecuado.
- Entrenar el chatbot para ser eficiente en la identificación de las necesidades del usuario, simplificando el proceso de brindar una solución y reduciendo el número de acciones requeridas mediante la detección de detalles relevantes dentro de la solicitud inicial e integrarlo en nuestra arquitectura.

2 METODOLOGÍA

2.1 DISTRIBUCIÓN DE TAREAS

1) Elección de la temática del trabajo.
2) Realización del estado del arte (búsqueda de información, instrucción sobre situación actual, posibles tecnologías punteras, etc.)
3) Elección de tecnologías y elaboración del marco teórico con la ayuda de los entendimientos adquiridos en el grado y en la preparación previa a este paso.
4) Creación y personalización de *front-end* en Angular.
5) Creación de tablas en BD.
6) Creación y modelización del chatbot en Dialogflow.

7) En paralelo, a recomendación de mi tutor, redacción de la memoria del trabajo y el desarrollo de este.

8) También en paralelo vamos haciendo pruebas para el correcto funcionamiento del trabajo.

9) Evaluación de posibles mejoras.

2.2 DIAGRAMA TEMPORAL

Tareas	Octubre	Noviembre	Diciembre	Enero	Febrero	Marzo
Elección de Temática						
Realización estado del arte						
Elección de tecnologías y elaboración marco teórico						
Creación/Modelización front-end (Angular)						
Creación/Modelización tablas (BBDD)						
Creación/Modelización Chatbot (Dialogflow)						
Redacción memoria del trabajo						
Pruebas de correcto funcionamiento						
Evaluación posibles mejoras						

Tabla 1 - Diagrama de Gantt

3 ESTADO DEL ARTE

3.1 CONCEPTOS GENERALES

3.1.1 Inteligencia artificial

Un concepto muy importante de detallar previamente a las definiciones de Bot o chatbot es el de Inteligencia Artificial (IA).

Es un concepto con definiciones no homogéneas, aunque sí similares. Si observamos estas en diferentes fuentes, ya sean la R.A.E o Wikipedia, veremos que existen ligeros matices distintivos.

Según la R.A.E la IA es la “Disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico.” [6]

Sin embargo, en Wikipedia “La inteligencia artificial es, en las ciencias de la computación, la disciplina que intenta replicar y desarrollar la inteligencia y sus procesos implícitos a través de computadoras.” [7]

Otra definición ofrecida por John McCarthy es: “Es la ciencia y la ingeniería de fabricar máquinas inteligentes, especialmente programas informáticos inteligentes. Está relacionado con la tarea similar de usar computadoras para comprender la inteligencia humana, pero la IA no tiene por qué limitarse a métodos que son biológicamente observables.” [8]

Es por eso por lo que al no haber un consenso sobre su definición se ha decidido preguntarle a un más que conocido chatbot sobre este término.

Según ChatGPT “La inteligencia artificial (IA) es la capacidad de un sistema (como una computadora, robot o software) para realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje, el razonamiento y la resolución de problemas. Esto se logra mediante algoritmos y técnicas de aprendizaje automático, que permiten al sistema mejorar su rendimiento con el tiempo.” [9]

3.1.2 Bot

Iniciamos el entendimiento de qué es realmente un bot y cuáles pueden ser sus fines.

La palabra bot proviene de robot, básicamente se definen como agentes automatizados que suelen funcionar en plataformas online. Pueden ser programas que se ejecutan de forma continua, formulan decisiones y pueden ser capaces de llegar a actuar sobre estas sin intervención humana, incluso adaptarse a ciertos contextos. [10] Estas aplicaciones software pueden tener distintos propósitos. Los más actuales en el mundo laboral serían la automatización de procesos/esfuerzos, la recopilación y procesamiento de información o ayuda/guía para los usuarios e incluso para los trabajadores, por poner ejemplos.

Nosotros nos centraremos en este último propósito. La ayuda al usuario o trabajador puede ser un concepto bastante general, ya que tiene muchos matices en los que ahondar. Vayamos a enumerar los tipos de bots principales con esta finalidad:

- **Chatbots:** diseñados para la interacción humano-máquina con la cual poder satisfacer necesidades mediante conversaciones por vía voz o texto.

- **Crawlers/Rastreadores:** su principal propósito es el de recabar mucha información en el menor tiempo posible y de distintas fuentes. Muy utilizados en la década de los 90 para acceder y archivar las páginas web que se creaban a diario.

- **Bot Informativo:** algunas veces parecidos a los chatbots, son bots encargados de ofrecer cierta información específica al usuario. [10]

Hay infinidad de tipos, pero estos suelen ser los más utilizados. Nuestro caso tratará sobre un chatbot.

3.1.3 Chatbot

Bien, habiendo resuelto la principal duda nos resalta otra: ¿Qué es un chatbot?

Un chatbot es un bot conversacional, es decir, un programa informático diseñado para simular una conversación con humanos a través de interfaces de texto o de voz, también denominados "sistemas de diálogo humano-ordenador con lenguaje natural". [11]

Los chatbots, o chatterbots, se basan en algoritmos de inteligencia artificial, como el procesamiento del lenguaje natural y el aprendizaje automático, para entender y responder las preguntas del usuario y así intentar proporcionar servicios de atención al cliente automatizados, como responder preguntas frecuentes, ayudar con transacciones y brindar información.

La mayoría de las empresas en las que uno puede pensar ya tienen incorporado en su red interna un chatbot para ayudar a sus empleados con la resolución de problemas, soporte técnico o dudas de carácter interno. "Grandes empresas como Lloyds Banking Group, Royal Bank of Scotland, Renault y Critoën ya utilizan asistentes automatizados en línea en lugar de *call centers* con humanos para proporcionar un primer punto de contacto con el cliente." [12]

Muchas de estas también incorporan uno en sus páginas web, para poder ayudar al cliente con búsquedas, productos, precios o como en el anterior caso, resolución de problemas o dudas con el negocio.

Nosotros desarrollaremos un pequeño chatbot orientado/enfocado hacia el cliente.

3.1.4 ChatGPT

En estos dos últimos años ha sido notable el gran avance y notoriedad que ha conseguido el mundo de la inteligencia artificial (IA). Tanto es así que reconocidas marcas tecnológicas mundiales apuestan por ella e incluso compran proyectos para seguir financiándolos y evolucionarlos a su estilo.

Dentro de todo este auge de tecnologías, apareció ChatGPT, un chatbot capaz de responder prácticamente cualquier tipo de pregunta/problema, con una rapidez asombrosa y con un acierto y sentido extraordinarios.

Su creación se remonta a varios años atrás, cuando un equipo de científicos y expertos en tecnología iniciaron un trabajo de construcción de un sistema capaz de procesar y comprender el lenguaje humano. Su objetivo era crear una plataforma de IA que tuviera la capacidad de interactuar con personas de manera natural y efectiva.

Para lograr dicho objetivo se utilizó un enfoque de aprendizaje automático basado en el análisis de grandes cantidades de datos de lenguaje natural, incluyendo textos, conversaciones, publicaciones en redes sociales y otros tipos de comunicación. A

medida que se iba alimentado el modelo mejoraba su capacidad de comprender y generar lenguaje natural.

Varios años más tarde de su investigación y desarrollo, el modelo de lenguaje se lanzó oficialmente en 2020 como una plataforma de IA llamada GPT-3 (Generative Pre-trained Transformer 3). Desde entonces, este ha sido utilizado por empresas y organizaciones de todo el mundo para una amplia variedad de aplicaciones, incluyendo asistentes virtuales, chatbots, análisis de datos, etc.

Uno de los grandes dilemas que se han presentado al aparecer esta tecnología tan avanzada y cuidada ha sido el de la Educación.

“El tema más controvertido con respecto a ChatGPT no está en la bondad de sus respuestas, sino en si se convertirá en la herramienta utilizada por quien necesite escribir un texto para hacerlo sin el esfuerzo humano necesario y, por tanto, sin adquirir aquellas competencias para las que fue diseñada la tarea intelectual.” [13]

Toda esta problemática se traslada también al mundo laboral, como ya hemos mencionado anteriormente, en tareas donde puede no haber una gran habilidad/interactividad humana los bots son muy útiles para reemplazar a los responsables de estas. Tareas como contactar mediante mensajería o llamadas telefónicas con el cliente (call centers), ya existen numerosos casos de empresas sustituyendo trabajadores por bots/chatbots que brindan a estas una oportunidad de rebaja en la inversión de personal. [14]

Otro sector que podría verse afectado en un futuro es el de desarrollo software. Contemplan la posibilidad de que ChatGPT pueda ser capaz de sustituir a programadores junior, debido a la facilidad y rapidez que tiene este en devolver código de una calidad medianamente razonable y en prácticamente cualquier lenguaje de programación.

Este aspecto de ChatGPT puede verse como un arma de doble filo, ya que sirve como un gran apoyo a principiantes/novatos que necesiten resolver ciertos problemas de forma inmediata y en cualquier momento del día, pero también implica una posible sustitución de su futuro trabajo en empresas de consultoría. Si bien es cierto que las consultoras necesitan de programadores seniors con experiencia, para conseguirlos necesitan dar tiempo y conocimientos a programadores juniors, con la esperanza de poder retenerlos lo suficiente para transicionarlos a cargos más altos. Por otro lado el gran ahorro de costes que puede suponer el pasar de 4-5 programadores juniors en cada proyecto a simplemente 1 o 2 puede decantar la balanza a favor de la IA.

Como vemos el avance de la tecnología no solo ofrece aspectos positivos en nuestra sociedad, con ellos siempre vienen nuevos desafíos para las capacidades laborales y empresariales, sobre todo de ciertos entornos más enfocados hacia la tecnología.

3.1.5 Entorno laboral operativo del bot

El entorno laboral en el cual va a operar nuestro chatbot es el mencionado en el título del proyecto. Nos referimos a una empresa de servicios de mantenimiento de robots enfocados a la manipulación logística. Para ello vamos a dar cierto contexto sobre la evolución en dichos entornos.

"La robótica industrial se desarrolló a mediados del siglo pasado para automatizar procesos industriales repetitivos que se caracterizaban por tener largas producciones." [15]

Debido a la eclosión que ha tenido este tipo de robótica se han abierto nuevos escenarios. Esto ha permitido a las medianas y pequeñas empresas mirar de otra forma el mercado, afrontando nuevos retos y evolucionarlo. [15] Todo este avance ha abierto una nueva oportunidad de negocio, el mantenimiento de esta tecnología. En el siguiente gráfico podemos observar el crecimiento de las ventas mundiales de robots industriales en un periodo de 18 años (Figura 1).

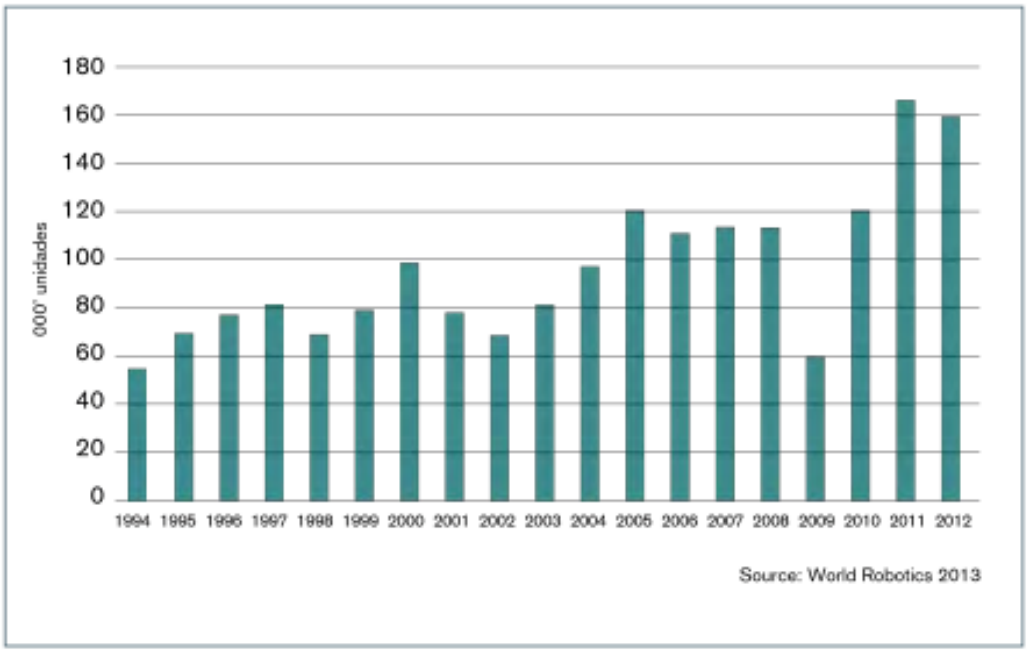


Figura 1 - Evolución de las ventas mundiales de robots industriales en el periodo 1994-2012 [24]

En los últimos años los fabricantes se han encontrado con diversas problemáticas, siendo hoy en día una de las más importantes la demanda de productos más personalizados. Esto añade complejidad a los niveles de producción automatizados y de unos años a esta parte la mano de obra ha ido disminuyendo, dejando en cierta medida desprotegidas a dichas empresas en picos de producción. Por esta razón los sectores con mayor utilización de la robótica son los de automoción y los de electrónica, acumulando aproximadamente 2 millones de robots a finales de 2019. [15]

Otra problemática que se planteó con la llegada de estas nuevas tecnologías son la escasez de habilidades capaces de otorgar a los robots. Algo que se ha ido resolviendo a lo largo del tiempo debido a la que denominan "robótica cooperativa". Este concepto

hace referencia a la compartición del espacio laboral entre humanos y máquinas, resultando en nuevas posibilidades de colaboración y aumento de productividad.

Se podría pensar que la aparición de empresas centradas en el mantenimiento de estos robots está más que justificada. Sin embargo, el rumbo que ha tomado esta industria ha sido algo más lógico. Las empresas encargadas de la venta o incluso de la creación de dichos robots, son las que ofrecen un servicio de soporte y mantenimiento.

Sí que es cierto que existen ciertas empresas dedicadas solamente al soporte de las máquinas, normalmente también encargadas de la distribución y venta de estas. Suelen ofrecen una amplia gama de servicios que incluyen la instalación, configuración, mantenimiento preventivo, reparación y actualización de los robots. Los técnicos especializados realizan inspecciones periódicas y pruebas de diagnóstico para detectar cualquier problema. Normalmente, estas empresas cuentan con servicios de formación y capacitación para los operadores de los robots, con el objetivo de maximizar la eficiencia y la seguridad en el manejo de estos. De forma complementaria realizan un asesoramiento en la selección y adquisición de robots, según las necesidades específicas de cada cliente.

Hoy en día, la robótica industrial es algo imprescindible en los entornos de producción de grandes, medianas e incluso ciertas pequeñas empresas y con ello va ligado, sin lugar a duda, su mantenimiento. Un pilar tan fundamental como

3.2 HERRAMIENTAS

Para llevar a cabo nuestro proyecto harán falta diversas herramientas para implementar cada uno de los apartados principales. Estos son la creación de nuestro *front-end*, la creación e implementación del chatbot y, de forma más breve, la configuración de nuestro *back-end*.

3.2.1 Herramientas para la creación de un ChatBot

Debemos entender en qué se basa un chatbot para su correcto funcionamiento e interacción con humanos. Como mencionaremos más adelante, muchas de las grandes empresas de software actuales (Google, Microsoft, Amazon, etc) han apostado por la creación de plataformas propias para el desarrollo de chatbots. A parte de los pesos pesados de la tecnología también existen muchas otras alternativas que ofrecen otras características interesantes, de esto hablaremos en los siguientes puntos. Al fin y al cabo, lo que ofrecen estas plataformas son distintas funcionalidades con respecto al procesamiento del lenguaje natural (NLP), estructuras de flujos para las conversaciones entre bot y humano, soporte para la correcta implementación y pruebas con distintos servicios y capacidades de conexión con sistemas de información. [16]

A parte de Inteligencia Artificial (IA), también se utilizan otros tipos de algoritmos y tecnologías para la creación de estos, algunos ya los hemos mencionado anteriormente, estos son algunos dependiendo de la finalidad de nuestro proyecto:

- **Procesamiento del Lenguaje Natural (NLP):** una de las más importantes, permite al chatbot entender y generar idioma natural, como el idioma hablado y escrito, lo que es esencial para simular una conversación humana.

- **Aprendizaje automático:** permite al chatbot aprender y mejorar con el tiempo, ya sea a través de datos etiquetados o mediante la retroalimentación del usuario.

- **Reglas y scripts:** permiten al chatbot responder a preguntas específicas y realizar tareas específicas, como reservar una mesa en un restaurante o comprar un producto en línea.

- **Integración con servicios externos:** permite al chatbot acceder a bases de datos y servicios web para proporcionar información y realizar tareas, como buscar un vuelo o consultar el tiempo.

- **Análisis de sentimiento:** permite al chatbot detectar y responder a las emociones del usuario, como la felicidad, la tristeza o la frustración.

Existe un gran espectro de herramientas disponibles, y en gran parte gratuitas, para la implementación de un chatbot en nuestro ecosistema tecnológico. Entre toda esta variedad destacan dos grupos a rasgos generales.

Las que permiten una mayor flexibilidad y control sobre el código en el que se basa un chatbot, como mayor personalización sobre el modelo de IA y NLP:

- **Botkit**

- **Rasa**

Herramientas con una mayor compatibilidad y fácil implementación con otro tipo de plataformas y soluciones, sacrificando una parte de control y particularización del proyecto. Predominan los pesos pesados de la tecnología moderna:

- **Dialogflow (Google)**

- **Microsoft Bot Framework (Microsoft)**

- **Amazon Lex (Amazon)**

- **WIT.ai (Facebook)**

En esta tabla se puede ver una comparación de todas las opciones de software disponibles para la creación de un chatbot mencionadas anteriormente. [16]

			Dialogflow (Google) [v2]	Watson (IBM) [v2]	Lex (Amazon) [07/06/2020]	Bot Framework + LUIS (Microsoft) [v4]	FlowXO [07/06/2020]	Landbot.io [07/06/2020]	Chatfuel [07/06/2020]	Rasa [10.1.2]	SmartLoop [07/06/2020]	Xenioo [07/06/2020]	Botkit (part of Bot Framework) [4.9.0]	LUIS [05/19/2020]	ChatterBot [1.0.5]	Pandorabots [07/06/2020]
Technical factors		1. Kind (Library, Framework, Platform, Service)	P	P	P	F	P	P	P	F	P	P	F	S	L	P
	Input processing	2. Regular expressions/patterns	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
		3. NLP for phrase match	✓	✓	✓	✓			✓	✓	✓	✓		✓	✓	
		4. Text processing to obtain phrase parameters	✓	✓	✓	✓				✓	✓	✓		✓		✓
		5. Number of languages: very high (≥50), high (≥10), some (<10), one (represented by U.S. flag)	h	h	🇺🇸	h	h		h	v		s		h	v	
		6. Sentiment analysis	✓	✓	✓	✓								✓		
		7. Speech recognition	✓	✓	✓	✓						✓		✓		
	Dialog	8. Storage of phrase parameters: volatile, persistent, both	b	b	b	b	b	v	v	b	v	v		v		v
		9. Support for intents	✓	✓	✓	✓	✓			✓	✓			✓		✓
		10. Support for entities: predefined, user-defined, both	b	b	b	b	p	p		b	b	b		b		
		11. Dialog structure: tree, follow-up intents, dsl	f	f	f	f	t	t	t	t	f	t			f	d
		12. Utterances to re-engage users					✓		✓		✓	✓				
		13. Specification of chatbot answers	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓
	Deployment	14. Integration with social networks/websites: high (≥10), some (<10), one (represented with logo)	h	s	s	h	s			h	s	s	s			s
		15. Interaction support for specific social networks	✓			✓						✓				✓
	System integration	16. Call to services from chatbot	✓	✓	✓	✓	✓	✓	✓	✓					✓	
		17. Chatbot usage via API	✓	✓	✓						✓	✓		✓		✓
Technical factors	Development and testing	18. Prebuilt components: chatbot templates, intents, small talks, services	cts	c	i	cs	c	c							c	t
		19. Version control: native, code based	n	n	n	c				c			c		c	
		20. Chat console for testing	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓
		21. Debug mechanisms	✓			✓				✓			✓		✓	
		22. Validation support	✓													
	Execution	23. Hosted deployment	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓		✓
		24. Support for analytics	✓	✓		✓	✓	✓			✓	✓				
		25. User message persistence	✓		✓	✓	✓	✓	✓			✓	✓			
	Security	26. Cloud security	✓	✓	✓	✓								✓		
Managerial factors	Organization	27. Pricing model: free, pay-as-you-go, quota, advanced feats	fp	fp	fp	fpa	fq	fa	fa	fa	fa	fq	f	fq	f	fqa
		28. Developer expertise: low, high	l	l	l	h	l	l	l	h	l	l	h	h	h	l
	Development	29. Code hosting: external, on-premises	e	e	e	o	e	e	e	o	e	e	o	o	o	e
		30. Group work				✓				✓			✓	✓	✓	
		31. i18n	✓													
		32. Open source								✓			✓		✓	
	Operational	33. New channels								✓			✓	✓	✓	
		34. No vendor lock-in								✓			✓	✓	✓	

Tabla 2 - Comparativa de librerías, frameworks, plataformas y servicios de un chatbot

En nuestro caso vamos a trabajar con la herramienta Dialogflow de Google.

Dicha herramienta nos ofrece ciertas ventajas sobre el resto, la principal es el programa de análisis en el que está basada, *Chatbase*. Este se respalda en la prestigiosa plataforma de *Google Analytics*. Además, Dialogflow también tiene ciertos puntos positivos que nos pueden facilitar el curso del proyecto, como por ejemplo su integración con servicios propios de Google (*Google Home*) o con servicios de terceros como *Twitter*, *Telegram* o *LINE*. Esta integración está diferenciada por servicios en su portal, donde también existe una opción llamada "Dialogflow Messenger", la cual permite una sencilla integración con prácticamente cualquier tipo de *front-end*. De todas formas, hablaremos de esto más adelante.

Su gran disponibilidad de agentes predefinidos, la capacidad de soportar múltiples lenguajes y la opción de editar el código de forma directa también son aspectos positivos a tener en cuenta para nuestro proyecto, ya que nos proporcionan herramientas para ajustar cómo se procesa la información a un nivel más detallado.

3.2.2 Tecnologías para el desarrollo web

Otra parte importante del proyecto se basa en el desarrollo web y su implementación con nuestro chatbot. Para entrar en contexto debemos comentar el avance exponencial en las tecnologías y métodos de desarrollo web en la última década. Avances impulsados, por ejemplo, por el uso de los teléfonos móviles para navegación web, que ha llevado a un aumento en la importancia de la responsividad y optimización para dichos dispositivos.

El gran aumento de Frameworks y librerías, ya sea Angular, React o Vlue.js, han simplificado y acelerado todo el desarrollo permitiendo crear aplicaciones más sofisticadas y con un mayor rendimiento. Con ellas vino la aparición de las *Progressive Web Apps* (PWA), diseñadas para combinar las mejores características de las aplicaciones móviles y las aplicaciones web para ofrecer un servicio de calidad en aplicaciones de navegador.

La experiencia de usuario y el diseño han tomado un gran peso en los últimos años, lo que ha llevado a la popularidad de herramientas y metodologías como *Material Design* y *Design Systems*. Como no, mencionar el *cloud computing*. La popularidad y la disponibilidad de servicios en la nube, como Amazon Web Services y Google Cloud, han permitido una mayor escalabilidad y flexibilidad en el desarrollo web.

Todo esto sumado con el avance técnico de recursos, mejorando la eficiencia, la escalabilidad y la accesibilidad de la tecnología, ha hecho que estemos disfrutando (o no) de la época dorada de esta en todos los aspectos. Una revolución que se espera siga progresando en fechas próximas.

La elección de un framework puede ser algo tedioso e incluso complejo si queremos que este se ajuste a los requerimientos de nuestro proyecto. El decantarnos por un framework inapropiado puede llevar a la pérdida de tiempo en el estudio de detalles en el lenguaje de programación correspondiente, el no cumplimiento de los tiempos marcados debido a la incomodidad en el desarrollo del proyecto y la también perdida de

tiempo si se decidiera cambiar de nuevo a otro framework, teniendo que migrar nuestro código y dependencias a este. [17]

Es por esto que teniendo en cuenta dichos detalles y otras consideraciones, para nuestro diseño web nos hemos apoyado en el framework de Angular, por su gran adaptabilidad y facilidad para la creación de un *front-end*, adecuándose a nuestros requerimientos.

"Angular es un Framework application-design y una plataforma de desarrollo para crear aplicaciones de una sola página eficientes y sofisticadas" [18]. Angular, desarrollado por Google, es una herramienta de código abierto para crear aplicaciones web. Es un Framework diseñado específicamente para construir interfaces de usuario en el lado del cliente, y utiliza el lenguaje de programación TypeScript. Angular se enfoca en la creación de aplicaciones de una sola página (SPA, por sus siglas en inglés) para sitios web.

"TypeScript es un superconjunto sintáctico de JavaScript que agrega escritura estática. Básicamente, esto significa que TypeScript agrega sintaxis además de JavaScript, lo que permite a los desarrolladores agregar tipos." [19]. TypeScript es una herramienta de programación de código abierto desarrollada por Microsoft, que se basa en JavaScript y agrega características de tipado estático y objetos basados en clases.

Dado que es una extensión de JavaScript, este código se puede integrar fácilmente en TypeScript, lo que significa que todo el código puede ser utilizado en este. TypeScript está diseñado para facilitar el desarrollo de proyectos de gran escala, con una sintaxis más concisa y óptima que permite a los usuarios escribir código de manera más eficiente. Luego, un compilador traduce todo el código a JavaScript para que sea interpretado por el motor utilizado en el navegador o servidor.

Las ventajas que ofrece Angular con respecto al resto de Frameworks son diversas, por eso vamos a enumerar algunas de sus características:

1. **Módulos:** se permite la organización y separación de la aplicación en módulos lógicos, lo que facilita la gestión y el mantenimiento del código.

2. **Two-Way Data Binding:** nos ofrece un enlace bidireccional de datos, lo que significa que los cambios en la vista se reflejan automáticamente en el modelo y viceversa. Esto reduce la cantidad de código necesario y mejora la eficiencia.

3. **Directivas:** Angular provee un conjunto de directivas que permiten manipular y transformar el HTML dinámicamente.

4. **Rendimiento:** se utiliza un sistema de detección de cambios inteligente que permite un rendimiento óptimo en aplicaciones complejas.

5. **Testing:** Angular proporciona una estructura de testing sólida y una gran cantidad de herramientas y librerías para realizar pruebas unitarias y de integración.

6. **Comunidad:** por último, pero no menos importante, Angular cuenta con una amplia comunidad de desarrolladores y una gran cantidad de recursos en línea,

lo que lo hace fácil de aprender, utilizar y en caso de necesidad, resolver problemas.

Es por esto y por su utilización en otros proyectos con anterioridad con dicho Framework, se opta por esta vía de trabajo.

4 DESARROLLO

4.1 DESARROLLO *FRONT-END*

Como hemos mencionado anteriormente, los proyectos de Angular se organizan por módulos. Estos módulos son una especie de contenedores que contienen todo el código dedicado a una aplicación, workflow o conjunto de capacidades. Dentro de estos se descompone en Componentes y eventualmente en Servicios (entre otros archivos de código). Se trata de una de las grandes ventajas que ofrece este Framework y que permite un rápido y eficiente diseño de páginas unificadas bajo el mismo servicio web.

Los componentes son uno de los elementos más importantes y principales para las aplicaciones Angular. Cada componente consta de:

- Una plantilla HTML donde se declaran que se representa en la página.
- Una clase de TypeScript que define el comportamiento.
- Un selector de CSS que define cómo se usa el componente en una plantilla.
- Opcionalmente, estilos CSS aplicados a la plantilla. [20]

Nuestro *front-end* estará formado por distintos componentes, cada cual realizará una función. El chatbot redirigirá al cliente al correspondiente apartado dependiendo de las indicaciones/peticiones de este.

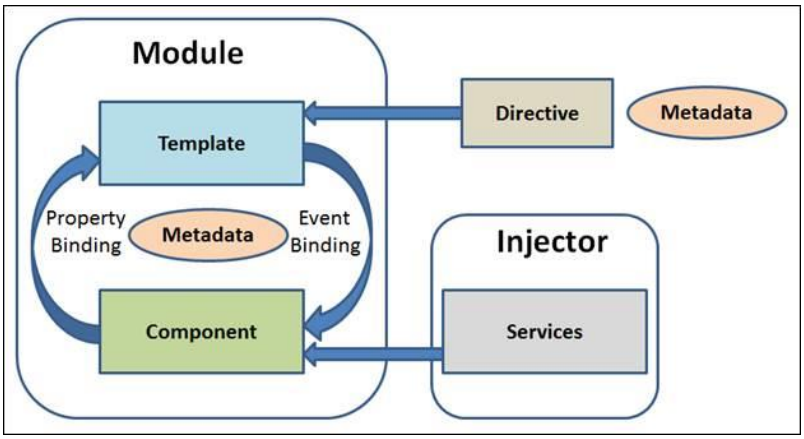


Figura 2 - Diagrama de la arquitectura Angular [23]

Este es un diagrama que plasma de forma adecuada el comportamiento de Angular (Figura 2), en nuestro caso, como hemos comentado, existirían diversos componentes y varios servicios. Todo esto se verá más en detalle en los siguientes subapartados, donde se va a realizar una explicación de cada componente que conformará nuestro *front-end*, sus servicios correspondientes, y sus archivos de código.

4.1.1 Componente "Home"

El componente principal será el que denominaremos ***"home"***. Como su nombre indica es el encargado de mostrar la página principal del proyecto. Este componente estará compuesto por los ficheros enumerados anteriormente, es decir:

- **home.component.html:** archivo HTML el cual representa el apartado visual de la página.

```
<div class="toolbar" role="banner">
  <a [routerLink]="['']">
    <img
    alt="UPV Logo"
    src="assets/img/elementoUPV-RO.png"
    />
  </a>
  <span>Trabajo Fin de Grado - Jorge Catalá Valenzuela</span>
    <div class="spacer"></div>
    <a aria-label="Twitter" target="_blank" rel="noopener" href="https://twitter.com/jorge_catala" title="Twitter">
      <svg id="twitter-logo" height="24" data-name="Logo" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 400 400">
        <rect width="400" height="400" fill="none"/>
        <path d="M153.62,301.59c94.34,0,145.94-78.16,145.94-145.94,0-2.22,0-4.43-.15-6.63A104.36,104.36,0,0,0,325,122.47a102.38,102.38,0,0,
      </svg>
    </a>
</div>

<div class="container">
  <h1>BIENVENIDOS</h1>
</div>

<div class="content">
  <df-messenger
    intent="WELCOME"
    chat-title="TFG"
    agent-id="62b56b1a-a8a8-43b3-8c4a-893de0e042b3"
    language-code="es"
  ></df-messenger>
</div>
<router-outlet></router-outlet>
```

Figura 3 - Código html componente "Home"

- **home.component.ts:** archivo TypeScript que define el comportamiento de la página, en este caso estará prácticamente vacío debido a la poca interactividad de esta.

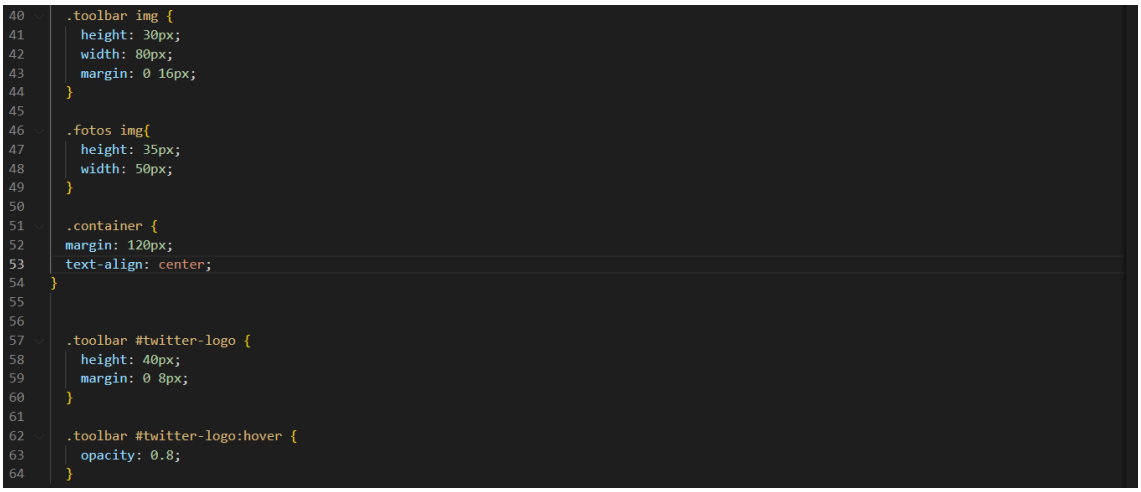
```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {

}
```

Figura 4 - Código ts componente "Home"

- **home.component.css:** archivo que contiene los estilos CSS aplicados (este archivo es igual en todos los Componentes del *front-end*).

```css
:host {
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji"
  font-size: 14px;
  color: #333;
  box-sizing: border-box;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

h1,
h2,
h3,
h4,
h5,
h6 {
  margin: 8px 0;
}

p {
  margin: 0;
}

.spacer {
  flex: 1;
}

.toolbar {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  height: 60px;
  display: flex;
  align-items: center;
  background-color: #022c57;
  color: white;
  font-weight: 600;
}
```

Figura 5 - Código css de todos los componentes (1/2)



```
.toolbar img {
  height: 30px;
  width: 80px;
  margin: 0 16px;
}

.fotos img{
  height: 35px;
  width: 50px;
}

.container {
margin: 120px;
text-align: center;
}


.toolbar #twitter-logo {
  height: 40px;
  margin: 0 8px;
}

.toolbar #twitter-logo:hover {
  opacity: 0.8;
}
```

Figura 6 - Código css de todos los componentes (2/2)

4.1.2 Componente "Catalogo"

El siguiente componente agregado es ***"Catalogo"***. Encargado de llamar mediante ficheros php alojados en el lado del servidor del *back-end* a los datos comprendidos en la tabla catálogo de nuestra base de datos.

Para este componente debemos introducir un nuevo término utilizado en el Framework de Angular, hablamos de los servicios.

"El servicio es una categoría amplia que abarca cualquier valor, función o característica que necesita una aplicación. Un servicio es típicamente una clase con un propósito bien definido. Debe hacer algo específico y hacerlo bien.

Angular distingue los componentes de los servicios para aumentar la modularidad y la reutilización." [21]

En términos ideales, la función de un componente consiste en facilitar/habilitar la experiencia del usuario. Para lograr esto, el componente debe ofrecer propiedades y métodos que permitan la vinculación de datos, actuando como intermediario entre la vista (diseño visual) y la lógica de la aplicación (modelo). Es decir, la vista representa el template y la lógica se refiere a la conceptualización del modelo.

Para tareas que no tengan relación con la vista o la lógica de la aplicación, es conveniente que un componente haga uso de servicios. Los servicios son útiles para tareas como, por ejemplo, de validación o de obtención de información del servidor. Si se definen estas tareas en una clase de servicio, estas pasan a ser disponibles para todos los componentes, aunque también es posible dividir subtareas en servicios distintos.

En nuestro caso vamos a generar dos servicios, uno para la obtención de datos de una tabla en concreto ("catalogo) de nuestra BD y otro con el mismo objetivo anterior (obtención de datos), y a parte la creación de valores en la tabla restante ("incidencias").

Por tanto, nuestro servicio se encargará de recoger la información almacenada en nuestra base de datos mediante una llamada HTTP (*get* para recoger o post para enviar).

En el lado del servidor alojaremos ciertos scripts php que al ser invocados realicen ciertas acciones sobre la BD y sus tablas.

En el caso de ***"catalogo"*** simplemente haremos un getAll para poder presentar todos los productos disponibles dentro de la tabla catálogo.

Explicamos brevemente su estructura:

- **catalogo.component.html:** archivo donde se representa el apartado visual de la página.

```
<div class="toolbar" role="banner">
  <a [routerLink]="['']">
    <img
    alt="UPV Logo"
    src="assets/img/elementoUPV-RO.png"
    />
  </a>
  <span>Trabajo Fin de Grado - Jorge Catalá Valenzuela</span>
    <div class="spacer"></div>
    <a aria-label="Twitter" target="_blank" rel="noopener" href="https://twitter.com/jorge_catala" title="Twitter">
      <svg id="twitter-logo" height="24" data-name="Logo" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 400 400">
        <rect width="400" height="400" fill="none"/>
        <path d="M153.62,301.59c94.34,0,145.94-78.16,145.94-145.94,0-2.22,0-4.43-.15-6.63A104.36,104.36,0,0,0,325,122.47a102.38,102.38,0,0,
      </svg>
    </a>
</div>

<div class="container">
  <h2>Listado de Productos</h2>
  <table mat-table [dataSource]="productos" class="mat-elevation-z8">

    <!--- Note that these columns can be defined in any order.
          The actual rendered columns are set as a property on the row definition" -->

    <!-- Name Column -->
    <ng-container matColumnDef="Nombre">
      <th mat-header-cell *matHeaderCellDef> Modelo </th>
      <td mat-cell *matCellDef="let productos"> {{productos.Nombre}} </td>
    </ng-container>
```

Figura 7 - Código html componente "Catalogo" (1/2)

```
    <!-- Weight Column -->
    <ng-container matColumnDef="Precio">
      <th mat-header-cell *matHeaderCellDef> Precio </th>
      <td mat-cell *matCellDef="let productos"> {{productos.Precio}}€ </td>
    </ng-container>

    <tr mat-header-row *matHeaderRowDef="['Nombre', 'Precio']"></tr>
    <tr mat-row *matRowDef="let row; columns: ['Nombre', 'Precio'];"></tr>

  </table>
</div>


<div>
  <df-messenger
    intent="WELCOME"
    chat-title="TFG"
    agent-id="62b56b1a-a8a8-43b3-8c4a-893de0e042b3"
    language-code="es"
  ></df-messenger>
</div>
```

Figura 8 - Código html componente "Catalogo" (2/2)

- **catalogo.component.ts:** archivo que define el comportamiento del componente. En este caso contiene algunas líneas más de código. En la clase CatalogoComponent, mediante CatalogoService, recogeremos los productos registrados en nuestra BD y los introduciremos en la variable "productos", para así poder hacer referencia a esta en el código HTML y poder mostrar los datos específicos.

```typescript
import { Component, OnInit } from '@angular/core';
import { CatalogoService } from '../catalogo.service';


@Component({
  selector: 'app-catalogo',
  templateUrl: './catalogo.component.html',
  styleUrls: ['./catalogo.component.css']
})

export class CatalogoComponent implements OnInit {

  productos:any

  constructor(private catalogoServicio: CatalogoService) {}

  ngOnInit() {
    this.recuperarTodos();
  }

  recuperarTodos() {
    this.catalogoServicio.getCatalogo().subscribe((result:any) => this.productos = result);
  }

}
```

Figura 9 - Código ts componente "Catalogo"

- **catalogo.component.css:** archivo que contiene los estilos CSS aplicados (este archivo es igual en todos los Componentes del *front-end*).

- **catalogo.service.ts:** archivo TypeScript encargado de albergar la lógica responsable de realizar una llamada HTTP (*get* en este caso) a nuestro servidor para ejecutar el código php correspondiente y así recuperar la información requerida de nuestra BD.

```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { environment } from 'src/environments/environment';
@Injectable({
  providedIn: 'root'
})
export class CatalogoService {
  baseUrl = environment.baseUrl

  constructor(private http: HttpClient) { }

  getCatalogo() {
    return this.http.get(`${this.baseUrl}/getAll.php`);
  }

  getRobot1() {
    return this.http.get(`${this.baseUrl}/get_robot1.php`);
  }

  getRobot2() {
    return this.http.get(`${this.baseUrl}/get_robot2.php`);
  }

  getRobot3() {
    return this.http.get(`${this.baseUrl}/get_robot3.php`);
  }

  getRobot4() {
    return this.http.get(`${this.baseUrl}/get_brazo.php`);
  }
}
```

Figura 10 - Código ts del servicio "Catalogo Service"

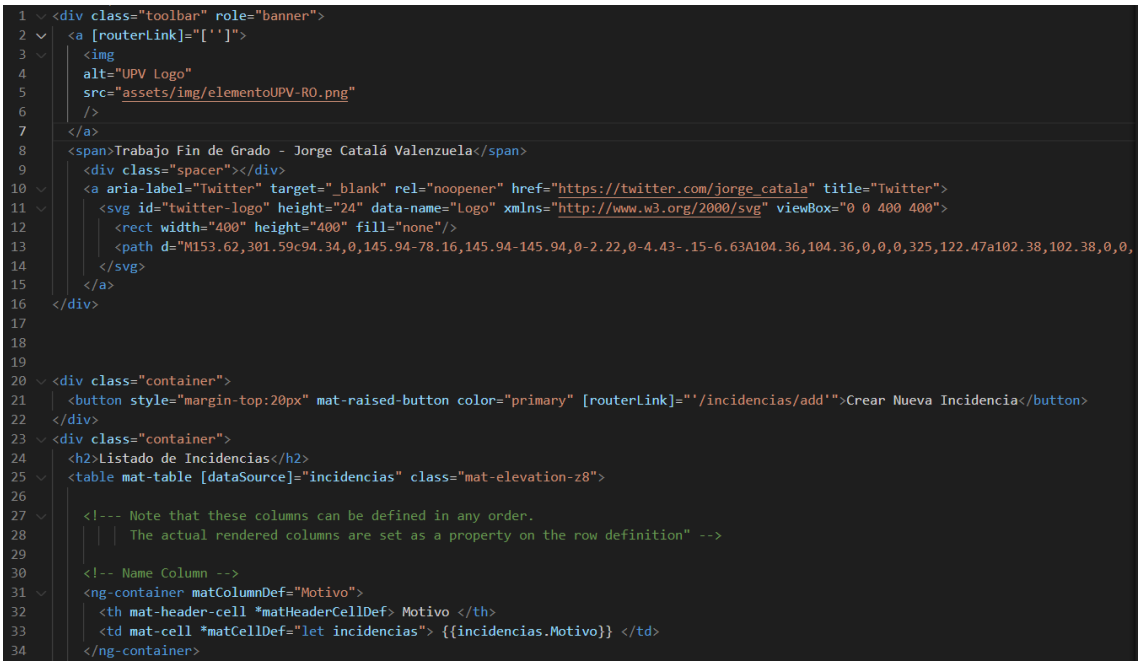
4.1.3 Componente “Incidencias”

Este componente es hermano del anterior (“Catalogo”), pero con una funcionalidad más añadida la cual permite redirigir al cliente a la página correspondiente para poder crear una incidencia nueva.

Como su nombre indica, en este componente vamos a listar las incidencias registradas por el cliente, mostrando el motivo de la incidencia, una descripción y la fecha de apertura de esta.

Su estructura será la siguiente:

- **incidencias.componente.html:** archivo donde se representa el apartado visual de la página. En este componente tenemos una peculiaridad, y es la de añadir un botón el cual redirige al cliente a una nueva página donde poder crear una nueva incidencia si así se deseara.



```
<div class="toolbar" role="banner">
  <a [routerLink]="['']">
    <img
    alt="UPV Logo"
    src="assets/img/elementoUPV-RO.png"
    />
  </a>
  <span>Trabajo Fin de Grado - Jorge Catalá Valenzuela</span>
    <div class="spacer"></div>
    <a aria-label="Twitter" target="_blank" rel="noopener" href="https://twitter.com/jorge_catala" title="Twitter">
      <svg id="twitter-logo" height="24" data-name="Logo" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 400 400">
        <rect width="400" height="400" fill="none"/>
        <path d="M153.62,301.59c94.34,0,145.94-78.16,145.94-145.94,0-2.22,0-4.43-.15-6.63A104.36,104.36,0,0,0,325,122.47a102.38,102.38,0,0,
      </svg>
    </a>
</div>



<div class="container">
  <button style="margin-top:20px" mat-raised-button color="primary" [routerLink]="'/incidencias/add'">Crear Nueva Incidencia</button>
</div>
<div class="container">
  <h2>Listado de Incidencias</h2>
  <table mat-table [dataSource]="incidencias" class="mat-elevation-z8">

    <!--- Note that these columns can be defined in any order.
          The actual rendered columns are set as a property on the row definition" -->

    <!-- Name Column -->
    <ng-container matColumnDef="Motivo">
      <th mat-header-cell *matHeaderCellDef> Motivo </th>
      <td mat-cell *matCellDef="let incidencias"> {{incidencias.Motivo}} </td>
    </ng-container>
```

Figura 11 - Código html del componente "Incidencias" (1/2)

```
    <!-- Weight Column -->
    <ng-container matColumnDef="Descripción">
      <th mat-header-cell *matHeaderCellDef> Descripcion </th>
      <td mat-cell *matCellDef="let incidencias"> {{incidencias.Descripcion}} </td>
    </ng-container>

    <ng-container matColumnDef="Fecha">
      <th mat-header-cell *matHeaderCellDef> Fecha </th>
      <td mat-cell *matCellDef="let incidencias"> {{incidencias.Fecha}} </td>
    </ng-container>

    <tr mat-header-row *matHeaderRowDef="['Motivo', 'Descripción', 'Fecha']"></tr>
    <tr mat-row *matRowDef="let row; columns: ['Motivo', 'Descripción', 'Fecha'];"></tr>

  </table>
</div>


  <div>
    <df-messenger
      intent="WELCOME"
      chat-title="TFG"
      agent-id="62b56b1a-a8a8-43b3-8c4a-893de0e042b3"
      language-code="es"
  ></df-messenger>
  </div>
```

Figura 12 - Código html del componente "Incidencias" (2/2)

- **incidencias.component.ts:** archivo que define el comportamiento del componente. Comportamiento idéntico al descrito con anterioridad en el componente Catalogo. En la clase IncidenciasComponent, mediante

IncidenciasService, recogeremos las incidencias registradas en nuestra BD y las introduciremos en la variable “incidencias”, para así poder hacer referencia a esta en el código HTML y poder mostrar los datos específicos.

```ts
import { Component } from '@angular/core';
import { IncidenciasService } from '../incidencias.service';

@Component({
  selector: 'app-incidencias',
  templateUrl: './incidencias.component.html',
  styleUrls: ['./incidencias.component.css']
})
export class IncidenciasComponent {

  incidencias:any

  constructor(private incidenciaServicio: IncidenciasService) {}

  ngOnInit() {
    this.recuperarTodos();
  }

  recuperarTodos() {
    this.incidenciaServicio.getIncidencias().subscribe((result:any) => this.incidencias = result);
  }

}
```

Figura 13 - Código ts del componente "Incidencias"

- **incidencias.component.css:** archivo que contiene los estilos CSS aplicados (este archivo es igual en todos los Componentes del *front-end*).

- **incidencias.service.ts:** archivo TypeScript encargado de albergar la lógica responsable de realizar llamadas HTTP (*get* o *post*) a nuestro servidor para ejecutar los códigos php, en este caso “getAll_incidencia.php”, y así recuperar la información requerida de nuestra BD.

```ts
import { Injectable } from '@angular/core';
import { environment } from 'environments/environment';
import { HttpClient } from '@angular/common/http';
import { Incidencia } from './incidencia';

@Injectable({
  providedIn: 'root'
})
export class IncidenciasService {

  baseUrl = environment.baseUrl

  constructor(private http: HttpClient) { }

  getIncidencias() {
    return this.http.get(`${this.baseUrl}/getAll_incidencia.php`);
  }

  addIncidencia(incidencia: Incidencia) {
    return this.http.post(`${this.baseUrl}/post_incidencia.php`, incidencia);
  }
}
```

Figura 14 - Código ts del servicio "Incidencias Service"

4.1.4 Componente “Add-Incidencias”

Este componente será el encargado de crear nuevas incidencias y registrarlas en nuestra BD. Esta tarea se realizará mediante llamadas HTTP (post) con el fin de enviar la información a nuestro servidor para que con esta pueda lanzarla contra nuestra BD con la ayuda de archivos php.

En este componente también debemos añadir un archivo (“incidencia.ts”) el cual reflejará el esqueleto de la incidencia. Podremos rellenar dicho esqueleto con los datos que adjunte el cliente y así tener una incidencia real y correcta la cual podremos guardar en nuestra BD.

Su estructura será similar a la de los anteriores componentes:

- **add-incidencias.component.html:** archivo donde se representa el apartado visual de la página. En este componente tenemos una diferencia con respecto al resto, y es la de añadir dos botones, uno para enviar la información y otro para cancelar la creación de la incidencia.

```
<div class="toolbar" role="banner">
    <a [routerLink]="['']">
      <img
      alt="UPV Logo"
      src="assets/img/elementoUPV-RO.png"
      />
    </a>
    <span>Trabajo Fin de Grado - Jorge Catalá Valenzuela</span>
      <div class="spacer"></div>
      <a aria-label="Twitter" target="_blank" rel="noopener" href="https://twitter.com/jorge_catala" title="Twitter">
        <svg id="twitter-logo" height="24" data-name="Logo" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 400 400">
          <rect width="400" height="400" fill="none"/>
          <path d="M153.62,301.59c94.34,0,145.94-78.16,145.94-145.94,0-2.22,0-4.43-.15-6.63A104.36,104.36,0,0,0,325,122.47a102.38,102.38,0,
        </svg>
      </a>
</div>
```

Figura 15 - Código html del componente "Add-Incidencias" (1/3)

```
<div class="container">
    <h2>Agregar Incidencia</h2>
    <form (ngSubmit)="onSubmit()" #formIncidencia="ngForm">
        <p>
            <mat-form-field hintLabel="Max 25 caracteres" appearance="fill" [style.width.%]="25">
                <input matInput #input maxlength="25" name="Motivo" [(ngModel)]="incidenciaModel.Motivo"
                    type="text"
                    placeholder="Motivo">
                <mat-hint align="end">{{input.value.length}}/25</mat-hint>
            </mat-form-field>
        </p>
        <p>
            <mat-form-field style="height: 400px;" [style.width.%]="100">
                <textarea matInput name="Descripcion" [(ngModel)]="incidenciaModel.Descripcion" type="text" placeholder="Descripción"  rows
                </textarea>
            </mat-form-field>
        </p>
        <h2>Fecha de Publicación: {{fecha_final}}</h2>
        <p>
            <button type="submit" mat-raised-button color="primary">Enviar</button>
        </p>
        <p>
            <button style="margin-top:20px" mat-raised-button color="warn" [routerLink]="'/incidencias'">Cancelar</button>
        </p>
    </form>
</div>
```

Figura 16 - Código html del componente "Add-Incidencias" (2/3)

```
<div>
    <df-messenger
      intent="WELCOME"
      chat-title="TFG"
      agent-id="62b56b1a-a8a8-43b3-8c4a-893de0e042b3"
      language-code="es"
  ></df-messenger>
  </div>
```

Figura 17 - Código html del componente "Add-Incidencias" (3/3)

- **add-incidencias.component.ts:** archivo que define el comportamiento del componente. Comportamiento distinto al descrito con anterioridad en el componente Catalogo e Incidencias. En la clase AddIncidenciasComponent, mediante IncidenciasService, registraremos los datos entregados por el cliente dentro de nuestro modelo “IncidenciaModel”, el cual solo es un molde para poder identificar correctamente los apartados necesarios para la creación de una nueva incidencia. Es así como el cliente solo debe introducir un motivo (no más de 25 caracteres) y una descripción detallada, la fecha será introducida por una lógica externa a este e inmodificable para poder identificar la fecha exacta de la creación de dicha incidencia.

```ts
import { Component, OnInit } from '@angular/core';
import { IncidenciasService } from '../incidencias.service';
import { Router } from '@angular/router';
import { Incidencia } from '../incidencia';
import { MatSnackBar } from '@angular/material/snack-bar';

@Component({
  selector: 'app-add-incidencias',
  templateUrl: './add-incidencias.component.html',
  styleUrls: ['./add-incidencias.component.css'],
})

export class AddIncidenciasComponent implements OnInit {


  constructor(private incidenciaService: IncidenciasService,
    public snackBar: MatSnackBar,
    private router: Router,
  ) { }

  todayISOString : string = new Date().toISOString();

  fecha = this.todayISOString.split("-", 3)

  year: string = this.fecha[0]
  month: string = this.fecha[1]
  day: string = this.fecha[2]

```

Figura 18 - Código ts del componente "Add-Incidencias" (1/2)

```
29    day_s = this.day.split("T")
30
31    day_final: string = this.day_s[0]
32
33    fecha_final: string = this.day_final+"-"+this.month+"-"+this.year
34
35    ngOnInit() {
36    }
37    incidenciaModel = new Incidencia("", "", this.fecha_final)
38
39    onSubmit() {
40      this.incidenciaService.addIncidencia(this.incidenciaModel).subscribe(() => {
41        this.snackBar.open('Incidencia Enviada', undefined, {
42          duration: 1500,
43        });
44        this.router.navigate(['/incidencias']);
45      })
46    }
47
48  }
49
50
```

Figura 19 - Código ts del componente "Add-Incidencias" (2/2)

- **add-incidencias.component.css:** archivo que contiene los estilos CSS aplicados (este archivo es igual en todos los Componentes del *front-end*).

- **incidencias.service.ts:** archivo TypeScript encargado de albergar la lógica responsable de realizar llamadas HTTP (*get* o *post*) a nuestro servidor para ejecutar los códigos php, en este caso "post_incidencia.php", y así poder registrar la incidencia del cliente en nuestra BD de forma correcta.

- **incidencia.ts:** archivo que define el esqueleto que tendrán las incidencias. Lo utilizaremos para rellenarlo con los datos aportados por el cliente y así hacer un envío y registro válidos para nuestra BD.

```
1  export class Incidencia {
2      constructor(
3          public Motivo: string,
4          public Descripcion: string,
5          public Fecha: string,
6          public Id?: number,
7      ) { }
8
9  }
```

Figura 20 - Código ts de la clase "Incidencia"

4.1.5 Componentes “ProductoX”

Estos componentes serán los encargados de mostrar la información de un producto en concreto de nuestro catálogo, diferenciándolos por su Id correspondiente en nuestra tabla de la BD.

Su estructura será:

- **productoX.component.html:** archivo donde se representa el apartado visual de la página. En este componente podremos visualizar el nombre del producto en cuestión y su precio por separado.

```
<div class="toolbar" role="banner">
    <a [routerLink]="['']">
      <img
      alt="UPV Logo"
      src="assets/img/elementoUPV-RO.png"
      />
    </a>
    <span>Trabajo Fin de Grado - Jorge Catalá Valenzuela</span>
      <div class="spacer"></div>
      <a aria-label="Twitter" target="_blank" rel="noopener" href="https://twitter.com/jorge_catala" title="Twitter">
        <svg id="twitter-logo" height="24" data-name="Logo" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 400 400">
          <rect width="400" height="400" fill="none"/>
          <path d="M153.62,301.59c94.34,0,145.94-78.16,145.94-145.94,0-2.22,0-4.43-.15-6.63A104.36,104.36,0,0,0,325,122.47a102.38,102.38,0,
        </svg>
      </a>
  </div>

  <div class="container">
    <h1>PRODUCTO</h1>
    <table mat-table [dataSource]="productos" class="mat-elevation-z8">

      <!-- Name Column -->
      <ng-container matColumnDef="Nombre">
        <th mat-header-cell *matHeaderCellDef> Nombre del Producto </th>
        <td mat-cell *matCellDef="let productos"> {{productos.Nombre}} </td>
      </ng-container>

      <tr mat-header-row *matHeaderRowDef="['Nombre']"></tr>
      <tr mat-row *matRowDef="let row; columns: ['Nombre'];"></tr>

    </table>
  </div>
```

Figura 21 - Código html del componente "ProductoX" (1/2)

```
<div class="container">
  <h1>PRECIO</h1>
  <table mat-table [dataSource]="productos" class="mat-elevation-z8">

    <!-- Weight Column -->
    <ng-container matColumnDef="Precio">
      <th mat-header-cell *matHeaderCellDef> Precio del Producto </th>
      <td mat-cell *matCellDef="let productos"> {{productos.Precio}}€ </td>
    </ng-container>

    <tr mat-header-row *matHeaderRowDef="['Precio']"></tr>
    <tr mat-row *matRowDef="let row; columns: ['Precio'];"></tr>

  </table>
</div>


<div>
  <df-messenger
    intent="WELCOME"
    chat-title="TFG"
    agent-id="62b56b1a-a8a8-43b3-8c4a-893de0e042b3"
    language-code="es"
></df-messenger>
</div>
```

Figura 22 - Código html del componente "ProductoX" (2/2)

- **productoX.component.ts:** archivo que define el comportamiento del componente. En la clase ProductoXComponent, mediante CatalogoService, recogeremos los productos registrados en nuestra BD haciendo una llamada al archivo php correspondiente (get_productoX.php), y los introduciremos en la variable “productos”, para así poder hacer referencia a esta en el código HTML y poder mostrar los datos específicos.

```typescript
import { Component } from '@angular/core';
import { CatalogoService } from '../catalogo.service';

@Component({
  selector: 'app-producto1',
  templateUrl: './producto1.component.html',
  styleUrls: ['./producto1.component.css']
})
export class Producto1Component {

  productos:any

  constructor(private catalogoServicio: CatalogoService) {}

  ngOnInit() {
    this.recuperarTodos();
  }

  recuperarTodos() {
    this.catalogoServicio.getRobot1().subscribe((result:any) => this.productos = result);
  }

}
```

Figura 23 - Código ts del componente "ProductoX"

- **productoX.component.css:** archivo que contiene los estilos CSS aplicados (este archivo es igual en todos los Componentes del *front-end*).

- **catalogo.service.ts:** archivo TypeScript encargado de albergar la lógica responsable de realizar una llamada HTTP (*get* en este caso) a nuestro servidor para ejecutar el código php correspondiente y así recuperar la información requerida de nuestra BD.

4.1.6 *Environments*

A parte de todos los componentes anteriores también debemos generar las variables de entorno necesarias para poder realizar la conexión con nuestro *back-end* (servidor).

Esta sección se llamará "*environments*" y contendrá dos archivos:

- **environment.ts:** contendrá las variables en el entorno de pruebas o en caso de fallo. Para poder ser el principal (por defecto) debemos indicarlo con production: true. En este caso no se cumple.

```typescript
export const environment = {
    production: true,
    baseUrl: ".",
}
```

Figura 24 - Código ts de la constante "environment"

- **environment.development.ts:** contendrá las variables en el entorno por defecto. Para poder ser el principal (por defecto) debemos indicarlo con production: true.

En este caso se cumple, añadiéndole la url donde debe atacar y donde se aloja nuestra BD.

```
export const environment = {
    production: false,
    baseUrl: "http://localhost:80"
}
```

Figura 25 - Código ts de la constante "environment.development"

4.2 DESARROLLO *BACK-END*

4.2.1 Tablas

Nuestro *back-end* estará formado por una pequeña base de datos alojada en nuestro servidor Apache. Esta BD será por tanto implementada en MySQL, donde podremos generar y gestionar nuestras diferentes tablas.

Dentro de este existirán dos tablas, a las cuales ya hemos hecho referencia con anterioridad:

- **Catalogo:** responsable de almacenar todos los productos disponibles de nuestro negocio ficticio. Conformada por las columnas “Id”, “Nombre” y “Precio”.

Id	Nombre	Precio
1	Robot1	55.79
2	Robot2	69.99
3	Robot3	49.59
4	Brazo Mecánico	89.99

Figura 26 - Ilustración de la estructura de la tabla "Catalogo"

- **Incidencias:** responsable de almacenar todas las incidencias creadas por el cliente en nuestro negocio ficticio. Conformada por las columnas “Id”, “Motivo”, “Detalle” y “Fecha”.

Id	Motivo	Descripcion	Fecha
1	Mal Funcionamiento Robot	El producto no funciona de manera correcta, se des...	27-02-2023
2	Apagado automatico	El robot se apaga automáticamente a los 10 minutos...	27-02-2023

Figura 27 - Ilustración de la estructura de la tabla "Incidencias"

4.2.2 Archivos PHP

Dentro de nuestro servidor alojaremos una lógica mediante archivos PHP para realizar operaciones sobre las tablas de nuestra BD.

Estos archivos serán invocados por llamadas HTTP (*get* o *post*) por nuestro *front-end*, el cual nos indicará que archivo debemos ejecutar y en caso necesario, que información necesaria nos otorga para poder registrarla en las tablas.

Estos archivos son los siguientes:

- **bd.php:** archivo encargado de realizar la conexión a nuestra BD con las credenciales correspondientes.

```php
<?php
$contraseña = "";
$usuario = "root";
$nombre_base_de_datos = "tfg";
try {
    return new PDO('mysql:host=localhost;dbname=' . $nombre_base_de_datos, $usuario, $contraseña);
} catch (Exception $e) {
    echo "Ocurrió algo con la base de datos: " . $e->getMessage();
}
```

Figura 28 - Código php del archivo "bd.php"

- **getAll.php:** archivo encargado de recoger toda la información almacenada en la tabla “catalogo” para después ser mostrada al cliente mediante el *front-end*.

```php
<?php
header("Access-Control-Allow-Origin: http://localhost:4200");
$bd = include_once "bd.php";
$sentencia = $bd->query("select Id, Nombre, Precio from catalogo");
$productos = $sentencia->fetchAll(PDO::FETCH_OBJ);
echo json_encode($productos);
```

Figura 29 - Código php del archivo "getAll.php"

- **getAll_incidencia.php:** archivo encargado de recoger toda la información almacenada en la tabla “incidencias” para después ser mostrada al cliente mediante el *front-end*.

```php
<?php
header("Access-Control-Allow-Origin: http://localhost:4200");
$bd = include_once "bd.php";
$sentencia = $bd->query("select Id, Motivo, Descripcion, Fecha from incidencias");
$incidencias = $sentencia->fetchAll(PDO::FETCH_OBJ);
echo json_encode($incidencias);
```

Figura 30 - Código php del archivo "getAll_incidencia.php"

- **post_incidencia.php:** archivo encargado de realizar un post en la tabla “incidencias”. El *front-end* enviará los datos introducidos por el cliente y este archivo será el encargado de registrarlos dentro de dicha tabla.

```php
<?php
header("Access-Control-Allow-Origin: http://localhost:4200");
header("Access-Control-Allow-Headers: *");
$jsonIncidencia = json_decode(file_get_contents("php://input"));
if (!$jsonIncidencia) {
    exit("No hay datos");
}
$bd = include_once "bd.php";
$sentencia = $bd->prepare("insert into incidencias(Motivo, Descripcion, Fecha) values (?,?,?)");
$resultado = $sentencia->execute([$jsonIncidencia->Motivo, $jsonIncidencia->Descripcion, $jsonIncidencia->Fecha]);
echo json_encode([
    "resultado" => $resultado,
]);
```

Figura 31 - Código php del archivo "post_incidencia.php"

- **get_robotX.php:** archivo encargado de recoger la información de un producto del catálogo en concreto, identificado por su Id.

```php
<?php
header("Access-Control-Allow-Origin: http://localhost:4200");
$bd = include_once "bd.php";
$sentencia = $bd->query("select Id, Nombre, Precio from catalogo WHERE Id=X");
$productos = $sentencia->fetchAll(PDO::FETCH_OBJ);
echo json_encode($productos);
```

Figura 32 - Código php de los archivos "get_robotX.php"

4.3 CONFIGURACIÓN DEL CHATBOT

En los siguientes subapartados haremos una explicación de los pasos a seguir para generar/configurar nuestro chatbot. También repasaremos los distintos elementos que pueden conformarlo, definiendo cómo se comportan y de qué manera haremos uso de estos para enriquecerlo.

4.3.1 Generación del chatbot

Para la creación de un bot con la herramienta Dialogflow es necesario realizar ciertas operaciones desde su página web[i]:

- Primero debemos registrarnos utilizando una dirección de correo con dominio en Gmail, la cual se sincronizará con una cuenta de Google Cloud.

- Una vez hayamos realizado este paso debemos dirigirnos al apartado "*CREATE AGENT*". Aquí tendremos que especificar ciertos parámetros como el nombre del agente, el idioma que se va a utilizar y la zona horaria (Figura 33). Si se desea, se pueden definir parámetros adicionales como una descripción, un avatar o el identificador del proyecto, todo esto desde el panel de control de nuestro agente (Figura 34).

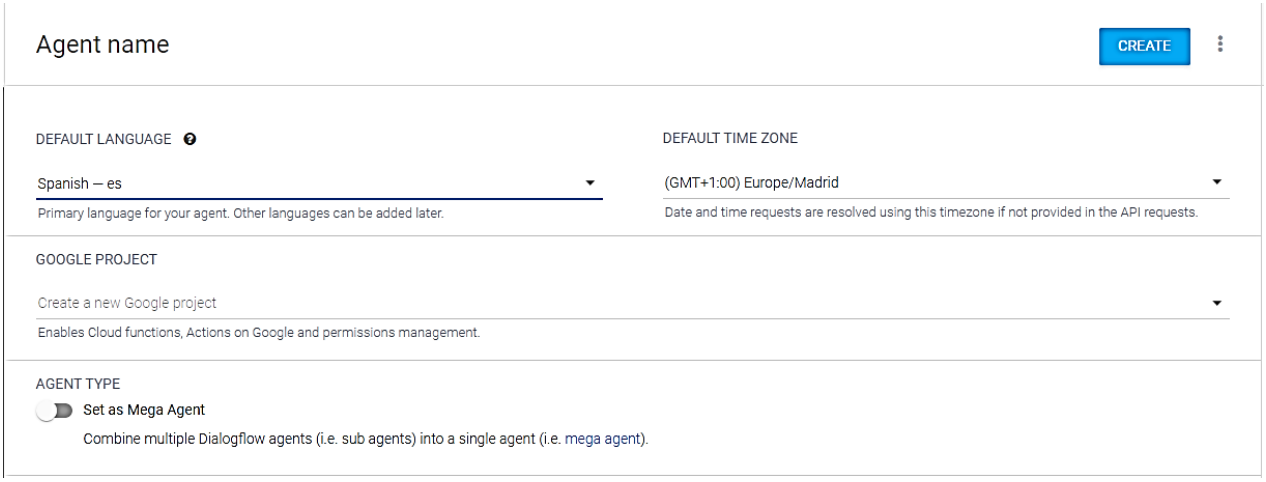


Figura 33 - Menú de creación de un nuevo agente

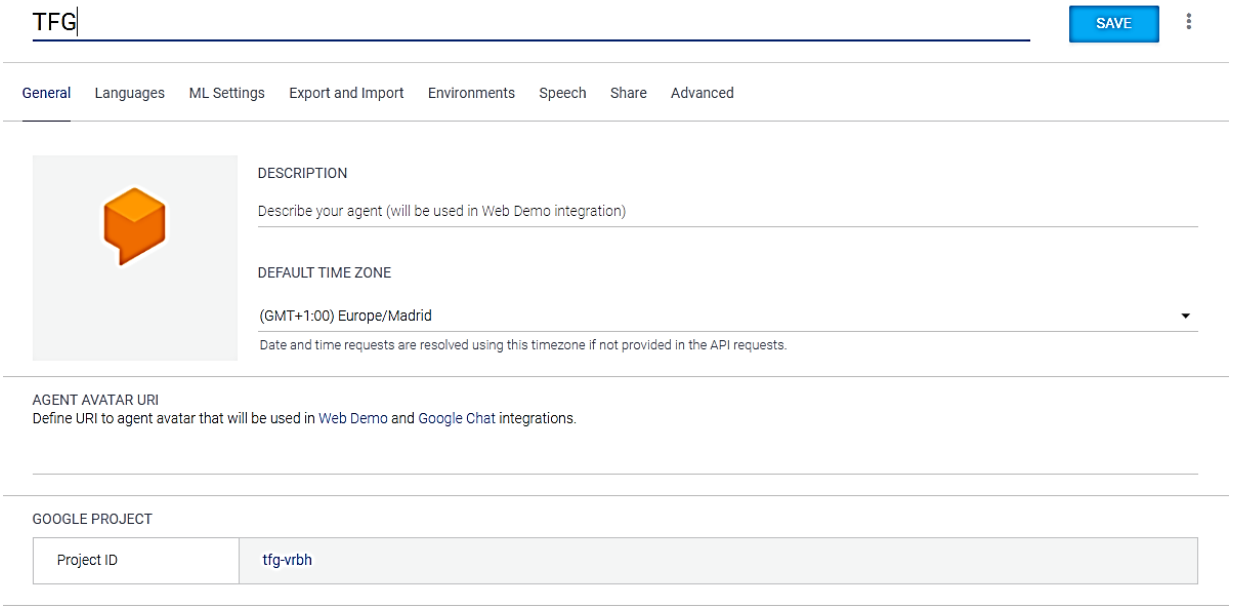


Figura 34 - Panel de control de nuestro agente

Con esto finalizamos la creación/generación y asignación de parámetros de nuestro agente. Los pasos siguientes se asocian a aumentar la riqueza de este en interacciones y respuestas.

4.3.2 Intenciones (*Intents*)

Dialogflow tiene diferentes tipos de interacciones. La más básica son las denominadas "*Intents*", estas se utilizan para contestaciones básicas sin la necesidad de facilitar un contexto o parámetro alternativo.

Un "*Intent*" contiene ciertos elementos que permiten al bot responder con gran acierto y rápidez. Realizaremos una explicación breve:

- **Frases de entrenamiento:** consisten en frases posibles por parte del usuario, Dialogflow las interpreta y establece relaciones entre ellas para establecer un patrón concreto y así poder ver coincidencias con los *intents*.

- **Acciones (opcional):** básicamente es una consecuencia tras un *intent*. Cada *intent* tiene una acción definida que se pueden utilizar para desencadenar la activación de otras.

- **Parámetros (opcional):** son valores que Dialogflow es capaz de extraer de las expresiones del usuario. Con estos valores podríamos diseñar alguna secuencia lógica o generar ciertas respuestas únicas. Ahondaremos en este apartado más adelante.

- **Respuestas:** son las contestaciones que se da a ciertos *intents*. Es un aspecto obligatorio que se debe definir, ya sea en forma de texto, voz o elementos visuales. También existe la opción de definir una respuesta como fin de una conversación.

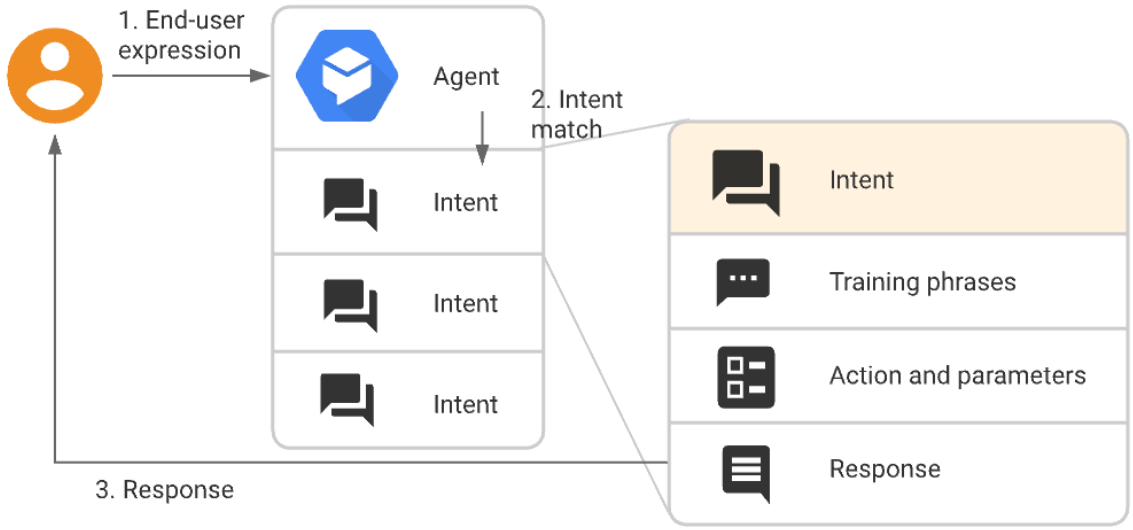


Figura 35 - Diagrama de un flujo básico de un intent/respuesta [22]

Aquí vemos (Figura 35) un diagrama del que podría ser perfectamente un flujo básico en una conversación con un chatbot.

1. Primero tendríamos el contacto con el bot, donde nuestras expresiones entran dentro del entorno del agente para ser analizadas.

2. Este recorre todas sus intenciones programadas para intentar detectar coincidencias. Vemos que cada intención puede estar formada por diversas fases, como acciones o recogida de parámetros, para su correcta compatibilidad con las expresiones introducidas.

3. Una vez se detecta la concordancia de la expresión con alguna de las intenciones, y después de realizar las acciones correspondientes, se genera una respuesta que se le envía al usuario.

4.3.3 Entidades (*Entities*)

Una *Entity* es un contenedor con datos definidos, ya sean personalizados o predefinidos por el propio Dialogflow.

Volvemos a los parámetros mencionados en el punto anterior. Estos serán utilizados para ser matcheados con los datos dentro de una *Entity* y poder dar una respuesta más específica relacionada con las frases facilitadas por el usuario.

¿Cómo funciona? En primer lugar, se genera una *Entity* y se definen nuestros valores deseados que serán referenciados por el usuario. En nuestro caso la *Entity* será "Productos". En esta encontraremos todos los productos existentes en nuestro catálogo y sinónimos de estos por si el cliente utilizara otra expresión para referirse a ellos.

Robot1	Robot1, robot 1, Robot 1, Robot numero 1
Robot2	Robot2, Robot 2, robot2, Robot numero 2
Robot3	Robot 3, robot 3, Robot numero 3
Robot4	Brazo Mecanico, brazo mecanico, Brazo Mecánico, brazo mecánico

Figura 36 - Entity "@Productos"

Una vez hemos definido nuestra *Entity*, crearemos un *Intent* y resaltaremos los nombres de nuestros productos en las frases que esperamos que introduzca el usuario, e indicaremos que tipo de respuesta queremos dar dependiendo del producto. En nuestro caso, la respuesta será:

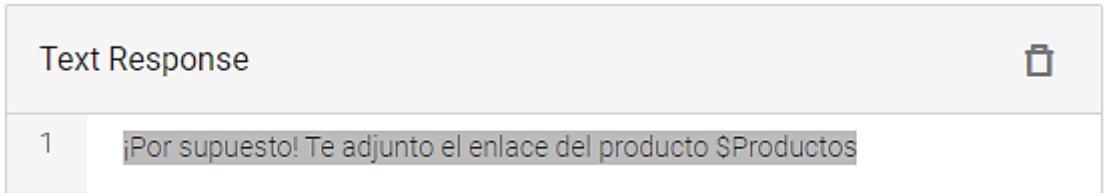


Figura 37 - Respuesta del Intent "get_products"

Donde "$Productos" es el parámetro que Dialogflow extraerá de la conversación con el usuario.

Además de generar una respuesta distintiva, también añadiremos un botón con la función de redireccionar al usuario a la página web encargada de representar dicho producto, así dependiendo del parámetro que nos proporcione el usuario podremos redirigirlo a un sitio u otro. El código que implementa este concepto se muestra en la siguiente imagen (Figura 38).

Custom Payload

```
{
  "richContent": [
    [
      {
        "options": [
          {
            "text": "$Productos",
            "link": "http://localhost:4200/producto/$Productos"
          }
        ],
        "type": "chips"
      }
    ]
  ]
}
```

Figura 38 - Código personalizado del Intent "get_products"

4.4 IMPLEMENTACIÓN CHATBOT ↔ *FRONT-END*

Dialogflow ofrece una gran variedad de formas de integración. En su portal, en el apartado *"Integrations"* podemos encontrar sus distintos apartados:

- **Telefonía en un click:** aplicaciones como Avaya[ii], SignalWire[iii], Voximplant[iv], AudioCodes[v], Twilio[vi] y una más genérica propia de Dialogflow llamada Dialogflow Phone Gateway[vii].

- **Telefonía:** referido a aplicaciones de telefonía más clásicas. Encontramos Genesys Cloud[viii] y Twilio.

- **Basadas en texto:** estas aplicaciones son, como en nuestro caso, las que requieren de un chatbot que se comunique mediante texto. Por eso podemos encontrar a Messenger (Facebook[ix]), Workplace (Facebook[x]), Google Chat[xi], Slack[xii], Telegram[xiii] y LINE[xiv]. A parte encontramos los métodos de integración genéricos de Dialogflow: Web Demo[xv] y Dialogflow Messenger[xvi].

- **Open source:** por último, están las aplicaciones de código abierto. Kik[xvii], Skype[xviii], Spark[xix], Twilio IP Messaging, Twilio (Text Messaging), Twitter[xx] y Viber[xxi].

Aplicaciones / Apartados	One-click Telephony	Telephony	Text based	Open source
Avaya	×			
SignalWire	×			
Voximplant	×			
AudioCodes	×			
Twilio (IP Messaging & Text Messaging)	×	×		×
Dialogflow Phone Gateway	×			
Genesys Cloud		×		
Messenger from Facebook			×	
Workplace from Facebook			×	
Google Chat			×	
Slack			×	
Telegram			×	
LINE			×	
Dialogflow Web Demo			×	
Dialogflow Messenger			×	
Kik				×
Skype				×
Spark				×
Twitter				×
Viber				×

Tabla 3 - Apartados de distintas integraciones

Nuestro chatbot está basado en la comunicación por vía texto con el usuario, ya que nuestra comunicación con este pasa por un chat vía mensajería en nuestra página web. Es por eso por lo que nos situamos en el apartado de aplicaciones basadas en texto (*text based*). Aquí encontramos muchas opciones, pero nuestro proyecto está constituido por un front-end genérico/personalizado en un framework concreto, es decir, no es una aplicación ya establecida y definida. Esto reduce las similitudes que podríamos tener con el resto de aplicaciones de la lista y las posibilidades de una integración del bot utilizando sus entornos.

Nos decantamos entonces por la opción de Dialogflow Messenger por las razones mencionadas anteriormente y por la gran facilidad que ofrece a la hora de su integración con páginas webs genéricas/personalizadas como es la nuestra.

Si hacemos click en este apartado podemos observar con que facilidad podemos integrar nuestro chatbot en nuestro front-end. Como resultado aparece un cuadro de configuración con un código html que pasamos a detallar en la siguiente imagen (Figura 39).

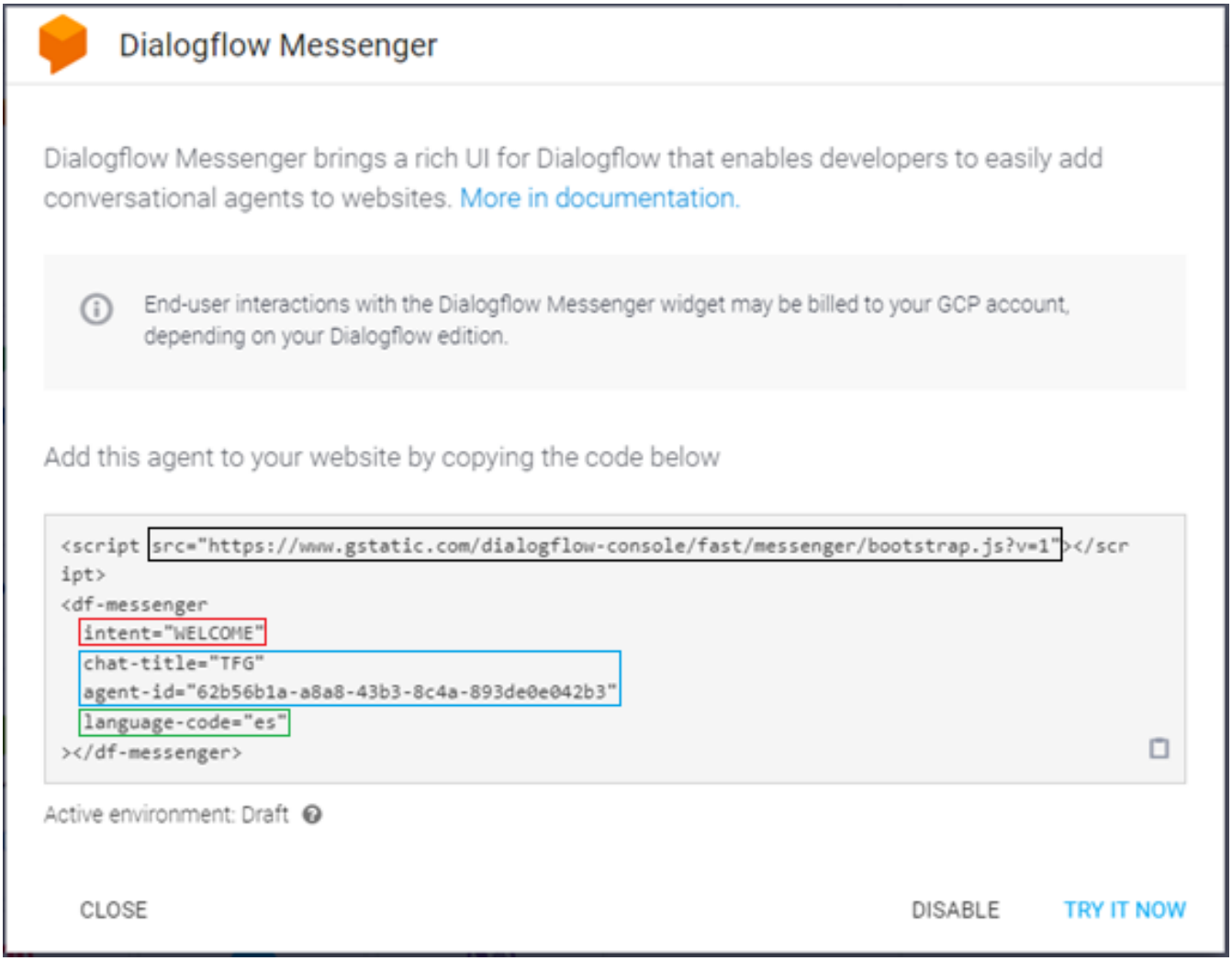


Figura 39 - Cuadro de opciones Dialogflow Messenger

- **Recuadro ROJO:** indica la intención (*intent*) principal con el cual el chatbot recibirá a los usuarios, en este caso "*WELCOME*".

- **Recuadro AZUL:** se indica el nombre de nuestro agente y su id asociado.

- **Recuadro VERDE:** se especifica el lenguaje en el que se espera que se establezca la conversación y en el que está configurado el chatbot. En nuestro caso será el español.

- **Recuadro NEGRO:** enlace del script que se ejecutará en nuestro *front-end* una vez hayamos añadido dicho código html.

Si observamos el código html adjunto en anteriores apartados de los componentes que conforman nuestro *front-end*, se pueden encontrar estas líneas de código integradas en cada uno de estos. Asegurándonos así de la aparición de nuestro chatbot en cada una de las páginas.

En nuestra opinión, la integración con diferentes aplicaciones está muy lograda por parte de Dialogflow. El método de integración es muy sencillo y sin apenas inconvenientes. También hay que resaltar que este método está en una versión BETA, por eso la personalización del icono del agente y de las ventanas de conversaciones no están muy logradas. Es muy probable que estas funcionalidades sean mejoradas próximamente.

5 CONCLUSIONES, LIMITACIONES Y FUTURAS LÍNEAS DE TRABAJO

5.1 CONCLUSIÓN

Para hacer una evaluación sobre la gestión adecuada del trabajo y la efectividad en los recursos utilizados, es necesario enfocarse en dos aspectos principales: el cumplimiento de los objetivos establecidos y los conocimientos previos enfocados al proyecto y los adquiridos durante este. Para llegar a conclusiones precisas, un buen ejercicio sería el de repasar los objetivos previamente definidos y analizar su nivel de cumplimiento.

1. El despliegue del front-end ha sido en cierta forma sencillo debido a lo aprendido en asignaturas del grado como pueden ser programación o ingeniería de sistemas telemáticos. Con la base proporcionada por estas, un período de tiempo en consultoras como desarrollador de software y el aprendizaje de ciertos aspectos durante el desarrollo del proyecto, este objetivo ha sido abarcado con cierta soltura y conocimiento.

2. La creación de una base de datos y tablas para poder alimentar al modelo se ha cumplido, en gran medida, gracias a la formación recibida en la asignatura de sistemas telemáticos para la gestión de la información (STGI). Aquí vimos todo lo relacionado con bases de datos y ha sido de gran apoyo para este apartado.

3. Este punto ha sido el más instructivo de todos debido a la gran variedad de herramientas para la creación de chatbots que se han estudiado durante la realización del marco teórico del proyecto. En nuestra opinión, hemos escogido la herramienta que más se adecuaba a nuestros requisitos (Dialogflow). Y siendo verdad que se podría haber profundizado más en todas las características que nos ofrece (cosa que se repasa en el siguiente apartado), pensamos que el objetivo ha sido cumplido con creces, al fin y al cabo, el tiempo que se puede dedicar a esta herramienta puede ser tan extenso como uno quiera.

4. Por último, el objetivo de entrenar al chatbot para que sea más eficiente e integrarlo en nuestra arquitectura. El cumplimiento de este punto ha sido completamente enriquecedor y sencillo para nosotros debido a las elecciones realizadas en puntos anteriores. El hecho de elegir un framework como Angular y una herramienta para nuestro chatbot como Dialogflow ha permitido que la integración de este último haya resultado menos compleja de lo que se esperaba en un principio. Y al respecto del entrenamiento del chatbot, la herramienta de Dialogflow tiene un método propio para aumentar las capacidades de sus agentes y refinar la recepción de expresiones y sus respuestas.

Después de hacer una revisión de los objetivos marcados al inicio del proyecto, podemos afirmar la adquisición de conocimientos debido a la utilización de tecnologías en las que no se había profundizado suficiente o directamente eran desconocidas, como también el cumplimiento de la finalidad del proyecto desde su inicio.

Finalmente, a modo personal, considero que este trabajo me ha ayudado a la expansión de mis conocimientos sobre distintos temas tecnológicos y también a la habilidad de poder distribuir el tiempo disponible y organizarlo de forma adecuada. Estos aspectos me han hecho crecer y dar un paso más adelante en mi entorno laboral.

5.2 LIMITACIONES Y FUTURAS LÍNEAS DE TRABAJO

5.2.1 Limitaciones

El principal aspecto limitante de este proyecto ha sido el tiempo, debido a circunstancias laborales y personales no se ha podido dedicar todo el tiempo que se habría deseado. De todas formas, agradecer enormemente a mi tutor el tiempo y ayuda dedicados para la realización del trabajo.

Una limitación que ha supuesto cierto cambio de rumbo ha sido la conexión de nuestra base de datos con la herramienta de Dialogflow. Se ha estudiado todas las posibilidades de esta herramienta y son muy extensas, pero por condicionamientos del framework

escogido no hemos podido realizar de forma efectiva la conexión de nuestro *back-end* con el chatbot, teniendo que mapear los elementos de uno en el otro.

5.2.2 Futuras líneas de trabajo

Con respecto a un trabajo futuro, como se ha indicado en los puntos anteriores los objetivos del proyecto han sido alcanzados. Aun así, hay un margen de mejora como podrían ser los siguientes aspectos:

- Mejora del diseño del *front-end*. Debido al tratarse de un trabajo de fin de grado, los plazos establecidos se deben cumplir con obligatoriedad, forzando a la selección concreta y acotada de los objetivos en función del tiempo disponible. Esto nos condiciona la dedicación al diseño de nuestra página web.

- Establecer una conexión directa entre el chatbot y el *back-end* para la posible realimentación de los datos comprendidos en las entidades (*entities*).

- Creación y configuración de una infinidad de intenciones (*intents*) para enriquecer aún más nuestro bot.

- Generación de aún más agentes que interactúen entre ellos para formar un entorno mucho más robusto para la recolecta y entrega de información al usuario.

- Una posibilidad de mejora sería el almacenar toda la arquitectura (*front-end* y *back-end*) en servicios cloud como podrían ser Google Cloud Platform[xxii], Microsoft Azure[xxiii] o Amazon Web Services[xxiv].

6 BIBLIOGRAFÍA

[1] A. S. T. &. S. N. Dhage, «Survey on Virtual Assistant: Google Assistant, Siri, Cortana, Alexa,» *International Symposium on Signal Processing and Intelligent Recognition Systems*, vol. 968, pp. 190-201, 2019.

[2] C. d. Wikipedia, «Asistente Virtual,» 27 10 2022. [En línea]. Available: https://es.wikipedia.org/wiki/Asistente_virtual. [Último acceso: 25 01 2023].

[3] C. d. Wikipedia, «Amazon Alexa,» 10 10 2022. [En línea]. Available: https://es.wikipedia.org/wiki/Amazon_Alexa. [Último acceso: 25 01 2025].

[4] C. d. Wikipedia, «Microsoft Cortana,» 10 01 2023. [En línea]. Available: https://es.wikipedia.org/wiki/Microsoft_Cortana. [Último acceso: 26 01 2023].

[5] C. d. Wikipedia, «Google Home,» 11 11 2022. [En línea]. Available: https://es.wikipedia.org/wiki/Google_Home. [Último acceso: 25 01 2023].

[6] R. A. ESPAÑOLA, «Definición de Inteligencia Artificial,» Diccionario de la Lengua Española de la Real Academia Española, 2023. [En línea]. Available: https://dle.rae.es/inteligencia#2DxmhCT. [Último acceso: 27 01 2023].

[7] C. d. Wikipedia, «Inteligencia Artificial,» 26 01 2023. [En línea]. Available: https://es.wikipedia.org/wiki/Inteligencia_artificial. [Último acceso: 27 01 2023].

[8] J. McCarthy, «WHAT IS ARTIFICIAL INTELLIGENCE?,» *Computer Science Department*, pp. 2-14, 2004.

[9] ChatGPT, «ChatGPT,» 2023. [En línea]. Available: https://chat.openai.com/chat/. [Último acceso: 27 01 2023].

[10] R. G. y. D. Guilbeault, «Unpacking the Social Media Bot: A Typology to Guide Research and Policy,» *Policy & Internet*, vol. 12, pp. 225-248, 2020.

[11] S. J. y. K. M. Guendalina Caldarini, «A Literature Survey of Recent Advances in Chatbots,» *MDPI and ACS Style*, vol. 13, pp. 1-41, 2022.

[12] C. d. Wikipedia, «Chatbot,» 24 02 2023. [En línea]. Available: https://en.wikipedia.org/wiki/Chatbot. [Último acceso: 25 02 2023].

[13] F. J. García-Peñalvo, «La percepción de la Inteligencia Artificial en contextos educativos tras el lanzamiento de ChatGPT: disrupción o pánico.,» *Education in The Knowledge Society (EKS)*, vol. 24, 06 02 2023.

[14] C. Baraniuk, «How talking machines are taking call centre jobs,» *BBC*, 24 08 2018.

[15] C. www.piab.com, «Damos habilidades a los Robots,» *Revista de Robots, Robotica y Automatización Industrial*, vol. 1, 2020.

[16] S. J.-P. E. G. y. J. d. L. Sara Pérez-Soler, «Choosing a Chatbot Development Tool,» *IEEE Xplore*, vol. 38, nº 4, pp. 94-103, 2021.

[17] G. A.-H. R. V.-G. L. R.-M. A. R.-G. y. J. L. L. C. María del Pilar Salas-Zárate, «Analyzing best practices on Web development frameworks: The lift approach,» *Science of Computer Programming*, vol. 102, pp. 1-19, 2015.

[18] A. Team, «Introduction to the Angular docs,» 2023. [En línea]. Available: https://angular.io/docs. [Último acceso: 10 02 2023].

[19] W. Team, «TypeScript Introduction,» 2023. [En línea]. Available: https://www.w3schools.com/typescript/typescript_intro.php. [Último acceso: 10 02 2023].

[20] A. Team, «Angular components overview,» 2023. [En línea]. Available: https://angular.io/guide/component-overview. [Último acceso: 15 02 2023].

[21] A. Team, «Introduction to services and dependency injection,» 2023. [En línea]. Available: https://angular.io/guide/architecture-services. [Último acceso: 25 02 2023].

[22] G. Developers, «Intents | Dialogflow ES - Google Cloud,» 05 07 2022. [En línea]. Available: https://cloud.google.com/dialogflow/es/docs/intents-overview?hl=es-419. [Último acceso: 28 02 2023].

[23] J. Trivedi, «Basic Architecture of Angular 2 Applications,» 01 09 2016. [En línea]. Available: https://www.c-sharpcorner.com/article/basic-architecture-of-angular-2-applications/. [Último acceso: 28 02 2023].

[24] S. T. Sil, «La robótica industrial en el ámbito de la automatización global: estado actual y tendencias,» *Técnica Industrial*, pp. 26-39, 12 07 2014.

[i] https://dialogflow.cloud.google.com

[ii] https://www.avaya.com/es/ - Empresa de telecomunicaciones especializada en el sector de la telefonía y call centers.

[iii] https://signalwire.com - API y plataforma de productos que ofrecen comunicaciones a nivel empresarial y de transmisión por todo el globo.

[iv] https://voximplant.com/es - Empresa latinoamericana que ofrece soluciones de voz, video y mensajería escalable. Desde tecnologías serverless a no-code.

[v] https://www.audiocodes.com/es - Proveedor de software, productos y soluciones de productividad de comunicaciones avanzadas.

[vi] https://www.twilio.com/ - Empresa que proporciona herramientas de comunicación programables para realizar y recibir llamadas telefónicas, enviar y recibir mensajes de texto y realizar otras funciones de comunicación mediante su servicio web API.

[vii] https://cloud.google.com/dialogflow/es/docs/integrations/phone-gateway?hl=es-419 – Es una función que proporciona una interfaz telefónica para tu agente. Se usa para crear soluciones de IVR (respuesta de voz interactiva) conversacionales que se integren en el resto de la red del centro de llamadas.

[viii] https://www.genesys.com - Empresa que vende experiencia del cliente y tecnología de centros de llamadas para medianas y grandes empresas. También vende software basado tanto en computación en la nube como on-premises.

[ix] https://www.facebook.com/messenger/ - Aplicación de mensajería desarrollada por Meta.

[x] https://es-la.workplace.com - red social desarrollada por Meta de tipo empresarial.

[xi] https://workspace.google.com/intl/es/products/chat/ - Software de comunicación desarrollado por Google, creado para equipos que proporciona mensajes directos y salas de chat de equipo, junto con una función de mensajería grupal.

[xii] https://slack.com/intl/es-es - Programa de mensajería instantánea diseñado por Slack Technologies. Desarrollado para comunicaciones profesionales y organizacionales, también se ha adoptado como una plataforma comunitaria.

[xiii] https://web.telegram.org - Telegram es una plataforma de mensajería y VOIP de origen ruso. La aplicación está enfocada en la mensajería instantánea, el envío de varios archivos y la comunicación en masa.

[xiv] https://line.me/en/ - Line es una aplicación de mensajería instantánea para teléfonos móviles, PC y Mac. Además de la mensajería básica, se pueden enviar imágenes, vídeos, mensajes de audio y hacer llamadas VoIP.

[xv] https://cloud.google.com/dialogflow/es/docs/integrations/web-demo?hl=es-419 - Proporciona una interfaz de usuario de chat de texto simple para tu agente.

[xvi] https://cloud.google.com/dialogflow/es/docs/integrations/dialogflow-messenger?hl=es-419 - Proporciona un cuadro de diálogo de chat personalizable para tu agente que se puede insertar en un sitio web.

[xvii] https://kik.com - Aplicación de mensajería instantánea gratuita para dispositivos móviles.

[xviii] https://www.skype.com/es/ - Software propietario distribuido por Microsoft. Permite comunicaciones de texto, voz y vídeo sobre Internet

[xix] https://sparkmailapp.com/es - Aplicación de correo electrónico para dispositivos Windows, iOS, macOS y Android de Readdle.

[xx] https://twitter.com/ - Empresa de comunicaciones cuyo principal producto es la red social Twitter.

[xxi] https://www.viber.com/es/ - Aplicación de comunicación que permite a los usuarios realizar llamadas gratuitas de teléfono y enviar mensajes de texto.

[xxii] https://cloud.google.com/ - Proveedor de servicios cloud de Google.
[xxiii] https://azure.microsoft.com/ - Proveedor de servicios cloud de Microsoft.
[xxiv] https://aws.amazon.com/ - Proveedor de servicios cloud de Amazon.