UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

GRADO EN INGENIERÍA INFORMÁTICA EN INGENIERÍA DE SOFTWARE

TRABAJO FIN DE GRADO

Bot conversacional de la Universidad de Extremadura (NexoBot)

UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

GRADO EN INGENIERÍA INFORMÁTICA EN INGENIERÍA DE SOFTWARE

TRABAJO FIN DE GRADO

Bot conversacional de la Universidad de Extremadura (NexoBot)

Autor: Vicente González Ortega

Tutor: José María Conejero Manzano

Co-Tutor: Juan Carlos Preciado Rodríguez

Resumen

En este documento se detalla todo el proceso de análisis, diseño e implementación del trabajo de fin de grado NexoBot.

NexoBot consiste en un asistente virtual dedicado principalmente a futuros alumnos o alumnos de la Universidad de Extremadura, utilizando datos recogidos de la Universidad de Extremadura, aunque algunas de las funcionalidades se han centrado principalmente en la Escuela Politécnica, la cual provee información interesante del alumno de forma rápida y eficaz.

En el sistema implementado de NexoBot existen una serie de microservicios que recogen la información de diferente forma, mediante web scraping[1] de la página de la Universidad de Extremadura y mediante consultas al portal de OpenData. Integra un servicio que permite recoger información que no ha encontrado en los otros servicios usando un motor de búsqueda como es Bing. Además, tiene incorporado dos servicios que utilizan la Inteligencia Artificial (IA) (Rouse, 2018) para ampliar el rango de respuestas del bot, y guiar al alumno de una forma satisfactoria.

El bot puede ser utilizado en plataformas como Telegram, Slack, Facebook, Skype, o incluso desde Alexa, perdiendo en este último algunas funcionalidades debido a su limitación utilizando únicamente comandos de voz.

[1] técnica utilizada a través de aplicaciones o librerías de software para extraer información de sitios web.

Abstract

This document details the whole process of analysis, design and implementation of the NexoBot.

The purpose of NexoBot was implementing a virtual assistant dedicated mainly to students or future students of the University of Extremadura, using data collected mainly from the University of Extremadura, although some of the functionalities have focused mainly on the Polytechnic School, which provides interesting information for the students quickly and efficiently.

In the system implemented by NexoBot there are a series of microservices that collect information in different ways, by scraping the page of the University of Extremadura, through queries to the OpenData portal, by integrating a service that allows you to collect information that has not been found in other services using a search engine such as Bing, and by integrating two services that use IA to expand the range of responses of the bot, and guide the student in a more effective way.

The bot can be used in platforms such as Telegram, Slack, Facebook, Skype, or even from Alexa, losing in the latter some functionalities due to it limitatuions base on using voice commands.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El principal propósito de este capítulo es que el lector comprenda los principales objetivos y alcance de este proyecto. Se iniciará hablando de la principal motivación por la que se ha decidido realizar este proyecto, también de los objetivos propuestos, y el alcance final en el caso de lograr todos los objetivos propuestos.

La universidad de Extremadura tiene una gran fuente de información muy relevante para todo tipo de públicos, estudiantes, profesores, etc. La consulta de dicha información se puede realizar de diferentes maneras, como por ejemplo ir físicamente a secretaría y realizar una consulta a los empleados allí presentes, consultar tablones de anuncios de las aulas, o acceder directamente a la página web disponible de cada centro.

La manera para consultar la información más utilizada es la online, debido a la gran comodidad de poder consultar la información de forma remota. En esta última parte es donde entra en juego el proyecto, con la principal idea de mejorar el sistema de transmisión de información, a través de un bot conversacional, sin necesidad de acceder a un sitio web en particular para recibir la información.

La idea principal, es poder ofrecer información solicitada por el usuario en tiempo real, gracias al uso de un bot. En muchos campos lleva años utilizándose, como son los videojuegos, el malware, etc. Pero todos persiguen un objetivo principal, el de imitar el comportamiento humano, y es lo que voy a intentar en este proyecto, realizarlo de

la forma más realista posible.

1.1. Motivación y Objetivos

El objetivo principal del proyecto consiste en reducir y hacer mas eficiente el tiempo de búsqueda que un alumno de la universidad tarda en encontrar una página dentro de la web de la Universidad de Extremadura, ya que en algunas ocasiones si la búsqueda se está realizando por primera vez puede tardar en encontrar exactamente lo que está buscando.

La solución es el uso de un asistente virtual (ChatBot) en lenguaje natural, de manera que el usuario pueda preguntar con oraciones completas, y el asistente pueda responder con la información que está buscando, o incluso con el enlace donde está la información, pero camuflados en lenguaje natural, para dar una breve descripción de donde le conducirá el enlace, siempre que se refiera a preguntas relacionadas con la universidad.

Al ser el objetivo principal muy genérico, se ha decidido dividir en los siguientes subobjetivos:

- Crear un chatbot conversacional sobre preguntas frecuentes de la Universidad de Extremadura utilizando la IA para ofrecer una respuesta correcta a preguntas formuladas de diferente manera pero que buscan la misma respuesta.

- El chat conversacional debe poder integrarse de forma simultánea con varias plataformas de mensajería instantánea, e incluso de comandos de voz.

- El chatbot debe responder preguntas basadas en el contexto de una conversación, convirtiendo el chat en una conversación realista.

- Los datos que se recolectarán serán recogidos de diferentes fuentes de información.

1.2. Alcance

El alcance inicial del proyecto es conseguir un asistente virtual que funcione como una aplicación autónoma, es decir que las respuestas que realice el chatbot estén completamente actualizadas y que solo necesite atención para solucionar problemas o añadir nuevas funcionalidades.

Para la realización de esta aplicación se ha pensado integrar el chatbot con alguna de las plataformas de Telegram o Skype, e incluso valorar la posibilidad de integrarlo como un Skill de Alexa, se ha descartado la idea de integrarse en los sistemas infomáticos de la Universidad de Extremadura por cuestiones de tiempo y de disponibilidad de acceso a dichas herramientas.

Capítulo 2

Estado del Arte

En los últimos años, el uso de la tecnología en tareas cotidianas está creciendo de forma exponencial, pudiendo acceder desde cualquier sitio a casi todos los datos que están disponibles en la red; noticias, emails, datos bancarios, trámites sanitarios y muchas cosas más que ahorran mucho tiempo a las personas.

En la actualidad, muchas de las grandes empresas tecnológicas se encuentran en una lucha constante por conseguir la mayor cantidad de información de usuarios, empresas, instituciones, etc. Pero la mayor lucha está en el dominio de las herramientas de mensajería instantánea, en las cuales se están desarrollando chatbots que consiguen ofrecer no solo un simple chat, si no que son capaces de ofrecer funcionalidades extras mucho más atractivas para el usuario.

Telegram es una de las plataformas de mensajería instantánea que mas ha avanzado en el desarrollo de bots, a finales de 2014 fue una de las primeras en integrar una API de desarrollo de bots bastante atractiva, y mas adelante empezó a introducir mejores funcionalidades como juegos o blogs, algo que otras plataformas de las mismas características aún no han integrado, como Whatsapp o Snapchat.

2.1. Bots

2.1.1. ¿Que son?

Los bots son software que ponen en marcha tareas repetitivas de forma autónoma intentando simular el comportamiento de un humano.

La definición anterior que ofrece lo que es un bot puede ser algo insuficiente, esto es debido a que un bot a día de hoy puede ser utilizado o aplicado en casi todos los ámbitos de la informática. Muchos de estos ámbitos pueden ir desde personajes en videojuegos con los que interactuamos, o recolectores de información de otras APIs (crawlers[1]). También pueden ser utilizados para prácticas delictivas como por ejemplo los Spam bots o para ataques de DNS en los cuales un gran número de bots realiza peticiones contra un servidor hasta que el servidor es colapsado.

Por lo que, como se puede apreciar, y como hemos mencionado anteriormente, el abanico es demasiado grande. Este trabajo se va a especializar en los chatbots, en el que su principal objetivo es mantener una conversación fluida entre máquina y ser humano.

El principal enfoque en este tipo de bots se debe a que en la actualidad, la mensajería instantánea, es el servicio de comunicación principal que más utilizan las personas.

Hablando de Bots tampoco se pueden olvidar, los asistentes virtuales, y en particular los de los sistemas operativos de las grandes empresas informáticas como son Siri y Cortana. Estos asistentes virtuales están cogiendo un gran peso ya que está demostrado que facilitan el día a día de los usuarios que están utilizando su software.

2.1.2. Historia

Todo viene del concepto de Robot, por lo que podríamos decir que los chatbots son el software que le dan vida a estos. El matemático Alan Turing (Martín, 2017) en 1950, considerado como unos de los padres de la ciencia de la computación y precursor de

[1]también conocido como rastreador, es un programa que analiza los documentos de los sitios web.

la informática moderna, fue el primero en proponer una teoría, en la cual decía que podía dar a las máquinas comportamiento inteligente, adelantando ya la idea de que éstas llegarían a pensar.

Este trabajo de Turing, inspiró a muchísimos informáticos y unos años mas tardes apareció el primer chatbot llamado ELIZA (YÚBAL, 2017) desarrollado por Joseph Weizenbaum en el año 1966. El objetivo de ELIZA era hacerse pasar por un psicólogo y mantener una conversación inteligente con los pacientes, los cuales llegaron a creer que realmente hablaban con un ser humano. Este chatbot era capaz de procesar las palabras clave, teniendo siempre un contexto mínimo y era capaz de generar respuestas adecuadas e incluso reaccionar ante la ausencia de palabras críticas o claves.

Pero hasta el año 2000, no llegó el bot conversacional que plantó las bases de lo que a día de hoy tenemos en los asistentes virtuales de casi todos los dispositivos electrónicos, este chatbot fue SmarterChild el cual a diferencia de los anteriores podía prestar ayuda al usuario facilitando información acerca del tiempo, de horarios de cine o incluso resultados deportivos, siendo compatible con servicios de mensajería instantánea como AIM, MSN Messenger, AOL Instant Messenger e ICQ.

2.1.3. Tipos de bots

Las funcionalidades que pueden realizar los bots, pueden ir desde la automatización de tareas repetitivas, hasta otras más complejas como pueden ser la toma de decisiones pasando por algún filtro o parámetro en el código. Por tanto, los bots pueden usarse tanto para hacer reservas en un restaurante, como para obtener toda la información de una página web, o entablar conversación con determinados usuarios ofreciéndoles algún servicio, que resuelve sus dudas.

Partiendo de esta base, vamos a categorizar los bots según las tareas que son capaces de realizar:

- Informativos: son encargados de recolectar la información que se van publicando en los diferentes canales de información. Existen muchas empresas que están apostando por ello, como son el caso de TechCrunch y la CNN, que disponen de

un bot que informa de los canales de noticias disponibles y donde los usuarios puede gestionar de cuáles recibir notificaciones.

- Indexadores de webs o Crawlers: están destinados a recopilar información de sitios webs o APIs. Este tipo de bot también puede ser utilizado para realizar otras tareas como pueden ser:
  - Detección y alerta de cambios de precios en sitios web (ecommerce).
  - Monitorización del estado de un sitio web (tiempo de actividad, errores y problemas de rendimiento).
- Chatbots: están diseñados para mantener una conversación con humanos a través de un sistema de diálogos. Los más sofisticados utilizan sistemas de procesamiento de lenguaje natural (PLN) y los más simples construyen sus respuestas en base a unas reglas. Se les suele dotar de una ”personalidad”similar a la de un humano.
- Gamebots: es uno de los recursos más utilizados en los videojuegos, ya que intentan simular a una persona realizando acciones con ese personaje. Dentro de este tipo de bot hay algunas variantes, ya que algunos usan IA, y dependiendo del tipo de juego, el bot sera de una manera u otra. En los juegos CRPG o Shooter el bot utiliza una IA,y para vencerle será necesario realizar una estrategia específica.
- Spam Bots: este tipo de bots es capaz de crear cuentas de correos, y hacer envíos masivos de correos electrónicos, con el fin de obtener beneficio con el envío de publicidad. También son utilizados para estafar a los usuarios con contenido fraudulento. Actualmente es fácil detectar este tipo de mails. Existen algunas restricciones para evitar este tipo de práctica como es la de la inserción del CAPTCHA[2] en muchos de los formularios de registro, aunque a día de

[2] siglas de Completely Automated Public Turing test to tell Computers and Humans Apart, se trata de una prueba para determinar cuando el usuario es humano o no

hoy ya existen bots que son capaces de vulnerar esas restricciones a través de Inteligencia Artificial.

- Zombie Bots: Este tipo de bot está destinado a ataques informáticos, en el que el zombie bot es un ordenador que ha sido comprometido de modo que una persona externa tiene el control de la máquina, que junto a miles de otros equipos forman una red bot. Están destinados a coordinar grandes ataques que ejecutan comandos enviados por el administrador de esa red, que normalmente son ataques de denegación de servicios. No son fáciles de detectar y la mayoría de equipos infectados por bots ni lo saben.

- Bot Imitadores : diseñados principalmente para imitar el comportamiento de una persona, y la mayoría se han usado para difundir propaganda, como por ejemplo el envío mensajes de una campaña política a través de twitter.

- Bot transancionales: la función de estos bots es actuar como agentes en nombre de los seres humanos e interactuar con sistemas externos para llevar a cabo una transacción específica, moviendo datos de una plataforma a otra. Este tipo de bots puede interactuar con cualquier API.

  Un ejemplo de este tipo de bot es X.ai, cuyo principal objetivo es encontrar reuniones entre equipos a través del envío de mails entre distintas personas.

- File-sharing Bots: son bots destinados a el uso de servicios P2P (torrent, ftp, descargas directas, etc), utilizan un término extraído de la consulta de un usuario (título de libro, de película, etc) y responden con una afirmación en caso de que tengan ese libro disponible para descargarlo. En el caso que quieran descargarlo el bot proporciona el enlace y genera el fichero, en este último paso, es donde pueden entrar los fines maliciosos, ya que puede inyectar contenido en la descarga y la víctima podría abrir el fichero y sin saberlo infectar su máquina.

2.1.4. Revolución de bots

Muchas de las grandes empresas tecnológicas no dejan de apostar por la Inteligencia Artificial, y esto es debido al gran crecimiento que ha experimentado este área en los últimos años.

Uno de los campos que ha experimentado un gran avance es el del procesamiento del lenguaje natural (PLN) (Wikipedia, 2018), campo que combina tecnologías de aprendizaje automático Machine Learning (ML) y de lingüística aplicada, teniendo como objetivo facilitar el procesamiento del lenguaje humano así como su comprensión.

Otro de los campos que tenemos que hablar es del Machine Learning (aprendizaje automático), cuya tecnología intenta que las personas y las máquinas trabajen de la mano, para que estas últimas sean capaces de aprender de la misma forma que un humano lo haría. Esto comenzó con filtros en emails para detectar emails fraudulentos, pero actualmente son capaces de hacer cosas aún mas complejas que cualquier humano podría hacer en un intervalo corto de tiempo, como puede ser, la detección de un cáncer, predicciones de trafico en intersecciones muy transitadas, o incluso generar mapas de zonas para grandes proyectos de construcción en tiempo real.

Estos avances, son un gran punto a favor para que se produzca este crecimiento de bots, porque esto, junto con el gran uso de las herramientas de mensajería instantánea son la combinación perfecta para cualquier sistema de comunicación.

A día de hoy, en internet podemos recibir y transmitir cualquier tipo de información, y tenemos disponible cualquier tipo de contenido: blogs, periódicos, redes sociales, etc, pero la conversación en redes sociales o vía mensajería instántanea es la que mayor tiempo nos ocupa. Y por tanto, el papel de los bots y, más en concreto los bots conversacionales, es vital en esta etapa.

En un futuro muchos de estos bots sustituirán a muchas herramientas que ofrecen servicios, ya que a veces es bastante engorroso, encontrar la solución a tu problema en el típico FAQ de las páginas web convencionales. ¿Por qué no tener un bot con el cual interactuar para que te resuelva los problemas de forma eficiente?

2.2. Chatbots

En este apartado nos centramos en el tipo de bot que se ha implementado en este proyecto, los chatbots.

2.2.1. Introducción

Un ChatBot es un programa informático desarrollado con la finalidad de comunicarse con seres humanos o con otras máquinas.

Su intención es simular que sea una persona quien está detrás de esa conversación. Sirve tanto para pedir u ofrecer información, como para llevar a cabo acciones. La mayoría de ellos, utilizan tecnologías basadas en inteligencia artificial, tanto para el uso de diferentes sistemas de lenguaje natural, como para las diferentes técnicas de aprendizaje automático.

La mayoría de los sectores se están apoyando cada vez más en estos bots conversacionales, puesto que dan un valor añadido a sus plataformas, por ejemplo, los clientes de hoteles como Marriot pueden realizar reservas directamente en más de 4.700 hoteles (Hervías, 2017), acceder a recomendaciones personalizadas en sus destinos o contactar con el Customer Engagement Center a través de Facebook Messenger, Slack o Google Assistant.

2.2.2. Historia

A lo largo de los años, los bots conversacionales han ido evolucionando a medida que se producían algunos avances tecnológicos.

En un principio, los primeros bots conversacionales, usaban un campo de la inteligencia artificial, conocido como procesamiento del lenguaje natural (PLN). Este campo se ocupa principalmente de mejorar la forma en que las máquinas se comunican con las personas a través del lenguaje natural.

Durante unos años, los bots conversacionales no tenían tanto peso, pero fue a partir de los años noventa, cuando se empezaron a combinar diferentes campos de la

inteligencia artificial, con el fin de mejorar la conversación del bot con seres humanos. Así surgieron propuestas como Clippy o ChatterBot.

Pero a día de hoy, solo tenemos dos formas de desarrollo de un bot conversacional, una simple y otra inteligente. En la opción simple, el bot presenta una serie de opciones y lleva de forma guiada al usuario por el contexto para el que ha sido creado. El usuario no puede realizar acciones o hablar de cosas fuera de ese contexto, ya que el bot no sabría por dónde ir, y de ahí su limitación, el cual depende plenamente de las reglas o acciones que el desarrollador haya preparado previamente.

La otra opción, es bastante más compleja. Se necesita una manipulación previa de los datos de entrada, y dotar al chatbot de comprensión aplicando alguna regla de procesamiento del lenguaje natural para permitir establecer un dialogo a partir de un lenguaje natural. Si a esto último le dotamos de herramientas de Machine Learning o Deep Learning, el bot conversacional podrá aprender y adaptarse de forma autónoma respondiendo a diferentes preguntas del usuario, aunque éstas estén formuladas de diferente forma, ya que para este tipo de bot no es necesario seguir una regla estricta como lo hacía el simple.

Un ejemplo de chatbot inteligente, y desarrollado en la Universidad de Murcia, es LOLA (1millionbot, 2017), cuya finalidad es guiar y ayudar a los futuros estudiantes de la UMU y de la Universidad Politécnica de Cartagena con todos sus trámites de preinscripción y matrícula. La tecnología que ha utilizado es TensorFlow/DialogFlow de la IA de Google la cual ha dado unos resultados que representan un avance muy sobresaliente en el potencial de los chatbots, gracias a sus técnicas de entrenamiento con un lenguaje natural general real. Este Trabajo de Fin de Grado, a diferencia del LOLA, abarcará temáticas muy diversas, evitando centrarse en un único tema como hace LOLA con la preinscripción y matricula. Además, será capaz de entender la temática principal del hilo de la conversación utilizando otras tecnologías de IA.

Viendo la importancia de la Inteligencia Artificial, en el mundo de los ChatBots, en los siguientes puntos se dará una explicación en la que entraremos más en detalle acerca de cada campo.

2.2.3. Inteligencia Artificial

La inteligencia Artificial (IA) se puede definir como la inteligencia que hace que las maquinas puedan imitar las funciones ”cognitivas” que las personas asocian con otras mentes humanas, como por ejemplo, percibir, razonar, aprender y resolver problemas.

Actualmente la IA se puede dividir en dos categorías:

- Inteligencia Artificial Fuerte (Wikipedia, 2017c): este tipo de inteligencia es la que intenta igualar o exceder la inteligencia promedio humano. Actualmente estamos lejos de ella, ya que es muy complicado conseguir una capacidad cognitiva superior a la de un humano, pero se van realizando avances, como por ejemplo el de la Universidad Tsinghua de Pekín que ha conseguido realizar un chip llamado Tianjic (Sarazen, 2019) que ha conseguido resolver el gran problema que tenían la mayoría de sistemas, la imposibilidad de aprender sobre la marcha. Un ejemplo claro, es cuando una máquina derrota a una persona en un juego, lo ha conseguido porque previamente ha jugado un millón de veces y es capaz de prever movimientos o acciones en base a ello, no porque aprenda de su oponente. El gran paso ha sido por el enfoque basado en neurociencias (chips neuromórficos), compuesto por cientos de miles de neuronas artificiales que operan intercambiando pequeñas señales eléctricas, como lo hace nuestro cerebro, gracias a esto es capaz de realizar una conducción completamente autónoma, mantener el equilibrio, esquivar obstáculos, etc, y todo aprendiendo sobre la marcha .

- Inteligencia Artificial Débil (Wikipedia, 2017b): También conocida como IA estrecha, es la que se usa en la mayoría de sistemas de la actualidad y en la que se basan la mayoría de chatbos del mercado, un ejemplo es SIRI, la cual opera dentro de un rango limitado que ha sido definido previamente, por lo que no hay ninguna inteligencia genuina, no tiene conciencia, y menos aún vida, aunque aparentemente esté bastante conseguida.

Para mejorar la Inteligencia Artificial, dotando las máquinas de inteligencia, los

grandes investigadores de este campo se han inclinado en dividir el principal ”proceso” en varios ”sub-procesos”.

- Aprendizaje: En este proceso entran las técnicas de aprendizaje automático que permiten mejorar a través de la experiencia. Existen tres tipos principales:
  - Aprendizaje Supervisado
  - Aprendizaje No Supervisado
  - Reforzamiento de aprendizaje
- Procesamiento del Lenguaje Natural: Se ocupa de la formulación e investigación de mecanismos eficaces para dotar a las máquinas de la capacidad de comunicación con humanos a través del lenguaje natural, como el español, el chino, etc. Este apartado es uno de los mas importantes para el desarrollo de los bots conversacionales por lo que se va a explicar mas detalladamente a continuación.

Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural (Mijangos, 2019b) es una rama del conocimiento de la inteligencia artificial que, pretende conseguir que una máquina comprenda lo que expresa una persona mediante el uso de una lengua natural. Esta ciencia siempre intenta buscar los mecanismos más eficaces para realizar dicha comunicación y evidentemente siempre que sean computacionalmente posibles.

Nace durante la década de los sesenta, donde todos sus métodos tuvieron una gran aceptación, pero más adelante empezaron a aparecer muchos problemas cuando se llevaron a la práctica, en entornos no controlados y con vocabularios genéricos. Algunos de los principales problemas fueron la polisemia y la sinonimia.

Muchas de las reglas utilizadas se realizaban de forma manual, y esto hacía difícil su implementación cuando aparecían entornos no controlados. Pero no es hasta finales de los años ochenta cuando se empezó a utilizar el aprendizaje automático para procesar el lenguaje. Además, con el avance de la tecnología los sistemas empezaron

a tener mas potencia computacional para tratar las grandes cantidades de datos que se necesitan.

Se empezaron a introducir algoritmos de aprendizaje automático como árboles de decisión, reglas de transformación manuales, autómatas, etc. Pero seguían teniendo el mismo problema y en entornos no controlados no funcionaban de la forma deseada.

El gran avance llego cuando se introdujo en el etiquetado gramatical (Wikipedia, 2017a) la aplicación de los Modelos Ocultos de Markov (HMM) (Mijangos, 2019a), cuya técnica fue desarrollada en gran parte a Baum and Petrie. Consistía en construir un modelo de lenguaje estadístico, que se usa para obtener a partir de una oración o frase de entrada la secuencia de etiquetados léxicos que tienen mayor probabilidad. Un ejemplo sencillo en el que hemos etiquetado una palabra como artículo, la siguiente palabra será un nombre con un 40 % de probabilidad, un adjetivo con otro 40 % y un numero el 20 % restante. Sabiendo esta información en una frase "El gato" gato es más probable que sea un nombre que un verbo.

Gracias a esto, lo que se está haciendo ahora es centrarse en los modelos estadísticos para que la toma de decisiones se base en los pesos asignados a cada dato de entrada. Está técnica se mejoró con el desarrollo de Steve DeRose y Ken Church con algoritmos de programación dinámica muy parecidos al algoritmo de Viterbi (Sipán, 2008) que reducían drásticamente el tiempo de etiquetado considerando todos los casos existentes que se pudieran dar, alcanzando una tasa de acierto del 95 %.

A esto hay que sumarle los algoritmos de aprendizaje que mencionamos en el punto anterior, los algoritmos supervisados y no supervisados. La forma de clasificar es bastante mas compleja en los algoritmos no supervisados, pero para este tipo de algoritmo se tiene una gran cantidad de datos, por lo que si se encuentra el patrón correcto, puede llegar a ser más efectiva que uno supervisado. La técnica que más se está usando en la actualidad es el Deep Learning basado en redes neuronales, en las que existen grandes cantidades de datos estructurados en un conjunto agrupados por unas determinadas características. Este tipo de técnica es capaz de identificar algunos escenarios mucho mejor que un humano, como por ejemplo un indicador de cáncer en

sangre, o detectar un tumor en una resonancia magnética.

2.2.4. Plataformas del mundo de los ChatBots

Educación

Los chatbots cada vez están siendo mas importantes en el mundo de la educación. Diferentes estudios realizados han concluido que la introducción de chatbots mejora la participación de los estudiantes, optimiza la administración de los centros, y aumentan la productividad (Nieves, 2018) .

El papel que puede tener un bots conversacional en el sector educativo, puede ser muy amplio, a continuación, se enumerarán algunos de los papeles más relevantes (Gende, 2019):

- Becas, Ayudas y Subvenciones, el bot podría sugerir las becas que puede solicitar el alumno, y dar algo de información de cómo realizar la inscripción.
- Información Académica, enviar mensajes de notas de proyectos, ejercicios, o exámenes, que estén disponibles en el campus virtual, de forma automática cuando un profesor ponga la nota.
- FAQ (preguntas frecuentes), el bot resolverá preguntas que pueden ser repetitivas pero que si se tienen localizadas puede ser muy útil, para evitar que un profesor tenga que repetir la misma respuesta muchas veces para diferentes alumnos, como por ejemplo el sistema de puntuación de la asignatura, cuando la información está disponible en la ficha 12A.
- Comunicación de los padres con los centros y profesores, gracias al chatbot podremos informarnos de las actividades realizadas, fechas y tareas que tienen pendiente, así como excursiones o vacaciones.
- Gestión de tutorías, el bot podría enviar un mensaje al profesor para preguntarle si tiene disponible la cita que le ha pedido el alumnos, si el profesor dice que sí,

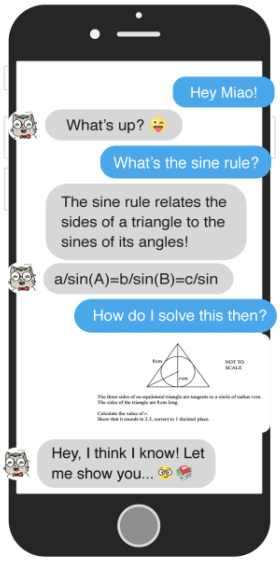


Figura 2.1: Miao

el bot contesta con la confirmación de la tutoría, en caso contrario, podría unir a ambos a la conversación para que concreten mejor la tutoría.

- Información de fechas importantes del alumno, como informar la fecha de un examen, o de un trabajo final que está almacenado en el campus.

A continuación, pondré algunos ejemplos de chatbots relevantes en el mundo educativo:

Miao (Academy, 2019), realizado en Singapur, el cual simula ser un estudiante de 16 años que estudia en Singapur, y cuya asignatura favorita es matemáticas. Ver en figura 2.1.

El objetivo principal del proyecto, es intentar ayudar a aquellas personas que tienen dificultades durante la clase para preguntar alguna duda al profesor, y al terminar, tampoco son capaces de decirle a sus compañeros que le expliquen el temario dado. Este bot, puede conversar contigo y resolverte las dudas como si de otro alumno se tratara.

Otro proyecto interesante es Duolingo (Duolingo, 2019), basado en la enseñanza de idiomas utilizando su aplicación de aprendizaje. Tiene una finalidad muy similar al

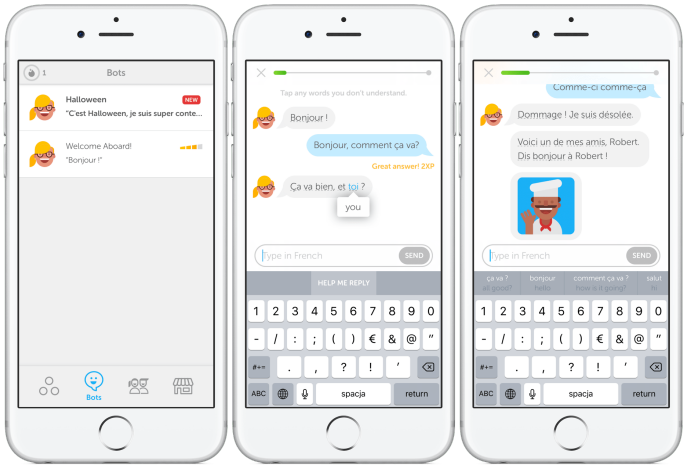


Figura 2.2: Duolingo

anterior proyecto, después de varios estudios realizados, se dieron cuenta que existía un gran problema al poner a dos personas físicas conectadas entre sí a aprender idiomas, ya que en muchos casos una persona tenía vergüenza y no podía continuar con el aprendizaje, y dejaba de utilizar la web. Para evitar este problema se desarrolló un bot para que el usuario pudiera entablar una conversación en el idioma que quiera aprender y, durante la conversación el bot pueda ir trazando patrones con los datos que va obteniendo de la conversación, para guiar y adaptar la conversación a tu forma de comunicarte. Ver en figura 2.2.

Alexa

Alexa es el asistente virtual controlado por voz que ha sido creado por Amazon, y lanzado en noviembre de 2014.

Su funcionamiento es muy parecido al de otros asistentes como Google Assistant, Siri y Cortana. Para arrancar es necesario decir su nombre, momento en el que el altavoz o dispositivo en el que esté integrado se pondrá a escuchar. Lo siguiente que se puede hacer es decirle un comando con tu voz, y el asistente reconocerá lo que le preguntas y te dirá una respuesta.

Las funciones de Alexa dependen de dos elementos claves. Por una parte están los comandos de voz que vienen integrados por defecto, y con los que puedes realizarle una

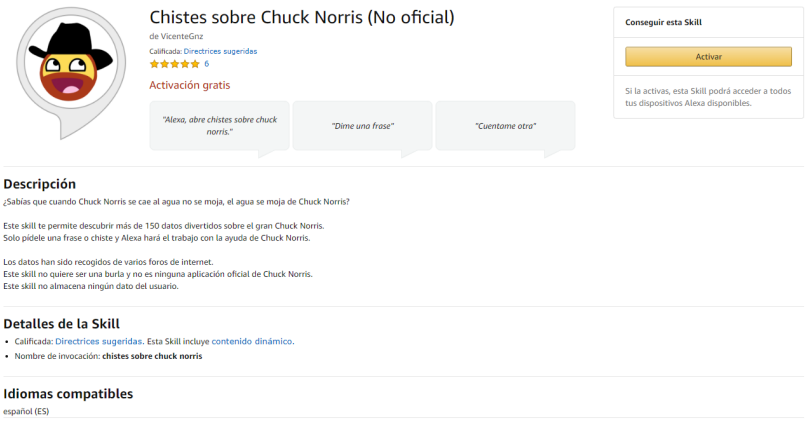


Figura 2.3: Chuck Norris Skill

gran variedad de peticiones, como cambio de divisas, consultar el tiempo, etc. Y luego están las skills, que son complementos realizado por desarrolladores tanto internos como externos a Amazon, que tienen funcionalidades extras y que se pueden instalar en el dispositivo. Existe una tienda oficial de Amazon Alexa dónde se encuentran todas las skills disponibles.

Los skills que tiene la tienda, son de temas variados, y cualquiera puede crear uno suyo, yo mismo durante la investigación para la realización del proyecto, realicé una, que puse y está actualmente en producción, dónde se han recibido algunas reseñas de usuarios que la ha utilizado, y existen métricas en la plataforma de desarrolladores de Alexa dónde se puede apreciar su uso. Esta skill se puede ver en la figura 2.3

Amazon ha introducido además *Alexa Skill Kit* (ASK), que es un conjunto de herramientas, documentación, fragmentos de código y APIs que pueden ser utilizadas para crear Skills de Alexa de forma rápida y sencilla .

Para la creación de una skill con una mínima funcionalidad, se requiere de conocimientos informáticos, por lo que vamos a tener en cuenta que el desarrollo de la funcionalidad ya está terminado, y solo necesitamos integrarlo con la plataforma ASK. Los pasos serían los siguientes:

- Acceder a https://developer.amazon.com/alexa/console/ask con tu usuario y contraseña, y darle a la opción Create Skill donde se podrá elegir el tipo de

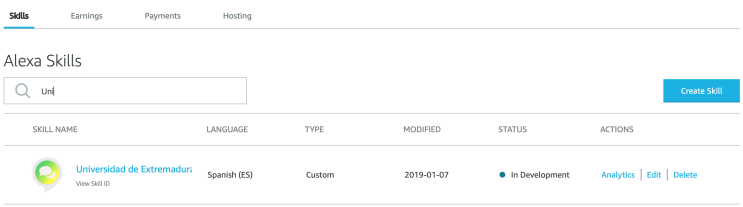


Figura 2.4: Skill de Alexa

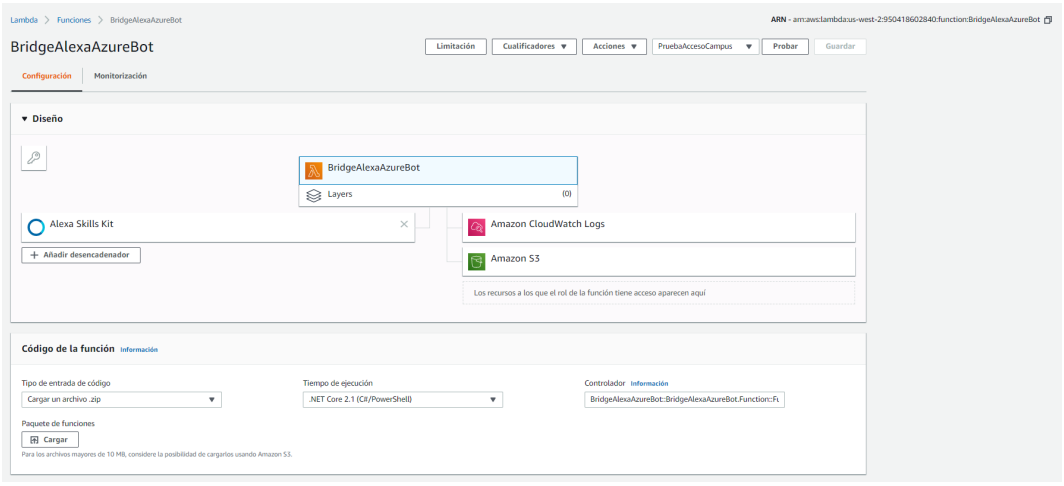


Figura 2.5: Amazon Web Services

Skill (Custom, Flash Briefing y Smart Home).

- Será necesario darle un nombre invocación, que será utilizado cuando Alexa reciba un comando de voz para abrir la aplicación.

- Previamente debemos tener subido nuestro código en un servidor, yo siempre utilizo Amazon Web Services, y lo que hay que subir es un archivo comprimido del proyecto e indicar una serie de características, si todo ha ido bien, se debe generar un código ARN, que esta en la parte superior derecha, ese sera el endpoint que necesitamos indicarle a nuestra skill, para que ejecute nuestro código. Ver figura 2.5.

- Una vez conocido el endopoint (ARN) será necesario configurar el endpoint donde estará nuestro código alojado de la skill desde el área ASK.

- Cuando se han realizado todos los pasos anteriores desde el área de Amazon Skill

Kit, se pueden realizar muchas cosas, como pruebas, asignación de permisos, validaciones con el objetivo de llevar tu skill al público, etc.

Alexa tiene un papel importante en el proyecto, por lo que más adelante, durante la explicación técnica, se entrará más en detalle de las pinceladas que se han dado en este punto acerca del abanico de posibilidades que nos ofrece.

Facebook Messenger

Facebook Messenger (Facebook, 2019) es una aplicación de mensajería instantánea que nació originalmente en 2008 como Facebook Chat, pero con el paso de los años, se lanzo un sitio web dedicado y las funcionalidades de mensajería se separaron de la aplicación principal de Facebook, obligando a todos sus usuarios a descargar una aplicación secundaría de mensajería si se quería utilizar en dispositivos móviles.

Durante 2015, Facebook permitió que las empresas y los usuarios comenzaran a interactuar entre sí con funciones como seguimiento de compras, recepción de notificaciones, o incluso para servicios de atención al cliente, etc. En 2016 introdujo una API para la construcción de chatbots en Facebook Messenger, para que los desarrolladores pudieran construir sus propios chatbots e integrarlos cómodamente. Los primeros chatbots utilizados en Facebook fueron principalmente de noticias.

Durante 2017 Facebook siguió apostando por esta tecnología y lanzó el asistente virtual M para sus usuarios (solo está disponible en EE.UU). Este asistente es capaz de escanear en los chats las palabras claves y a partir de ella sugerir acciones determinadas. Además, incorporó chatbots en los grupos de Messenger como ”Chat Extensions” donde añadió una nueva pestaña ”Discovery” cuyo principal objetivo es la mejora de búsqueda de bots disponibles, para lograr esto habilitó códigos QR especiales que, cuando se escanean, llevan al usuario a un bot específico.

A diferencia que para Alexa, existen actualmente herramientas que permiten crear chatbots totalmente compatibles con Facebook sin necesidad de conocimientos informáticos, como son ChatFuel (Chatfuel, 2019) que es gratuita o ManyChat(ManyChat, 2019) que es de pago.

Los pasos a realizar son los siguientes:

- Darte de alta en una herramienta para creación del BOT como Chatfuel o Manychat.
- Crear fan page: https://www.facebook.com/pages/create, donde nos ofrecerán varios tipos de páginas (Organización, Empresa, Entretenimiento, etc).
- Crear cuenta en facebook for developers en caso de no tenerla y dar de alta un producto de tipo Messenger.
- Confirmar el acceso de esta nuevo producto con tu página de fans. Para eso es necesario introducir el identificador de nuestra fan page creada en el producto de tipo Messenger.
- Por último quedaría configurar los WebHooks, donde debe ir la URL donde está alojado nuestro código, esto puede depender de donde se haya realizado el chatbot, además será necesario añadir los permisos y un token para garantizar la identidad.

En un estudio realizado durante este año (Tomás, 2019), algunos de los bots conversacionales que han tenido mayor éxito han sido:

- KAYAK: Ayuda a buscar información sobre vuelos, hoteles, coches de alquiler y actividades en cientos de webs de viajes. Además es capaz de actualizar y gestionar datos de tus viajes.
- 1-800-FLOWERS: Te permite encargar flores o contactar con el servicio de atención al cliente sin abandonar Messenger.
- UBER: Da la posibilidad de solicitar un viaje desde cualquier conversación simplemente haciendo click en un icono con una imagen de un coche que aparece en la conversación. Los usuarios también pueden iniciar la conversación con Uber directamente para solicitar un viaje, o consultar disponibilidad.

- KLM AIRLINES: Puedes recibir tu confirmación de reserva, notificaciones, tarjetas de embarque y actualizaciones sobre el estado de tu vuelo a través de una conversación fluida con el bot.
- CNN: un chatbot que nos informa de todas las noticias que nos interesen.

Como se puede apreciar la creación de un bot conversacional en Facebook es algo muy sencillo y se puede hacer incluso sin conocimientos informáticos, y es por esto, que es una de las servicios de mensajería instantánea donde existe mayor uso de bots conversacionales, junto a Telegram, de la que vamos a hablar a continuación.

Telegram

Telegram Messenger (Cabello, 2016), es un aplicación de mensajería instantánea desarrollada desde el año 2013 por los hermanos Nikolái y Pável Dúrov. Entre sus funcionalidades principales destacan el envío de varios archivos de gran tamaño, gestión de contactos, encuestas, canales de difusión, y comunicación en masa. El servicio lo administra una organización auto financiada cuya sede principal opera en Dubái, Emiratos Árabes.

Está aplicación tiene algunas diferencias notables con las de su competencia, entre ellas destacan, que su código fuente y sus APIs son publicas (excepto el lado del servidor) para que los desarrolladores puedan trabajar en la comunidad libremente, y el uso de chat secretos donde los mensajes entre emisor y receptor están cifrados con un algoritmo exclusivo, y sobre todo la más relevante para nosotros, la plataforma de bots.

Desde mediados de 2015, Telegram cuenta con esta plataforma de Bots, y a día de hoy se pueden diferenciar en dos tipos de bots, los originales y los denominados “Inline Bots”, la diferencia entre ellos es muy simple. Para el uso de los originales es necesario abrir una ventana de chat e incluirlo en el grupo o conversación que estamos usando, y lo llamaremos añadiendo una ”@” al principio, a diferencia los ”Inline Bots”, no necesitan incluirse en ninguna conversación o grupo para poder usarlo, se pueden usar en cualquier momento.

Para los bots existen algunas características que no están disponibles, algunas de ellas son:

- No pueden iniciar conversaciones con usuarios, solo es el usuario el encargado de empezar la conversación.
- Su nombre siempre debe acabar en ”bot” como por ejemplo @NexoBot @Educa-Bot.
- No tienen estado ni última visualización.
- El almacenamiento de los mensajes en la nube es limitado, debido a la gran cantidad de datos que pueden generarse, se limpian cada poco tiempo.

Para la creación de un chatbot en Telegram (*Telegram APIs*) se necesitará iniciar la conversación con otro chatbot el cual nos guiará en todo el proceso de creación, este chatbot se llama @BotFather (ver en figura 2.6), y estos son los pasos a realizar con él:

- Utilizar el comando ”/newbot” para iniciar un dialogo con el cual podremos configurar el chatbot.
- Introducir el nombre del chatbot, que será el que aparezca en los detalles de contacto.
- Introducir el nombre de usuario, que será el nick con el que los usuarios nos encontrarán para añadirnos a una conversación.
- Ahora es necesario conectar, el bot creado con nuestro código fuente, y por lo tanto debemos tenerlo alojado en un servidor, para que Telegram pueda realizarle peticiones. Solo necesitamos acceder a este enlace https://api.telegram.org/bot=TOKEN/setWebhook?url=URLSERVIDOR donde TOKEN es la key facilitada al terminar con @BotFather, y URLSERVIDOR es donde tenemos alojado nuestro código.

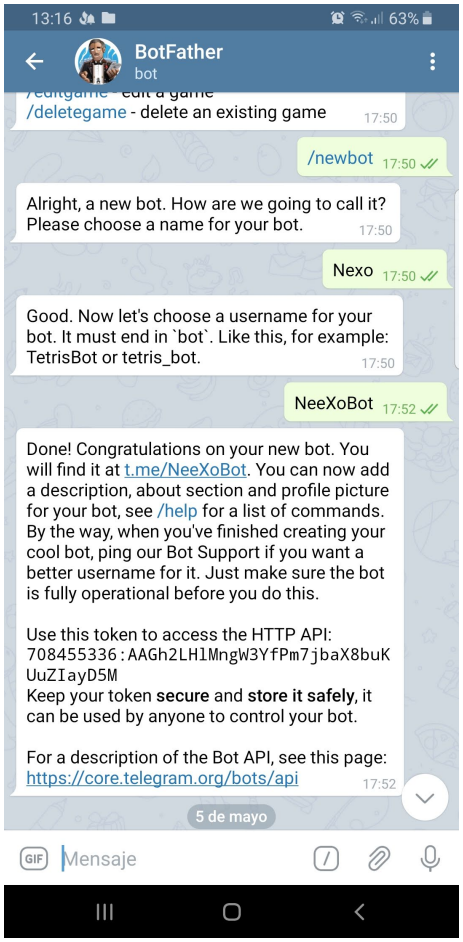


Figura 2.6: Bot Father

Algunos de los bots conversacionales que han tenido mayor éxito en esta plataforma (Santos, 2019) han sido:

- @AlertBot: Bot que te permite programarlo para que te envíe un mensaje determinado a una hora concreta, a modo de recordatorio.
- @WeatherMan: Puedes obtener una previsión meteorológica del tiempo en tu ciudad, para mañana o para los próximos cinco días.
- @AmazonGlobalBot: Es uno de los bots ”Inline” puedes preguntarle el precio de cualquier producto de Amazon, y te muestra hasta un gráfico de la evolución de precio.
- @Storebot: un chatbot conocido como ”el bots de bots”, permite recoger los bots mejor valorados, los mas recientes, incluso puede agrupar hasta por categorías.

Como hemos visto, la creación de un bot en esta plataforma es una tarea muy sencilla, gracias a la API que nos han facilitado. No se necesita conocimientos técnicos para un bot básico pero si queremos darle algo más de forma, las posibilidades que nos ofrecen son muy amplias, ya que adopta cualquier tipo de lenguaje y hay un gran número de librerías disponibles para los desarrolladores que contienen las funciones básicas del bot. Uno de los lenguajes con mayor peso es Python, aun que yo me he inclinado, por la librería de Microsoft ”Microsoft Bot Framework V4 SDK”.

2.3. Portal de Información UEx

Las universidades cada vez apuestan más por una infraestructura donde se almacenen grandes cantidades de datos, para ofrecer una gran cantidad de información tanto interna para profesores y alumnos, como externa para equipos de investigación basándose en informes reportados por la Universidad, como pueden ser planes de estudio, servicios disponibles, etc.

Esta infraestructura debe estar formada por un gran centro de datos y una red potente sobre la que se depositen todos los servicios y el trabajo se pueda realizar de forma fluida.

Actualmente, la Universidad de Extremadura cuenta con un portal de datos abiertos (*Open Data*), tanto de la comunidad universitaria como de ciudadanos, con datos identificativos, académicos, económicos, y de cualquier otra índole que puede servir de gran utilidad a la sociedad, y que pueda fomentar la transparencia de las instituciones.

Se ha realizado una gran labor para que los datos se puedan descargar en diferentes formatos, y siguiendo el estándar W3C para que cualquier aplicación de software pueda utilizarlos sin ningún problema. El conjunto de datos que utiliza es ”open linked data”.

He querido mencionar este portal, por que tiene un papel importante en el desarrollo del chat bot, ya que recoge datos de los centros, de las asignaturas, profesores y de los grados disponibles de toda la Universidad de Extremadura. La url de acceso al índice donde se encuentran todos estos datos es `http://opendata.unex.es/dataset`.

También se ha utilizado para recolectar datos la página principal de la Universidad de Extremadura conocida como `https://www.unex.es/`.

Capítulo 3

Solución Propuesta para NexoBot

En este capítulo, se hablará de las técnicas que se han evaluado y han sido utilizadas para realizar el desarrollo del Bot.

La decisión de qué tecnologías utilizar se realizó en una etapa temprana, y el objetivo principal del proyecto era hacer un bot lo más realista posible, para que los que lo utilicen no piensen que están hablando con una máquina, si no con un humano.

El ecosistema del bot es muy amplio, por lo que se hará una división para tratar cada tecnología utilizada de forma independiente. Ver figura 3.1. El núcleo principal donde está el corazón del bot es la API REST, para su desarrollo se tuvo que decidir por qué framework optar. Actualmente existen muchísimos frameworks y herramientas para la creación de bots.

Las primeras herramientas o constructores de bots que fueron descartadas, son las que vienen de páginas web online, como Chatfuel, ManyChat, etc. Estas fueron descartadas porque carecen de características PNL, y por lo tanto no podríamos realizar nuestro bot conversacional teniendo en cuenta los objetivos marcados. Aparecieron tres opciones principales que permitían que fuera viable lograr los objetivos marcados por el proyecto, estos fueron Botkit, RASA NLU y Microsoft Bot Framework. El primero en descartar debido a la poca influencia de la comunidad de desarrolladores y de la poca información que había al respecto, a pesar de su gran soporte de PNL, fue RASA NLU. Las diferencias entre Microsoft Bot Framework y BotKit no eran

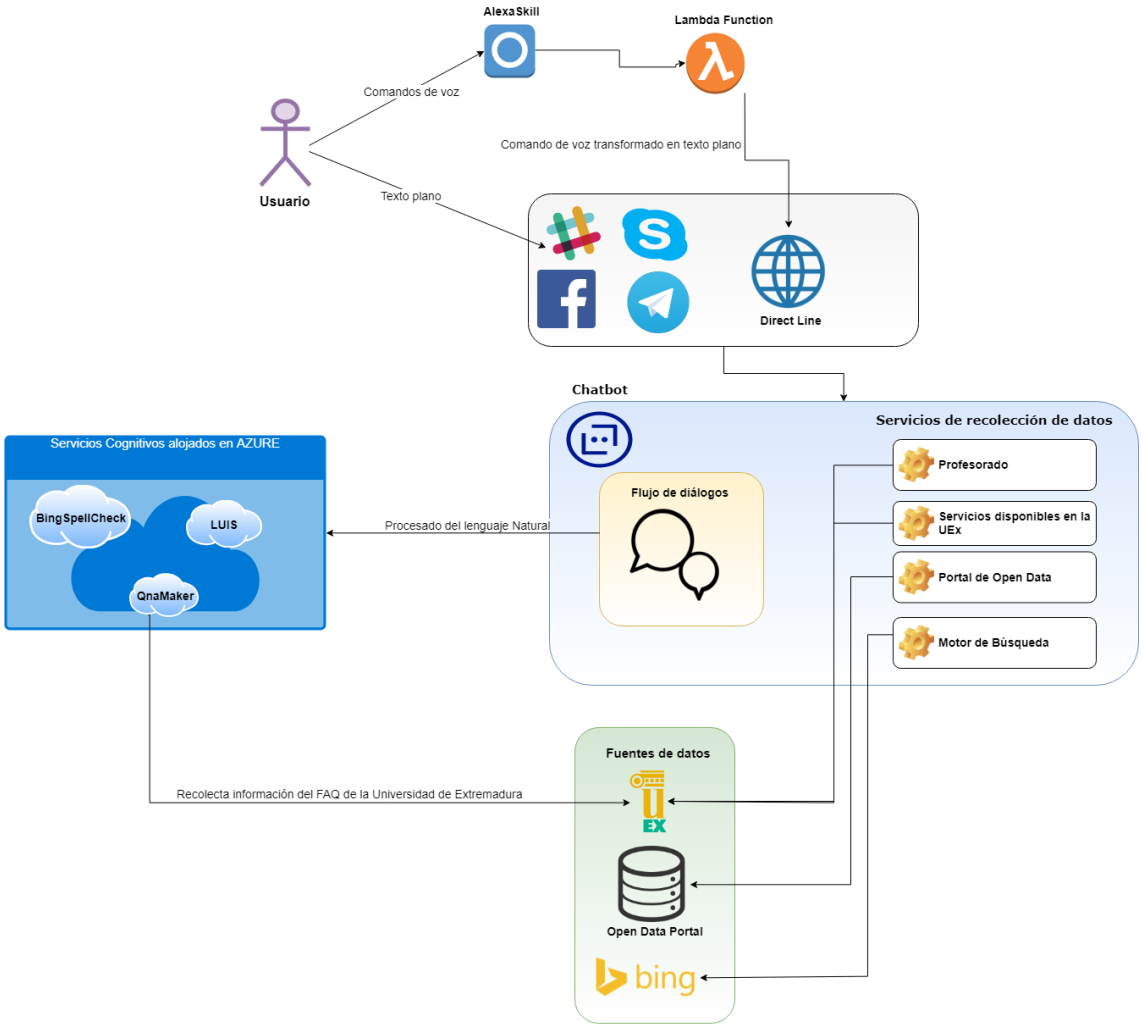


Figura 3.1: Ecosistema de NexoBot

tan fuertes como para decantarme por uno u otro, ambos tenían un gran apoyo de la comunidad de desarrolladores, buena documentación, y soporte con PNL, no integrado en el framework, pero sí a través de servicios IA muy fáciles de acoplar. Tendiendo en cuenta estos servicios IA, Microsoft Bot Framework era el perfecto, ya que a pesar de tener mucha dependencia de sus servicios de IA, cumplía a la perfección lo necesario para el desarrollo del proyecto, sin embargo BotKit se quedaba algo corto con los servicios IA que se podían integrar, algunos de ellos con bastantes bugs. Ver resumen en figura 3.2.

Una de las grandes ventajas de elegir la tecnología de Microsoft sobre chatbots es el gran número de herramientas y facilidades que tenemos en el portal de Azure, como

	Botkit	RASA	Azure Bot Service
Integración con plataformas de mensajería instantánea	✓	✗	✓
Soporte PNL	✗ Añadiendo Middlewares	✓	✓ Integrado con LUIS.AI
Lenguajes de programación	Javascript(Node.js)	Python	Javascript(Node.js), C#
Soporte de voz	✗	✗	✓
PROS	• Buena documentación y comunidad creciente. • Fácil de implementar.	• Mucho control del PLN. • Fácil desarrollo al utilizar un lenguaje como Python. • Puede imitar a LUIS.AI si se usa como servidor local.	• Muy enfocado a bots "asistentes". • Sencilla integración con servicios y plataformas de bots. • Buena documentación y gran comunidad(open-source).
CONTRAS	• Los middlewares son muy inestables.	• Exigentes requisitos en los servidores. • Curva de aprendizaje . • No se integra fácilmente con otras plataformas.	• Alta dependencia de servicios de Microsoft. • Difícil de usar para desarrollos muy complejos.

Figura 3.2: Tabla resumen de comparativa entre los diferentes frameworks de desarrollo de bots.

despliegues, integración con diferentes canales de comunicación, alta disponibilidad del sitio web, etc. Pero antes de entrar en detalle sobre las tecnologías, si seguimos un orden, en primer lugar necesitamos tener un gran volumen de datos que van a ser necesarios para el funcionamiento de nuestro bot, seguido de la forma en la que recolectarlos y del aprendizaje autónomo una vez realizado el análisis de los datos. Y por último, la implementación tecnológica del bot.

3.1. Fuente de datos

Para el funcionamiento del bot, es necesario que la fuente de datos siempre esté disponible, ya que si no está disponible no podemos ofrecerle al usuario todas las opciones. A continuación, se explicará detalladamente todas las funcionalidades necesarias así como las distintas tecnologías disponibles para hacer posible su desarrollo, y cuáles son los motivos por las que han sido elegidas.

3.1.1. Acceso a datos

Para el acceso a datos, se barajaron varias opciones, una de ellas era realizar un proceso que se encargara de leer los datos mediante web scrapping o de diferentes formas de recoleccion de datos y volcarlos en una base de datos, pero el volumen de datos era limitado por lo que no merecía la pena. Se optó por la opción de implementar pequeños microservicios dentro de la API del bot, almacenando los datos en memoria, pero evitando realizar la misma consulta varias veces con la implementación de una caché. En este sentido, uno de los objetivos, era abstraer lo máximo posible cada funcionalidad del bot, consiguiendo que el bot no conozca de donde se están sacando los datos, lo único que conoce es la existencia de un microservicio que se encarga de darle la lista de profesores, la lista de asignaturas, etc. Gracias a esto, si el día de mañana se produce un cambio en el servicio de la Universidad de Extremadura que provee los datos por otro tipo de servicio, como puede ser SOAP o REST, el bot no necesita ninguna modificación, siendo únicamente necesario, realizar una nueva

implementación del microservicio que debe estar totalmente desacoplado del bot.

3.1.2. Recolector de datos

Introducción

El proceso de recolección de datos tiene un papel importante para nuestro chatbot. La cantidad de información debe ser suficiente como para poder entablar un dialogo con un usuario.

Para la recolección de información la solución fue sencilla y estaba tomada desde el principio, ya que se conocía la existencia del portal de datos abiertos de la Universidad de Extremadura. El único problema encontrado, era que no todos los datos estaban disponibles en el portal, datos como fichas de asignaturas, horario de tutorías, etc.

Como solución no viable, se podría introducir toda está información en el portal y que el bot solo consuma esa fuente de datos, y sea su principal proveedor. Pero por temas de recursos y tiempo se descarta, ya que se sale fuera de los terrenos del proyecto.

Por último, se optó por una alternativa para la recolección de esos datos no existentes en el portal de Open Data, utilizando una técnica denominada Web Scraping, que será explicada a continuación.

Web Scrapping

El web scraping es una técnica que sirve para recoger información de páginas web de forma automatizada. De ahí su traducción al español “escarbar una web” (Martí, 2016).

Su objetivo es conseguir grandes cantidades de información sin teclear una sola palabra, a través del uso de algoritmos de búsqueda que rastrean centenares de webs extrayendo sólo aquella información que necesitamos.

El funcionamiento básico de esta tecnología, consiste en transformar el HTML de la página web del que queremos obtener sus datos en un formato válido del que

se pueda procesar la información. La mayoría de las veces se utilizan expresiones regulares para la obtención de los campos que se necesitan.

La mejor librería de .NET utilizada para esto, es HTML Agility Pack, he usado esta librería, porque maneja bastante bien el HTML mal formado, donde no hay etiquetas de cierres, hay elementos mal anidados, etc. Además, tiene soporte para convertir el HTML en XML, y permite procesar los nodos de forma más eficiente. Esto es un punto importante, puesto que si se hacen muchas consultas en un rango corto de tiempo el rendimiento podría verse afectado.

A pesar de haber elegido esta tecnología, es la que más inconvenientes puede tener. El primero de estos inconvenientes, y el que más nos afecta, es que estamos acoplados a la estructura original del HTML. Con esto quiero decir, que si la página sobre la que estamos haciendo la recolecta de información, cambiase la estructura, por un tema meramente estético, nosotros tendríamos que cambiar también nuestro código para poder leer la nueva estructura.

Otro problema es el tiempo de estos algoritmos de búsqueda. Cada vez que se quiera obtener información de la página habrá que realizar una búsqueda dentro de su contenido, página por página y de forma independiente. Si la página tiene una estructura pequeña, no se notará la búsqueda, pero si la estructura de la página es enorme, puede que tengamos problemas de rendimiento.

Microservicios recolectores de datos

Los microservicios han sido divididos por funcionalidad puesto que cada microservicio tiene un único propósito. Los microservicios utilizados para la recolecta de datos son:

- Servicios ofrecidos en la UEx: es un servicio que utilizando la librería HtmlAgilityPack se encarga de recolectar datos sobre todos los servicios disponibles en la Universidad de Extremadura[1] .

[1] `https://www.unex.es/organizacion/servicios-universitarios`

- Servicio de Profesorado: el principal objetivo de este servicio es obtener los datos de los profesores de la Universidad de Extremadura, utilizando la misma librería que el servicio anterior. La principal diferencia de este servicio, es que existe un número elevado de profesores de diferentes carreras, y en la forma de extraer los datos es necesario generar una URL dinámica donde se indique el identificador del centro, para poder obtener todos los profesores de ese centro. La url que se debe generar de forma dinámica es la siguiente `https://www3.unex.es/inf_academica_centro/index.php?mod=profesores&file=index&id_centro=16`, siendo el valor 16 el identificador de la Escuela Poltécnica.

- Servicio del portal de Open Data: este servicio se encarga de recoger los datos del portal de Open Data de la Universidad de Extremadura, a partir de querys de sparql hacia la API del portal, obteniendo en formato JSON información de asignaturas, titulaciones y centros.

- Servicio de motor de búsqueda: este servicio es encargado de realizar búsqueda a través del motor de búsqueda de Bing, realizando un filtro con el dominio adecuado. Siendo capaz de devolver mucha información sobre cada uno de los resultados obtenidos. El primer resultado es el que tiene mayor relevancia por lo que será devuelto como respuesta del bot.

- Otros Servicios: hay otros tipos de servicios que no funcionan como recolectores de datos, estos son LuisService y SpellCheckService, cuyo objetivo común es el de dotar inteligencia artificial al bot, realizando tratamiento de datos, en vez de recolección. La única excepción es QnaService, que además de dotar de IA, también recolecta datos del FAQ de la Universidad de Extremadura. Se hablará más en detalle acerca de estos servicios.

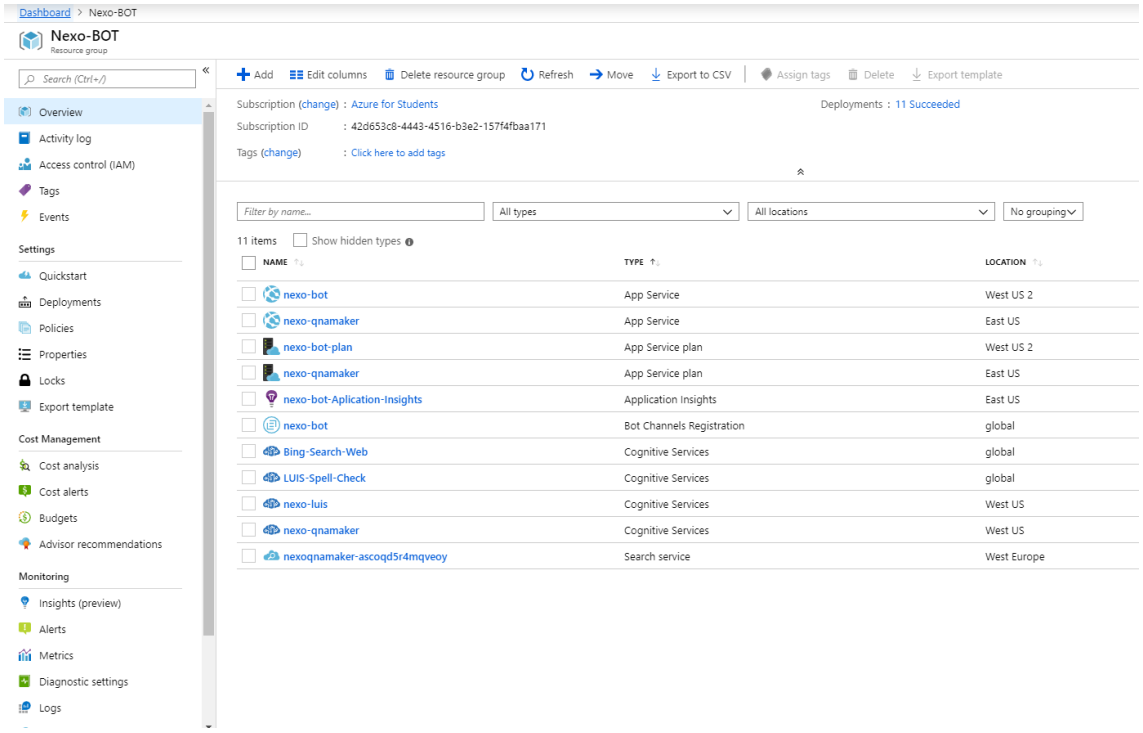


Figura 3.3: Panel de servicios de NexoBot

3.2. Portal Azure

Azure es una nube pública, donde se alojarán todas las aplicaciones y servicios que utilizará el bot que se va a implementar y será el lugar donde administrar todos estos servicios de forma rápida y sencilla. Ver figura 3.3.

En Azure existen diferentes servicios de infraestructura y plataformas para el desarrollo de aplicaciones, proporcionando almacenamiento, redes, máquinas virtuales, etc. Además, dispone de gran seguridad y protección de datos, con respaldo por posibles errores consiguiendo así una disponibilidad del 100 % para el usuario.

Uno de los servicios que ofrece Azure es la creación y administración de bots donde destacan las siguientes funcionalidades:

- Pruebas: Gracias al chat web integrado, se puede comprobar el funcionamiento del bot.
- Análisis: Es posible la recolección de datos para mostrar gráficas de nº de usuarios, tráfico, latencia, etc...

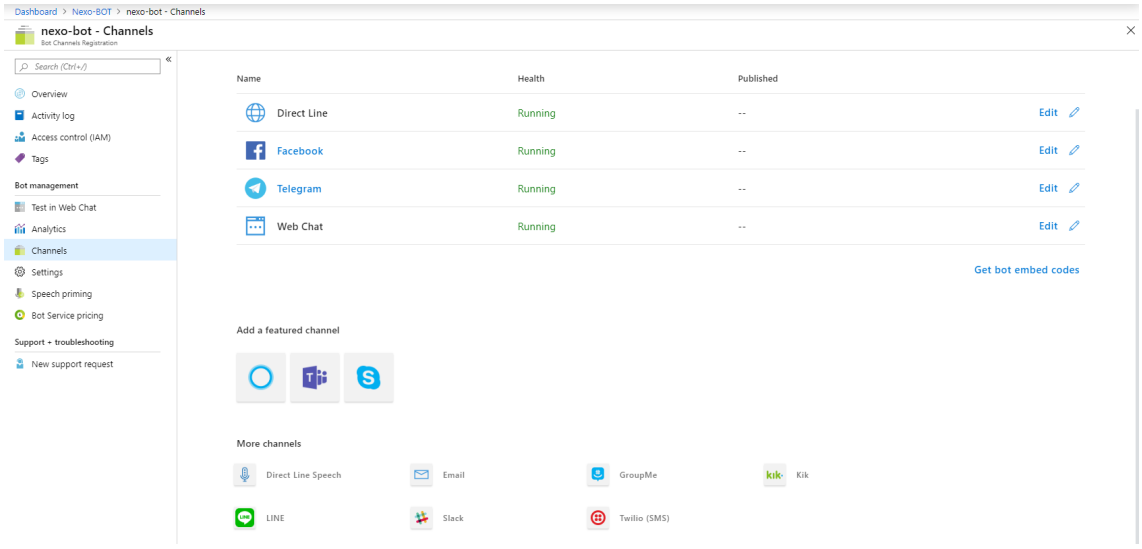


Figura 3.4: Canales de comunicación de NexoBot

- Comunicación: Se puede configurar y administrar los canales disponibles para la comunicación entre bot y usuarios. Ver figura 3.4.

Para la implementación del bot en Azure utilizando Microsoft Bot Framework y Azure existen tres opciones:

- Edición en línea: Es posible modificar el código de la aplicación sin necesidad de utilizar un entorno de desarrollo integrado.
- Descarga de código: Para trabajar con el código del bot en un entorno de desarrollo integrado y hacer pruebas en local si fuese necesario. Una vez realizado los cambios habría que volver a subir el código.
- Integración continua: Con Visual Studio el desarrollador será capaz de trabajar localmente y una vez verificados los cambios, al utilizar la función publicar estos se aplicarán automáticamente en la nube.

3.3. Microsoft Bot Framework

Está desarrollado en C# que es un lenguaje de programación orientado a objetos dentro de la plataforma de .NET y ofrece un SDK conocido como Bot Builder V4

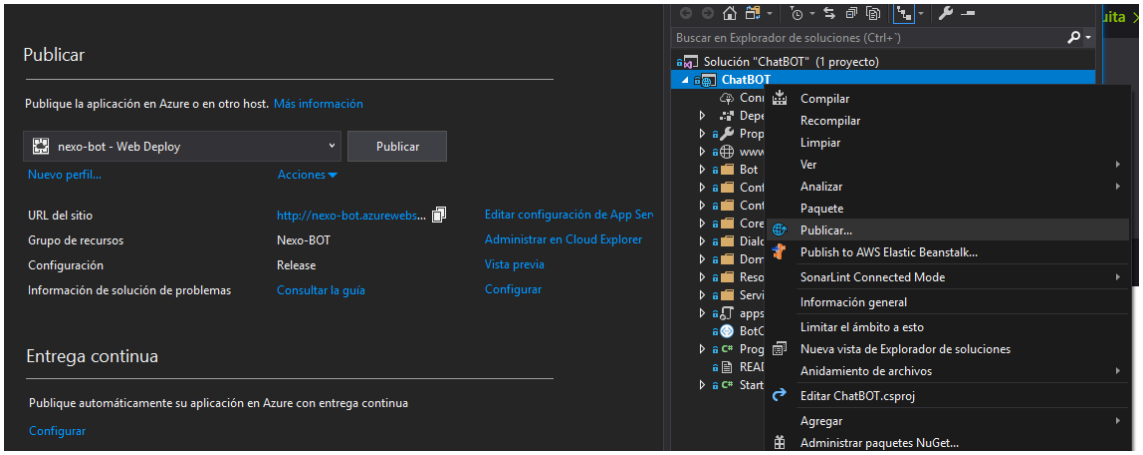


Figura 3.5: Publicar cambios en Azure

compuesto por un conjunto de herramientas para el desarrollo de bots. Cuenta con una librería de manejo de diálogos que proporciona diálogos en cascada para facilitar la administración de la conversación de un bot.

Algunos de los conceptos claves para familiarizarse con el framework son:

- Connector: Proporciona una API REST permitiendo que un bot se comunique a través de múltiples canales. Facilita la comunicación entre bot y usuario transmitiendo mensajes a los diferentes canales.

- Activity: El objeto Activity es usado por el conector (Connector) para pasar información entre bot y canal (usuario). El tipo más común de actividad (Activity) es el mensaje, pero existen muchos otros tipos de actividad.

- Dialog: Cuando se crea un bot usando Bot Builder SDK, podemos usar "Dialogs" para modelar una conversación y administrar el flujo de conversación. Este puede estar compuesto por otros cuadros de diálogo y así maximizar la reutilización. Existe una pila de diálogos donde se almacenan los diálogos activos y la ruta entre diálogos.

Además cuenta con un número elevado de servicios disponibles para integrar con el framework, muchos de ellos importantes para el bot, como son LUIS, Bing Web Search, Bing Spell Check o QnaMaker, siendo todos de estos servicios cognitivos con aplicación de PNL.

3.3.1. Servicios Cognitivos

Dentro de los servicios cognitivos de Azure existen APIs y SDKs que ayudan a los desarrolladores a crear aplicaciones inteligentes sin necesidad de realizar inteligencia artificial o ciencia de datos desde cero. Los servicios cognitivos de Azure permiten a los desarrolladores agregar fácilmente funciones cognitivas a cualquier tipo de aplicación.

La integración de Azure Cognitive Services en el bot es crucial para mejorar todas las tareas de procesamiento del lenguaje natural, para que nuestro bot, comience a hablar, comprender e incluso comenzar a razonar.

El catálogo de servicios dentro de Azure Cognitive Services se puede clasificar en cinco pilares principales: visión, voz, lenguaje, búsqueda y decisión, pero nuestro bot se centrará exclusivamente en dos, lenguaje y búsqueda.

Bing Spell Check

Este servicio cognitivo es el más simple de los utilizados en el desarrollo de nuestro bot, pero es uno de los más necesarios. Dada la cantidad de faltas de ortografía que se producen en los chats de mensajería instantánea, es muy difícil realizar inteligencia artificial sobre un lenguaje con faltas de ortografía, ya que puede haber infinitas posibilidades de escribir una palabra. La primera iteración que hará el bot, será realizar una llamada a la API ”Bing Spell Check”, con el mensaje enviado por el usuario, y devolver ese mismo mensaje pero sin faltas ortográficas, con esto garantizamos que el diálogo sigue el flujo con palabras pertenecientes al lenguaje natural, y de forma transparente para el usuario, ya que todo se hace internamente.

Como ventaja, cuenta con una fuente de datos enorme, que está actualizándose constantemente, ya que incluso es capaz de reconocer nuevas marcas, títulos o expresiones populares según van apareciendo en nuestros lenguajes cotidianos.

LUIS

Language Understanding (LUIS) (Microsoft, 2019a), es un servicio API basado en la nube, que aplica inteligencia artificial personalizada de aprendizaje automático al

texto del lenguaje natural de un usuario, para predecir el significado general y extraer información relevante.

La aplicación de LUIS tiene un modelo de lenguaje natural específico que puede ser previamente construido o basado en entidades personalizadas. El servicio de LUIS que utiliza el bot, tendrá entidades personalizadas según demanda.

Este tipo de entidades personalizadas se pueden llamar también intenciones (intents), que representan acciones que el usuario quiere realizar en base a una serie de expresiones.

En la figura 3.6 se pueden ver todas las intenciones creadas para nuestro bot.

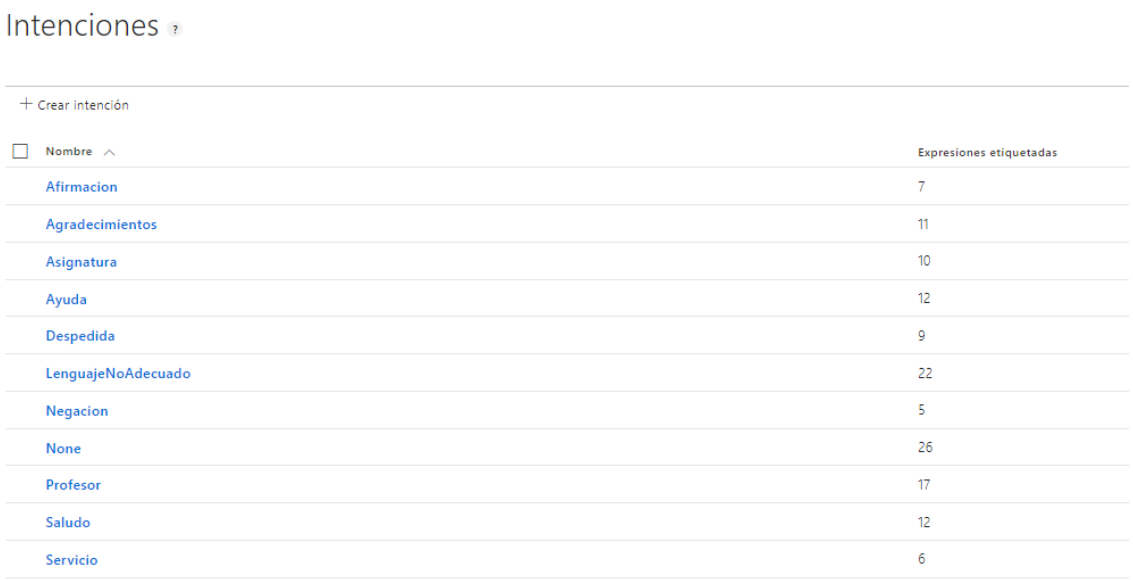


Figura 3.6: Intenciones NexoBot en LUIS

Estas intenciones serán de utilidad para conocer qué tipo de acción o de dialogo tiene que llevar a cabo el bot. Cada intención tiene asignada una lista de expresiones las cuales tienen una puntuación (ver figura 3.7). Cuando un usuario envía una frase a la API de LUIS, este responde con la intención que más puntos le ha dado para esa frase. Viendo la anterior figura, si le escribiéramos una frase al bot, y pusiéramos ”afirmativo” casi con total seguridad LUIS devolvería en su respuesta que la acción que se debe ejecutar es la relacionada con la intención ”Afirmación”.

Como ventaja, remarcaría la posibilidad de mejorar la precisión de la predicción,

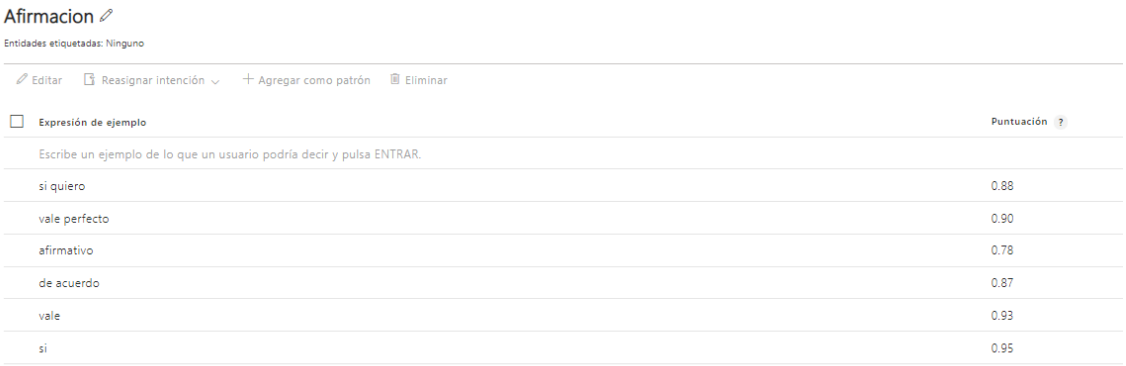


Figura 3.7: Expresiones de una intención en LUIS

a través de métodos como el aprendizaje activo de enunciados de punto final, lista de frases para incluir palabras ”comodín” y patrones para reducir la cantidad de expresiones asociadas a intenciones. Y como desventaja de este servicio es que cada vez que se quiera introducir alguna intención nueva, será necesario modificar el código.

QnaMaker

Es un servicio API alojado en la nube capaz de crear una capa de conversación, sobre preguntas y respuestas a partir de una fuente de datos (Microsoft, 2019b).

La función principal que tiene es la de crear una base de conocimiento (KB) a partir de su contenido semiestructurado, a partir de URLs de preguntas frecuentes (FAQ), manuales de productos, documentos de soporte y preguntas y respuestas personalizadas.

El servicio QnA Maker responde las preguntas de lenguaje natural de sus usuarios al compararlas con la mejor respuesta obtenida de su base de conocimiento. Por lo tanto, será de mucha utilidad para el bot, ya que podrá responder preguntas basándose en la base de conocimiento que se ha generado a través de FAQ de la Universidad de Extremadura, aplicando PNL e inteligencia artificial para poder relacionar el lenguaje natural de una pregunta con otra aunque se hayan formulado de diferente forma.

La arquitectura de QnA Maker se divide en dos componentes (ver figura 3.8):

- Servicios de gestión de QnA Maker: creación inicial, actualización, capacitación

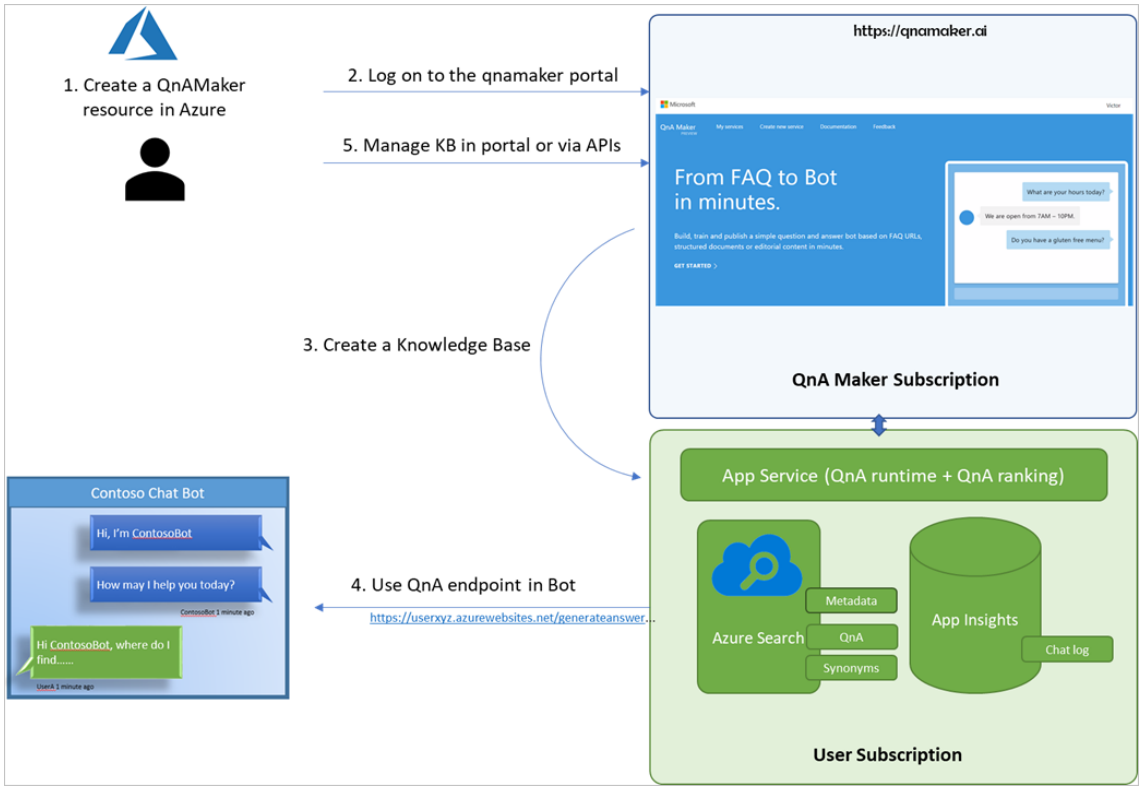


Figura 3.8: Arquitectura de QnaMaker extraído de Microsoft, 2019b, Microsoft

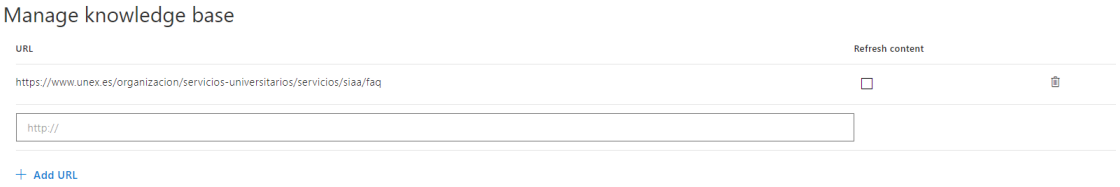


Figura 3.9: Configuración de URL del FAQ de la UEx

y publicación de la base de conocimiento. Estas actividades se pueden realizar a través del portal (ver figura 3.9) o las API.

- Datos y tiempo de ejecución de QnA Maker: esta parte se necesita hacer desde la suscripción de Azure en la región especificada. Todo el contenido KB se almacena en Azure Search y el endpoint se implementa como un servicio de aplicaciones.

Bing Web Search API

Este servicio de inteligencia artificial, se usará de comodín cuando las preguntas relacionadas con la Universidad de Extremadura, no estén en el servicio de QnaMaker, ya que con Bing Web Search API tenemos un servicio RESTful que proporciona respuestas a consultas de usuarios, de los resultados del motor de búsqueda Bing, estos datos son proporcionados en formato JSON y se puede decidir la relevancia con la que son mostrados (*Bing Web Search Api Docs*).

Alguna de las características destacables que tiene el servicio, es que se puede filtrar y restringir el contenido, en nuestro caso se ha restringido que solo realice búsquedas en páginas web donde el dominio sea ”unex.es”. Así garantizamos que la información es de la Universidad de Extremadura y no de otra universidad de España.

También puede localizar datos por región, país, y eliminar caracteres especiales de los resultados de búsqueda como por ejemplo los caracteres con formato Unicode.

3.3.2. Diálogos

La forma más útil de gestionar una conversación con el usuario es a través de los diálogos. Cada diálogo está diseñado en el bot para realizar una tarea específica, y siguiendo un orden específico. La forma de invocación puede variar, puede ser a partir de la respuesta de un usuario, a respuestas de estímulos externos, o invocados desde otro diálogo.

Los diálogos tienen una serie de componentes, que son de mucha utilidad para el flujo de la conversación de nuestro chatbot.

- Colección de diálogos: pueden ser diálogos en cascada, diálogos de componentes, etc. Cada dialogo es añadido a la colección con un ID, este ID será luego utilizado por si se quiere ir de un diálogo a otro.

- Contexto del dialogo: contiene la información del diálogo, y si el usuario ha interactuado con el diálogo, este contexto contiene, el diálogo padre, y el estado actual del dialogo, muy útil para guardar información de forma persistente.

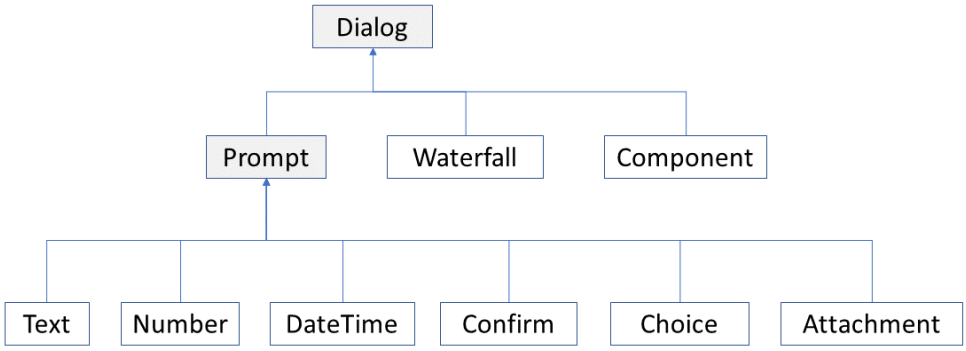


Figura 3.10: Tipos de diálogos extraído de Microsoft

En la figura 3.10 se pueden apreciar los diferentes tipos de diálogos, pero para nuestro bot, nos centraremos principalmente en dos de ellos, y son los diálogos de tipo aviso y en cascada.

Diálogos de aviso (prompt)

Este tipo de diálogo tiene como principal objetivo pedir información al usuario y evaluar su respuesta de la forma más fácil posible (*Dialogs in Azure Bot Service*). Por ejemplo, para una pedir un número , se especifica la pregunta o la información que está solicitando, y la solicitud comprueba automáticamente si recibió una respuesta de número válida. Si lo hizo, la conversación puede continuar; de lo contrario, volverá a solicitar al usuario una respuesta válida.

Para el desarrollo del chatbot este tipo de diálogo se usa muy a menudo, ya que en muchas ocasiones es necesario realizar preguntas sobre el centro del qué se quiere obtener la información, el nombre de la asignatura, el nombre de un profesor, etc.

Dependiendo del tipo de aviso que utilicemos, podemos configurarlo de diferentes formas, algunas de ellas son; con el uso de sinónimos, o utilizando expresiones

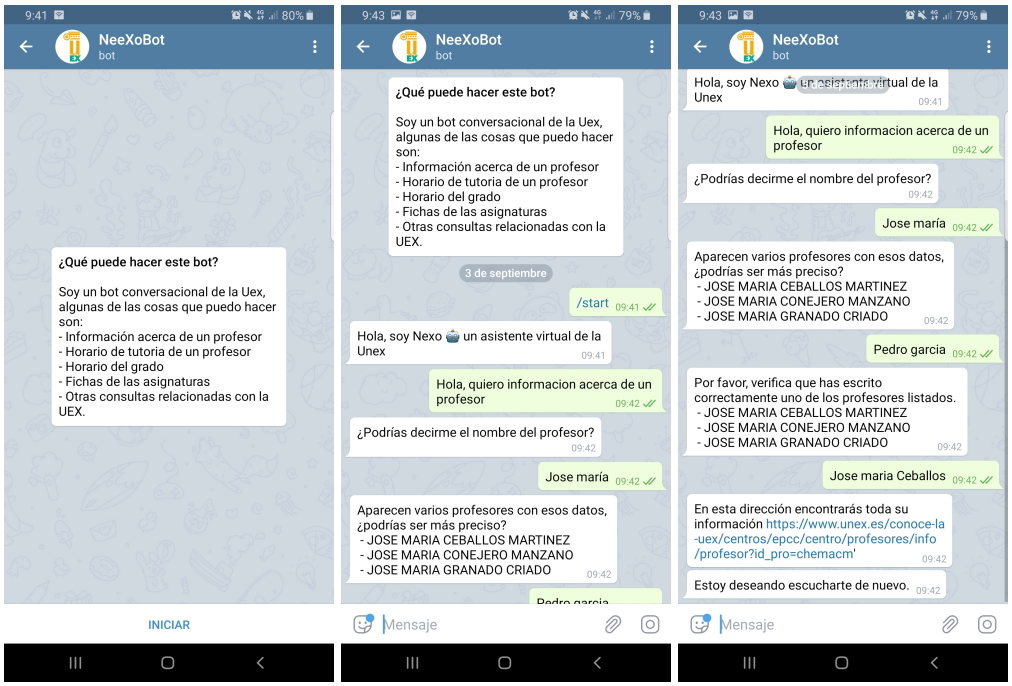
regulares para la obtención del resultado.

Pero una de las funciones más útiles es la de hacer tu propio validador sobre esos datos, ya que puedes pedir la edad de la persona, y esperar un número, dado que no sería válido cualquiera, se puede crear un validador y asignar un rango.

Diálogos en cascada

Para poder realizar un flujo de conversación guiado por pasos, era necesario el uso de los diálogos en cascada en el cual, cada paso de la conversación se implementa como una función asíncrona que toma un parámetro de contexto (WaterFallStep).

Cuando se define un diálogo, normalmente se introducen una serie de pasos, estos pasos son denominados WaterFallStep, y son las acciones que el bot va ejecutando de manera ordenada. Ver en figura 3.11. En cada paso, el bot solicita al usuario una entrada



(a) Diálogo inicial. (b) Diálogo intermedio. (c) Diálogo final.

Figura 3.11: Ejemplo real de diálogo en cascada del bot conversacional.

de datos (o puede comenzar un diálogo secundario, pero a menudo es un diálogo de tipo aviso realizando una pregunta), espera una respuesta y luego pasa el resultado

al siguiente paso. El resultado de la primera función se pasa como argumento a la siguiente función, y así sucesivamente. De este modo, conseguimos una conversación totalmente guiada y transparente para el usuario, simulando que se está teniendo una conversación con un humano y no con un chatbot.

La figura 3.12 es un diagrama de secuencia del diálogo en cascada puesto de ejemplo anteriormente. En él se observan los diferentes ”steps” por los que pasa el bot durante la conversación.

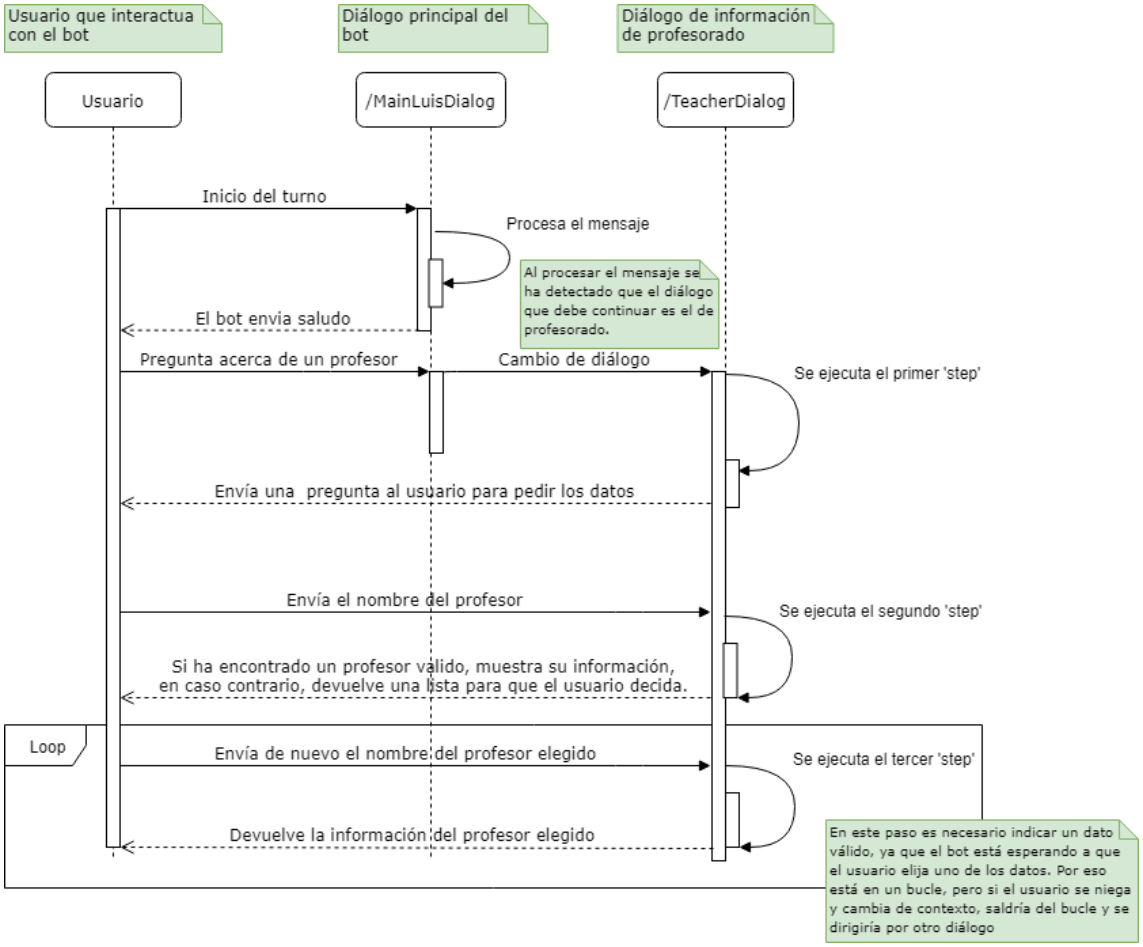


Figura 3.12: Diagrama de secuencia de un hilo de conversación en cascada.

3.3.3. Language Generation

Es una nueva librería que se puede integrar en el SDK de Bot Builder que ofrece la posibilidad de definir varias frases sobre un mismo contexto que va asociándolas a

un identificador. Cuando un usuario quiera obtener una frase a través del identificador, esta librería mandará como respuesta una de las frases configuradas de forma aleatoria, proporcionando una experiencia conversacional más natural entre el bot-humano (*Language Generation Documentation*).

Actualmente se encuentra en fase de desarrollo, y no hay mucha documentación al respecto. Para el desarrollo de este Trabajo de Fin de Grado he tenido que hacer algunas modificaciones directamente desde el código de la librería de algunas cosas que he ido viendo que se podían mejorar, y que podían ser de bastante utilidad. Alguna de ellas, ha sido aprobada por desarrolladores de Microsoft, y está incluida en las nuevas versiones.

La generación de lenguajes, se realiza a través de la evaluación de plantillas. Estas plantillas tienen un formato particular, para que el framework pueda interpretar correctamente todos los datos. Un ejemplo de una plantilla de nuestro bot sería el siguiente:

```
# Gratitude
- [GratitudePrefix], [GratitudeSuffix]

# GratitudePrefix
- No tienes porque agradecerlo
- De nada
- No hay de que

# GratitudeSuffix
- para eso estamos.
- estoy encantado de ayudarte.
- me encanta compartir mi conocimiento.
```

Cuando desde el código se quiera evaluar el valor de ”Gratitude”, no se sabe con certeza cuál es el valor que va a devolver, y es por eso por lo que tiene un sentido más natural.

Este tipo de plantillas, cada vez está evolucionando más, ahora es posible hacer ”switch cases”, evaluar condiciones, etc.

3.4. Alexa y Amazon Web Services

Uno de los principales objetivos marcados, era integrar el bot en el mayor número de canales de comunicación existentes, o al menos los mas utilizados, y Alexa es uno de los más utilizados en el mundo de los asistentes virtuales.

El primer reto, era integrar la API de Bot Framework con una skill de Alexa, pero como hemos visto en la figura 3.4, no tiene soporte para este canal de comunicación desde el panel de administración de Azure.

Aquí aparece el primer problema entre el SDK de Microsoft Bot Framework y el SDK de Alexa. En una primera opción se intentó poner el endpoint del bot de azure en Alexa, pero los modelos de datos eran muy diferentes entre ambos frameworks.

Para solucionar este problema, se utilizó un canal de comunicación que el SDK de Microsoft tiene de comodín, Direct Line. Este canal permite la comunicación directa entre una aplicación cliente y el bot, las solicitudes se realizan mediante HTTPS y son enviadas mediante POST. El único requisito de Direct Line son las credenciales, que pueden ser mediante una clave secreta que se puede obtener del panel de administración de Azure o desde un token generado en tiempo de ejecución. Estas credenciales deben viajar en la cabecera de la solicitud en el campo ”Authorization”.

Pero seguíamos con el mismo problema, la solicitud enviada por el SDK de Alexa no era la esperada, y el modelo del bot no estaba preparado para transformar ese objeto. Como primera idea, se propuso modificar el modelo de solicitudes y respuestas del bot de microsoft, pero esto se descartó enseguida viendo que rompía la compatibilidad con los otros canales de comunicación.

La solución final por la que se optó, fue la de crear un ”puente” entre el bot y Alexa que haga una conversión de los datos y los haga compatible entre ellos. La forma más sencilla y eficiente de hacer esto, siguiendo la documentación de desarrollo de Amazon, es la de crear una función AWS Lambda, que no es más que un servicio

informático que ejecuta, en nuestro caso, una única función independientemente del lenguaje de programación utilizado, sin necesidad de administrar servidores, permitiendo que solo nos preocupemos de la parte del código fuente. Además, una de las grandes ventajas es que cuando el número de peticiones es muy elevado, Amazon hace automáticamente un escalado[2].

El lenguaje de programación elegido fue C# y con el framework de .NET Core, debido a que las librerías que se necesitaban tenían esas dependencias. Estas librerías fueron Alexa.NET para conseguir los modelos de las solicitudes procedentes de la Skill de Alexa, y la librería de Direct Line para la creación de un cliente que realice peticiones al canal de comunicación de Direct Line activado en el bot, transformando la solicitud de Alexa en una solicitud de Direct Line, y una vez recibida la respuesta del canal de comunicación del bot, transformar de nuevo la respuesta en un modelo válido para Alexa, funcionando de intermediario.

Una de las grandes ventajas de esta decisión, es la escalabilidad, pudiendo hacer modificaciones y funcionalidades únicas para Alexa, sin necesidad de modificar el bot, solo la función AWS Lambda.

3.5. Planificación y gestión del proyecto

La planificación del desarrollo de este Trabajo de Fin de Grado se ha realizado siguiendo algunos de los principales pilares de la metodología ágil Scrum. Esta metodología, ayuda a los equipos de software a resolver los problemas y tareas de los proyectos de la forma más productiva posible, intentando hacer entregas con el mayor valor posible.

Algunas de las características principales de esta metodología, y que fueron de principal ayuda para su elección, son las siguientes:

- El objetivo principal del proyecto estaba muy bien definido.
- Mucho trabajo se podía desglosar en pequeñas partes.

[2]ampliar el número de nodos del clúster donde está alojada la aplicación para mejorar el rendimiento.

- Organización semanalmente o mensualmente. Durante el desarrollo del Trabajo de Fin de Grado he estado en Madrid, por lo que era muy complicado tener un seguimiento diario.

- Entregas periódicas para recibir retroalimentación y poder realizar mejoras o modificaciones sobre el producto entregado, consiguiendo hacer entregas con mayor valor.

La herramienta de gestión utilizada ha sido Trello, por su sencillez y por lo completa que es su versión gratuita. Permite crear un tablero, donde se organiza todo le trabajo, compuesto por listas independientes que contienen tarjetas. Las listas son creadas para un ámbito concreto, en nuestro caso, se ha decidido crear cuatro listas relacionadas con la metodología utilizada, y las tareas irán asociadas a cada lista, dependiendo de su estado. Ver en figura 3.13.

Una vez mencionada la metodología utilizada, y la herramienta de gestión donde se ha planificado el Trabajo de Fin de Grado, se hablará de en que han consistido las distintas fases en las que se ha dividido. Investigación, diseño, implementación y validación.

3.5.1. Investigación

Una vez definidos los objetivos principales del proyecto, en esta primera parte se dedicó el tiempo a investigar acerca del mundo de los bots conversacionales, probando las diferentes tecnologías y realizando estudios de cuál era la solución que más se adaptaba a estos objetivos propuestos.

Se realizaron estudios a fondo de las diferentes APIs existentes y las plataformas a las que podía ir dedicado el bot conversacional. Las principales tareas abordadas en esta fase se pueden observar en la tabla 3.1.

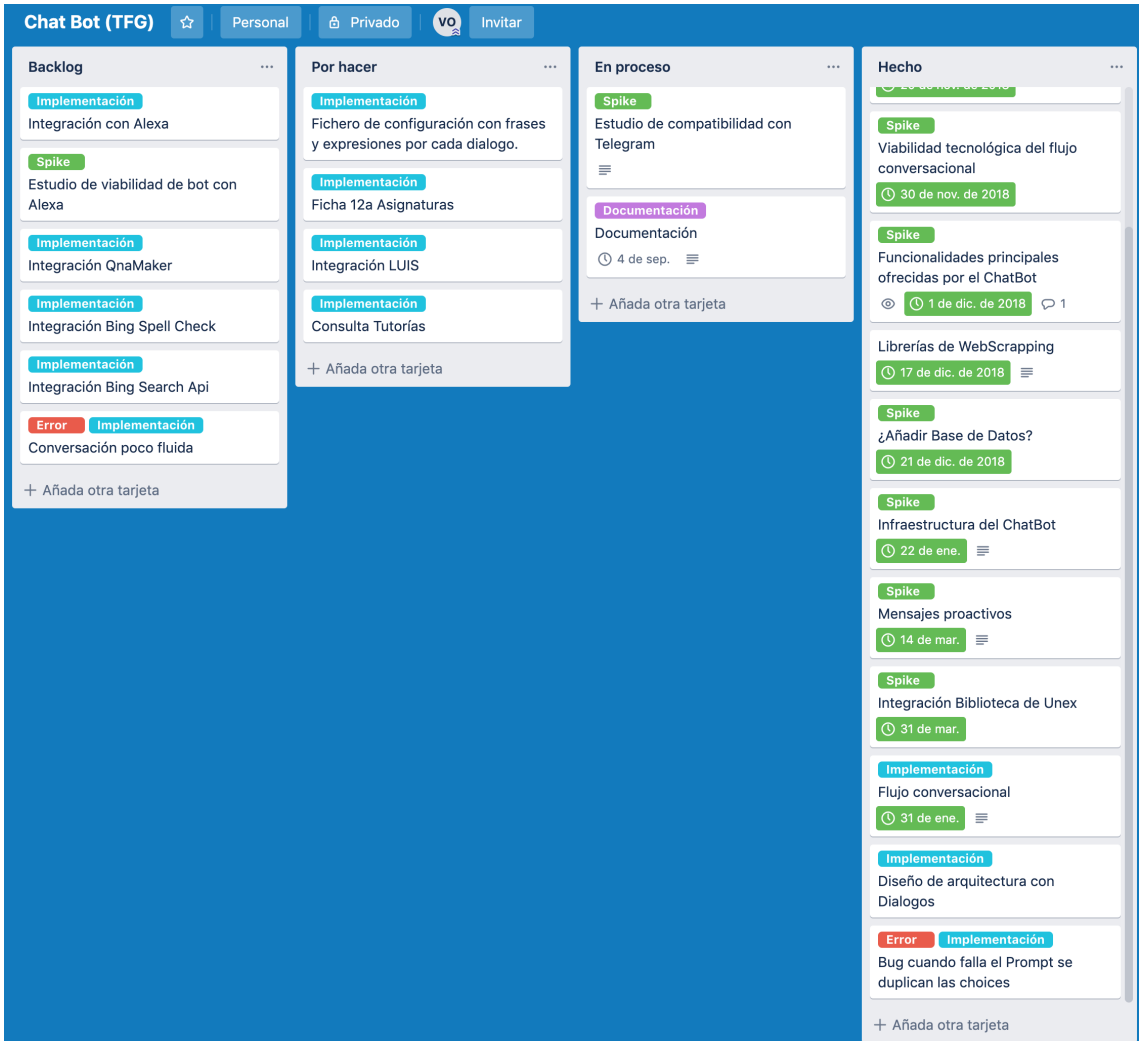


Figura 3.13: Ejemplo de tablero durante la fase de implementación.

3.5.2. Diseño

Tras haber finalizado la fase de investigación y saber que tecnologías se van a utilizar en el desarrollo del Trabajo de Fin de Grado, empieza la fase de diseño, donde se han decidido alguno de los aspectos técnicos que van a definir la funcionalidad de forma muy general, realizando diagramas de los componentes elegidos en la anterior fase, y guiándolos a su futura implementación, que se realizará siguiente fase. Las tareas que se han abordado en esta pueden apreciarse en la tabla 3.2.

Tarea	Duración
Estudio de los diferentes frameworks disponibles del chatbot	15 horas
Estudio de las diferentes tecnologías de flujo conversacional	15 horas
Funcionalidades principales que ofrece el Bot	5 horas
Librerías de Web Scraping	5 horas
Viabilidad con plataforma Alexa	15 horas
Estudio de servicios cognitivos y herramientas de PNL	20 horas
Estudio de las diferentes fuentes de información disponibles	15 horas
Estudio de los diferentes objetos procedentes de las fuentes de información	5 horas
Creación de diferentes bots para estudiar su comportamiento	15 horas
Creación de diferentes flujos conversacionales a modo comparativo	15 horas
Creación de Skill de Alexa para estudiar su comportamiento	15 horas
Comparativa con ejemplos de los diferentes servicios cognitivos elegidos	15 horas
Estudio de alojamiento del bot conversacional y sus dependencias	10 horas
-	165 horas

Tabla 3.1: Tabla con las principales tareas de la fase de investigación.

3.5.3. Implementación

En esta tercera fase se realiza todo el desarrollo de la estructura del bot, integrando las principales funcionalidades. Esta fase puede ir solapada con la anterior, puesto que la sencillez de algunas tareas hace que se puedan hacer a la vez. También es necesario, que todos los desarrollos estén completamente validados, para poder garantizar que se han cumplido todos los objetivos propuestos del bot conversacional.

Las principales tareas abordadas en esta fase, ordenadas cronológicamente se pueden ver en la siguiente tabla 3.3.

3.5.4. Resumen global

La suma del número de horas dedicadas en cada fase del proyecto no sería suficiente para conocer el tiempo real que se ha dedicado al proyecto, ya que de forma paralela a estas fases, se ha dedicado tiempo a la escritura de la memoria. A continuación, en la tabla 3.4 se puede observar el computo global del número de horas empleadas para la realización del Trabajo de Fin de Grado.

Tarea	Duración
Diseño de la principal arquitectura del bot	25 horas
Diseño de las distintas acciones que podrá realizar el Bot	15 horas
Diseño de los comportamientos que tendrá el bot para cada comando	10 horas
Diseño de los diferentes módulos internos del bot	10 horas
Diseño del flujo conversacional basado en diálogos	25 horas
Diseño de los diferentes comportamientos fuera del marco definido	15 horas
Diseño de la estructura de los datos y fuentes de información	15 horas
-	115 horas

Tabla 3.2: Tabla con las principales tareas de la fase de diseño.

Tarea	Duración
Desarrollo del framework del chatbot	15 horas
Implementación de módulos internos	15 horas
Alojamiento del bot en Azure	15 horas
Conexión del chatbot con los servicios cognitivos	15 horas
Creación de los comandos que puede realizar el bot	5 horas
Implementación del flujo conversacional	5 horas
Integración con las diferentes plataformas disponibles	15 horas
Creación de una API que hace de puente entre Alexa y el Bot	15 horas
Implementación de los servicios de recolección de datos	15 horas
Proveer con datos los servicios cognitivos a partir de las pruebas	10 horas
-	125 horas

Tabla 3.3: Tabla con las principales tareas de la fase de implementación.

Fase	Duración
Investigación	165 horas
Diseño	115 horas
Implementación	125 horas
Memoria	50 horas
-	455 horas

Tabla 3.4: Tabla con el computo global de horas del Trabajo de Fin de Grado.

Capítulo 4

Manual del desarrollador

En este capítulo se explicará detalladamente los principales aspectos técnicos del desarrollo del chatbot.

4.1. Arquitectura

La principal arquitectura del chatbot se ha realizado siguiendo algunos de los principios SOLID, realizando una arquitectura lo más limpia posible, evitando acoplarnos a librerías y haciéndolas fácilmente sustituibles, haciendo todo el código testeable y totalmente desacoplado de la parte de interfaz de usuario. Las clases implicadas en este desarrollo se pueden observar en el diagrama de clases de la siguiente figura 4.1.

Los diferentes elementos que componen esta arquitectura y su conexión entre ellos, se pueden observar en la figura 4.2. A continuación, se hablará en detalle de estos componentes.

4.1.1. Flujo conversacional

La principal arquitectura del flujo conversacional está basada en diálogos, como se ha mencionado anteriormente, en la siguiente figura 4.3 se pueden apreciar los diferentes diálogos que la componen.

El diálogo inicial del bot conversacional y el encargado de orquestar los diferentes diálogos, es MainLuisDialog. Este diálogo hace uso del servicio de LUIS, enviando el mensaje recibido del usuario a este servicio, para que LUIS pueda realizar un análisis de la expresión , y devolver el ”intent” que más puntuación obtenga. Está intención siempre estará categorizada en el código fuente por lo que provocará la ejecución del método ReplaceDialogByIntent() donde se cambiará completamente de diálogo manteniendo el contexto del diálogo de donde partíamos.

Cada diálogo tiene definido diferentes pasos, por lo que los podemos dividir en dos categorías: simples y compuestos.

Los diálogos simples, son los mas sencillos, puesto que solo tienen interacción con el fichero de lenguajes, algunos de estos diálogos son, GratitudeDialog, GoodByeDialog, HelpDialog, etc. Estos diálogos no hacen uso de la IA, ya que el análisis lo ha realizado previamente MainLuisDialog, y gracias a ese análisis, el dialogo se ha bifurcado por esta rama.

Los diálogos más complejos son los que tienen más de un paso, y al igual que los simples, son principalmente invocados por MainLuisDialog, pero a excepción de estos, desde estos diálogos también se pueden producir bifurcaciones, ya que también hacen uso de la IA. Dada la importancia de estos diálogos se explicará en detalles qué hace cada uno de ellos.

- MainLuisDialog, aunque ya se ha hablado de él, cabe destacar que cualquier error producido en la aplicación, o cualquier flujo de diálogo no definido, acabará en este diálogo, para que pueda quedarse a la espera del próximo mensaje del usuario y poder hacer la bifurcación correspondiente. Se puede ver su función en el siguiente diagrama 4.4.

- QuestionDialog, este diálogo hace uso de tres servicios principales, SpellCheckService, QnaMakerService y SearchService, el fin principal de este servicio es poder responder a cualquier pregunta relacionada con el ámbito de la Universidad de Extremadura.

- TeacherDialog, es el encargado de ofrecer la información principal de un profesor de la UEx, utiliza el servicio TeacherService que es el que le provee la información.

- UnexFacilitiesDialog, ofrece los servicios de la Universidad de Extremadura que están disponibles por categorías, haciendo uso del servicio UnexFacilitiesService.

- SubjectDialog, facilita la información relacionada con centros, asignaturas y grados de la Universidad de Extremadura, ofreciendo la Ficha docente de la asignatura en el caso de que esté disponible, haciendo uso del servicio SubjectService.

4.1.2. Servicios

En este apartado, se entrará más en detalle acerca de los servicios utilizados por los diálogos que se han mencionado anteriormente.

BingSearchService

Este servicio contiene un cliente de la API de Bing para realizar búsquedas, denominado WebSearchClient. Para utilizarlo se necesita añadir una clave que se puede obtener de la plataforma de azure.

El método GetResultFromSearch, extrae los mejores resultados del mensaje introducido por el usuario a través del motor de búsqueda Bing, pero realizando una serie de ajustes para afinar aún más esa búsqueda, añadiéndole ”site:unex.es”, la opción de búsqueda segura, y el idioma español, garantizando que la búsqueda es apta para cualquier persona.

La arquitectura de este servicio se puede ver en la figura 4.5.

SpellCheckService

El corrector ortográfico es utilizado para cualquier petición que se haga hacia los servicios cognitivos, garantizando que el mensaje enviado es apto para el lenguaje que se está utilizando. Para realizar la consulta a la API, es necesario utilizar el cliente SpellCheckClient, que al igual que BingSearchApi, necesita la clave del servicio contratado. La configuración elegida para realizar la corrección es muy simple, indicándole el idioma y el modo de corrección. Existen dos modos de corrección, ”proof” que es el más exhaustivo puesto qué es capaz de corregir errores gramaticales como las mayúsculas o minúsculas. El otro modo es el que he utilizado y es ”spell” que no es tan exhaustivo pero funciona bien para el lenguaje cotidiano.

La clases principales que implementan este servicio se pueden ver en la figura 4.6.

OpenDataService

Este servicio es el encargado de recoger la información acerca de los centros, grados y asignaturas, a partir del portal de Open Data. Tiene dependencia de un fichero de configuración denominado DegreeConfig.json, donde se almacenan las consultas SPARQL que se realizan hacia el portal de Open Data para la obtención de los resultados y también contiene códigos de los centros que usa la página web de la Universidad de Extremadura para diferenciarlos entre ellos. A continuación, se pude apreciar un ejemplo de la consulta SPAQL acortada y la asociación de códigos.

```
"SubjectsQuery": "http://opendata.unex.es/sparql?....&format=json",
"StudyCentres": {"OpenDataCode": 15,"UnexPageCode": "cum"}
```

Esta asociación se ha creado para la generación de URLs dinámicas de la UEx a partir de la información obtenida del portal de Open Data.

En un principio la arquitectura de este servicio era como aparece en la siguiente figura 4.7, pero para mejorar el tiempo de respuesta durante la conversación se hizo una implementación nueva que hace uso de una caché basándose en el diseño de un patrón proxy como se puede ver en la figura 4.13 y que se detallará más adelante.

TeacherService

La librería principal que se ha utilizado en este servicio es HtmlAgilityPack para realizar la recolecta de información de profesores a través de la página de la Universidad de Extremadura. Además, es necesario tener la URL de donde se sacarán los datos de los profesores. 4.8,

UnexFacilitiesService

Este servicio es muy parecido al servicio de profesores, utilizando la librería de HtmlAgilityPack, la única diferencia es que en este servicio, los servicios que ofrece la Universidad de Extremadura, están categorizados. Por lo que se hace un previo filtrado agrupando cada servicio, para que al ofrecérselo al usuario, pueda filtrar por una categoría u otra. Las clases principales de este servicio se pueden ver en la figura 4.9.

BotServices

En los inicios del SDK Bot Builder, tanto QnaMaker como LUIS venían integrados en el framework, por lo que la propia librería te ofrecía conectores y utilidades para usar estos servicios de una forma muy sencilla. Con el paso del tiempo, estos servicios se han ido sacando de esta librería para poder ser utilizado en otras aplicaciones. El principal propósito por el que lo he mantenido así era porque el propio framework lo adaptó para poder tener diversos servicios de LUIS o de QnaMaker, funcionando a la vez. Por eso, existen dos objetos públicos que son un diccionario clave-valor, siendo el valor una instancia de cada uno de ellos. Por motivos de tiempo y de limitaciones con la subscripción de estos servicios, no se han creado más servicios de IA, pero habría sido interesante, usar un LUIS por cada diálogo, entendiendo exactamente las expresiones asociadas a cada uno de ellos.

La clase principal de este servicio se puede ver en la figura 4.10.

4.1.3. Función Lambda de Amazon Web Services

La conexión entre Alexa y NexoBot se ha realizado con una función Lambda que hace de intermediario entre ambos. En la figura 4.11 están las clases implicadas en este desarrollo.

El punto de comunicación con Alexa se encuentra en la clase Function y más en concreto en el método FunctionHandler, que recibe un objeto de tipo ”SkillRequest” y devuelve otro de tipo ”SkillResponse”. Siendo estos objetos totalmente compatibles con la API de Alexa.

En el objeto SkillRequest viene la información del intent[1] que ha reconocido Alexa. Para este desarrollo a excepción de los intents obligatorios de Alexa (cancel, stop y help) siempre se utiliza el mismo intent en la comunicación con el bot.

Cuando la función Lambda detecta una solicitud con este intent se ejecuta el método GetPhrase() enviando el mensaje de Alexa recibido en el objeto SkillRequest al bot. Este método realiza la comunicación con el bot a través de la clase DirectClientService.

En la primera comunicación entre la función Lambda y el bot es necesario iniciar la conversación. Para esto es necesario realizar una solicitud enviando en la cabecera la clave secreta de DirectLine configurada para el bot. Cuando el bot reciba una solicitud de este tipo, devolverá un identificador de la conversación que se utilizará durante todo el flujo conversacional entre el usuario, Alexa, y el bot.

La clave secreta debe viajar en la cabecera en todas las peticiones que se realicen entre ambas partes.

Para el envío de mensajes de Alexa hacia el bot se utiliza la función SendMessageAsync, cuyos parámetros son el identificador de la conversación y el mensaje a enviar. Y para la recepción de mensajes se utiliza la función GetMessagesAsync, también con el parámetro del id de la conversación, y una marca de agua para garantizar que no se han perdido mensajes en el camino, aunque este último es opcional.

[1] intención configurada en la plataforma de Alexa para tipificar el comando de voz utilizado por el usuario.

Es importante tener en cuenta, que esta función Lambda no tiene lógica ni realiza tratamiento de datos, solo se dedica a comunicar o proveer datos válidos entre dos clientes cuyos canales de comunicación eran incompatibles. Cualquier error tanto controlado como no controlado en la parte del bot, será transformado en un mensaje de texto que será devuelto a la función Lambda la cual lo enviará a Alexa. A continuación, se hablará más en detalle de esta gestión de errores.

4.1.4. Manejo de excepciones

La gestión de errores del chatbot se ha realizado implementando un handler encargado de manejar las peticiones HTTP del bot. La librería Microsoft.Bot.Builder tiene una serie de adaptadores que interceptan todas estas peticiones. Para su desarrollo se ha creado una clase denominada ”AdapterWithErrorHandler” que hereda de la clase ”BotFrameworkHttpAdapter”. Esta clase creada lo único que necesita es un constructor, donde se le indica el comportamiento que tiene que seguir cada petición.

El comportamiento por defecto que se ha modificado es la función ”OnTurnError”, que se ejecuta únicamente cuando se produce un error no controlado en el flujo de la conversación. El comportamiento que se le ha dado a está función es mostrar un mensaje de error del tipo ”Algo ha ido mal.” haciendo uso de la librería de LanguageGeneration mencionada en anteriores puntos. Además, se eliminará toda la información de la conversación y se iniciará de nuevo la conversación con el usuario.

4.1.5. Configuración

Para poder ejecutar algunas serie de acciones y poder recolectar datos de forma correcta, es necesario indicar al bot algunos parámetros de configuración.

En .NET Core existe un inyector de dependencias incorporado que utiliza un contenedor denominado ServiceCollection para registrar los servicios que se van a utilizar. La forma más sencilla de realizar una inyección de dependencia en .NET Core es de la siguiente forma

```
services.AddSingleton<ISearchService, BingSearchService>()
```

agregando al contendor de servicios la instancia, provocando que para cualquier uso de la interfaz ISearchService, sea resuelta en tiempo de ejecución con la instancia de la clase BingSearchService.

Para seguir con la forma en la que se registran los servicios, y evitar la lectura constante del fichero, que con los ficheros de configuración asociados a un modelo de datos no se puede realizar, puesto que no existe una función de añadir al colector de servicios la dupla Object y String, dónde Object es el objeto del modelo de datos, y String es la ruta del fichero de configuración. Se ha implementado un método de extensión de ServiceCollection denominado AddConfiguration(), para registrar cada fichero de configuración en la colección de servicios, siguiendo la dupla mencionada anteriormente. A continuación, se puede ver un ejemplo de como queda la nueva función de extensión creada.

```
services.AddConfiguration(ConfType.DegreeConfig, "opendata.json");
services.Configure<DegreeConfigModel>(
            services.GetConfiguration(ConfType.DegreeConfig)
                                    );
```

Con esto nos garantizamos una única lectura del fichero, y poder utilizarlo en cualquier ámbito de la aplicación ya que el colector de servicios está siempre disponible. Para su uso, solo habría que utilizar la interfaz IOptions<T> que es la utilizada por .NET core para almacenar la configuración principal de la aplicación.

```
protected readonly DegreeConfigModel _degreeConfigModel;


public OpenDataService(IOptions<DegreeConfigModel> degreeConfigModel)
{
    _degreeConfigModel = degreeConfigModel.Value;
}
```

Algunos de los datos que se han almacenado en estos ficheros de configuración son

las URLs absolutas de la Universidad de Extremadura con alguna modificación para generar de forma dinámica los enlaces de información de centros, profesores, o incluso ficha de asignaturas.

Un ejemplo de estas URLs es el siguiente, " https://www.unex.es/conoce-la-uex/centros/0 " siendo 0 el código del centro del que se quiere generar el enlace.

4.2. Diseño e implementación

En esta sección se presentan los principales patrones de diseño utilizados en el proyecto.

4.2.1. Singleton

Este patrón de diseño, muy utilizado en el desarrollo de software, se ha aplicado en el proyecto en diferentes escenarios, siendo el aplicado a ficheros de configuración del chatbot el escenario clave de uso de este patrón, ya que con su uso, una vez arrancada la aplicación, se tiene una única instancia del los ficheros de configuración. Todos los usuarios que utilicen el bot y que realicen una conversación dónde es necesario leer algún fichero de configuración, no será necesaria la creación por parte del servidor de una nueva instancia para leer los datos del fichero de configuración, dado que la instancia de ese modelo de datos se creo en el inicio de la aplicación y es compartida por todos los usuarios.

En la figura 4.12 se puede observar el diagrama del patrón singleton aplicado a este escenario.

4.2.2. Proxy

Por el uso de las APIs de Telegram y Alexa en el desarrollo de la aplicación, surgieron una serie de problemas que había que cubrir, uno de ellos era la obligación de realizar solicitudes con un tiempo de respuesta inferior a 8 segundos. Debido a esto,

surgió la necesidad de realizar una caché para evitar hacer una sobrecarga del servidor y poder controlar todas las peticiones obteniendo primero el objeto de la cache si está disponible y en el caso de que no lo esté se realizaría la petición contra el servidor.

Para este desarrollo se tomó la decisión de usar un patrón de diseño denominado ”proxy”, que fue aplicado para todas las peticiones que se realizan en el portal de Open Data, donde tenemos la mayor cantidad de datos.

En la figura 4.13 podemos ver el sencillo diagrama de clases de la implementación de este patrón.

4.2.3. Strategy

Todos los servicios creados a excepción de BotServices, están adaptados para el uso de este patrón de diseño. Con el uso de este patrón se pueden realizar inyecciones de dependencias para definir comportamiento. Estos servicios se han realizado de esta forma, para que si en un futuro, la fuente de información de los datos de uno de los servicios cambia, solo sea necesario crear una nueva implementación de ese servicio recogiendo los nuevos datos, garantizando que los modelos de datos que utiliza el bot no se han cambiado.

4.3. Guía de despliegue

En este apartado se hablará de los pasos a seguir para la puesta en marcha del bot conversacional.

4.3.1. Requisitos previos

En primer lugar es necesario cumplir una serie de requisitos.

- Utilizar un sistema operativo Windows/Linux.
- Tener instalado el SDK de .Net Core 2.2
- Tener instalado BotFramework-Emulator para la realización de las pruebas.

- Tener cuenta de Azure donde estarán los servicios cognitivos utilizados. (Opcional)

- Versión de Visual Studio 2015 o superior.

4.3.2. Configuración

Una vez cumplidos estos requisitos, se puede descargar el código fuente del repositorio[2] desde Visual Studio utilizando la rama de desarrollo.

Como la configuración del bot actual está asociada a mi cuenta de Azure, si el desarrollador no quiere realizar modificaciones de los servicios cognitivos utilizados, no será necesario modificar la configuración. En el caso de que quiera hacer uso de sus servicios cognitivos, tendrá que realizar modificaciones en los ficheros de configuración, para poder utilizar sus servicios en lugar de los creados por mi inicialmente.

El fichero de configuración donde se encuentra esta información es BotConfiguration.bot que está en la raíz del proyecto. A continuación, se puede visualizar el estado actual de mi configuración en este fichero para el servicio de LUIS y de QnaMaker.


```
{
    "type": "luis",
    "appId": "dee10eee-de9e-4faf-83e1-6ad79ff985ed",
    "authoringKey": "0bdc84c3b1a64a9e82aeb967fac51b45",
    "id": "2",
    "name": "HelpService",
    "region": "westus",
    "subscriptionKey": "0bdc84c3b1a64a9e82aeb967fac51b45",
    "version": "0.1"
```


[2] https://github.com/vicentegnz/ChatBOT

```
},
{
    "type": "qna",
    "endpointKey": "df338b6a-6d76-4168-9780-09e64fd83fbf",
    "hostname": "https://nexo-qnamaker.azurewebsites.net/qnamaker",
    "id": "3",
    "kbId": "4015c6bb-b141-438f-9989-d41c7314b841",
    "name": "FrequentlyAskedQuestions",
    "region": "westus"
}
```

4.3.3. Ejecución

Una vez completada toda la configuración de los servicios cognitivos. Se podrá arrancar la aplicación. Si todo ha ido bien, se deberá visualizar la página de principal del Bot como se puede apreciar en la figura 4.14 y donde aparece un enlace de descarga del emulador que se necesita para poder probar la funcionalidad del bot. Una vez descargado e instalado el emulador, se debe utilizar la opción de abrir bot a partir de un fichero de configuración, y buscamos en el directorio del ChatBot el fichero BotConfiguration.bot (Ver en figura 4.15). En algunas ocasiones es necesario introducir el AppId y el AppKey del ChatBot, puesto que el emulador no lo obtiene por defecto. Estas credenciales se encuentran en el mismo fichero de configuración.

Si todo se ha realizado correctamente, se podría iniciar una conversación con el bot desde el emulador.

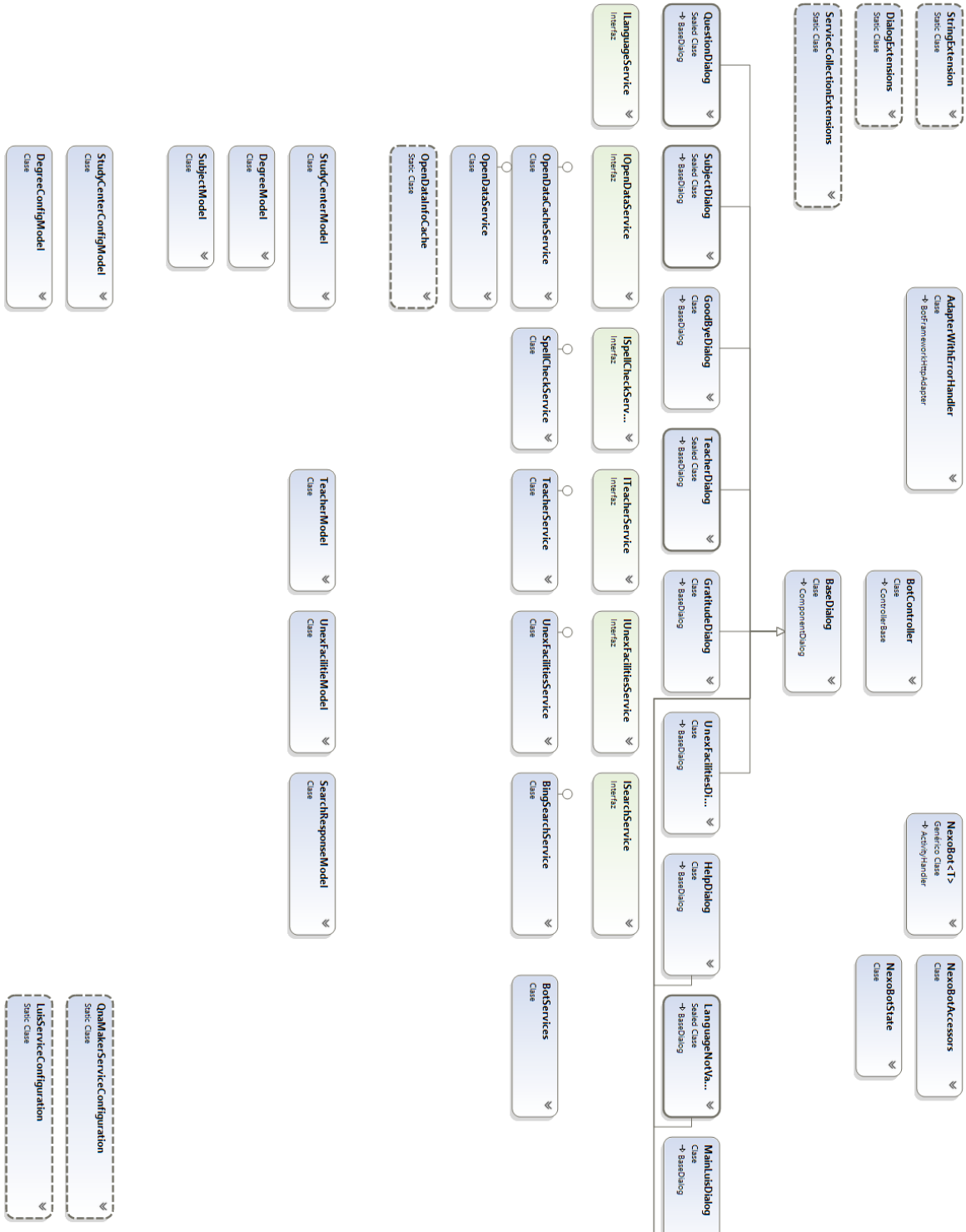


Figura 4.1: Diagrama de clases del ChatBot

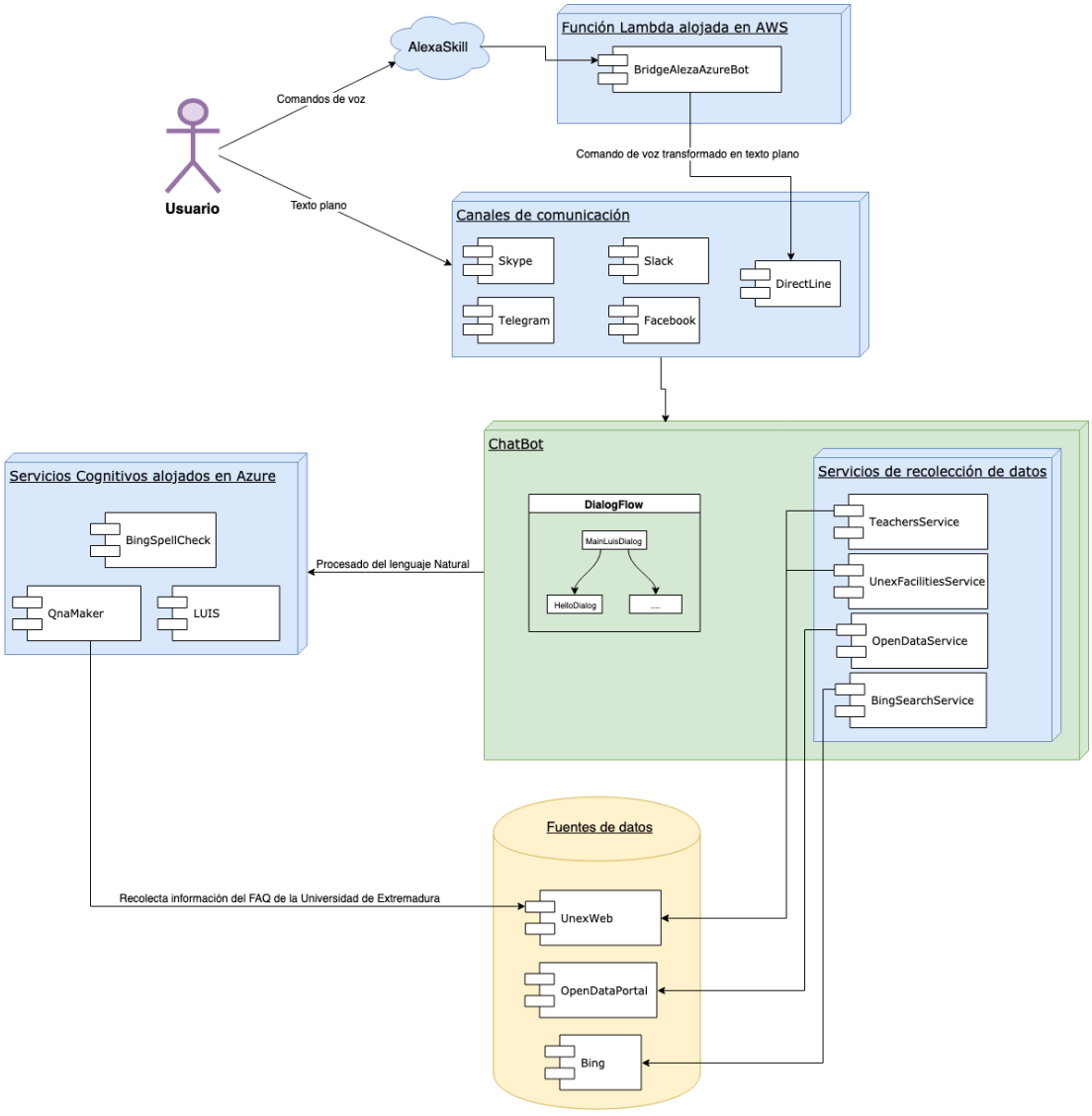


Figura 4.2: Diagrama de componentes del ChatBot

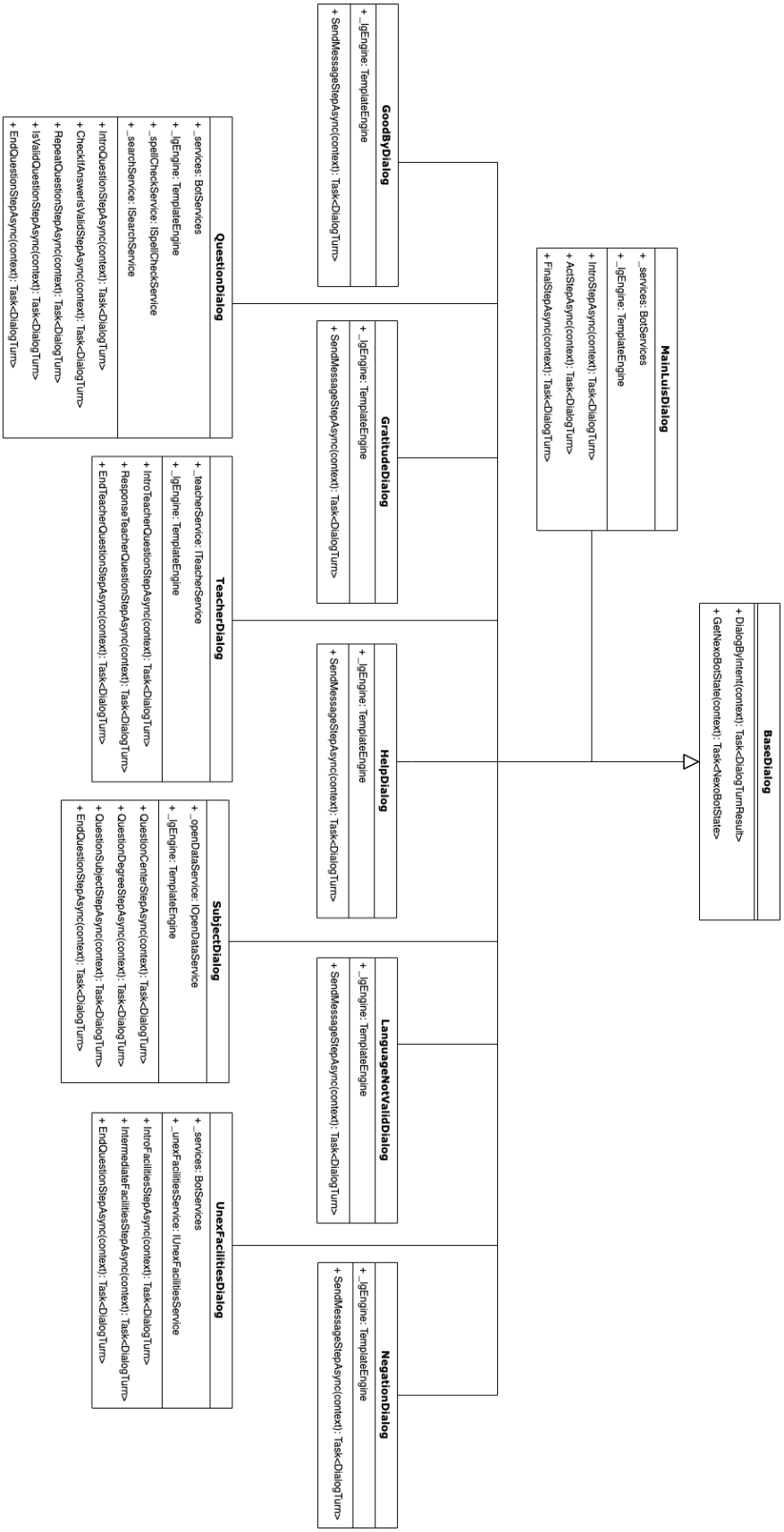


Figura 4.3: Arquitectura con los diferentes diálogos del bot.

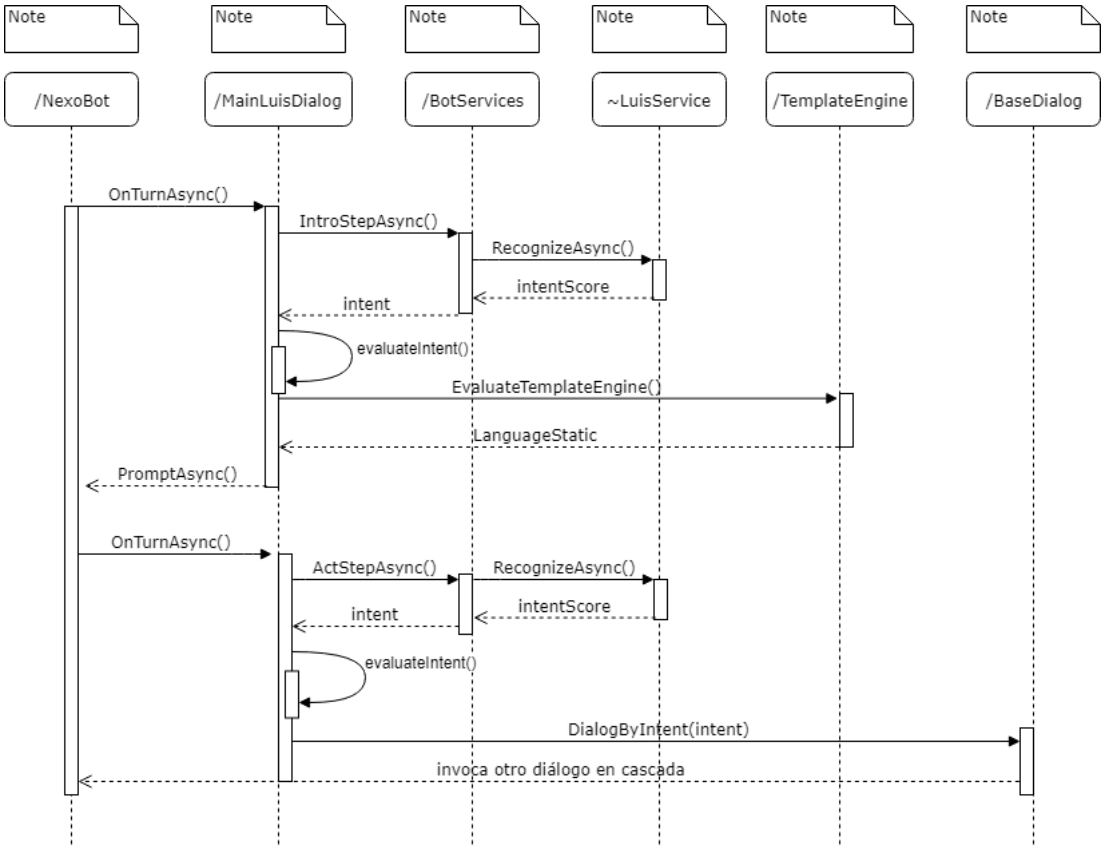


Figura 4.4: Diagrama de secuencias del diálogo principal.

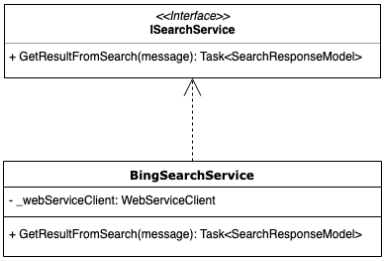


Figura 4.5: Arquitectura del servicio de búsqueda.

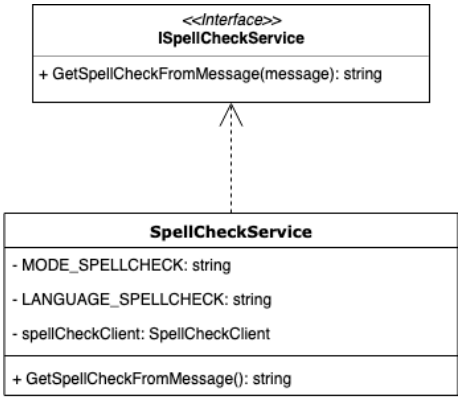


Figura 4.6: Clases del servicio de corrección ortográfica.

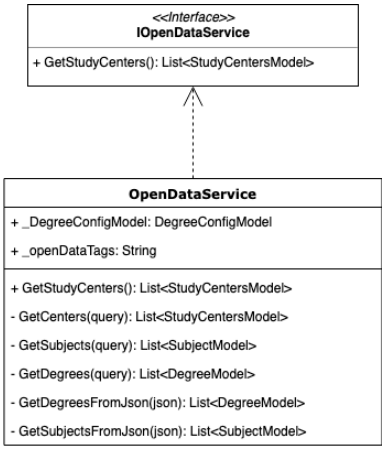


Figura 4.7: Clases del servicio de recolección de datos del portal de Open Data.

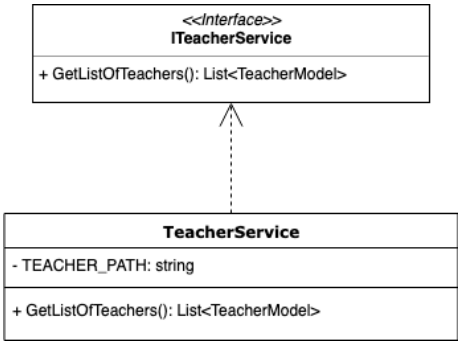


Figura 4.8: Clases del servicio que recolecta la información de profesores de la UEx.

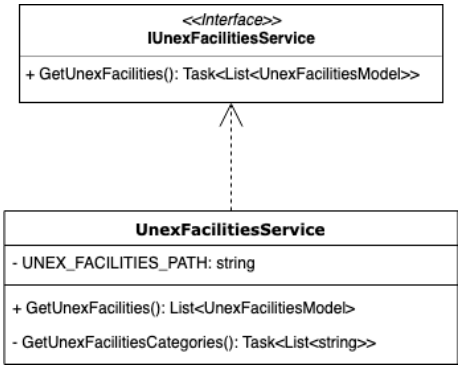


Figura 4.9: Clases del servicio que recolecta la información de los servicios diponibles en la UNEX.

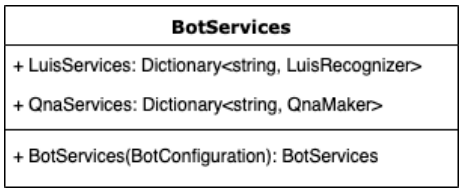


Figura 4.10: Clase principal de la colección de servicios de IA del bot.

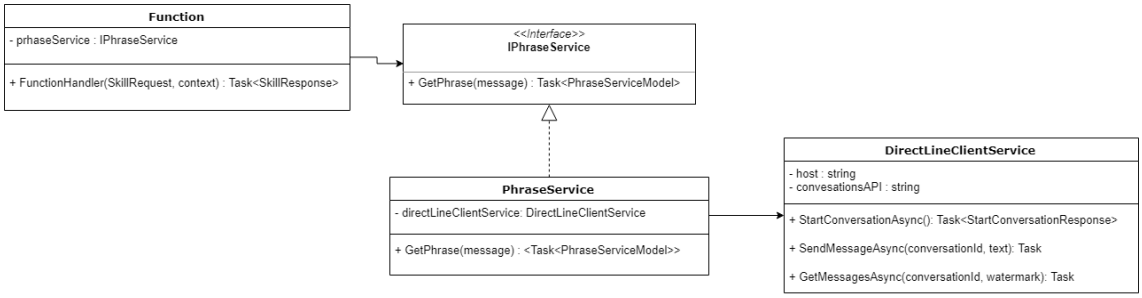


Figura 4.11: Clases implicadas en la función Lambda entre Alexa y el bot.

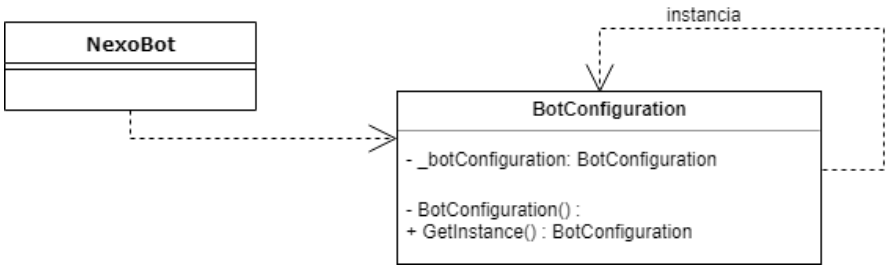


Figura 4.12: Diagrama de clases patrón singleton.

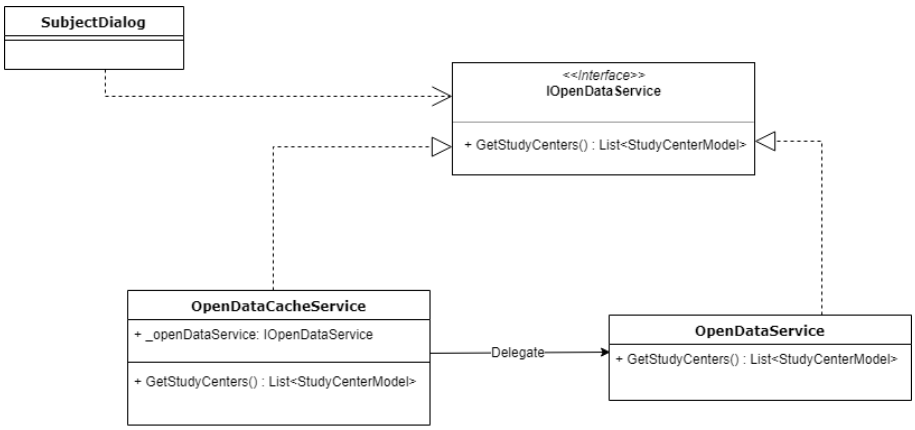


Figura 4.13: Diagrama de clases patrón proxy

Figura 4.14: Página principal de inicio de NexoBot.

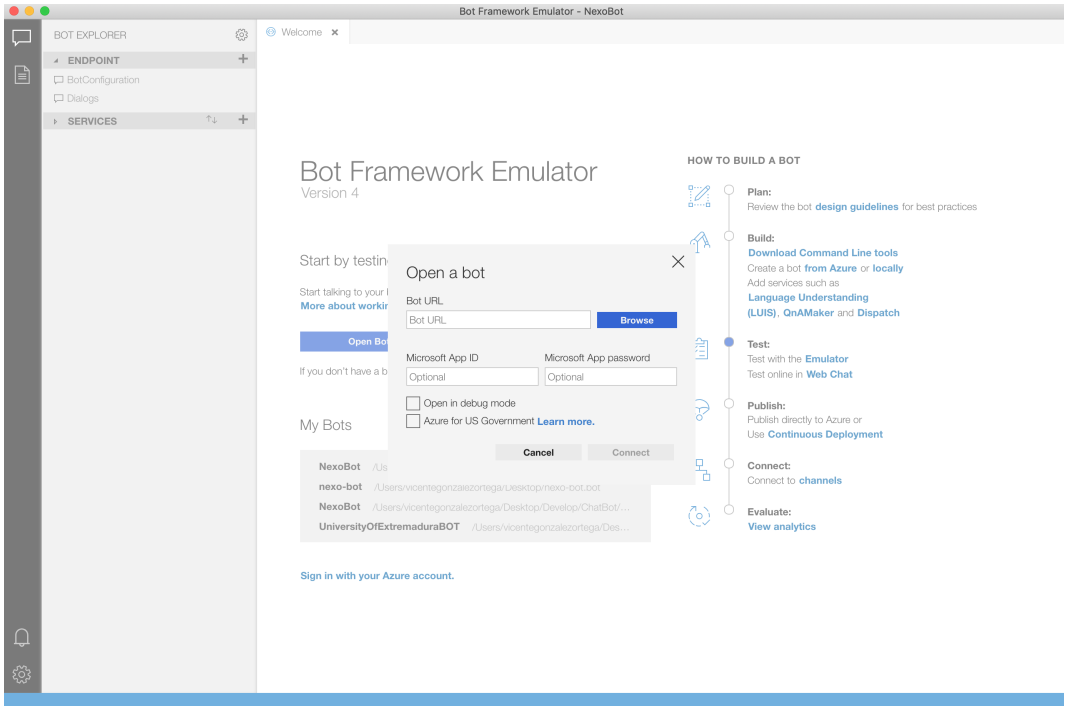


Figura 4.15: Emulador utilizado para las pruebas del chatbot.

Capítulo 5

Manual de usuario

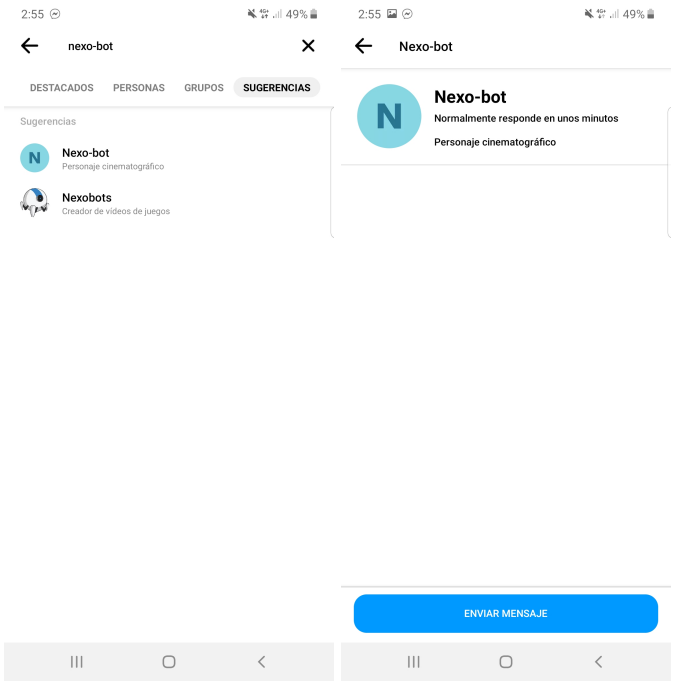
En este capítulo se explicará como utilizar el bot conversacional implementado en este Trabajo de Fin de Grado. La usabilidad y las funciones principales pueden estar limitadas dependiendo de la plataforma elegida, por lo que, a continuación se explicará detalladamente como utilizar NexoBot en cada una de estas plataformas.

5.1. Facebook

El uso de NexoBot en la plataforma de Facebook es muy simple, solo es necesario hacer uso de Facebook Messenger y realizar una búsqueda por el nombre del bot, que en este caso es ”Nexo-Bot”. Ver en figura 5.1. Una vez encontrado el bot, solo es necesario iniciar una conversación. La conversación se puede iniciar de cualquier forma, por lo que los usuarios son libres de preguntar lo que deseen. En la figura 5.2 se muestra un ejemplo de una conversación realizada en esta plataforma.

5.2. Telegram

Para utilizar NexoBot en la plataforma de Telegram, los pasos a seguir son muy similares a los de Facebook. En primer lugar se hace una búsqueda del bot por el nombre ”neeXoBot”. Ver en figura 5.3.



(a) Búsqueda de Nexo-Bot. (b) Inicio de conversación.

Figura 5.1: Ejemplo de búsqueda e inicio del bot en Facebook Messenger.

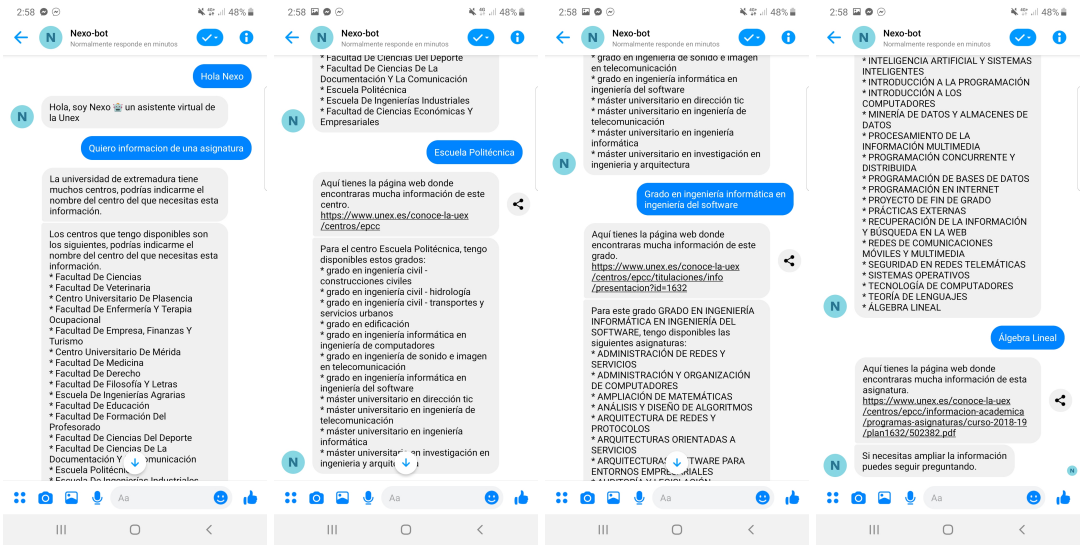
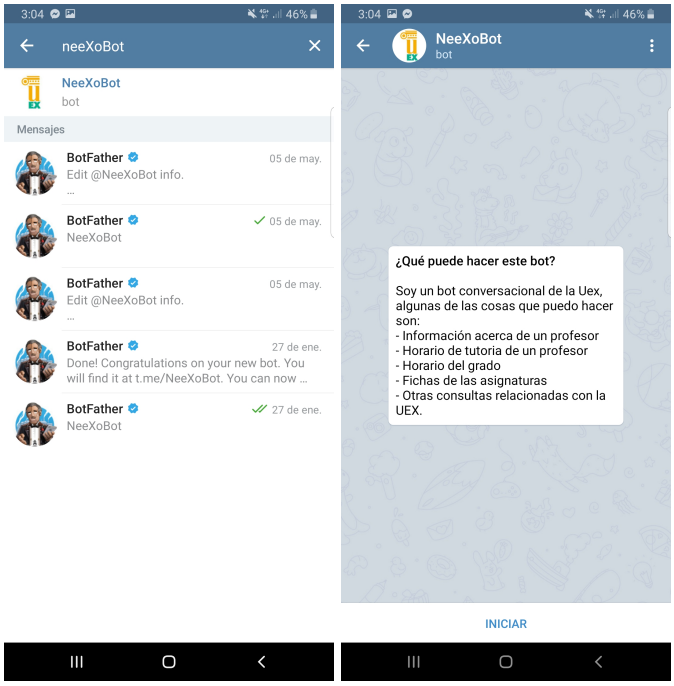


Figura 5.2: Ejemplo real de diálogo en cascada del bot conversacional en Facebook.



(a) Búsqueda de Nexo-Bot. (b) Inicio de conversación.

Figura 5.3: Ejemplo de búsqueda e inicio del bot en Telegram.

En este tipo de plataforma, las funcionalidades son exactamente las mismas que para la plataforma de Facebook Messenger. Para esta plataforma se ha decidido realizar un diálogo diferente al ejemplo de la plataforma anterior, para que se puedan observar las diversas formas que existen de interactuar con el bot. Ver figura 5.4.

5.3. Alexa

Para el uso del bot en esta plataforma, es necesario tener permisos de usuario de pruebas en el correo vinculado a los dispositivos de Alexa en los que se quiera utilizar el bot. Este tipo de permisos se asignan desde el panel de desarolladores de Amazon Alexa (*aSK*). Una vez estén asignados los permisos, habría que dirigirse a la tienda de Skills y buscar ”Universidad de Extremadura” para realizar su activación. Cuando la skill se encuentre activa (ver figura 5.5), habrá que abrir la skill conversando con Alexa, viendo cual es la oración que inicia esa skill. En las pruebas siempre se ha utilizado ”Abrir Universidad de Extremadura”.

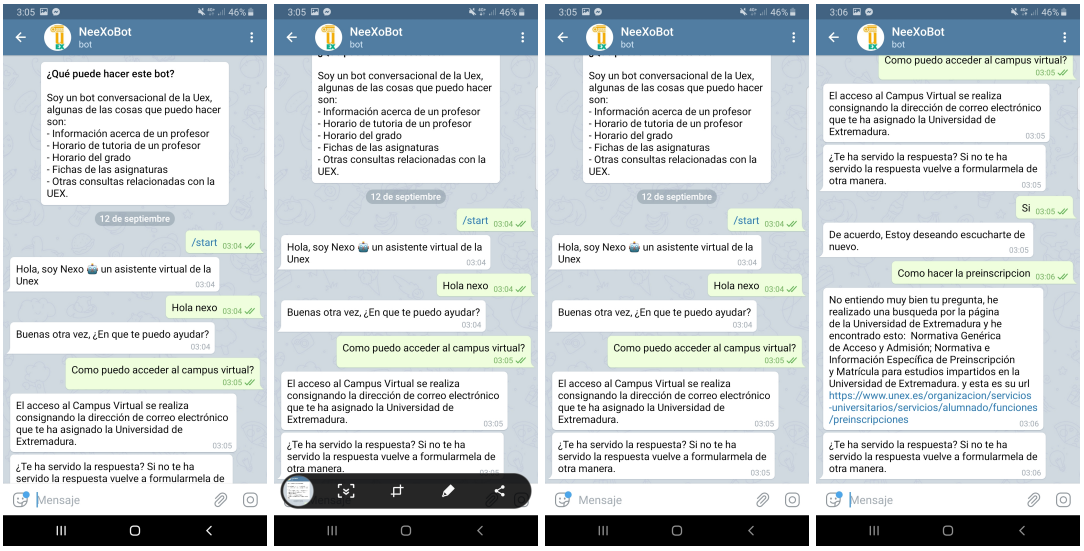


Figura 5.4: Ejemplo real de diálogo en cascada del bot conversacional en Telegram.

Hay que tener en cuenta que en este tipo de plataforma existen algunas limitaciones de algunas características del bot. Un caso de estas limitaciones es en el envío de links, que en anteriores plataformas, solo es necesario pulsar el enlace para poder acceder a la página web. Alexa, solo se encargará de mencionarla, pero dependiendo del dispositivo, existirán casos en los que no se pueda acceder, y la información no sea de utilidad.

El principal propósito de integrarlo con esta plataforma, era la de ofrecer diferentes posibilidades, y evitar que siempre la comunicación se realice mediante texto.

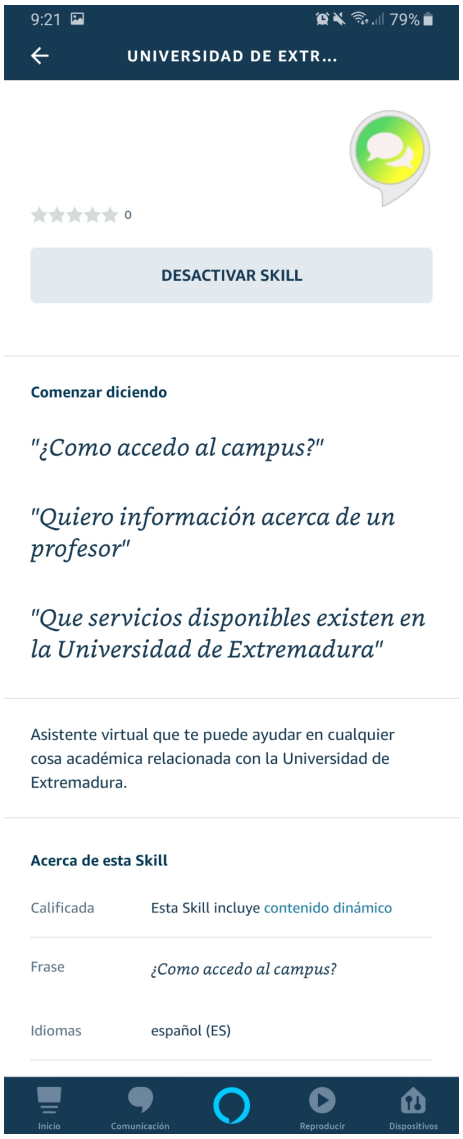


Figura 5.5: Skill activa de la Universidad de Extremadura en Alexa.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

Este proyecto partió de la idea de ofrecer un servicio de información extra al alumno de la Universidad de Extremadura, utilizando las nuevas tecnologías, y haciendo uso de herramientas o aplicaciones que estén al alcance del alumno, como es la mensajería instantánea. Durante el desarrollo, los requisitos y las decisiones que se iban tomando en la fase de diseño, hacían cambiar constantemente las posibilidades que el servicio de información podía ofrecer. A pesar de esto, se puede decir que el resultado del proyecto ha sido satisfactorio.

Son varios los objetivos marcados para el desarrollo del proyecto, algunos de ellos tienen más importancia, ya que son vitales para el funcionamiento del bot. Estos objetivos han sido realizados con éxito y entre ellos se encuentran:

- Creación del bot conversacional que pueda responder a preguntas del FAQ de la Universidad de Extremadura.
- Integración con diferentes plataformas, tanto de texto, como de comandos de voz.
- Recolección de datos de distintas fuentes de información.
- Flujo conversaciónal realista gracias a tecnologías de IA y herramientas de PLN.

En el alcance inicial del proyecto, apreció la idea de conseguir que la aplicación sea totalmente autónoma, pero no se ha podido conseguir plenamente para uno de los servicios cognitivos utilizado. Para el servicio de QnaMaker se ha podido lograr, actualizando automáticamente la información recolectada del FAQ de la UNEX, pero para LUIS es necesario hacer uso de la herramienta online si queremos añadir nuevas oraciones o frases que procesen la información y lo transformen en comandos del bot, por lo qué no se puede dar por cumplido el objetivo.

Cuando comencé con el desarrollo de este proyecto, mis conocimientos acerca de estas tecnologías era mínimo, puesto que la mayoría de ellas no las había usado nunca a lo largo de mis estudios. En ocasiones esto complicaba las cosas, pero una vez obtenido resultados era muy gratificante, ya que no solo el aprendizaje era de las tecnologías usadas si no qué, también era de las tecnologías que hemos probado para hacer las comparaciones y elegir la que más se adapta a los objetivos propuestos.

Una de las ideas principales que más peso ha tenido y que más valor me ha aportado es hacer uso de la IA para que el bot conversacional, fuera algo más que una máquina. Con el uso de esta tecnología aparecieron técnicas como el procesado del lenguaje natural que empezaron a tener más relevancia en el proyecto y que cuanto más avances tenía mi Trabajo de Fin de Grado, más consciente era del potencial que tenían estás tecnologías, y que estoy seguro que será la tendencia principal en las empresas de hoy en día, tanto para realizar tareas administrativas, como para dar asistencia haciendo un enfoque similar al de mi Trabajo de Fin de Grado.

En un principio las ideas iniciales del proyecto eran más ambiciosas de lo que se ha conseguido implementar, debido a la limitación de tiempo, y a la labor de investigación que se ha tenido que hacer para llevar a cabo la implementación, pero a pesar de esto, se ha dejado una puerta abierta a futuros trabajos después de conocer a fondo las posibilidades que ofrecen todas las tecnologías utilizadas.

6.2. Trabajos futuros

En este último apartado se comentarán algunas mejoras que se podrían ir añadiendo al bot en un futuro, algunas de estas mejoras han ido surgiendo durante la realización del proyecto, pero por motivos técnicos o de tiempo no se han podido llevar a cabo.

- Integración del bot en la página web de la Universidad de Extremadura a modo de asistente virtual, respondiendo a las mismas consultas que responde por los otros medios de mensajería instantánea. Para realizarlo, solo habría que realizar la interfaz gráfica, y para la funcionalidad sería realizar las solicitudes a la API de ”Direct Client” del bot.

- Integración del bot con toda la información disponible en el Aula Virtual de cada alumno, pudiendo guardar información relevante que ayude al bot a ser más intuitivo y rápido en la búsqueda de respuestas del usuario. Se podría almacenar en las cookies información particular del usuario, como por ejemplo las asignaturas matriculadas, o también añadir un modo recordatorio en el bot que sin iniciar la conversación pueda notificar fechas próximas a exámenes o trabajos, o incluso notificaciones de notas publicadas en el tablón, para las asignaturas matriculadas.

- Permitir al bot acceder a la información de la biblioteca de la Universidad de Extremadura `http://lope.unex.es/` para facilitar la consulta la disponibilidad de libros, y poder realizar reservas.

- Si el número de datos comienza a crecer, utilizar una base de datos que será enriquecida con un proceso de BackEnd que recolecte todos los datos cuando estén disponibles. Podría hacerse en una hora nocturna durante todos los días. De este modo el bot solo consultaría la base de datos, la cual siempre tendrá datos disponibles, ya que si falla el proceso de recolección de datos, los anteriores datos siguen en la base de datos, y con esto se garantiza alta disponibilidad.

- Utilizar todas las posibilidades que ofrecen las tecnologías cognitivas utilizadas, ya que algunas de ellas durante la realización del proyecto han ido evolucionando muy rápidamente. En este sentido, algunas de estas tecnologías no han podido ser aprovechadas al máximo debido a que algunas de ellas han salido en la fase final del desarrollo del proyecto, pero después de haber leído todas estas nuevas funcionalidades se puede garantizar que mejorarán notablemente las conversaciones entre el chatbot y el usuario.

- Una funcionalidad extra muy interesante, podría ser la de gestión de tutorías, en la que el chatbot le muestra los horarios disponibles al alumno, el alumno elige el horario que mejor le conviene, y el chatbot le envía un mensaje con toda la información al profesor para que confirme la cita, en el caso de el profesor confirme la cita, el chatbot le contesta al alumno confirmando que el profesor ha aceptado la tutoría. Para su implementación se haría uso de los mensajes proactivos de la librería de Bot Builder SDK de Microsoft (*Mensajes proactivos*, Microsoft), utilizado en la mayoría de los casos para realizar reservas. El bot inicia una tarea en segundo plano totalmente independiente al flujo de conversación, y cuando esta tarea finalice, se envía la notificación al usuario a través del id de la conversación almacenado previamente.

Referencias Bibliográficas

1MILLIONBOT, 2017. *LOLA* [https://1millionbot.com/chatbot-lola-umu/].

ACADEMY, M., 2019. *MIAO* [http://miaoacademy.org/].

AMAZON. *Alexa Skill Kit* [https://developer.amazon.com/es/alexa-skills-kit].

AMAZON. *aSK* [https://developer.amazon.com/alexa/console/ask].

CABELLO, C., 2016. *¿Qué son los bots de Telegram? Cómo conseguir infinidad de funcionalidades para nuestras conversaciones?* [https://www.nobbot.com/redes/los-bots-telegram-conseguir-infinidad-funcionalidades-nuestras-conversaciones/].

CHATFUEL, 2019. *Chatfuel* [https://chatfuel.com/].

DUOLINGO, 2019. *Duolingo* [https://es.duolingo.com/].

FACEBOOK, 2019. *Facebook Messenger* [https://es.wikipedia.org/wiki/Facebook_Messenger].

GENDE, I. M., 2019. *¡Sácale partido (educativo) a los chatbots!* [https://www.unir.net/educacion/revista/noticias/sacale-partido-educativo-a-los-chatbots/549203682749/].

HERVÍAS, A. E., 2017. *Que aporta un chatbot* [https://www.analiticaweb.es/chatbot-aporta-estrategia-marketing/].

MANYCHAT, 2019. *ManyChat* [https://manychat.com/].

MARTÍ, M., 2016. *Qué es el Web scraping? Introducción y herramientas* [https://sitelabs.es/web-scraping-introduccion-y-herramientas/].

MARTÍN, C. G.-O., 2017. Historía de los bots [https://blog.ferrovial.com/es/2017/08/bots-conversacionales-historia/].

MICROSOFT. *Bing Web Search Api Docs* [https://azure.microsoft.com/es-es/services/cognitive-services/bing-web-search-api/].

MICROSOFT. *Dialogs in Azure Bot Service* [https://docs.microsoft.com/es-es/azure/bot-service/bot-builder-concept-dialog?view=azure-bot-service-4.0].

MICROSOFT. *Language Generation Documentation* [https://github.com/Microsoft/BotBuilder-Samples/tree/master/experimental/language-generation].

MICROSOFT. *Mensajes proactivos* [https://docs.microsoft.com/es-es/azure/bot-service/dotnet/bot-builder-dotnet-proactive-messages?view=azure-bot-service-3.0].

MICROSOFT, 2019a. *LUIS* [https://docs.microsoft.com/es-es/azure/cognitive-services/luis/what-is-luis].

MICROSOFT, 2019b. *Qna Maker* [https://docs.microsoft.com/es-es/azure/cognitive-services/qnamaker/overview/overview].

MIJANGOS, V., 2019a. *Modelos Ocultos de Markov* [https://drive.google.com/file/d/1cpUYomQUBsxfwg3lFnRe9mzovhtfVs_4].

MIJANGOS, V., 2019b. *Procesamiento del lenguaje Natural* [Febrero]. **urlalso**: \url{https://sites.google.com/site/victormijangoscruz/cursos/procesamiento-de-lenguaje-natural}.

NIEVES, B., 2018. *3 maneras en que los chatbots están revolucionando la educación de tus hijos* [https://planetachatbot.com/chatbots-educacion-f2cfd9d4b345].

ROUSE, M., 2018. *Inteligencia artificial* [https://searchdatacenter.techtarget.com/es/definicion/Inteligencia-artificial-o-AI].

SANTOS, E., 2019. *Los 13 mejores bots que puedes añadir a Telegram* [https://www.nobbot.com/redes/los-bots-telegram-conseguir-infinidad-funcionalidades-nuestras-conversaciones/].

SARAZEN, M., 2019. *Tianjic* [https://medium.com/syncedreview/nature-cover-story-chinese-teams-tianjic-chip-bridges-machine-learning-and-neuroscience-in-f1c3e8a03113].

SIPÁN, P., 2008. *Algoritmo de Viterbi* [http://hdl.handle.net/10251/1501].

TELEGRAM. *Telegram APIs* [https://core.telegram.org/].

TOMÁS, A., 2019. *7 casos de éxito en facebook messenger* [https://www.ecommerce-nation.es/7-casos-de-exito-de-chatbots-en-facebook-messenger/].

UEX. *Open Data* [https://opendata.unex.es/].

WIKIPEDIA, 2017a. *Etiquetado Gramatical* [https://es.wikipedia.org/wiki/Etiquetado_gramatical].

WIKIPEDIA, 2017b. *Inteligencia Artificial Débil* [https://es.wikipedia.org/wiki/Inteligencia_artificial_debil].

WIKIPEDIA, 2017c. *Inteligencia Artificial Fuerte* [https://es.wikipedia.org/wiki/Inteligencia_artificial_fuerte].

WIKIPEDIA, 2018. *Procesamiento del lenguaje Natural* [https://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales].

YÚBAL, 2017. *Así era ELIZA* [https://www.xataka.com/historia-tecnologica/asi-era-eliza-el-primer-bot-conversacional-de-la-historia].