BotMentor
Bot de ayuda al estudiante en la plataforma Telegram

Trabajo de fin de grado

Eloy González Acedo
Pedro Sánchez Ramírez

Ingeniería de Software
Facultad de Informática
Universidad Complutense de Madrid
Director: Simon Pickin

13 de septiembre de 2017

Documento maquetado con TEXIS v.1.0.

Este documento est preparado para ser imprimido a doble cara.

BotMentor
Bot de ayuda al estudiante en la plataforma Telegram

Informe técnico del departamento
Ingeniería del Software e Inteligencia Artificial
IT/2017/6

Versión 1.0

Ingeniería de Software
Facultad de Informática
Universidad Complutense de Madrid
Director: Simon Pickin

13 de septiembre de 2017

A todos los profesores que, durante estos años, han hecho que entendamos y amemos esta ciencia y profesión.

Agradecimientos

A nuestras familias, compañeros y amistades por su apoyo prestado. A nuestro tutor Simon por creer en nuestra idea y guiarnos a través de este viaje.

Resumen

0.1. Resumen

Este documento detalla el proceso de análisis, diseño e implementación del proyecto de fin de grado BotMentor, así como otras etapas necesarias para su realización.

Este proyecto consiste en una plataforma pensada inicialmente para alumnos, que, haciendo uso de los sistemas de información de la universidad y más en concreto de la facultad de informática de la complutense, provea a éstos de la información relevante que puede interesar a un alumno y la provee de manera rápida y eficaz. Dicho sistema se basa en una plataforma de bots.

Para su desarrollo se ha optado por la elaboración de un servicio que mediante scraping recolecta información de la web de la facultad, almacenándola en una base de datos, un servicio web que consulta esa base de datos devolviendo la información útil solicitada y un bot en la plataforma Telegram que permite el acceso seguro y rápido a dicha información.

0.2. Summary

This document details all the process like planning, design and construction, needed to make ours degree final project: BotMentor.

The purpose of this project was a system initially designed for students, who using the university's information systems and more specifically those of the faculty, provides interesting information. This system is based in a bot platform.

The development was thinked as the creation of a service who scraps the webpage of the faculty and stores the interesting data in a DB, a web service uses this DB to select and send the requested information to the Bot, made in the Telegram's platform.

0.3. Palabras clave (keywords):

Bot, Telegram, WebService, Scraping, Machine-learning, Telepot, Python, Java, Chatbot, Parser

Índice

Índice de figuras

Capítulo 1

Introducción

El objetivo de este capítulo es que el lector comprenda algunos de los conceptos básicos sobre los que se basa este proyecto. Por esta razón, se seguirá un orden cronológico en la explicación. Empezando por la motivación que nos ha llevado a realizarlo, así como los objetivos propuestos para el proyecto, para finalizar con el alcance, es decir, el punto al que sería posible llegar de lograrse los objetivos propuestos.

Actualmente, para acceder a la información relevante para el desarrollo del curso académico, debemos acudir físicamente a los tablones de la facultad o a la secretaría de alumnos y de forma remota a la web de la facultad o a las plataformas de campus virtual. En muchos casos se hace uso de estas tecnologías remotas debido a su comodidad. Ahí es donde este proyecto pone su énfasis. La idea es mejorar el sistema que se utiliza para comunicar la información de manera remota al alumno mediante un sistema que, aún siendo antiguo, cada vez está más de moda y a la orden del día:"El Bot".

Un bot se define como una aplicación que imita el comportamiento humano. El uso de bots afecta a muchos ámbitos (videojuegos, mensajería masiva, malware, chat automatizado...). En nuestro caso nos referimos a un bot capaz de gestionar peticiones de los usuarios y enviarles la información solicitada en tiempo real.

1.1. Motivación

La idea de este proyecto no es más que facilitar el acceso a la información relevante de la universidad a todos sus integrantes, tanto alumnos como profesores y personal no docente. A través de una sencilla aplicación que podrán llevar en cualquier dispositivo inteligente, tendrán acceso a toda la información que suele utilizarse día a día en la universidad. Ésto es especiálmente interesante para nuevos alumnos, puesto que, sin conocer los portales de la

universidad, éstos pueden resultar a veces liosos o sobrecargados de secciones o información, por lo que resultaría tremendamente interesante poner a su disposición un sistema de fácil acceso, sencillo y que va a la información esencial que un alumno estándar puede precisar en su día a día.

Con esta herramienta se busca ante todo facilitar la integración de los nuevos alumnos en la universidad, pero también puede resultar útil para cualquier alumno veterano debido a su sencillez, evitándoles navegar por varias webs hasta encontrar lo que necesitan y con una interfaz amigable para cualquier dispositivo (dado que hasta hace poco, muchas webs de la universidad no estaban adaptadas a dispositivos móviles).

Hoy en día el uso de bots es algo muy extendido en diversos ámbitos. Desde un uso político para intentar influir en la opinión pública en temas sensibles como el reciente "Brexit" (Cadwalladr, 2017) a otros en el ámbito educativo con buenos resultados, como ejemplos podemos citar el caso del asistente de enseñanza creado por Ashok Goel, que ayudaba a sus alumnos sin que estos supieran que era una IA (Maderer, 2016) fabricada por IBM. En el capítulo dedicado al Estado del arte (sección 2.2.7) se hace mención a este y otros temas relacionados con la historia y el uso actual de los bots.

1.2. Alcance

El alcance inicial del proyecto es conseguir una aplicación autónoma, es decir que se mantuviese ella sola y no necesitase atención a menos que fuese para solucionar problemas o añadir nuevas funcionalidades. Ésta aplicación debería poder integrarse en los sistemas informáticos de la universidad y así facilitar otro cauce a través del cual obtener información e incluso realizar gestiones sencillas, en caso de no poder obtener acceso a dichos sistemas, se pensaría en la forma de creación de un sistema propio de información que "bebiese" del de la universidad. Para la realización de esta aplicación se ha pensado en el uso de una plataforma en constante desarrollo y expansión como podrían ser las API de desarrollo de bots de Telegram o Facebook, siendo el de Telegram el más llamativo para esta tarea.

1.3. Resumen introductorio sobre el documento

En el presente documento se viene a detallar, los fundamentos sobres los que se basa este proyecto (Capítulo 2), así como las tecnologías elegidas para llevarlo a cabo (Capítulo 3) y todos los detalles sobre su funcionalidad (Capítulo 4), implementación (Capítulo 6), análisis y diseño. Así como los detalles sobre la gestión de la planificación (Capítulo 5), configuración y calidad (Capítulo 7). Para finalizar encontraremos las conclusiones (Capítulo

8) obtenidas del desarrollo del mismo y las posibles actualizaciones o mejoras que han ido surgiendo a lo largo de este, dando lugar a muchas posibilidades que podrían llevarse a cabo en futuros proyectos que tomen este como base. Las aportaciones de cada miembro al proyecto (Capítulo 9.1. Al final del documento, también hay unos apéndices con la especificación de requisitos (Apéndice 4.2.2), así como una selección de diagramas UML (Apéndice A).

1.4. Introduction

The purpose of this chapter is for the reader to understand some of the basic concepts on which this project is based. For this reason, a chronological order will follow in the explanation. Starting with the motivation that has led us to do it, as well as the objectives proposed for the project, to finish with the scope, that is to say, the point at which it would be possible to achieve the proposed objectives.

At present, in order to access information relevant to the development of the academic course, we must physically go to the faculty boards or the student secretariat and remotely to the faculty website or the virtual campus platforms. In many cases these remote technologies are used because of their convenience. That is where this project puts its emphasis. The idea is to improve the system used to communicate information remotely to the student through a system that, although old, is increasingly fashionable and the order of the day: "The Bot."

A bot is defined as an application that mimics human behavior. The use of bots affects many areas (video games, mass mailing, malware, automated chat ...). In our case we refer to a bot capable of managing requests from users and send them the information requested in real time.

1.5. Motivation

The idea of this project is simply to facilitate access to the relevant information of the university to all its members, both students and teachers and non-teaching staff. Through a simple application that can be carried in any smart device, they will have access to all the information that is used day by day in the university. 'This is especially interesting for new students, since, without knowing the university portals, these can sometimes be lousy or overloaded with sections or information, so it would be tremendously interesting to put at your disposal an easily accessible, simple system and that goes to the essential information that a standard student may need in his day to day life.

With this tool, it is primarily intended to facilitate the integration of new students in the university, but it can also be useful for any veteran student due to its simplicity, avoiding surfing several websites to find what they need and a friendly interface for any device (since until recently, many university websites were not adapted to mobile devices).

Nowadays the use of bots is very widespread in several areas. From a political use to try to influence public opinion on sensitive issues such as the recent "Brexit"(Cadwalladr, 2017) others in the educational field with good results, as examples we can cite the case of the assistant of creation created by Ashok Goel, who helped to their students without them knowing that it was an AI made by IBM. (Maderer, 2016). In the chapter dedicated to the State of the art (section **??**) it is mentioned to this and other subjects related to the history and the real use of the bots.

1.6. Scope

The initial scope of the project is to achieve an autonomous application, that is to say that it was kept alone and did not need attention unless it was to solve problems or add new functionalities. This application should be able to be integrated into the university's information systems and thus provide another channel through which to obtain information and even carry out simple steps, in case of not being able to obtain access to such systems, it would be thought in the form of creation of an own system of information that "would drink.ºf the one of the university. For the realization of this application has been thought of the use of a platform in constant development and expansion as they could be the APIs of development of bots of Telegram or Facebook, being that of Telegram the most striking to this task.

Capítulo 2

Estado del arte

El uso de la tecnología en las tareas cotidianas está creciendo de manera exponencial en las últimas décadas. Con la expansión de los smartphones cualquiera puede acceder, en cualquier lugar, a todos los datos de interés disponibles en la red: noticias, correo electrónico, entretenimiento, etc. Además también ha facilitado la capacidad de realizar trámites bancarios, administrativos, sanitarios, entre otros, que ahorran tiempo e incluso a veces conocimiento a las personas.

En los últimos años, existe una guerra entre las redes sociales para hacerse con el control del tráfico de información entre usuarios, instituciones, empresas, etc. Esto implica también una guerra por el dominio de las herramientas más usadas en estas redes: la mensajería instantánea. En éstas plataformas se están desarrollando bots, los cuales consiguen ofrecer funcionalidades extras a un simple chat y así hacerlos más atractivos a los usuarios.

Una de las plataformas que más ha revolucionado en los últimos años el panorama de mensajería es Telegram. Esta herramienta intenta diferenciarse de sus competidores aportando valor añadido a los simples chats de otras plataformas como Whatsapp, Facebook Messenger o Snapchat. Además de compartir documentos, fotos, ubicaciones o sonidos, ha implementado recientemente una api para poder desarrollar bots (tel, 2015), juegos e incluso blogs (tel, 2016) de manera sencilla y totalmente integrados con la interfaz del chat, manteniendo su esencia y sencillez.

2.1. Bots

2.1.1. ¿Qué es?

Un bot es una aplicación software que realiza distintos tipos de tareas de forma autónoma, siendo su objetivo primordial simular a una persona.

Podemos observar que la definición de lo que es un bot nos ofrece un gran abanico de posibilidades, dándonos la oportunidad de utilizar ésta tecnología prácticamente en todos los ámbitos de la informática. Por ejemplo podemos encontrarnos con plataformas de entretenimiento como los videojuegos, en los cuales los bots simulan ser personajes con los que interactuamos, o la utilización de estos para recolectar información de otras APIs o webs mediante indexadores (crawles). También como toda tecnología es utilizada para prácticas delictivas, como los "Spam bots" que son creados para enviar spam de forma masiva.

Como hemos podido observar esta tecnología presenta una amplia gama de utilidades, pero nuestro trabajo se enfoca principalmente en los chatbots, siendo estos un tipo determinado de bot que busca llevar a cabo una conversación fluida entre una máquina y un ser humano.

Actualmente las aplicaciones de mensajería instantánea son la forma principal de comunicación entre la mayoría personas y, es por este motivo por el que hemos decidido enfocar nuestro trabajo a este tipo de bots.

También debemos de tener en cuenta a los asistentes virtuales, que cada vez van asumiendo más responsabilidades en los entornos informáticos más relevantes, como por ejemplo en los sistemas operativos de las empresas tecnológicas más importantes del momento, ya sea "Siri" en iOS, "Google Assistant" en Android o "Cortana" en Windows. Todos ellos van obteniendo un peso muy relevante en la interacción del usuario con el software que les rodea a diario.

2.1.2. ¿Cómo surgen los bots?

La idea de bot surge a partir del concepto de robot, siendo los bots una versión software de los mismos.

La primera aparición de los bots fue durante la década de los 50, cuando Alan Turing, considerado uno de los padres de la ciencia de la computación, menciono la posibilidad de que las máquinas llegasen a tener pensamientos propios. Fue entonces cuando diseño el famoso Test de Turing, que se basa en probar la habilidad de una maquina a la ahora de simular el comportamiento de un humano.

Uno de los primeros en llevar a cabo la idea de Turing fue el profesor en informática teórica Joseph Weizenbaum en el año 1966 con el proyecto Eliza (Busacca, 1998), el cual fue uno de los primeros en procesar lenguaje natu-

ral, de esta forma era capaz de reconocer las palabras claves y usarlas para construir preguntas. 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.

Fue así como Joseph Weizenbaum creo él primer bot conversacional el cual influyo en el posterior desarrollo de nuevos proyectos sobre esta nueva tecnología como el Albert One o A.L.I.C.E. Pero no fue hasta la creación del proyecto SmarterChild en el 2001 cuando empezaron a ser algo más que simples chatbots.

SmarterChild nació con la idea de ofrecer un servicio a los usuarios de mensajería instantánea de MSN Messenger, AOL y lo SMS. Lo que hizo diferencial a este chatbot fue su capacidad, no solo de conversar con los usuarios, sino que además era capaz de realizar distintos tipos de tareas, como informar sobre el tiempo, sobre los horarios del cine, las noticias, etc.

Podemos decir que gracias a SmarterChild se establecieron las bases de los que hoy en día son los asistentes virtuales que encontramos en casi todos los dispositivos electrónicos que nos rodean. (FM, 2017)

2.1.3. Tipos de bots según sus funcionalidades

Los bots se basan en la ejecución de tareas automatizadas, estas tareas suelen ser simples y repetitivas, a una velocidad mayor a la que podría realizarla un ser humano.

Las funcionalidades de los bots dependen de las tareas a realizar, por lo que los dividimos en diferentes categorías: (García, 2016):

- Comunicación

  Algunos bots se comunican con otros usuarios de servicios basados en Internet, a través de mensajería instantánea (IM), Internet Relay Chat (IRC) u otra interfaz web como los bots de Facebook y Twitter. Estos bots conversacionales responden a las preguntas de los usuarios, además pueden manejar muchas tareas, como información meteorológica, del código postal, resultados deportivos, etc.

  Otra utilidad de los bots en IRC es estar atento a la conversación de un canal, comentando ciertas frases escritas por los participantes, esto se utiliza como un servicio de ayuda para los nuevos usuarios, o para censurar algunos comentarios no permitidos, es decir actúan como un moderador.

- Comercial

  Una de las aplicaciones de los bots es su uso comercial, como por ejemplo para buscar ofertas en una página web para poder indexarlas, o las granjas de bots que tienen como objetivo manipular las posiciones o aumentar las calificaciones de las tiendas de aplicaciones en línea, esto sucede por ejemplo en la App Store de Apple y en Google Play.

- Indexador web o Crawlers

  Este tipo de bot inspecciona y recolecta la información que hay en las páginas webs y en las APIs de forma automatizada. (Patil y Patil, 2016)

  El bot araña comenzara con un grupo inicial de direcciones, las cuales visitara y analizara mediante un grupo de reglas establecidas previamente, buscando direcciones de páginas nuevas que visitar y almacenar, para posteriormente realizar de nuevo el proceso. Las páginas almacenadas serán procesadas por un motor de búsqueda que indexara las páginas y proporcionara un sistema rápido de búsqueda. Los bots indexadores más famosos son el googlebot y el bingbot. A parte de este tipo de funcionalidad los bots araña también son usados paro otros tipos de tareas, como:

  - Crear el índice de una máquina de búsqueda.
  - Monitorización del estado de una página web, para hallar cualquier tipo de error (alertbot).
  - Recolectar información sobre precios de productos para recopilar un catálogo (ecommerce).

- Bots informativos

  Los bots de carácter informativo se encargan de gestionar y publicar la información que se les ha otorgado, podemos observar que cadenas tan importantes como la CNN usan esta tecnología para simplificar las tareas informativas.

- Game Bots

  En los videojuegos los bots son un recurso muy utilizado, y es que estos simulan ser un jugador más del juego. Una de las características más importantes de estos bots es su inteligencia artificial (IA) y, dependiendo del tipo de juego los bots serán diferentes, por ejemplo en los first-person shooter (FPS) estos tendrán la capacidad de analizar el mapa y establecer una estrategia de juego, o los massively multiplayer online role-playing game (MMORPG) que se dicaran a tareas más automatizadas y repetitivas como el farmming.(Wikipedia, Game bots)

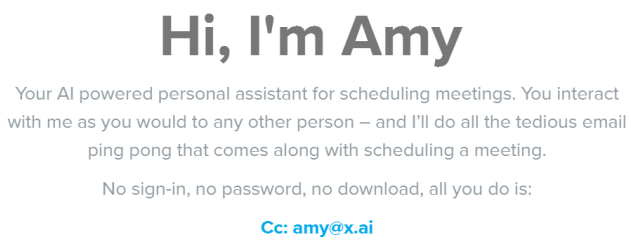


Figura 2.1: Anuncio de x.ai

- Bots transaccionales

  La función de estos bots es realizar transacciones específicas de datos entre plataformas, simulando así el papel de un agente que interactúa con sistemas externos. Además pueden comunicarse con cualquier API.

  Un ejemplo sería x.ai, "el asistente personal que programa reuniones para usted" cuyas funciones muestra la figura 2.1.

  Este asistente con el que interactuamos será el que prepare nuestras reuniones, encontrando los horarios adecuados para la misma.

- Chatbots

  Los bot conversacionales tienen como principal función entablar una conversación inteligente con los usuarios. Estos suelen usar el lenguaje natural y el machine learning, pero también tenemos bots más simplificados que basan sus repuestas en unas reglas simples.

  Los Chatbots son utilizados sobretodo en medios de mensajería como Telegram, Facebook Messenger, etc. Aunque cada vez se van ampliando más sus funcionalidades, para que ofrezcan más servicios.

- SpamBots

  Los spambots (urbanas, 2016) son bots que envían de forma automática un determinado spam, creados con el fin de obtener un benéfico mediante él envió de publicidad no deseada. Algunos de estos tienen un propósito más dañino para el usuario, como por ejemplo los que envían contenido fraudulento o todavía más graves como son las estafas online. Generalmente es fácil detectar este tipo de prácticas gracias a sus patrones de repetición.

  Tipos de spam según al lugar donde se envíen:

  - Correo no deseado en el email.

- Mensajes en foros y chats.
- Mensajes en comentarios de blogs, portales web y redes sociales.
- En videojuegos multijugador.
- En plataformas de stream gaming

- Zombie bot

  En primer lugar se considera Zombie a un ordenador que ha sido infectado con algún tipo de malware, con el cual una tercera persona puede ejecutar actividades hostiles, sin que el dueño del ordenador sepa lo que está sucediendo en ningún momento.

  De esta forma aparecen los Zombie bots, los cuales son capaces de controlar redes enormes de ordenadores Zombies (botnets), realizando con ellos ataques coordinados a gran escala, usualmente se utilizan para provocar una denegación de servicios a la víctima.

- Bots de intercambio de archivos

  Los bots también han llegado a los servicios P2P (torrent, descargas directas, etc), para detrimento de los usuarios ya que su utilidad es maliciosa, siendo su objetivo inyectar malware en los archivos que los usuarios comparten.

  Cuando el usuario realiza una consulta (título de película, serie, etc), el bot confirma que dispone del archivo consultado y, le proporciona un enlace. Al mismo tiempo genera un archivo falso y le inyecta el malware. El usuario descarga y abre el archivo, infectando así su ordenador.

- Fraud bots

  Los fraud bots se basa en engañar al sistema para que se crear que se ha generado un clic, se utilizan para todo tipo de plataformas, como por ejemplo generan visitas en Youtube o en las páginas que usan sistemas de pago por clic (PPC).(RODRIGUEZ, 2012)

2.1.4. ¿A qué se debe el boom de los bots?

Existen varias razones que influyen en que los bots hayan llamado la atención de las grandes empresas, las cuales están apostando por esta tecnología.

En primer lugar queremos hablar de la mejoras en la inteligencia artificial y en concreto en dos de sus ramas más importantes, como lo son el procesamiento del lenguaje natural (PLN) y el Machine Learning.

En los que se refiere al Machine Learning (aprendizaje automático), esta disciplina se ocupa de que las maquinas sean capaces de auto-programarse, es decir, que aprendan de su propia experiencia. En concreto ha habido un avance espectacular en una de sus técnicas el Deep Learning con el cual somos capaces de utilizar las redes neuronales para el aprendizaje de la máquina.

Con los avances en el procesamiento de lenguajes naturales (PLN), campo que combina tecnologías como el aprendizaje automático y la lingüística aplicada, con el objeto de hacer posible la compresión y el procesamiento del lenguaje humano.

Estos nuevos avances aportan capacidades indispensables a los bots como la posibilidad de aprender, mejorar, adaptarse y entender de forma fluida las necesidades del usuario.

Por último debemos mencionar un factor vital para esta nueva revolución de los bots y no es ni más ni menos que la comunicación.

La comunicación entre los humanos ha sido la herramienta con la que transmitir a quienes nos rodean nuestras ideas, impresiones y sentimientos.

Con las tecnologías hemos ido mejorando los sistemas de comunicación, como por ejemplo con el telégrafo con el cual se podían enviar mensajes cortos a grandes distancias o el teléfono que posibilitaba una conversación fluida entre sus interlocutores a grandes distancias. En la actualidad es internet el medio de comunicación predominante por el cual los usuarios pueden transmitir y recibir todo tipo de información y, es que tenemos a nuestra disposición todo tipo de contenidos como blogs, redes sociales, periódicos, plataformas de video, traductores, enciclopedias, etc. Son estas y muchas más actividades las que podemos realizar en la red, ¿pero cuál es la actividad que ocupa el mayor parte de nuestro tiempo?

La conversación, esta es la actividad que más ocupa nuestro tiempo, desde el uso del SMS hasta la llegada de los nuevos sistemas de mensajería, ya sea WhatsApp, Facebook Messenger, Telegram, etc. Es por tanto la conversación entre los bots y los usuarios el gran aliciente que presenta esta tecnología. ¿Por qué? Pues creemos que la respuesta es bastante simple: por primera vez, los bots te permitirán interactuar instantáneamente con el mundo que te rodea.

En definitiva creemos que los bots en un futuro no muy lejano sustituirán a las apps de hoy en día y que nos ofrecen distintos tipos de servicios

y soluciones, con la ventaja de evitar la necesidad de aprender su interfaz y funcionamiento, simplemente conversando con el bot obtendremos lo que necesitamos. Un ejemplo muy claro seria a la hora de reservar entradas de un partido, no necesitaríamos descargarnos una app concreta para esa situación, sino que iniciaremos una conversación con nuestro bot con el cual obtendríamos la reserva deseada.

2.2. Bots conversacionales

2.2.1. Introducción

Un bot conversacional o Chatbot(Shevat, 2017) es un programa desarrollado con el objetivo de comunicarse con otras máquinas o seres humanos. La comunicación puede ser tanto de forma oral como escrita, siendo esta última la más usada, ya que es el método con el que las máquinas han sido desarrolladas y con el que interaccionamos con ellas.

Como mencionamos en apartados anteriores este tipo de programas buscan simular de forma convincente a un ser humano, y para ello precisan de sofisticados sistemas de lenguaje natural y avanzadas tecnologías enfocadas en el aprendizaje. También es cierto que la gran mayoría de chatbots utilizan sistemas más sencillos, que se basan en la búsqueda de palabras clave con las cuales dar una respuesta más adecuada.

Los Chatbots suelen ser utilizados en entornos en los cuales se entabla un diálogo, este es el motivo por el cual las plataformas de mensajería como Facebook invierten en el desarrollo de esta tecnología. Pero no es el único ámbito en el cual se ha establecido con firmeza, ya que por ejemplo los famosos asistentes virtuales como Siri en iOS o Cortana en Windows implementan en sus servicios a los chatbots.

2.2.2. Evolución

Las tecnologías utilizadas para desarrollar los bots conversacionales han ido evolucionando a lo largo de los años, mejorando enormemente las capacidades de los bots.
En los inicios se utilizaba una herramienta de la inteligencia artificial (IA) que era especialmente útil para entablar una comunicación entre máquina-persona, el procesamiento del lenguaje natural (PNL). El método consistía en el uso de complejas reglas escritas a mano que el bot utilizaba para conversar con un humano.

No fue hasta finales de la década de los ochenta cuando se empezó a combinar distintas disciplinas derivadas de la inteligencia artificial (IA), como el aprendizaje automático, para mejorar la comunicación entre los bots y las

personas. Es cierto que por aquella época los chatbots empezaban a tomar forma, pero no ha sido hasta hace pocos años que se empezado a desarrollar de una forma más amplia esta tecnología.

Podemos diferenciar dos corrientes tecnológicas a la hora de desarrollar un Chatbot, estas se distinguen por la forma en la que se crean siendo una más compleja que la otra y por las capacidades que estos presentan, en definitiva unos son bots no inteligentes (simples) y los otros si, ya que presentan rasgos de inteligencia.

La primera forma de desarrollo y más sencilla se basa en el uso de unas reglas simples con las que el usuario podrá seleccionar una serie de opciones que le presentaremos en forma de dialogo, estas opciones son estáticas por lo que no cambian durante la interacción del usuario con el Chatbot. Para que este tipo de Chatbot pueda tener un marco más amplio de actuación necesita una base de datos más grande y un considerable aumento de las reglas, que deberán ser escritas a mano por el desarrollador. Por supuesto con este modo de creación nuestro chatbot tendrá muchas limitaciones, ya que no podrá salirse del marco establecido por el contexto para el que fue creado. Como ejemplo podríamos poner los bots de imágenes y vídeos de Telegram.

La otra metodología de desarrollo se basa en técnicas de inteligencia artificial mucho más complejas de utilizar y por tanto de explicar. En primer lugar debemos entender que para la creación de este tipo de chatbot necesitamos resolver una serie de problemas, a continuación enumeraremos los más esenciales:

- Datos: se necesitan técnicas para formalizar los datos de entra y así poder manipularlos de forma correcta.
- Comprensión: se ha de establecer una serie de reglas con las que estableceremos un vocabulario para así formalizar la comunicación entre el bot y el usuario, digamos que tenemos que enseñar lenguaje natural al bot para que pueda establecer un dialogo. Como herramienta utilizaremos el paradigma de procesamiento de lenguaje natural (PLN).
- Aprendizaje: para que un chatbot pueda mejorar y adaptarse necesita aprender de su entorno, para ello se ha de crear un modelo con el cual el bot podrá entrenar. Cuanto más entrene mejor podrá responder a las preguntas del usuario, además de poder adaptarse de forma dinámica al contexto del dialogo. Para esta tarea tenemos varias herramientas y formas de conseguir nuestro objetivo, en especial hablaremos de “Machine Learning” y el “Deep Learning”.

Un claro ejemplo de la actualidad de los chatbots inteligentes es el desarrollo del proyecto Mitsuku (Worswick, 2016), un chatbot creado por Steve Worswick que simula ser una joven de 18 años con la que podrás entablar una conversación inteligente, para su desarrollo se basó en el uso de la tecnología de lenguaje natural AIML technology (A.L.I.C.E.), añadiendo complementos que mejoran la experiencia. Dentro de sus funcionalidades incluye la capacidad de razonar sobre temas específicos. Un ejemplo práctico seria a la hora de que el usuario preguntara "¿Puedes comerte una casa?", y entonces Mitsuku buscara las propiedades de çasa", esta encontraría el valor de "hecho dez este establece un "ladrilloz como sabe que el atributo ladrillo no se puede comer le responde con una respuesta negativa. Podemos ver un ejemplo de conversación en la figura 2.2.

Mitsuku ha conversado con millones de personas en todo el mundo a través de la web y otras aplicaciones, ha sido premiada en dos ocasiones por la competición Loebner Prize, la cual tiene como objetivo superar el Test de Turing.

Son ejemplos como el de Mitsuku los que nos hacen reflexionar sobre la potencia escondida en este tipo de tecnología y como el desarrollo de las herramientas provenientes de la IA son fundamentales para ello, por lo tanto y viendo la relevancia de estos conceptos en la realidad de los Chatbots actuales hemos decido explicarlos de forma más detalla.

2.2.3. Inteligencia Artificial

La inteligencia artificial (IA) (Wikipedia, IA) se encarga del estudio de los “agentes inteligentes”: cualquier dispositivo que perciba su entorno y tome una serie de decisiones que se puedan definir como correctas o exitosas en base a un objetivo propuesto. Podemos decir que resolver problemas o aprender son las capacidades que la inteligencia artificial busca simular en una maquina.

Debemos diferenciar dos corrientes de la IA: la robusta y la aplicada.

- Strong AI: Podemos decir que esta inteligencia es la más futurista y la que se encuentra más lejos de nuestras capacidades inventivas, ya que esta propone una capacidad cognitiva similar a la de los humanos.

- Weak AI: Este tipo es mucho más realista y cercano a nuestros tiempos y, es aquí donde utilizamos algoritmos de aprendizaje guiado mediante el Machine Learning y el Deep Learning.

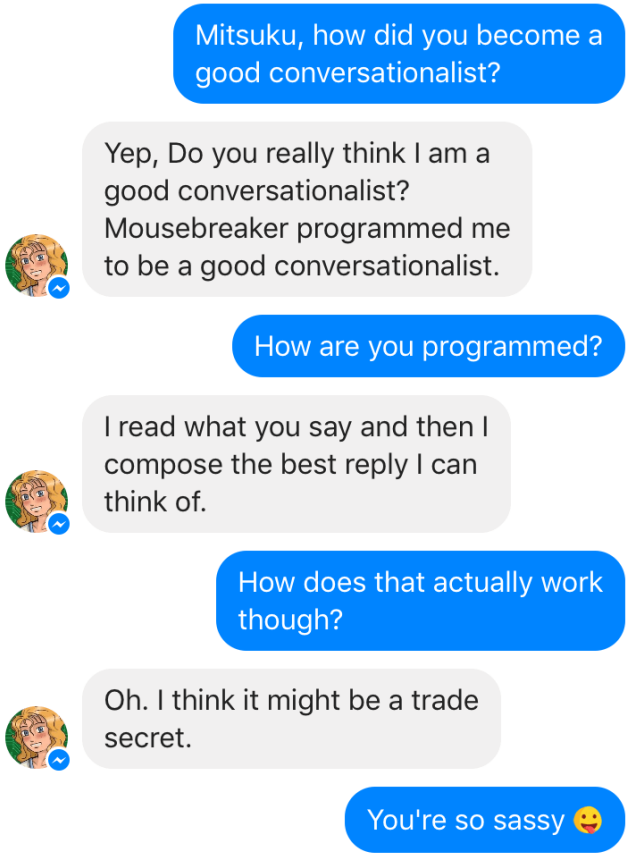


Figura 2.2: Ejemplo de conversación con Mitsuku

Los investigadores de la inteligencia artificial buscan desarrollar tecnologías que permitan a las maquinas funcionar de forma inteligente. El problema principal al que se enfrenta la comunidad científica es simular una inteligencia artificial, este se ha dividido en sub-problemas según las capacidades que se vayan a estudiar. En este apartado hablaremos de los sub-problemas más relevantes en el universo de los chatbots.

- Resolución de problemas: durante los primeros inicios de la IA los algoritmos desarrollados para resolver problemas usaban la técnica de "setp-by-setp" que quiere decir paso a paso, esta técnica se basa en el uso de deducciones lógicas, de una manera secuencial. El principal problema es que los algoritmos consumen enormes recursos computacionales, que los hacen inviables para problemas más complicados. Son estos los motivos que impulsaron a los investigadores hacia la búsqueda de algoritmos más eficientes.

  Estos algoritmos de la lógica matemática y búsqueda tienen como obje-

tivo resolver problemas basados en representaciones de alto nivel "simbólico", este enfoque no se ajusta realmente a la forma en la que los humanos resolvemos problemas, lo más normal a la hora de resolver un problema es mediante el uso de juicios rápidos e intuitivos. En consecuencia la IA en la actualidad se ha enfocado en la computación sub-simbólica, la cual está basada en redes neuronales, estadística, etc.

- Representación del conocimiento: podemos decir que el conocimiento es un conjunto de conceptos dentro de un dominio y las relaciones entre esos conceptos. El principal problema que la IA a de resolver es la cantidad de datos a clasificar, ya que por ejemplo si durante la conversación con un chatbot surge un tema relacionado con una casa, este sabe los conceptos básicos, como que es una construcción, que tiene paredes, etc. Pero ninguna de estas cosas definen al resto de casas, por lo que no sería una verdad absoluta. Este problema surgió cuando empezaron a definir las reglas sobre algún campo a representar y, conforme se iban definiendo empezaban a aparecer innumerables excepciones. La tarea de clasificar todo lo que nos rodea es costosa y prácticamente imposible con los medios disponibles hoy en día.

- Aprendizaje: El aprendizaje automático o "Machine Learning" en inglés, es un tipo de algoritmo que mejora de forma automática a través de la experiencia. Los problemas de esta rama de la IA se dividen en sub-problemas según su funcionalidad (supervisada, no supervisada, semisupervisado, etc), al ser esta una rama tan fundamental en el universo de los chatbots la explicaremos más detalladamente en los puntos siguientes.

- Procesamiento del lenguaje natural: El procesamiento del lenguaje natural otorga a las maquinas la capacidad de entender el lenguaje humano. Al igual que el aprendizaje, el lenguaje natural es de vital importancia para el funcionamiento de los chatbots por lo que hemos decidido explicarlo de forma más detalla a continuación.

2.2.4. Procesamiento Del Lenguaje Natural

El procesamiento de lenguajes naturales (PLN) (Wikipedia, Procesamiento del lenguaje natural) es una de las disciplinas en las ciencias de la computación, inteligencia artificial y lingüística que estudia las comunicaciones entre los humanos (mediante el lenguaje natural) y las computadoras. El objetivo de esta ciencia es la búsqueda de mecanismos más eficaces con los que realizar la comunicación y que así puedan llegar a ser computacionalmente viables.

Antes de la llamada "revolución estadística" la mayoría de sistemas PLN se basaban en un conjunto de reglas diseñadas a mano, lo que requiere de una gran labor a la hora de desarrollar las reglas que cada vez se vuelven más complejas, además de la dificultad de controlar y adaptar las variaciones que van surgiendo en los lenguajes naturales. Fue entonces cuando a finales de los años ochenta se introdujo el aprendizaje automático para el procesamiento del lenguaje. También hubo otros factores que impulsaron la revolución en los PNL, como el aumento de la potencia computacional de las maquinas.

En los inicios de la integración de los algoritmos de aprendizaje en los sistemas PNL, como los arboles de decisión, produjeron reglas complejas y similares a las escritas a mano. En cambio el etiquetado gramatical introdujo el uso del modelo oculto de Márkov (HMN)(Gomez, 2009), este modelo estadístico asume que el sistema a modelar es un sistema de Márkov de parámetros desconocidos, este sistema busca determinar los parámetros desconocidos de una cadena a partir de parámetros observables. Un HMM se puede considerar como la red bayesiana dinámica más simple. La red de Bayes es un modelo grafo probabilístico (estático) que representa un conjunto de variables aleatorias y sus dependencias a través de un grafo.

Es por esto que las investigaciones se han centrado más en los modelos estadísticos, los cuales toman decisiones probabilísticas basadas en la asignación de pesos (importancia) a los datos de entrada. Un ejemplo claro son los sistemas de reconocimiento de voz que ahora utilizan modelos estadísticos.

En los últimos tiempos el desarrollo del PNL se ha enfocado en los algoritmos de aprendizaje, en concreto en los tipos no supervisados y semisupervisados. Estos son capaces de aprender de los datos que no han sido clasificados previamente. Normalmente la tarea de clasificar (no supervisado) datos es más complicada que las tareas que generan y entrenan un modelo (supervisado), además de ser menos efectivas. Sin embargo al contar con una enorme cantidad de datos no clasificados, estas técnicas compensan los resultados inferiores.
En la actualidad se ha puesto de moda el uso de la técnica de aprendizaje profundo (Deep Learning), la cual ha logrado resultados muy satisfactorios en muchas tareas del PNL como el modelado de lenguaje o el parsing.

El funcionamiento de este tipo de algoritmos se basa en proporcionar grandes cantidades de datos que serán estructurados en un conjunto de "features" (características). Son varias las ventajas que aporta este paradigma en comparación con las reglas creadas a mano. En primer lugar este tipo de algoritmos se centran de forma automática en los casos más comunes, en cambio las reglas hechas a mano no siempre son capaces de hallar el carácter

más común en los casos propuestos.

Otra virtud es el uso de la estadística de inferencia que puede detectar de forma más eficiente una entrada errónea (palabras erróneas o mal escritas). Cuando intentamos evitar errores usando el método manual observaremos que es mucho más difícil y menos efectivo. Tal vez la característica más relevante de todas sea su capacidad para crear reglas más precisas mediante el aumento de los datos a procesar lo que ofrece una mayor potencia, en cambio las reglas hechas a mano solo pueden mejorar esta capacidad haciendo las reglas más complejas y aumentando así su dificultad a la hora de ser desarrolladas.

2.2.5. Machine Learning

El aprendizaje automático (del inglés, "Machine Learning") es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. La manera en la que trabajan estos algoritmos es extrayendo patrones en los datos, aprendiéndolos y utilizándolos para ser capaces de etiquetar/clasificar datos no vistos antes.

El aprendizaje automático surgió en los inicios de la IA, como una de las ramas principales a investigar, ya que es fundamental para que una maquina sea inteligente. En un principio se utilizaron las “redes neuronales” y el razonamiento probabilístico para intentar hacerlas aprender.

Sin embargo, durante los siguientes años los sistemas probabilísticos presentaban problemas tanto teóricos como prácticos a la hora de adquirir los datos y representarlos. Además las investigaciones se empezaron a enfocar en otro tipo método, llamado Inteligencia Artificial Simbólica, esto produjo una fisura entre la IA y el aprendizaje automático.

En 1980 los sistemas expertos dominaban prácticamente la mayoría de investigaciones sobre la IA, mientras por otro lado la estadística no tenía casi repercusión. Los trabajos sobre el aprendizaje basado en conocimiento / simbólico continuaron, siendo estos liderados por la programación lógica inductiva, además la parte más estadística se desprendió y con ella sus investigaciones. Las comunidad de investigadores perdieron interés en las redes neuronales, aunque en otras disciplinas se siguió investigando, como en la Física y la Mecánica Estadística. No fue hasta mediados de los ochenta con la reinvención de la retroprogramación (ESCOM, 2009), la cual consiste en un algoritmo de aprendizaje supervisado que entrena a las redes neuronales, que empezaron a resurgir la redes neuronales.

Tras expandirse a otros campos distintos a la inteligencia artificial, el aprendizaje automático empezó a enfocarse en objetivos diferentes, más prácticos. Se alejo en parte del estudio del a IA simbólica, y se empezó a enfocar en métodos y modelos provenientes de la estadística y la teoría de la probabilidad. Además coincidió con el aumento y disponibilidad de información digital gracias a Internet.

Es importante mencionar la minería de datos, ya que comparte muchas metodologías con el aprendizaje automático, ésta se basa en descubrir patrones en grandes volúmenes de información. Ambos usan métodos similares pero con objetivos distintos, para la minería de datos el objetivo principal es descubrir las propiedades (desconocidas) de los datos, en cambio el aprendizaje automático se centra en la predicción mediante el entrenamiento, por ejemplo usa métodos de la minería de datos como el "aprendizaje no supervisado" o como un paso de preprocesado para mejorar el entrenamiento.

También se utilizan funciones de optimización, como las funciones de perdida, que muestran las discrepancias entre las predicciones del modelo que se está entrenando y las instancias reales del problema, un ejemplo sería a la hora de clasificar, en este proceso se intenta asignar una etiqueta a las instancias y con los modelos creados se entrena para predecir las etiquetas pre-asignadas de un conjunto de ejemplos.

Los algoritmos de aprendizaje intentan imitar la forma de aprender de las personas, como ya es sabido las personas aprendemos a partir de las experiencias y con ellas generalizamos conceptos. La idea de "generalizar" en las maquinas, es la capacidad para clasificar nuevos ejemplos/tareas no vistas con anterioridad, utilizando lo aprendido en la fase de creación del modelo. Este modelo general es construido a base de ejemplos de entrenamiento que provienen de una distribución de probabilidad desconocida.

Son varios los tipos de algoritmos que se usan en el aprendizaje automático y cada uno de ellos tiene un propósito concreto, estos son los tipos más relevantes en la actualidad:

- Los algoritmos supervisados, son una tarea del aprendizaje automático que es capaz de generar una función o modelo a través de la extracción de los datos de entrenamiento etiquetado. Estos algoritmos se usan principalmente para tareas de regresión y clasificación. Los datos de entrenamiento son un conjunto de ejemplos de entrenamiento, estos ejemplos consisten en un objeto de entrada (vector) y un valor de salida deseado. Entonces el algoritmo analiza los datos de entrenamiento y produce un modelo, el cual se utilizara para generar nuevos ejemplos.

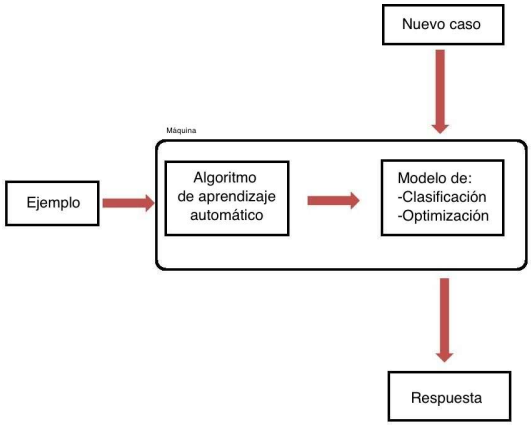


Figura 2.3: esquema de algoritmo de aprendizaje extraido de wikimedia por Debora.riu

Uno de los mayores problemas de este proceso es la etiquetación de los datos, ya que esta ha de hacerse a mano para que el algoritmo pueda aprender, lo cual es un proceso muy costoso.

Un ejemplo en el mundo de los Chatbots sería un servicio de atención al cliente donde el bot debe resolver las dudas y problemas del usuario, comprendiendo las peticiones que le hace el usuario y retroalimentaciones de estas (aprendiendo).

- Los algoritmos no supervisados, son una tarea del aprendizaje automático capaz de generar un modelo para definir la estructura oculta a partir de los datos no etiquetados. Es por esto que el algoritmo ha de ser capaz de reconocer los patrones y con estos establecer las etiquetas de las nuevas entradas. Este tipo es utilizado para producir probabilidades condicionales, es decir, se utiliza para realizar el etiquetado que se necesitaría en los algoritmos supervisados. Uno de sus mayores problemas es la estimación de la densidad en las estadísticas
- Los algoritmos semisupervisado este tipo es una mezcla de las dos técnicas antes vistas.
- Los algoritmos por refuerzo se basan en la recompensa a la hora de llevar a cabo una acción determinada, lo que confluirá en una maximización de los objetivos a alcanzar.

En la figura 2.3 vemos de manera simplificada el proceso de aprendizaje automático. Como hemos podido observar los algoritmos de aprendizaje

tienen sus ventajas y sus desventajas, pero quizás el mayor problema es su limitada efectividad y poca potencia. Con la aparición del Deep Learning algunos de estos problemas se resolvieron, es por esto que en la siguiente sección analizaremos en detalle.

2.2.6. Deep Learning

El Deep learning surge como una técnica concreta de Machine learning, la cual se fundamenta en el uso de unas redes neuronales artificiales para tareas de aprendizaje que contienen más de una capa (layer) oculta. Dentro de esta técnica encontramos los tres tipos de algoritmos planteados anteriormente, pero nosotros nos enfocaremos principalmente en el algoritmo no supervisado, el motivo principal es la necesidad que tienen los chatbots de etiquetar/clasificar su entorno para generar el modelo con el que aprenderán. El aprendizaje profundo ha supuesto un nuevo impulso para las aplicaciones de la IA, revolucionando campos como la visión artificial o el procesamiento del lenguaje natural (PNL).

Los chatbots han de ser capaces de responder preguntas complejas no solo utilizando el contexto dentro de dialogo establecido con la el interlocutor, sino que ha de ser capaz de usar información externa, la cual ha de conocer previamente.

En primer lugar para desarrollar un chatbot que pueda mantener conversaciones inteligentes necesitamos clasificar o estructurar la información a utilizar. Es un hecho que el aprendizaje profundo requiere de una gran cantidad de datos estructurados para generar los modelos adecuados para el aprendizaje y, aunque hoy en día se dispone de una gran cantidad de datos, estos requieren de una etiquetación manual, lo que supone una tarea muy costosa.(Posts by Mazdak Rezvani, 2017)

La metodología que promete superar la barrera de la etiquetación manual, es el aprendizaje no supervisado, este tipo de algoritmo es capaz de aprender sacando conclusiones sobre la semántica intrínseca en los datos, es decir, lo que se busca es procesar millones de datos no estructurados y construir de forma autónoma un estructura.

Esto lo hace mediante las redes neuronales, las cuales son capaces de detectar los atributos más significativos de los datos a procesar, esto sería semejante al proceso de etiquetamiento, una vez se obtienen los atributos es cuando se inicia la fase de clasificación (supervisado) y optimización del modelo. Por esto es vital la implementación de este tipo de algoritmo en el paradigma del aprendizaje profundo.

El aprendizaje profundo está enfocado en la recopilación de conjuntos grandes y estructurados, con el cual se le creara una red neuronal, tras el entrenamiento la red mapea directamente de la entrada a un conjunto fijo de salidas que se conocen de antemano. Con este enfoque tan estático sería imposible mantener una conversación realista entre el chatbots y una persona.
Son muchas las posibilidades que ofrece esta tecnología pero por si sola no será capaz de superar los desafíos a los que se enfrentan los chatbots. El entendimiento del contexto, la falta de ambigüedad entre las diferencias en los lenguajes, el empleo del razonamiento lógico, etc.

Aun nos encontramos en las primeras etapas de lo que será la revolución conversacional impulsada por la IA. Es por esto que debemos combinar múltiples disciplinas, como el PNL, las estadísticas, el aprendizaje profundo y otras, creando una corriente tecnológica capaz de adaptarse a las necesidades del consumidor.

2.2.7. Chatbots para la educación

En el ámbito de la educación los Chatbots presentan nuevas formas de transmitir el conocimiento a los estudiantes, administra la información para que sea más fácil acceder a ella, mediar chats entre los alumnos, etc.

A continuación enumeramos posibles utilidades de los Chatbots en el ámbito de la educación:

- 1. Admisiones

  Los bots serán los encargados de tramitar la admisión de un alumno a la universidad y confirmar si ha sido o no seleccionado. Una vez confirmada su plaza el alumno recibirá toda la documentación necesaria sobre inscripción a la universidad, con la información de su grado, horarios, pagos de matrícula, etc.

- 2. Académicos

  El bot proporcionara toda la información académica del estudiante, como las asignaturas que tiene asignadas, los créditos matriculados, los horarios de las asignaturas, las notas, etc.

- 3. Comunicación

  La comunicación entre los alumnos y sus profesores es vital, por eso creemos que es importante que sea fluida y fácil de usar. Con esta aplicación se podrían crear grupos de chat entre los alumnos de la clase y también con el profesor que las imparte. Otra aplicación es a la hora de

mediar en un debate entre los alumnos evitando así comportamientos poco correctos.

- 4. Becas

  Con la información obtenida al solicitar la admisión del alumno, el bot podría informar a los estudiantes de las posibilidades de ayudas financieras. Informales sobre los plazos de suscripción y la información relevante para acceder a la beca recomendada por el bot.

- 5. Selección de asignaturas optativas

  El bots se basara en el horario de las asignaturas del estudiante para recomendarle las optativas que más se ajustan a su horario actual. El estudiante elige la optativa que más le agrade y el bot lleva a cabo la inscripción al curso y actualiza el horario del estudiante.

- 6. Gestión de las tutorías

  El estudiante podrá seleccionar las tutorías a las que quiere asistir y el bot notifica al profesor para concertar una reunión e informara sobre el tema que tratara en esta. El profesor comunicara si es posible realizar la reunión y en caso de ser así se añadirá al calendario, si esta no fuera posible el bot abriría un chat entre el profesor y el alumno para que acordasen una cita más adecuada.

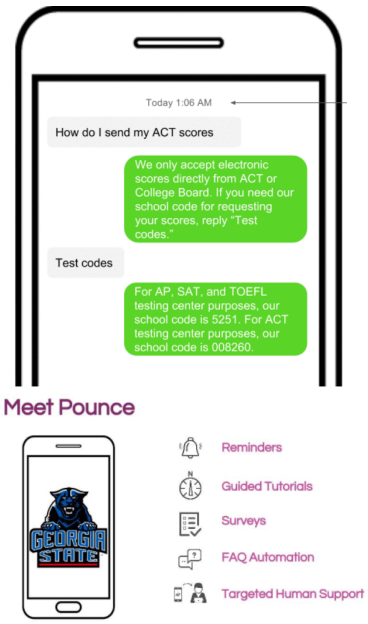
- 7. Gestión de pruebas académicas

  El bot informa de las evaluaciones inminentes de las asignaturas cursadas, el horario y la clase donde se darán lugar.

En este apartado enseñaremos proyectos concretos donde los Chatbots se han hecho un hueco en la educación actual.

En primer lugar queremos hablar sobre el proyecto CourseQ (universidad de Cornell, 2017) utilizado por la universidad de Cornell que tiene como objetivo ayudar tanto a profesores como alumnos, ofreciendo información sobre el campus de una forma más fácil y rápida. CourseQ ofrece varias funcionalidades, como mandar mensajes de texto a la clase, al departamento o a un grupo de alumnos. También gracias a la mensajería pueden realizarse de forma sencilla encuestas, por no hablar del fácil y rápido acceso a información de la universidad, programar tareas, fechas de vencimiento y un largo etcétera.

En este mismo ámbito se creó el Chatbot AdmitHub (universidad de Georgia, 2017) en la universidad de Georgia que intenta hacer el cambio del colegio a la universidad más llevadero a los estudiantes, proporcionando a los nuevos estudiantes tutoriales guiados, recordatorios, encuestas, etc.



Bot de AdmitHub

En la Universidad Tecnológica de Georgia han dado un paso más en lo que se refiere a la integración de los chatbots en la educación. Jill Watson (el chatbot) es una profesora de la universidad que imparte clases online sobre diseño de programas informáticos, entre sus habilidades encontramos que puede responder preguntas de una forma inteligente al alumno, dar explicaciones y plantear dudas. Pero lo realmente curioso de este chatbot es que se ha hecho pasar por un profesor de verdad durante meses y ningún alumno llego a percatarse de que su profesora era en realidad un chatbot.

Jill Watson es un chatbot creado por los investigadores del Georgia Tech con la ayuda de la plataforma de inteligencia artificial Watson de IBM. El año previo a su publicación los investigadores pusieron a analizar 40.000 comentarios de un foro de discusión denominado "Piazza", con lo cual el bot obtuvo un aprendizaje previo que le otorga una serie de preguntas y las respuestas relacionadas, haciendo que el bot fuese bastante consistente desde sus inicios. Cuanto más conversaba con los alumnos más precisas eran sus respuestas.

Un ejemplo distinto a los anteriores seria Duolingo, es una plataforma de aprendizaje de idiomas que basa su actividad en enseñar distintos tipos de idiomas utilizando su aplicación de aprendizaje. Como cada estudiante es diferente y tiene una forma distinta de aprender, esta empresa ha desarrollado un bot con el cual el usuario podrá conversar en el idioma que se quiere aprender y, conforme vayas chateando con el bot este ira trazando patrones con los datos que va obteniendo de la conversación, así el bot empezara a dialogar de una forma más fluida y adaptada a tu forma de comunicarte

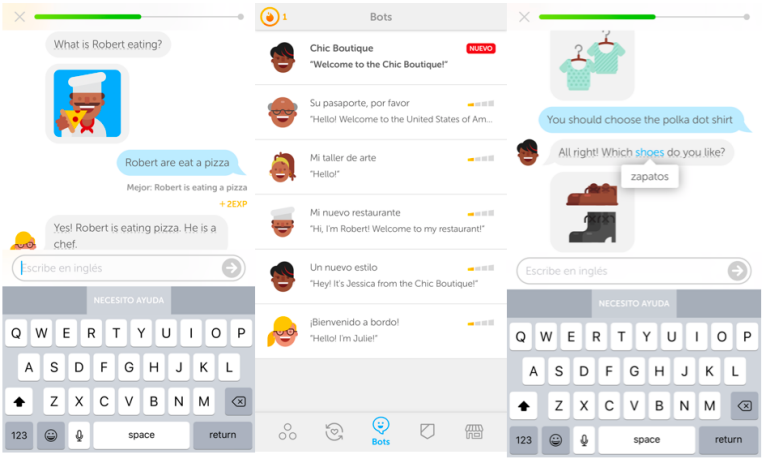


Figura 2.4: Capturas de la aplicación Duolingo

y aprender. Podemos ver un ejemplo de funcionamiento de dicho bot en la figura 2.4

2.2.8. Chatbot de Facebook Messenger

Facebook Messenger (Facebook, 2017) es un servicio de mensajería instantánea, donde sus funciones más destacas son, él envió de mensajes entre los usuarios, las llamadas de voz y video llamadas tanto con dos usuarios como en conversaciones grupales.

En abril de 2016, Facebook introdujo una API para poder desarrollar chatbots en Messenger, y así dotar a los desarrolladores y empresas de una herramienta que les permita exprimir al máximo las posibilidades de Facebook Messenger en sus negocios, como por ejemplo mejorar el servicio de atención al cliente o un bot que dé el parte meteorológico.

Al año siguiente Facebook demostró de nuevo que está apostando por esta tecnología, habilitando un asistente virtual M para sus usuarios (por ahora solo en EE.UU). El cual es capaz de escanear en los chats las palabras claves y con ellas sugerir acciones determinadas. Además, incorporó chatbots en los grupos de Messenger como Çhat Extensionsz una pestaña "Discovery"para mejorar la búsqueda de bots y habilitando códigos QR especiales que, cuando se escanean, llevan al usuario a un bot específico.

Para poder crear nuestro bot en Facebook necesitamos realizar una serie de pasos:

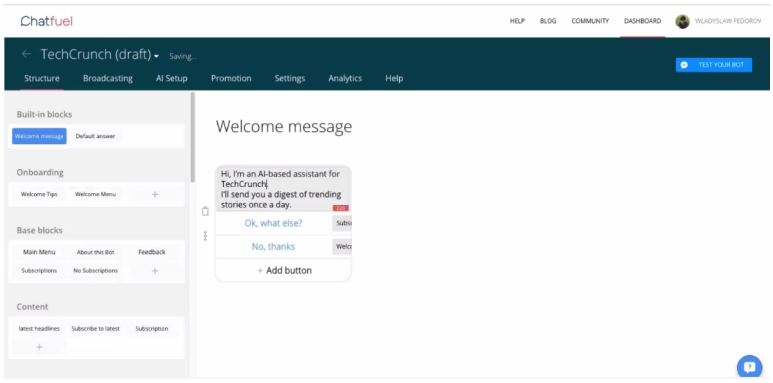


Figura 2.5: Captura de la plataforma ChatFuel

- Crear fan page: https://www.facebook.com/pages/create, donde nos ofrecerán varios tipos de páginas (Lugar o negocio local; Empresa, organización o institución; entretenimiento; etc)
- Registrarse en facebook for developres y creamos una aplicación Messenger
- Token: añadimos nuestra página y se nos genera un token que nos identifica.
- Webhook: esta es una "url" de nuestro proyecto que va a estar a la escucha de eventos de terceros, en nuestro caso nos enfocamos en "messages"

Con estos sencillos pasos ya tenemos configurado nuestro bot para poder empezar a programar sus diversas funcionalidades. Dependiendo del alcance de nuestro chatbot podemos desarrollarlo de una forma más compleja o más simple. Para un proyecto de mayor envergadura necesitaríamos tener conocimientos de informática avanzados, en cambio si el proyecto no requiere funcionalidades complejas este puede ser desarrollado mediante una aplicación de terceros con lo cual no necesitaremos tener conocimientos de programación programar, como por ejemplo chatfuel(chatfuel, 2017) como vemos en la camptura de la figura 2.5 o botsify(botsify, 2017).

2.2.9. Chatbot de Telegram Messenger

Telegram Messenger(Telegram, 2017c) es un servicio de mensajería por internet que a diferencia del resto de servicios ha compartido tanto su código como su API(Telegram, 2017b) con los desarrolladores. Son varias las funcionalidades que han hecho famosa a esta aplicación, las más destacables son los "chat secretos" y la plataforma de bots.

En los inicios de la plataforma los bots de Telegram emulaban servicios, como por ejemplo la gestión de los "stickers". Fue entonces cuando utilizaron esta misma tecnología para ayudar a los usuarios a desarrollar sus propios bots de forma sencilla.

Los bots que encontramos Telegram (Telegram, 2017a) son como usuarios autónomos, es decir, simulan ser una persona que utiliza la plataforma (como nosotros), pero con varias diferencias relevantes, que los diferencian de las personas, como:

- Los bots no tienen estado en línea ni última visualización, la interfaz muestra la etiqueta 'botén su lugar.
- Los bots tienen un almacenamiento en nube limitado: los mensajes antiguos pueden ser eliminados por el servidor poco después de haber sido procesados.
- Bots no puede iniciar conversaciones con usuarios. Un usuario debe agregarlos a un grupo o enviarles primero un mensaje. Las personas pueden usar "telegram.me/ <bot-username>" enlaces o búsqueda de nombre de usuario para encontrar su bot como se ve en la figura 2.6.
- Los nombres de usuario de los bot siempre terminan en 'bot'(por ejemplo, @TriviaBot, @GitHub-bot).
- Cuando se añaden a un grupo, los bots no reciben todos los mensajes de forma predeterminada.
- Los bots nunca comen, duermen o se quejan (a menos que hayan sido programados de otra manera).

Lo primero que tenemos que hacer para interactuar con un chatbot de Telegram es añadirlo al grupo de conversaciones o enviarle un mensaje. También podemos ir al perfil del bot y utilizar los botones "Añadir a grupo", "Compartir". Una vez hemos iniciado una conversación podemos interactuar con ellos de distintas formas, por ejemplo mediante botones interactivos (menú), con palabras clave " horarios. . . " o mediante una conversación fluida (inteligencia artificial) como podemos ver en la figura 2.7.

Lo que más nos ha llamado la atención de esta plataforma es la facilidad con la que podremos crear un chatbot a través de la API de Telegram Bot.

En primer lugar el desarrollador no requiere conocimientos avanzados sobre protocolos de conexión u otras técnicas, ya que la plataforma provee al usuario con todos estos elementos fundamentales para la conexión y comunicación. El protocolo utilizado por Telegram para la conexión es el MTProto (servicio para transportar archivos), este servicio gestiona la encriptación y

Figura 2.6: Ejemplo de busqueda de bots en la aplicación para smartphone de Telegram

comunicación con la API de Telegram. Además tenemos a nuestra disposición un kit de desarrollo para la implementación del chatbot.

El proceso de creación de un chatbot será guiado mediante otro chatbot llamado @BotFather, en primer lugar usaremos el comando "/newbot", con este iniciaremos un dialogo con el cual podremos configurar el chatbot. Lo primero que hará @BotFather es preguntarnos por el nombre de nuestro chatbot, este nombre será el que figure en los detalles del contacto, después nos pedirá el nombre de usuario (@testbot), este será link con el que los usuarios de Telegram podrán añadir nuestro bot a una conversación. Como podemos observar en la figura 2.8 con estos sencillos pasos ya hemos creado y configurado el chatbot.

Una vez configurado podremos empezar a darle forma y, es que son varias las posibilidades que se nos ofrecen para desarrollar el bot, tanto en los lenguajes que podemos utilizar, como las librerías que contienen las funciones básicas del bot. Uno de los leguajes más comunes para el desarrollo es el de Python, por otro lado "Telepot" es la librería con más apoyo actualmente.

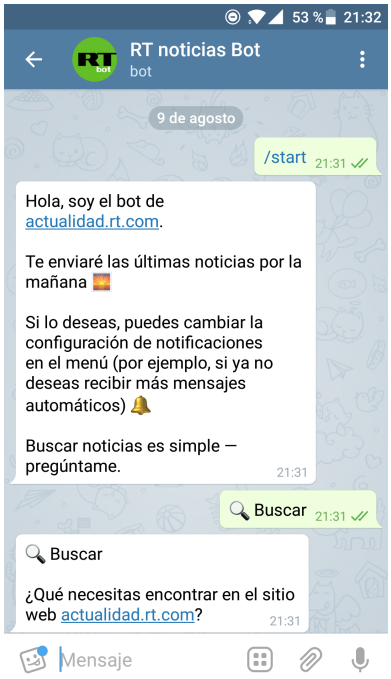


Figura 2.7: Ejemplo de uso de bot RT Noticias

2.2.10. Chatbots de interés de estas dos diversas plataformas

Facebook Messenger (Serrano, 2017)

- Sequel Stories: es un juego donde debemos seguir la historia que nos esta contado el bot y elegir entre las opciones que nos va ofreciendo.
- Job Pal: este chatbots está enfocado a encontrarte trabajo, le contamos que tipo de trabajo buscamos, la zona, etc y él nos ofrecerá una serie de ofertas que podemos elegir.
- HealthTab: es un chatbots que ofrece consultas médicas, con previo disclaimer, en la contamos nuestros síntomas al bot y él nos sugiere unas posibles soluciones a nuestros problemas, y si no quedamos satisfechos podemos pedir al chatbot que nos ponga en contacto con un médico con el que poder chatear.
- Instalocate: es un bots que nos informa sobre nuestro vuelo, si se ha retrasado, cuando sale, donde, etc.
- Memegenerator bot: este chatbot genera memes a partir de una imagen seleccionada y de un texto.

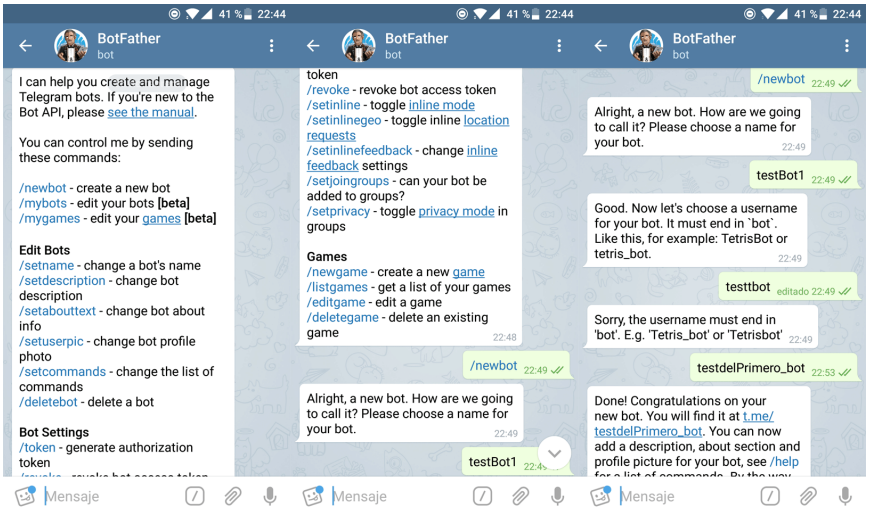


Figura 2.8: Proceso de creacion del Bot con BotFather

- CNN: un chatbot que nos informa de todas las noticias que nos interesen.

Telegram (García, 2017)

- Gmail bot: con este bot somos capaces de enviar y recibir emails, descargar o adjuntar documentos, en definitiva es como tener nuestra aplicación email en un chat.
- Twitter: con este chatbot podremos prácticamente sustituir nuestra aplicación de twitter, ya que nos permite desde revisar nuestro timeline, hasta publicar tuits.
- YouTube Audio Downloader: este bot nos ofrece la posibilidad de descargar fácilmente el audio de los videos publicados en youtube.
- Feed Reader Bot: un lector de RSS que nos permite añadir feeds (url), como facebook o cualquier página public.
- Babelgram: es un bot que imita a la aplicación de Google Translate, con la cual podremos traducir palabras de cualquier idioma.
- Alert bot: con el cual podremos programar alertas.

Como podemos observar todos estos tipos de chatbots buscan de una u otra forma sustituir a las aplicaciones originarias, ofreciendo dentro de un mismo entorno todas las funcionalidades que tenemos de forma dispersa en distintas aplicaciones.

2.3. Sistema de información de la UCM

2.3.1. Introducción

En los tiempos actuales, es imprescindible que las universidades dispongan de una infraestructura sólida, que permita la comunicación tanto interna (entre administraciones, profesorado y alumnos) como externa (publicaciones, información sobre servicios y planes de estudio, transparencia en la gestión) y facilite los trámites y el desarrollo tanto de las actividades docentes como de las labores de investigación. Esta infraestructura debe incluir un centro de datos y una red potente sobre la que se cimienten todos los servicios y así permitir la desenvolvencia fluida de las numerosas actividades que se llevan a cabo en todos los departamentos del campus.

2.3.2. Historia

La UCM fue pionera en ofrecer servicios de informáticos en la comunidad educativa. El Centro de Procesamiento de Datos (CPD) fue creado en los años 60, en aquella época el servicio estaba fuertemente centralizado y requería el desplazamiento a las instalaciones, al no existir una red extensa como la actual. Durante los siguientes años, sobretodo en los 80 gracias a la generalización de la informática gracias a los PC, se fueron integrando más y más equipos en muchas áreas administrativas y académicas de la universidad. Pronto se planteó la necesidad de interconectar todos los equipos, así durante el final de los años 80 y principio de los 90, paulatínamiente debido a la seguridad y fiabilidad necesaria para esta infraestructura, se comenzó a crear una red de apoyo tecnológico para la universidad.

(de Nuevas Tecnologías, 2003)

2.3.3. Sistemas actuales

Diferentes sistemas que posée en la actualidad, arquitectura, distribución, funcionalidades, etc. Como se ve en la figura 2.9 el sistema actual es muy extenso y abarca varios campus, colegios mayores y otras instituciones interconectadas.

La universidad ha consolidado sus sistemas informáticos, no sin mantenerse en constante mantenimiento y mejora tan necesaria en los tiempos que corren. Entre su catálogo de servicios se encuentran:

- Servicios de apoyo a la información en la red: servicio de correo electrónico, sistemas de publicación y consulta de información institucional en la web.
- Apoyo al puesto de trabajo: instalación, puesta en marcha y mantenimiento (hardware, SSOO, drivers, etc.) de los equipos necesarios para

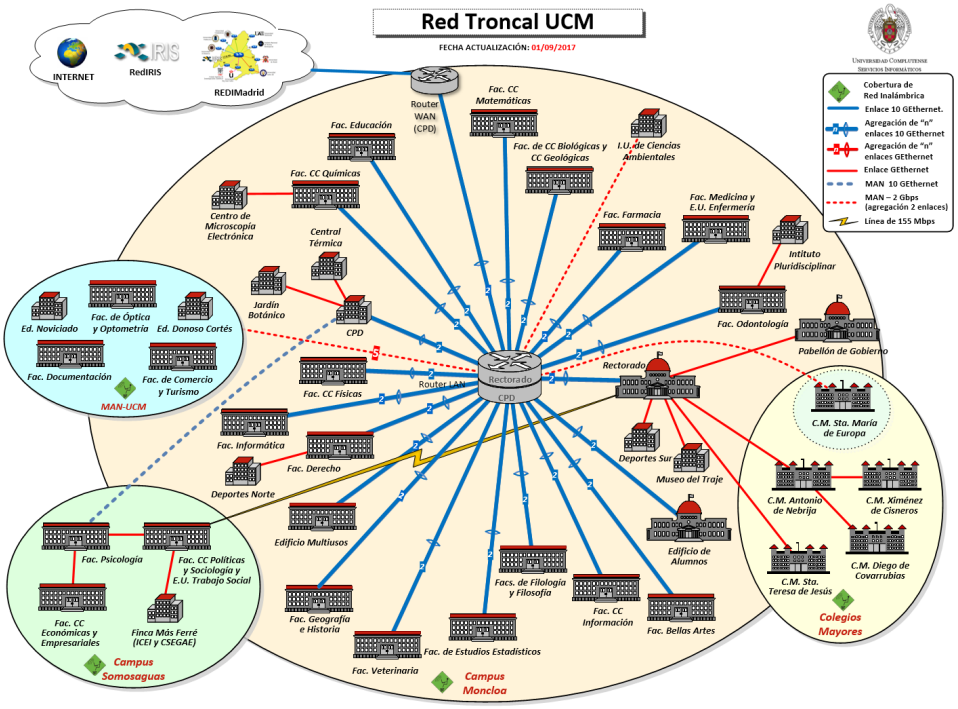


Figura 2.9: Red troncal de la UCM, extraido de la Web de los sistemas de Información de la UCM

todo el personal de la universidad.

- Distribución de Software: Gestión y distribución de licencias (tales como antivirus, software científico (SISOFT).
- Programas de formación: incluidas desde las aplicaciones más comunes y utilizadas como suites ofimáticas hasta software más específico de uso científico.
- Servicios de la biblioteca de la UCM (BUC): web y sistemas de la biblioteca, asi como el desarrollo y puesta en marcha de mejoras en los sistemas de la biblioteca (facilidad de uso, automatización de los procesos, etc.).

Además de estos hay muchos más. Nosotros, debido al ámbito de nuestro proyecto debemos centrarnos en los servicios que se ofrecen a los alumnos (información docente, de instalaciones e infraestructuras y de bibliotecas).

Capítulo 3

Tecnología del BotMentor

En este capítulo hablaremos de la técnicas utilizadas para el desarrollo, la implementación y en general, el funcionamiento del Bot. La decisión de utilizar dichas tecnologías se realizo en etapas iniciales y su finalidad era la de implementar el sistema completo de la forma más realista posible, no contemplando posibles ideas o problemas que fueron surgiendo a medida que avanzó el desarrollo. El sistema completo necesita varias tecnologías en función de cada componente del mismo, por lo que lo dividiremos en 3 partes, cada una de las cuales hace referencia a uno de los subsistemas que propician y permiten el funcionamiento/sostenimiento del bot.

Por un lado se encuentra la fuente de datos para el bot, en nuestro caso es un servicio web. Por otro lado el proceso encargado de recolectar los datos de manera autónoma, para así mantener siempre los datos actualizados en el bot. Finalmente hablaremos de la propia tecnología que utiliza el bot en su implementación.

3.1. Fuente de datos

Para la labor y el funcionamiento de nuestro bot, es imprescindible tener siempre disponible una fuente de datos a consultar, gracias a ella puede ofrecer a los usuarios todas las funcionalidades que posee. Para este proyecto existen multitud de posibilidades para esta finalidad, a continuación, vamos a explicar tanto la funcionalidad necesaria como las distintas tecnologías disponibles para llevarlo a cabo y los motivos que nos han llevado a utilizar las que finalmente fueron elegidas.

3.1.1. Almacén de datos

La información obtenida por el recolector ha de ser almacenada para su posterior consulta. Puesto que la información que precisa el bot, a día de

hoy resulta bastante simple, podría haber sido una buena opción almacenarla con ficheros de texto, debido a su sencillez y/o rapidez. Pero para permitir futuras mejoras en cuanto a funcionalidad del mismo, que requirieran una mayor complejidad de operaciones o cruzamiento de datos, optamos por no utilizarlos. También podríamos haber utilizado fichero XML con estructura o diversos tipos de bases de datos (relacionales, no relacionales, documentales, XML, etc.). Finalmente nos decantamos por una base de datos debido a su facilidad de uso y el dominio que tenemos sobre ellas.

3.1.2. Acceso a los datos

Para acceder a los datos, el bot podría simplemente conectarse remotamente a la base de datos (o cualquiera de los soportes de datos). También se podría desarrollar una API o servicio web. Nosotros decidimos desarrollar un servicio web, para abstraer la lógica de las peticiones, es decir, el bot genera una petición de información y recibe una respuesta, de manera encapsulada , sin saber donde, ni de que manera se encuentra almacenada dicha información. Como se ha mencionado anteriormente, esto es útil para encapsular y abstraer esta parte del código y así permitir nuevas fuentes con tecnologías más modernas. Nosotros estamos usando un servicio web, pero se pueden incluir aquí todo tipo de APIs (REST, WSDL, SOAP, XML) o estándares de computación distribuida como CORBA o RMI. (W3C, 1999)

3.1.3. Servicio web

Nuestra opción ha sido la de la utilización de un servicio web mediante el protocolo SOAP escrito en JAVA. Utilizamos JAVA debido a la sencillez en la creación de estos sistemas, la experiencia poseída en este tipo de aplicaciones y debido a su seguridad y portabilidad. Hemos utilizado el protocolo SOAP debido a su uso extendido, la posibilidad de ser usado con cualquier lenguaje de programación/plataforma y puede ser transmitido por muchos protocolos (HTTP, TPC, SMTP...).

Éste servicio envía mensajes SOAP codificados en XML, en función del método al que se llame, este envía clases JAVA (profesor, tutoría, clase) con diferentes campos:

Ejemplo de mensaje SOAP:

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:consultarGruposResponse xmlns:ns2="http://presentacion/">
            <return>
                <curso>3</curso>
```

```
                <grado>GII</grado>
                <grupo>A</grupo>
                <id>46</id>
            </return>
            <return>
                <curso>3</curso>
                <grado>GII</grado>
                <grupo>B</grupo>
                <id>47</id>
            </return>
            <return>
                <curso>3</curso>
                <grado>GII</grado>
                <grupo>C</grupo>
                <id>48</id>
            </return>
            <return>
                <curso>3</curso>
                <grado>GII</grado>
                <grupo>D</grupo>
                <id>49</id>
            </return>
            <return>
                <curso>3</curso>
                <grado>GII</grado>
                <grupo>E</grupo>
                <id>50</id>
            </return>
            <return>
                <curso>3</curso>
                <grado>GII</grado>
                <grupo>I</grupo>
                <id>51</id>
            </return>
        </ns2:consultarGruposResponse>
    </S:Body>
</S:Envelope>
```

Mensaje SOAP devuelto por el método ConsultarGrupos transportado como un mensaje SMTP.

Algunos métodos utilizan sobre-escritura en función de los parámetros de entrada, que pueden variar en función del uso que se haga de ellos. A continuación definimos los métodos con sus parámetros de entrada y salida. La funcionalidad del servicio web incluye los siguientes métodos:

- Consultar Horario

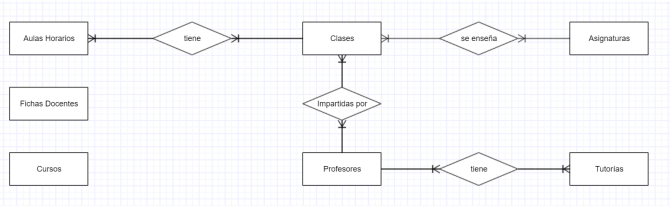


Figura 3.1: Diagrama de entidad-relación

- Consultar Tutorías
- Consultar Profesor
- Consultar Clase
- Consultar Fichas docentes

El servicio web consulta estos datos de una base de de datos SQLite, decidimos utilizar esta base de datos debido a la potencia, ligereza y portabilidad que brinda este sistema, no es necesario montar un servidor para poder utilizarla, tan solo tener acceso al fichero en el que se almacena y los drivers de conexión JDBC.

3.2. Base de datos

Como antes mencionamos, el recolector almacena los datos en la base de datos, el servicio web permite efectuar las operaciones necesarias de consulta sobre ella y devolver toda la información que solicita.

Para ello, se decidió utilizar una base de datos relacional3.1, lo más sencilla posible. Entre las opciones más comunes (MariaDB, MySQL, ORACLE, SQLSERVER, HSQL, etc.) se decidió usar SQLite, por su facilidad de uso, porque no requiere instalación alguna, se almacena en un solo fichero, esto le permite portabilidad absoluta, tan solo es necesaria la librería con los drivers de JAVA integrados en la aplicación. Además de estas ventajas, acepta bases de datos de hasta 2 TB y es más ágil y tiene menor latencia, debido al uso de subrutinas y funciones en lugar de procesos. Adicionalmente tiene librerías para la mayoría de lenguajes de programación que se usan habitualmente (JAVA, VB, .Net, Python, PHP, C/C++...) y de hecho se usa en multitud de aplicaciones y sistemas como Android, Google Chrome, Skype, IOS...

3.3. Recolector de datos

3.3.1. Introducción

La recolección de datos es un proceso fundamental para el correcto funcionamiento del chatbot. Como ya mencionamos en capítulos anteriores los chatbots requieren de una gran cantidad de datos estructurados para poder mantener un dialogo con el usuario. Para resolver el problema de la recolección de información, hay diferentes tipos de soluciones.

En primer lugar se pensó en la posibilidad de que el cuerpo técnico de la facultada de informática implementara una solución API Rest o un servicio web para poder solicitar la información relevante. Esta posibilidad se descarto debido a varios inconvenientes, para empezar se requiere tiempo y recursos para desarrollar el servicio web o la API Rest, supusimos que el cuerpo técnico no dispone de tal tiempo y recursos para un proyecto de TFG, además se tendría que hacer con muchísimo cuidado para evitar abrir agujeros de seguridad y crear problemas de integridad de datos ya que estarían dando un acceso bastante directo a una parte de sus datos de manera descontrolada. Esta hubiese sido la opción más práctica y con miras al futuro de la expansión y continuidad del bot, la más realista, dado que abstendría al equipo de desarrollo del bot de como se almacena o se gestiona toda la información.

La segunda alternativa propuesta al personal técnico es la cesión por parte de la facultad de una serie de documentos XML con la información relevante para el proyecto. Pero debido a una serie de circunstancias solo se puede obtener una pequeña parte de la información requerida, por lo que necesitamos recurrir a otro tipo metodologías, para completar todos los datos fundamentales. Esta alternativa hubiera sido menos cómoda que la anterior, debido a que tendría que automatizarse la aportación de los XML actualizados o bien hacerlo manualmente. No obstante la ventaja de los XML es que al tener todo etiquetado y ser un documento estructurado, sería fácil de manipular e indexar para ser usado por el sistema del bot.

La tercera y última alternativa que planteamos en el proyecto para la obtención de los datos, es la técnica denominada Web Scraping que explicaremos a continuación.

3.3.2. Web Scraping

El web scraping es un conjunto de técnicas capaces de obtener el código HTML de una o varias páginas web, con el objetivo de recopilar grandes cantidades de información que posteriormente serán analizadas y empleadas

en diferentes aplicaciones, como por ejemplo para obtener una serie de contactos, para la indexación de páginas web que utiliza google en su buscador, la minería web, la monitorización de precios en línea o la comparación de ellos, etc.

En definitiva el web scraping es una metodología para copiar el contenido de las páginas web y almacenarlo, para luego recuperarlo y analizarlo. En nuestro caso queremos extraer el contenido de la página web de la Facultad de Informática, analizarlo y guardarlo en la base de datos racional. Para esta complicada tarea hemos elegido Beautiful Soup una librería de Python que nos facilitara mucho la labor de scraping, esta es capaz de extraer datos de archivos HTML y XML.

3.3.3. Procedimiento

En primer lugar capturamos los archivos HTML de la página web. Una vez tenemos el código HTML deberemos analizar la información y extraer la que más nos interese. Para la extracción usaremos una serie de métodos predeterminados (librería) con los cuales podremos extraer las etiquetas y sus datos.

Código HTML (una parte del código)

```
<ul>
    <li> Informaci n relevante1 </li>
    <li> Informaci n relevante2 </li>
    <li> Informaci n relevante3 </li>
</ul>
```

Ejemplo:

"exli = html.find-all('li')"

La variable "exli" tendrá una array con todas la etiquetas "li", si recorremos mediante un for el array podremos sacar el texto relevante de cada "li".

exli = [
Información relevante1 , Información relevante2 , Información relevante3]

Ejemplo:

```
for  l in exli:
       p = l.find(text=True);
```

Al recorrer el array podremos ir sacando el texto concreto de cada etiqueta (html)

p = Información relevante1

3.3.4. Inconvenientes de esta tecnología

El web scraping presenta varios inconvenientes que hacen poco robusta la recolección de datos y su manutención a largo plazo. Como hemos observado en el apartado anterior para obtener los datos de interés necesitamos analizar el código HTML y sacar los datos relevantes. Este es el primer problema que surge, ya que el procedimiento es muy tedioso y lento, que además se deberá efectuar en cada página que contenga datos relevantes para el proyecto. Si el proyecto es pequeño esto no llegaría a ser un problema, pero si este es de mayor envergadura la tarea de análisis podría llegar a ser inviable.

El segundo problema y más relevante para el proyecto está relacionado con el cambio, porque el cambio es constante en las páginas web y, esto supondría un continuo re-análisis que no haría sostenible el proceso de recolección de datos. Por ejemplo durante el desarrollo del proyecto la Universidad impuso un cambio de diseño en las páginas web de la FDI pero afortunadamente, no afectó mucho a la representación de los datos de interés en la versión implementada de la aplicación.

En nuestro caso debemos tener en cuenta el año que se cursa para poder analizar las páginas y automatizar la selección de páginas a scrapear, pero si por cualquier motivo el código html cambiase deberíamos re-analizar esa página en concreto. Esto supone un problema a largo plazo que podría afectar a la eficiencia del proyecto.

3.3.5. Programación de la Tarea

La programación de la tarea de scraping se hará mediante el software nativo del Nas synology, visible en la figura 3.2 en el cual estarán subidos todos los ficheros relacionados con el proyecto BotMentor. Con esta aplicación podemos programar cualquier script para que se ejecute cada día a una hora determina. En nuestro caso hemos decidido scrapear las páginas web de la Facultad de Informática de la UCM todos los días a las 00:00 horas.

3.4. Sistema del bot

El proyecto BotMentor está pensado para funcionar en varias plataformas y tecnologías diferentes, con el objetivo de ser una aplicación versátil.

Lenguaje de programación: Python es un lenguaje multiparadigma, debido a que soporta tanto la orientación a objetos, como la programación interactiva. También es un lenguaje interpretado, que usa tipado dinámico y

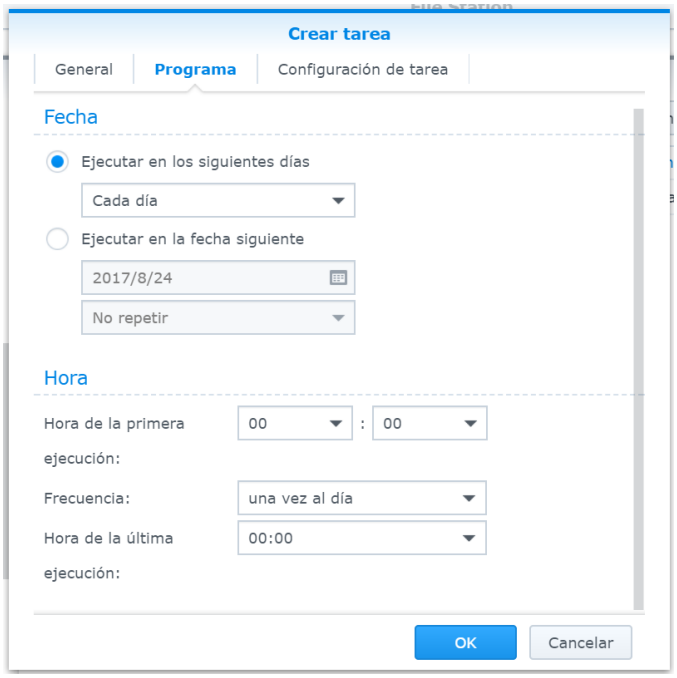


Figura 3.2: Formulario de creación de tarea mediante el software de Nas Synology

es multiplataforma. Se ha elegido este lenguaje, debido a la variedad de frameworks y documentación existente para el desarrollo del bot. En particular nos facilita enormemente el uso de la API de Telegram.

Plataformas: La plataforma Telegram Bot API permite a los desarrolladores conectar fácilmente el bot con la aplicación de mensajería Telegram. Estos bots son cuentas especiales que no requieren de un número de teléfono adicional.

El servidor de Telegram provee la comunicación y encriptación con la API de Telegram, utilizando el protocolo MTProto, que como ya comentamos en el último capítulo el uso de este sistema de comunicación se abstrae, mediante un kit de desarrollo que facilita enormemente la comunicación.

Librerías:

- Telepot:una librería que facilita la utilización de la API de Telegram mediante un framework implementado en Python de los métodos de la API, sin tener que preocuparse de los mensajes JSON que se envían y

reciben a los servidores de Telegram.(Lee, 2015b)

- Suds: una librería de Python, el cual está desarrollado como cliente ligero para el uso de mensajes SOAP y, así poder facilitar la interconectividad entre el bot y el webservice.(Ortel, 2015)
- ChatterBot: (Cox, 2017) es una librería de Python que utiliza algoritmos de aprendizaje para generar respuestas "inteligentes" al usuario. Esta librería nos facilita el desarrollo de nuestro chatbot y nos ayuda automatizar las conversaciones entre el bot y los usuarios.

  Las características más importantes de esta librería son la independencia del idioma y el aprendizaje continuo, que la convierten en una librería muy versátil.

  En primer lugar el idioma que vaya usar nuestro bot es totalmente independiente del diseño de la librería, esto nos da la capacidad de usar cualquier idioma para el aprendizaje del chatbot y su posterior conversación con el usuario. Por otro lado está la posibilidad de ir mejorando el chatbot conforme se interactúa con él, ya que el aprendizaje del bot nunca termina.

  El chatbot comienza sin ningún tipo de conocimiento, es decir la base de datos esta vacía. Conforme va avanzando su aprendizaje la base de datos va aumentando, cuanto más le enseñemos, más conocimientos tendrá.

  El bot puede recibir un aprendizaje previo con las preguntas-respuestas predeterminadas, o puede ser lanzado directamente para que el usuario lo utilice, lo más recomendable es instruir previamente al chatbot para que desde un inicio puede entablar una conversación con el usuario. Cada vez que un usuario introduce una sentencia, la librería guarda el texto que se introdujo y la respuesta del chatbot. Conforme va avanzando la conversación el chatbot ira incrementa sus datos, por lo que las repuestas del bot empezaran a ser más precisas.

  El programa selecciona las frases (de la base de datos) que más coincidan con la frase de entrada, después de tener las frases seleccionadas elegimos la frase que más coincidencias tenga, de la frase elegida cogemos su respuesta correspondiente, hemos de recordar que cada pregunta tiene asociada una o varias respuestas. Este algoritmo de aprendizaje es parecido al K vecinos más cercanos (en inglés, k-nearest neighbors) (Abdelmalik Moujahid, 2017).

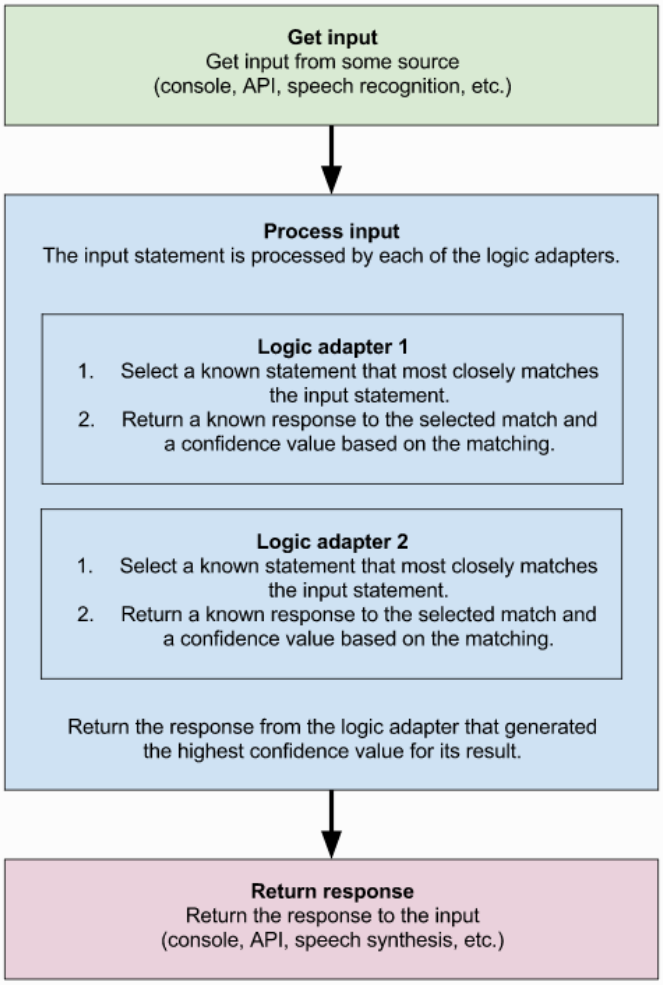


Figura 3.3: Proceso de selección de respuestas del chatterBot, sacada de la página web de ChatterBot

Capítulo 4

Funcionalidad y requisitos

Desde el principio, se fueron estableciendo unos límites y unos mínimos en la funcionalidad del proyecto, puesto que su alcance depende, sobre todo, de la imaginación y del tiempo que se le pueda dedicar. Al ser un sistema que ayuda a los alumnos de la universidad, este se podría ampliar a todas las facultades, a toda su infraestructura y entorno (bibliotecas, eventos, actividades deportivas...).

Decidimos que, al menos, este debía servir a los alumnos de la facultad de informática, proveyendo la información más utilizada por los alumnos a la hora de acceder a la web de la facultad, de manera rápida e intuitiva.

Para definir los requisitos, utilizamos el estándar IEEE Std. 830-1998 (IEEE, 2008).

4.1. Definición del sistema

Este sistema está pensado para necesitar administración mínima, a no ser que se cambien las fuentes de datos o se quiera ampliar sus funcionalidad, y ser autónomo, por lo que sus únicos usuarios serían los (alumnos) que iniciaran y/o utilizaran el bot.
El sistema está dividido en varios módulos:

- Módulo de datos:
  Éste módulo es el encargado de proveer la información que utiliza el bot para realizar sus funciones. Ya sea el servicio web, una API REST, etc., éste módulo debe tener los métodos de consulta bien definidos para facilitar y abstraer la labor del bot de la lógica y de la tecnología del sistema en el que se almacenan los datos.

- Recolector de datos:
  Éste módulo complementa al servicio web, mediante scraping (Martí,

2016) se encarga de, periódicamente y a horas con baja demanda, actualizar la base de datos. Gracias a él la información proveída por el bot es tan veraz como la web de la facultad.

- Bot de Telegram:
  La parte principal del sistema, ya que se encarga de consultar y proveer toda la información a los usuarios. Es la interfaz de usuario que, mediante comandos y/o botones de navegación en la pantalla, permite al usuario configurar su información personal (que grado estudia, en que curso o grupo está matriculado) para consultar datos relacionados con este u otras actividades docentes.

4.2. Especificación del sistema

4.2.1. Diagramas de análisis

A continuación se muestran algunos de los diagramas necesarios durante la fase de análisis del proyecto:

4.2.1.1. Diagrama de despliegue

De manera gráfica, podemos definir el sistema como un conjunto de subsistemas, en el diagrama de despliegue (A.2).

4.2.1.2. Diagramas de clase

Como diagrama de clase fundamental y descriptivo exponemos el modelo de dominio en el diagrama (A.1).

4.2.1.3. Diagramas de casos de uso

A continuación se referencia al único diagrama de caso de uso. Ya que el usuario solamente interviene en su interacción con el bot, el resto de procesos son autónomos. Este diagrama se encuentra en el Apéndice B, sección A.3

Además, como es habitual en todo desarrollo de software, se realizó un análisis de requisitos. A continuación se encuentra la especificación de requisitos (SRS).

4.2.2. SRS

- Bot al servicio web:

  - **Función:** Consultar Horarios
    ID: sw01

Prioridad: Alta
Estabilidad: Alta
Descripción: El usuario consulta horarios de una asignatura o curso.
Entrada: Curso, grupo, grado o nombre de asignatura.
Salida: Mensaje SOAP con información sobre los días, horas y aulas/laboratorios de la(s) asignatura(s).
Origen: Petición HTTP.
Destino: Sistema.
Necesita: Estar conectado a la red.
Acción: Una vez validados los datos, se consulta la BBDD y se devuelve la información requerida.
Precondición: Datos de asignatura o curso válidos.
Postcondición: Envío de mensaje SOAP.

- **Función:** Consultar Tutoría
  ID: sw02
  Prioridad: Alta
  Estabilidad: Alta
  Descripción: Un usuario envía petición de consulta sobre la tutoría de una clase concreta, de todas las asignaturas con un nombre dado o de un profesor en concreto.
  Entrada: Asignatura, curso, grupo y grado o Profesor o Asignatura
  Salida: Mensaje SOAP con información sobre los días, horas y lugares de la(s) tutorías(s).
  Origen: Petición HTTP.
  Destino: Sistema.
  Necesita: Estar conectado a la red.
  Acción: Una vez validados los datos, se consulta la BBDD y se devuelve la información requerida.
  Precondición: Datos de asignatura válidos.
  Postcondición: Envío de mensaje SOAP.

- **Función:** Consultar Profesor
  ID: sw03
  Prioridad: Alta
  Estabilidad: Alta
  Descripción: Un usuario envía petición de consulta sobre el profesor de una asignatura.
  Entrada: Nombre del profesor
  Salida: Mensaje SOAP con información sobre el profesor (tlfn. e-mail, despacho, etc.).

Origen: Petición HTTP.
Destino: Sistema.
Necesita: Estar conectado a la red.
Acción: Una vez validados los datos, se consulta la BBDD y se devuelve la información requerida.
Precondición: Datos de asignatura válidos.
Postcondición: Envío de mensaje SOAP.

- **Función:** Consultar Clase
  ID: sw04
  Prioridad: Alta
  Estabilidad: Alta
  Descripción: Un usuario envía petición de consulta sobre una clase, es decir, una ïnstancia.ªsignatura que se cursa actualmente en un grado.
  Entrada: Asignatura, curso, grupo y grado.
  Salida: Mensaje SOAP con información sobre la clase.
  Origen: Petición HTTP.
  Destino: Sistema.
  Necesita: Estar conectado a la red.
  Acción: Una vez validados los datos, se consulta la BBDD y se devuelve la información requerida.
  Precondición: Datos de asignatura válidos.
  Postcondición: Envío de mensaje SOAP.

- **Función:** Consultar Asignatura
  ID: sw05
  Prioridad: Alta
  Estabilidad: Alta
  Descripción: Un usuario envía petición de consulta sobre una asignatura
  Entrada: Nombre de asignatura o siglas, curso, grupo y grado.
  Salida: Mensaje SOAP con información sobre el profesor de la asignatura.
  Origen: Petición HTTP.
  Destino: Sistema.
  Necesita: Estar conectado a la red.
  Acción: Una vez validados los datos, se consulta la BBDD y se devuelve la información requerida.
  Precondición: Datos de asignatura válidos.
  Postcondición: Envío de mensaje SOAP.

- **Función:** Consultar Ficha docente

ID: sw06
Prioridad: Alta
Estabilidad: Alta
Descripción: Un usuario envía petición de consulta la ficha docente de una asignatura o de un curso completo.
Entrada: Asignatura, curso, grupo y grado.
Salida: Mensaje SOAP con URL de acceso a la ficha docente, en PDF, de la asignatura.
Origen: Petición HTTP.
Destino: Sistema.
Necesita: Estar conectado a la red.
Acción: Una vez validados los datos, se consulta la BBDD y se devuelve la información requerida.
Precondición: Datos de asignatura válidos.
Postcondición: Envío de mensaje SOAP.

- Web Scraping:

  - **Función:** Scraping Tutorías
    ID: ws01
    Prioridad: Alta
    Estabilidad: Alta
    Descripción: Scraping de la página web de Profesores y Tutorías de la facultad de informática de la UCM.
    Entrada: Codigo HTML
    Salida: Matriz que contiene los datos relevantes
    Origen: Petición
    Destino: Sistema.
    Necesita: Estar conectado a la red.
    Acción: Capturar código html, analizarlo y guardarlo en la BBDD.
    Precondición: Página web correcta.
    Postcondición: Información relevante.

  - **Función:** Scraping Asignaturas
    ID: ws02
    Prioridad: Alta
    Estabilidad: Alta
    Descripción: Scraping de la página web Lista de Asignaturas de la facultad de informática de la UCM.
    Entrada: Codigo HTML
    Salida: Matriz que contiene los datos relevantes
    Origen: Petición
    Destino: Sistema.

Necesita: Estar conectado a la red.
Acción: Capturar código html, analizarlo y guardarlo en la BBDD.
Precondición: Página web correcta.
Postcondición: Información relevante.

- **Función:** Scraping Cursos
  ID: ws03
  Prioridad: Alta
  Estabilidad: Alta
  Descripción: Scraping de la página web Horarios por curso y grupo de la facultad de informática de la UCM.
  Entrada: Codigo HTML
  Salida: Diccionario que contiene los datos relevantes
  Origen: Petición
  Destino: Sistema.
  Necesita: Estar conectado a la red.
  Acción: Capturar código html, analizarlo y guardarlo en la BBDD.
  Precondición: Página web correcta.
  Postcondición: Información relevante.

- **Función:** Scraping Fichas Docentes
  ID: ws04
  Prioridad: Alta
  Estabilidad: Alta
  Descripción: Scraping de las páginas webs que contienen las fichas docentes de la facultad de informática de la UCM.
  Entrada: Codigo HTML
  Salida: Diccionario que contiene los datos relevantes
  Origen: Petición
  Destino: Sistema.
  Necesita: Estar conectado a la red.
  Acción: Capturar código html, analizarlo y guardarlo en la BBDD.
  Precondición: Página web correcta.
  Postcondición: Información relevante.

Capítulo 5

Planificación, control y seguimiento

5.1. Planificación Temporal

El proyecto BotMentor se dividió en dos fases a la hora de realizar la planificación, el control y el seguimiento, estas dos fases comprenden el curso desde su inicio en Octubre hasta su fin en Septiembre. Esta división de fases se debe a que podemos diferenciar dos etapas en el desarrollo del proyecto, la primera fue desde el inicio del curso hasta la primera entrega (junio) y la segunda etapa que es desde julio hasta la segunda entrega (septiembre). Queremos diferenciar estas dos etapas porque los modelos de proceso de software que utilizamos son distintos y estos tienen un peso vital en el proyecto.

5.2. Metodología de desarrollo

La primera fase se basó en un modelo de proceso del software clásico, este fue elegido en un principio por la familiaridad y experiencia que los componentes del proyecto sobre este tipo de proceso. En concreto usamos el modelo de cascada que nos permitía ordenar por etapas los procesos a desarrollar, estas etapas son secuenciales, por lo que no se puede empezar una nueva etapa hasta que se acabe la anterior, una vez terminadas todas las etapas se hace una revisión final.

Este modelo nos presentó una serie de ventajas como su probada efectividad y su sencillez, por otro lado este modelo no se adecuaba a las necesidades del equipo, por lo que a lo largo del desarrollo de la primera fase fueron apareciendo compromisos e inconvenientes (provenientes de la actividad académica) que nos llevaron a no poder seguir el modelo de planificación tradicional.

Tras terminar la primera fase, la cual no fue todo lo bien que se podía esperar el equipo decidió reestructurar el planteamiento inicial, lo que nos lleva a la segunda fase.

En la fase II decidimos adoptar una metodología ágil, debido a todos los inconvenientes surgidos. Al ser solo dos los integrantes del grupo y debido a la escalabilidad del proyecto, se decidió utilizar una adaptación de la metodología Scrum (Alexander Menzinsky, 2016), muy utilizado en los tiempos que corren para el desarrollo incremental, por ejemplo, de aplicaciones para smartphone y desarrollos similares al nuestro.

Se eligió debido a que permite un desarrollo incremental del sistema y sus subsistemas. La metodología de Scrum define una serie de roles como (ScrumMaster, Stakeholder, Product Owner...) dado que somos solo dos personas en el equipo, hemos tenido que compartir los roles, por lo que no hay ninguno que encaje específicamente con ninguno de los integrantes. Nosotros mismo definimos los requisitos del proyecto e incluso la idea (Stakeholder). Como ScrumMaster podríamos incluso incluir a nuestro tutor, ya que nos ha ayudado en la elección de tecnologías y a solventar algunos problemas que han ido surgiendo.

Para el desarrollo del proyecto nos hemos marcado varias etapas o "Sprints" de duración variable, normalmente entre una o dos semanas en función de la dificultad o la carga de trabajo que teníamos de los demás temas académicos (exámenes, practicas, proyectos, etc.), realizamos reuniones semanales, que podríamos llamar "weeklys", para ir definiendo las tareas a realizar (de análisis, diseño e implementación) durante el siguiente "Sprint" o consultar el avance de las tareas actuales. En caso de haber problemas con alguna de las tareas, estudiábamos la situación para intentar solventarlo y en caso necesario, solicitábamos ayuda al tutor. Además de las reuniones, estábamos en contacto mediante herramientas de comunicación como Telegram o correo electrónico para seguir el avance de las tareas.

5.3. Gestión de la configuración

Para la realización del proyecto, tuvimos necesidad de utilizar diferentes repositorios o servicios online. Se decidieron utilizar los siguientes sistemas para el control de versiones y acceso a los ficheros:

- Google Drive: para documentación, diagramas y código.
- ShareLaTex: para la documentación relacionada con la memoria.

- GitHub: como repositorio para el código.

5.4. Lineas base

La línea base ha sido una de las mediciones más difíciles de realizar, debido a que las tecnologías y plataformas utilizadas para el desarrollo del proyecto eran desconocidas para los integrantes del grupo, por lo que el diseño inicial era algo difuso. Estos motivos no nos permitieron implantar una línea base, con lo cual no pudimos establecer comparaciones a lo largo del desarrollo, esto supuso un cambio continuo a lo largo del proyecto y un avance más lento.

5.5. Herramientas de planificación y seguimiento

En la primera fase se plantearon varias herramientas de planificación y seguimiento, como Trello, pero tras mucho debatir los pros y los contras de los distintos servicios, llegamos a la conclusión de que necesitábamos una herramienta potente para controlar de forma más exhaustiva la planificación y seguimiento. Es por lo cual que elegimos el software de administrador de proyectos Microsoft Project, que entre sus características principales encontramos la capacidad de asignar tareas, dar seguimiento al progreso y analizar las cargas de trabajo.

La planificación del proyecto se ha dividido en diferentes módulos según su proceso de desarrollo, en concreto hemos querido separar la planificación en dos módulos principales, uno de ellos es el vinculado a la memoria 5.1, por otro lado tenemos el enfocado a la implementación del proyecto 5.2. Al mismo tiempo hemos dividido este en submódulos con los que controlar el desarrollo por etapas concretas según la funcionalidad que vayamos a implementar en dicho momento.

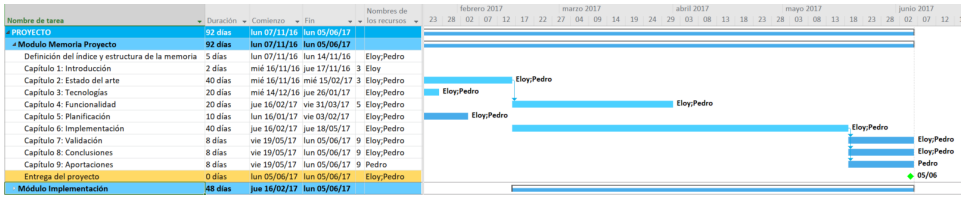


Figura 5.1: Captura del diagrama de Grantt. Módulo memoria.

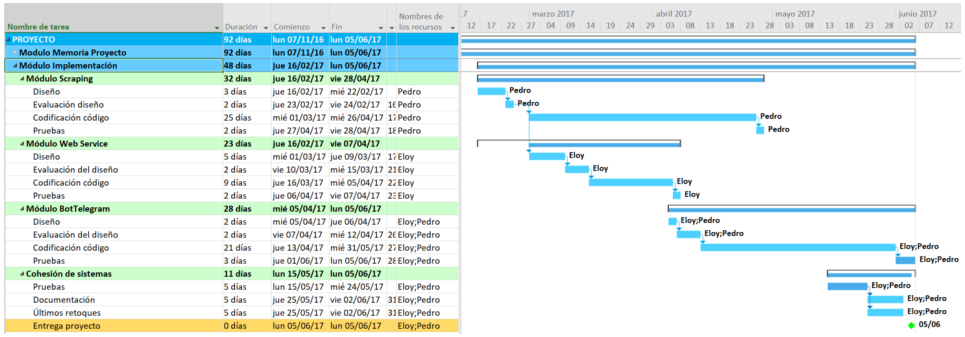


Figura 5.2: Captura del diagrama de Grantt. Módulo implementación.

Como podemos observar en las imágenes los submódulos de la implementación se dividen en cuatro partes, cada una de ellas tiene una relevancia singular en el proyecto.

Los diagramas que mostramos a continuación (figuras 5.3 y 5.4) son referidos a la etapa que concreta el inicio del curso y el final del primer plazo de entrega (Junio).

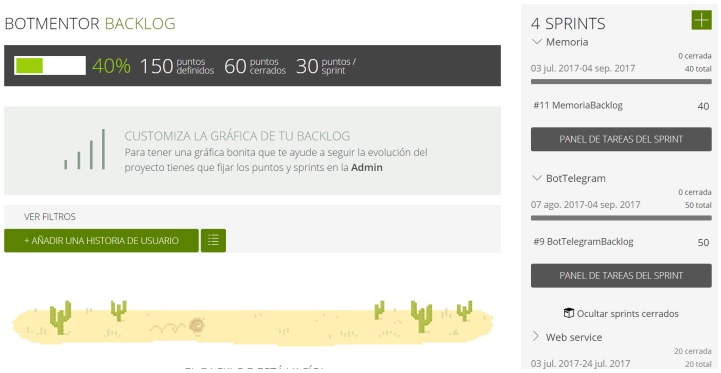


Figura 5.3: Captura de la aplicación Taiga. Backlog y Sprints.

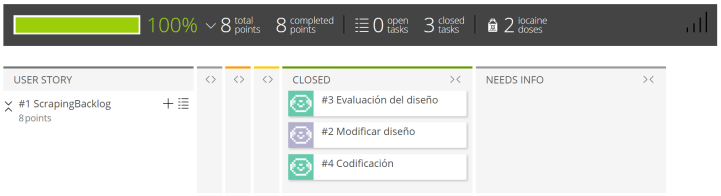


Figura 5.4: Captura de la aplicación Taiga. Task.

En la segunda fase se decidió utilizar una herramienta más acorde al nuevo modelo de proceso, para ello utilizamos la plataforma para manejo de proyectos ágiles Taiga.

Mediante el uso de esta plataforma podemos crear un proyecto el cual se divide en dos etapas claras, la primera es en la que se han de definir los "Backlogs", como se ve en la figura 5.3, en estos estarán los módulos del proyecto, una vez tenemos esta etapa terminada debemos crear los "Sprints", ver figura 5.4, con los cuales seguiremos las tares y su evolución a lo largo del tiempo.

Capítulo 6

Diseño e Implementación

En este capítulo detallaremos los aspectos más técnicos del desarrollo del bot. Para ello, mostramos algunos diagramas y enlazamos a otros diagramas incluidos en el Apéndice A

6.1. Arquitectura

La arquitectura de sistema del BotMentor podría resumirse con el siguiente diagrama: A.2

6.1.1. Patrones de diseño

Hemos utilizado muchos patrones de diseño para el desarrollo, la mayoría en Java, pero también algunos en Python, los más importantes son los siguientes:

6.1.1.1. Singleton

Este patrón es utilizado para permitir una única instancia de la clase FactoriaDAO y FactoríaCommand. (Welicki, 2017). En la figura 6.1 podemos ver el sencillo diagrama de clases.

6.1.1.2. Transfer

Todas las clases del servicio web (asignatura, tutoría, horario, profesor, etc.) tiene un POJO definido (Fowler, 2000), es decir, una clase sin comportamiento que, únicamente tiene los atributos (datos) que la definen. Éste se

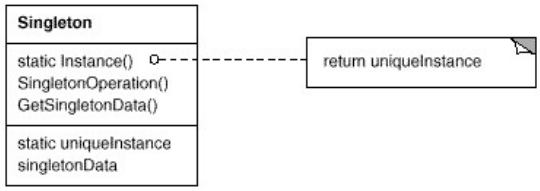


Figura 6.1: Diagrama de clases del patrón Singleton fuente: (of Minnesota Duluth, 2017)

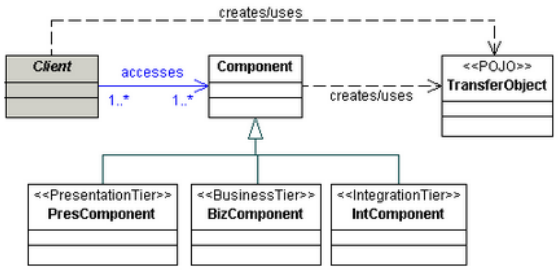


Figura 6.2: Diagrama de clases del patrón Transfer por Miguel Castellanos (Castellanos, 2013b)

utiliza para enviar y recibir datos mediante las comunicaciones SOAP. Podemos consultar su diagrama de clases en la figura 6.2

6.1.1.3. Factory

Se emplea para construir los DAOs y los comandos, esto permite recibir una instancia de la clase DAO/command que se necesite de manera sencilla (**?**). Esta clase utiliza el patrón singleton para que solo pueda ser instanciada una vez.

6.1.1.4. DAO

(Data Access Object) (Chuwiki, DAO) Es un patrón que se utiliza para encapsular el acceso a la base de datos, haciendo que el servicio web sea independiente de la tecnología o base de datos que utiliza. Este provee las operaciones básicas necesarias para consultar toda la información que precisa. Se puede consultar su diagrama de clases en la figura 6.3

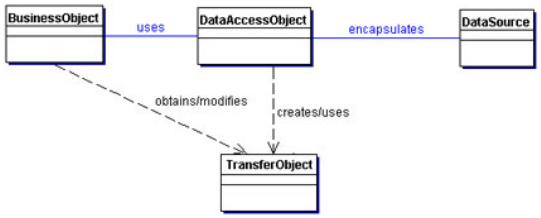


Figura 6.3: Diagrama de clases del patrón Dao por Miguel Castellanos (Castellanos, 2013a)

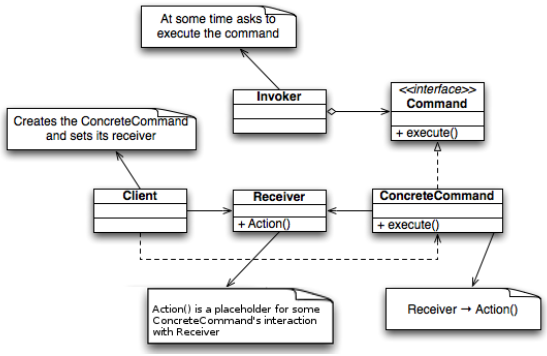


Figura 6.4: Diagrama de clases del patrón Command hecho por el usuario JoaoTrindade de WikiMedia

6.1.1.5. Command

(Erich Gamma y Vlissides, 1994) Es un patrón que permite fácilmente la extensión de las operaciones del bot. Para ello se hizo uso de la herencia, mediante la implementación de la interfaz (aunque en python no existen las interfaces propiamente dichas), para tener el mismo comportamiento en todos los comandos (adaptado en cada uno de ellos).

6.1.2. Frameworks

6.1.2.1. Telepot

Este framework desarrollado en Python y alojado en GitHub, es el encargado de facilitar la implementación del Bot, permite el uso de la API de bots de Telegram de manera nativa en Python. Recibe actualizaciones frecuentes (cuando es actualizada la propia API de Telegram) que permite

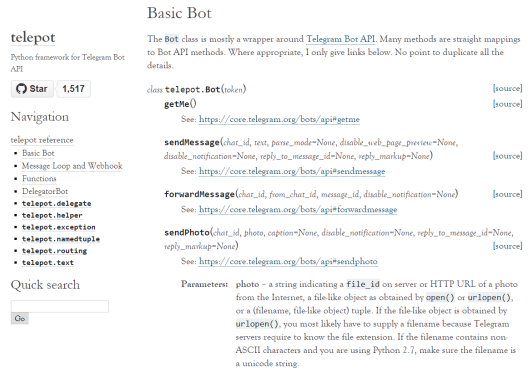


Figura 6.5: Documentación de Telepot(Lee, 2015a)

probar las frecuentes novedades que se van incluyendo. (Lee, 2015b) Dispone de documentación sobre todos los métodos de la API implementados, así como multitud de ejemplos de uso como podemos apreciar en la figura 6.5.

6.2. Diagramas de implementación

Como todo desarrollo de software, ha sido necesaria la realización de diagramas UML para definir su diseño, por lo que en esta sección se agrupan diagramas necesarios para su implementación, como son los diagramas de despliegue, clase, secuencia, actividad, etc.. A continuación mostraremos ejemplos de los diagramas fundamentales:

En esta sección expondremos una muestra de los diagramas más representativos del proyecto, dado que en para la mayoría de funciones del bot el desempeño de las funciones es similar, no aportaría demasiado incluir todos los diagramas de secuencia/actividad.

6.2.1. Diagramas de Secuencia

Diagrama de secuencia de “Consultar horarios”: A.5

6.2.2. Diagramas de actividad

Diagrama de actividad de “Consultar horarios”: A.4

6.2.3. Diagramas de base de datos

Modelo entidad-relación: Figura 6.6:

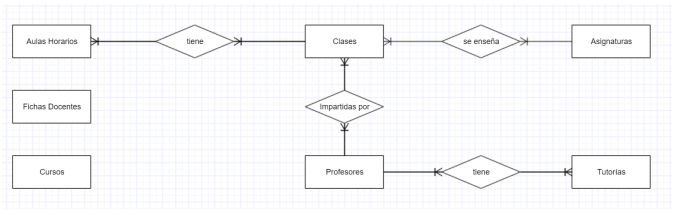


Figura 6.6: Diagrama de entidad-relación

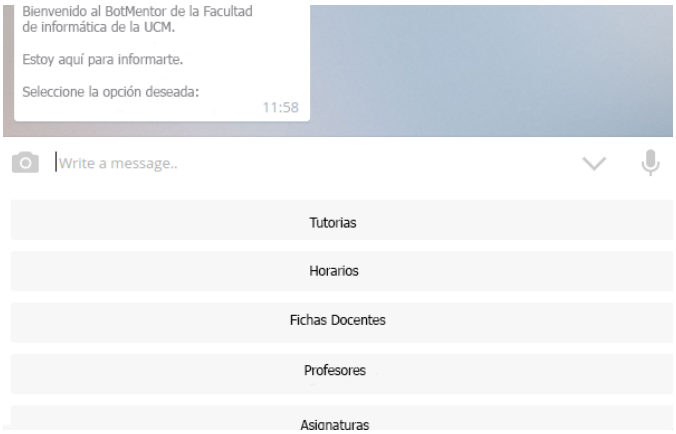


Figura 6.7: Menú contextual con custom Keyboard en el Bot

6.3. Interfaces gráficas

Para el desarrollo de las interfaces gráficas, se han probado diversos tipos permitidos por la API bot de Telegram, entre ellos constan los Custom Keyboards, que sustituyen al teclado alfanumérico convencional del smartphone/tablet y que permiten poner botones para generar menús y submenús como se aprecia en la figura 6.7. También existen los inline keyboards, mas recientes, que aparecen directamente en el chat, asociados a un mensaje (su id) y que por tanto permiten mayor flexibilidad, dado que están más destinados a una conversación grupal, pudiendo centrarse en un mensaje de un usuario concreto, un ejemplo de estos menús podemos verlo en la figura 6.8
.

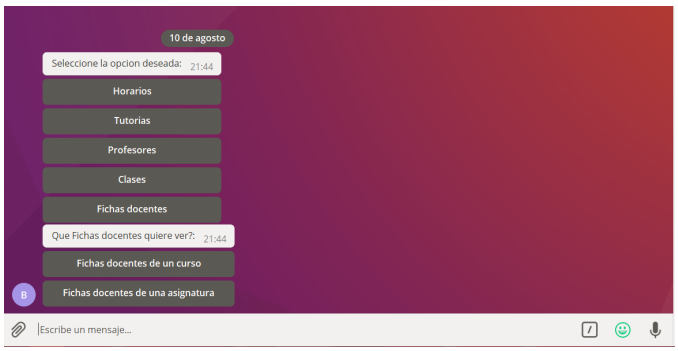


Figura 6.8: Menú contextual con inline Keyboard en el Bot

6.4. Herramientas

6.4.1. Herramientas de modelado

En el diseño del sistema hemos usado herramientas para el diseño de diagramas UML (casos de uso, dominio, actividad, secuencia, etc.) se han utilizado diferentes plataformas dependiendo de la facilidad para generar los diagramas de algunas herramientas o de la parte del sistema que se estaba diseñando :

- IBM Rational Software Architect: Debido a que el servicio web está desarrollado con java, este IDE basado en eclipse permite directamente la creación de código a partir de los diagramas y viceversa. Además de ser muy completo y frecuentemente utilizado en otros proyectos de la universidad.
- Plataforma Gliffy, Sistema online que permite el diseño de diagramas UML, así como Entidad-Relación. (Kohlhardt, 2017)

6.4.2. Herramientas de implementación

Para la implementación del servicio web hemos utilizado el IDE NetBeans 8.2, ya que es un editor muy completo y cómodo. Éste facilita mucho el uso de Glassfish para montar el servidor en el que se aloja el servicio web y su testeo.
Además utilizamos phyton para desarrollar el Bot y el scraping (recolector de datos)

Para la implementación del bot, usamos el IDE PyCharm en su versión gratuita Comunnity Edition, este entorno de desarrollo desarrollado por JetBrains, nos permitió codificar, depurar e integrar el bot con los demás componentes del sistema de manera sencilla. Se puede apreciar en la figura 6.9

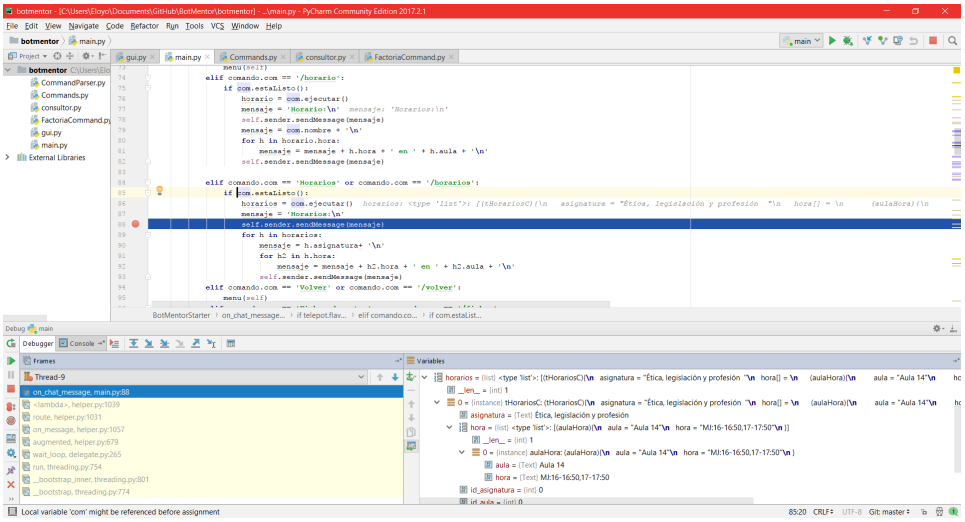


Figura 6.9: Vista del IDE PyCharm mientras se depura

la utilidad como método de depuración.

6.4.3. Herramientas de documentación

Para la redacción de la memoria, decidimos usar el lenguaje LaTeX, este nos permite realizar un documento limpio y profesional. Por sugerencia de nuestro tutor, decidimos también utilizar la plantilla TeXiS(Gómez-Martín y Gómez-Martín, 2009), creada por personal docente de nuestra facultad. Gracias a TeXiS(y Pedro Pablo Gómez Martín, 2009), disponemos del formato típico para tesis, trabajos de fin de carrera y publicaciones de la UCM. Así mismo, para la edición de la memoria, recurrimos a una plataforma web llamada ShareLaTex (ShareLaTex, 2017), que nos proporciona compilación y previsualización online, así como un control de versiones y revisión ortográfica del contenido.

En ocasiones también se ha hecho uso de Google Drive para compartir imágenes, diagramas y versiones previas de la memoria.

6.4.4. Herramientas de comunicación

Como medios de comunicación interna entre los integrantes del proyecto utilizamos Telegram y correo electrónico, hemos elegido Telegram debido a que es multiplataforma (se puede ejecutar simultáneamente en un móvil, un portátil e incluso desde su cliente web, sincronizando mensajes y archivos en todos los dispositivos)(tel, 2014), seguro y permite enviar ficheros de cual-

quier tipo. Para la comunicación con el tutor se ha utilizado únicamente el correo electrónico.

Capítulo 7

Verificación y Validación

Para asegurar que el sistema cumple con los principios de Boehm:

- Verificación: El sistema está construido de forma correcta, es decir, satisface las condiciones impuestas al comienzo de la fase de desarrollo.

- Validación: El sistema construido es correcto, por lo que satisface la especificación de requisitos.

Por lo que para que estas actividades se lleven a cabo, es fundamental haber definido correctamente los requisitos del sistema, así como haber analizado las condiciones que debe soportar en la mayoría de situaciones.
Según los cuatro niveles de integridad que plantea el estándar IEEE 1012-2004, el sistema en general requiere un nivel dos, es decir, el sistema debe ejecutarse correctamente o podría producir comportamientos no deseados, causando consecuencias menores y permitiendo una mitigación con relativa facilidad. (IEEE, 2004)
A continuación se detallaran los planes de validación y verificación que se deben llevar a cabo para asegurar la corrección y la validez del sistema.

7.1. Revisiones

Durante el desarrollo del sistema, fuimos realizando revisiones periódicas, con el mero propósito de ver que se cumplían los requisitos marcados al inicio del proyecto. Estas revisiones comenzamos a hacerlas cuando el sistema ya comenzaba a realizar funciones simples, es decir, en un estado intermedio del desarrollo. Lejos de asemejarse a una RTF convencional, consistían en revisar el funcionamiento de los subsistemas comparándolo con lo documentado en la SRS y en el diseño concebido al inicio del proyecto.

7.2. Pruebas

Al ser un sistema que está distribuido en varios servidores, las interdependencias entre ellos hacen que las pruebas a realizar sean diversas para garantizar su funcionamiento.

Para determinar la calidad de los sistemas tenemos que tener en cuenta diversos puntos críticos en su funcionamiento, si cualquiera de estos no satisface las medidas estimadas, lastrará la experiencia de usuario o incluso la volverá imposible. Entre estos puntos, debemos destacar:

- Tiempo de respuesta del bot: Para una experiencia de usuario satisfactoria, el bot debe tener un tiempo de respuesta aceptable, es decir, entre 1-3 segundos de retardo entre que el usuario envía la petición al bot y éste la responde (suponiendo una buena conexión a internet en el dispositivo desde el que se hace la consulta). Este punto depende a su vez de la fuente de datos, pero el servidor en el que se aloja el servicio que controla el bot, debe tener, aún así, un rendimiento óptimo, por lo que la complejidad de las funciones que utiliza debe ser a lo sumo lineal.

- Tiempo de respuesta de la fuente de datos: Aquí está la clave del buen funcionamiento del sistema, puesto que, suponemos, que es el que mayor carga de trabajo tiene y por tanto un tiempo de respuesta mayor. Independientemente de la fuente de datos que utilicemos (de las antes mencionadas: servicio web, API, etc.), se ha de garantizar un tiempo de respuesta menor que el propuesto para el bot, es decir, menos de 3 segundos. En el caso principal que nos atañe, utilizando un servicio web propio, debemos asegurar que el tiempo de acceso a la base de datos debe ser eficiente.

- Capacidad de usuarios simultáneos: Los servidores en los que se alojan los sub-sistemas que componen el proyecto, deben ser capaces de atender un numero suficiente de peticiones de usuarios. Si el sistema colapsa o se ralentiza demasiado, corremos el riesgo de hacerlo inútil. De los servidores utilizados, los menos susceptibles a colapsarse o tener problemas de ralentización son la propia nube de Telegram y el recolector de datos, por lo que debemos poner el énfasis fundamentalmente en la fuente de datos y el servicio del bot. Estimando la cantidad de usuarios potenciales del sistema (2043 alumnos según la web de la facultad) (FDI, 2017) y con los recursos de los que disponemos, consideramos suficiente una capacidad para 20 usuarios simultáneos.

- Actualización recurrente de los datos: Partiendo de la base del uso del scraping de la web de la facultad para la obtención de recursos del bot.

El sistema de recolección de datos debe funcionar de manera autónoma y ha de garantizarse que no sufra problemas que impidan la correcta actualización de los datos. Esta tarea, debe realizarse en horas de poca utilización de los sistemas (de madrugada), para así no interrumpir el correcto funcionamiento de los mismos. Como depende de la web de la facultad, esto es algo difícil de asegurar, no obstante se pueden implementar mecanismos de comprobación y de repetición de las tareas en caso de un fallo puntual en los sistemas de la FDI.

- Validez de los datos utilizados: El sistema debe proveer datos válidos, así como tener mecanismos para desechar datos incompletos o incorrectos e informar de la fecha de obtención de los datos mostrados por el bot, para hacer una idea al usuario de cuán actual/antigua es la información que está recibiendo. Una vez más, en nuestro caso principal, esto depende de que la información suministrada por la web de la facultad sea correcta y esté actualizada. Por lo que la manera de que no sucedan situaciones indeseadas, únicamente puede ser mediante mecanismos de control del propio bot o del recolector.

Todos estos puntos críticos engloban diferentes tipos de pruebas: de carga, robustez, integración, monitorización y prestaciones.

A continuación se exponen los tipos de planes diseñados para realizar todas ellas.

7.2.1. Pruebas activas

En primer lugar, el hecho de realizar pruebas con amigos/familiares/-compañeros voluntarios, que nos aporten su visión sobre el funcionamiento, la interfaz y robustez del sistema, nos permitiría tener una fuente objetiva de pruebas, así como un feedback interesante para ver que cosas mejorar, cambiar o solucionar en futuras versiones.

Además de esto, convendría buscar alguna herramienta que permitiese realizar pruebas a gran escala directamente sobre el Bot de Telegram, pero dado que para tener una cuenta de Telegram con la que poder realizar pruebas, es necesario un número de teléfono, no es sencillo disponer de un gran número de usuarios con los que probar de manera simultanea el funcionamiento del sistema.

Sin embargo, unas pruebas mas sencillas de realizar, fueron un conjunto de pruebas sobre el servicio web desarrollado por nosotros como fuente de datos. Gracias a la herramienta SoapUI (SoapUI, 2017),como podemos ver en las figuras 7.1 y 7.2, generamos un banco de pruebas para comprobar

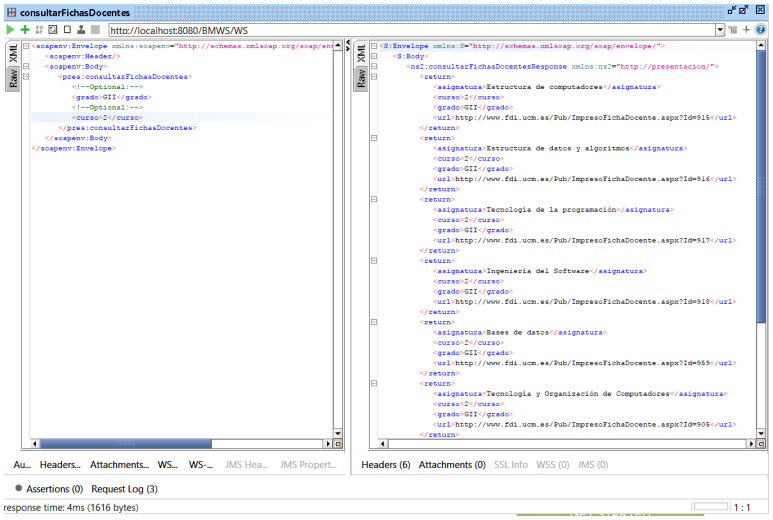


Figura 7.1: Ejemplo de testeo con SoapUI

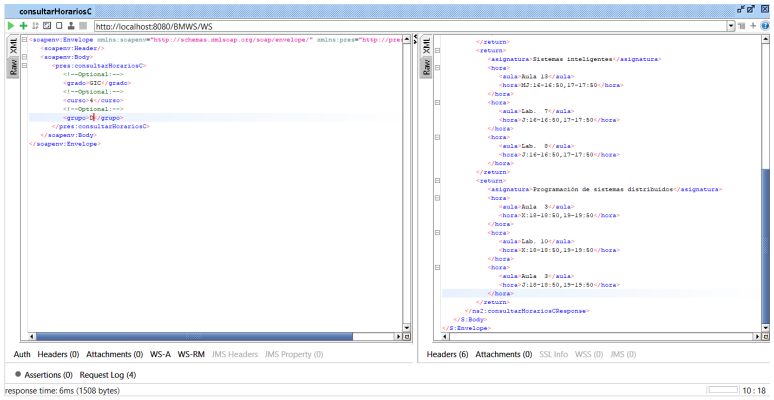


Figura 7.2: Otro jemplo de testeo con SoapUI

tanto el tiempo de respuesta como la capacidad teórica soportada (en condiciones ideales).

7.2.2. Pruebas pasivas

Se realizaron, además pruebas de monitorización durante días consecutivos sobre el recolector de datos, más específicamente sobre el cron/tarea programada, gracias a la cual se inicia para comprobar que los datos recolectados eran los esperados, estos fueron satisfactorios. También se monitorizó la estabilidad del servicio web, revisando los logs del servidor GlasshFish.

7.2.3. Pruebas unitarias

Puesto que se ha usado Java para el desarrollo del servicio web, se pensó en realizar pruebas utilizando JUnit para ir probando por separado cada uno de los módulos, aunque finalmente por falta de tiempo no se pudieron llevar a cabo.

Capítulo 8

Conclusiones

Recapitulando, este proyecto comenzó como un sistema más de información para los alumnos de la universidad, queriendo hacer uso de nuevas tecnologías al alza en el presente y permitiendo una mayor simplicidad en las consultas cotidianas de cualquier alumno. Nuestro interés principal era el desarrollo de una utilidad social, no comercial. A raíz de esto fue evolucionando y cambiando conforme los requisitos variaban, las decisiones de diseño afectaban a las tecnologías utilizadas y viceversa. Finalmente ha resultado un largo viaje durante este curso, durante el cual hemos dedicado innumerables horas.

Son varios los objetivos que nos hemos propuesto alcanzar en este proyecto, los cuales hemos dividido en dos categorías según su relevancia para el proyecto. En la primera categoría están los objetivos principales, de estos depende toda la estructura del proyecto. En la segunda están los objetivos secundarios, los cuales son mejoras significativas para el proyecto, pero no fundamentales para su funcionamiento.

Los objetivos principales fueron realizados con éxito, entre ellos se encuentran, la recolección de datos, el servicio web, el bot de Telegram y la integración de todos ellos. Estos han sido tanto especificados, como implementados.

Los objetivos secundarios son:

- Implementación de una interfaz más amigable en el bot (menús). Por motivos de tiempo este objetivo no se pudo cumplir.
- Inclusión de más información de la página web que pueda ser relevante para el alumno, por ejemplo Cisne. Catálogo de la biblioteca. Por motivos de tiempo este objetivo no se pudo cumplir.
- Integración de inteligencia artificial en el bot. Este objetivo ha sido

cumplido con éxito.

Cuando empezamos a pensar la idea del proyecto, que nos llevo varios días definirla y explicarla al tutor. Al ser una idea de proyecto original nuestra, de cuya tecnología no era muy conocedor y tener otros compromisos con más proyectos, se ofreció a tutelarnos de manera más autónoma. Cuando la idea estuvo definida y después de varios contactos por parte de nuestro tutor hacia el departamento encargado de los sistemas de información de la facultad (página web en este caso), nos fue imposible conseguir acceso a una fuente de información ya construida y completa (aunque si recibimos ficheros XML con algunos datos de la web), ya que no era algo prioritario y precisaba tiempo y recursos no disponibles, lo que nos llevó a centrar parte de nuestros esfuerzos a idear la nuestra propia.

Después tuvimos que indagar en tecnologías que no habíamos utilizado nunca a lo largo de nuestros estudios, véase LaTeX, TeXiS, Python (con sus frameworks y librerías para scraping, consumo de mensajes soap entre otros), SQLite, etc. lo que ha sido en ocasiones difícil, pero muy gratificante a la hora de ver los resultados de nuestro empeño. Ha habido diferentes momentos de investigación que nos han llevado a probar distintas plataformas o maneras de implementar nuestro proyecto, por ejemplo a la hora de elegir el framework para el desarrollo del bot, se probaron multitud de ellos en varios lenguajes como JAVA, C++ y finalmente Python, hasta dar con el que mejores resultados daba, aún a falta de ejemplos y documentación para el desarrollo de cosas complejas con la API de Telegram.

Uno de los conceptos o ideas que surgió a lo largo del proyecto fue la necesidad de utilizar técnicas de inteligencia artificial (IA), para convertir nuestro chatbot en algo más que un simple menú con botones. Conforme más investigábamos sobre las tecnologías y fundamentos que rodean a los bots, más relevancia iba adquiriendo la IA para nuestro proyecto.

Llegados a un cierto punto del desarrollo decidimos implementar estas técnicas en nuestro chatbot, ya que la forma más óptima para transmitir la información relevante de la página web de la facultad es mediante la conversación y, para esta labor tan compleja son fundamentales técnicas como el Procesamiento de lenguaje natural (PNL), el Machine learning o el Deep learning.

Toda esta labor nos hizo darnos cuenta del potencial que tienen estas tecnologías y que muy probablemente en un futuro próximo, muchas empresas e instituciones las utilizaran para dar servicio (tanto de atención como de gestión) a la gente.

Inicialmente las pretensiones del proyecto eran mayores a lo que finalmente se llegó a implementar, debido a la presión durante el curso de practicas en empresa, exámenes y la investigación necesaria, citada anteriormente. No obstante, dejamos la puerta abierta a futuras mejoras del sistema gracias a todas las posibilidades que posee la plataforma de Telegram, en la que regularmente se añaden funcionalidades y herramientas muy interesantes.

8.1. Lineas futuras

En esta sección trataremos algunas de las ideas que pueden mejorar o complementar al bot, estás, o bien no han podido implementarse por falta de tiempo y dificultades técnicas, o han ido surgiendo a medida que se desarrollaba el proyecto, no estando contempladas al inicio del mismo.

- Inclusión de un servicio de información propio de la UCM. Esto permitiría una mejor integración con los datos de la universidad, más fiabilidad por parte de los usuarios y de la comunidad académica en general y mayor cantidad de información (no solamente la referida a la web de la facultad si no a la de todas las facultades o servicios de la universidad). Para ello podrían crear una API rest fiable y de acceso restringido, con diferentes niveles de seguridad para acceder a un tipo u otro de datos, siempre de consulta, de toda información que se publica en las webs de las facultades, en la biblioteca e incluso en geaportal/ucmnet.

- Integración con las plataformas de aulas/campus virtuales (como Moodle), lo que permitiría a los profesores y alumnos interactuar de manera más cómoda y notificar en tiempo real a los usuarios de los avisos, actividades y demás asuntos relacionados con la actividad docente.

- Funcionalidades extras que aportarían mayor valor al bot, entre ellas podríamos citar:

  - Acceso al catalogo de la biblioteca de la UCM, lo que permitiría a los usuarios consultar el número o la ubicación de los ejemplares disponibles en la red de bibliotecas de la universidad.
  - Notificación de eventos de la facultad. Se podrían enviar avisos sobre las conferencias, torneos, talleres y demás actividades que se organizan frecuentemente en la universidad, permitiendo incluso al usuario seleccionar sobre que temas/actividades desea ser notificado.
  - Mejora de la interfaz de usuario para el bot. Permitir diferentes maneras de interacción con el mismo, ya sea por comandos, botones inline o en el teclado personalizado. Durante el desarrollo del

mismo se implantaron los comandos, para después poder usarlos con los menús, incluyendo en los atributos de los comandos la información que se iba recibiendo de los botones pulsados por el usuario (por ejemplo: al pulsar el botón "Tutorías" en el menú principal, se crearía un "comandoTutorias", después al seleccionar un grado, se almacenaría ese grado en los atributos y así hasta rellenar todos los atributos necesarios para la consulta, momento en el cual se ejecutaría la petición) permitiendo aprovechar la infraestructura creada con estos sistemas de interacción y con futuros que pudieran incluirse en la API de Telegram.

- Inclusión de información del usuario a modo de "cookies" para dinamizar las consultas o funcionalidades extras del bot. Gracias a que cada usuario posee un id propio, el sistema podría almacenar en una base de datos, información interesante para evitar tener que introducir información una y otra vez en el sistema. Ésta información podría ser: una lista de asignaturas matriculadas, el curso, grupo y grado que se cursa actualmente, los tipos de eventos sobre los que se desea información o los prestamos de la biblioteca (éstos últimos tomando como base las posibles mejoras anteriormente citadas). Gracias a esta información el bot sería mucho más intuitivo, rápido y amigable para el usuario.
- Mejora de los algoritmos de aprendizaje utilizados, en concreto queremos en un futuro implementar redes neuronales artificiales, es decir, queremos incluir en el proyecto técnicas como el Deep learning, que son capaces de mejorar enormemente las capacidades conversacionales de nuestro chatbot. Para esta labor utilizaríamos herramientas como KERAS, esta es una librería con una API de redes neuronales de alto nivel, basada en Python. Sus principales virtudes son:
  - Permitir un prototipado fácil y rápido (a través de la facilidad de uso, la modularidad y la extensibilidad).
  - Soporta red neuronal convolucional y redes recurrentes, así como combinaciones de los dos.
  - Se ejecuta sin problemas en la CPU y la GPU.

8.2. Conclusions

To recapitulate, this project began as a more information system for the students of the university, wanting to make use of new technologies upward in the present and allowing a greater simplicity in the daily consultations of any student. Our main interest was the development of social utility, not commercial. As a result of this, it evolved and

changed as requirements varied, design decisions affected the technologies used and vice versa. It has finally been a long journey during this course, during which we have spent countless hours.

There are several objectives that we have set out to achieve in this project, which we have divided into two categories according to their relevance to the project. In the first category are the main objectives, of these depends the entire structure of the project. In the second are the secondary objectives, which are significant improvements for the project, but not fundamental for its operation.

The main objectives were successful, among them are data collection, web service, the bot of Telegram and the integration of all of them. These have been both specified and implemented.

The secondary objectives are:

- Implementation of a friendlier interface on the bot (menus). For reasons of time this objective could not be complete.
- Inclusion of more information of the web page that may be relevant to the student, for example Cisne. Library catalog. For reasons of time this objective could not be complete
- Integration of artificial intelligence in the bot. This objective has been successfully fulfilled.

During the last year, we spent a lot of time to make this project real, first to define and detail the idea and after to explain it to our tutor. He was agree with it and decided to leads our project. The firts problem was to gain access to any data system of the UCM to provide us the necessary data. Finally this wasn't possible and we take the decission to make our own system by scraping the webpages of the faculty. All these things make us to center ours efforts some time.

After that, we had to investigate the use of technologies that we had never used before like LaTeX, TeXiS, Python (some frameworks and libraries for scraping and consume of SOAP messages), SQLite. It was dificult but rewarding when we began to see progress.

One of the concepts or ideas that emerged throughout the project was the need to use artificial intelligence (AI) techniques, to make our chatbot more than just a menu with buttons. As we further investigated

the technologies and fundamentals surrounding the bots, more relevance was acquiring AI for our project.

At a certain point of development we decided to implement these techniques in our chatbot, since the most optimal way to transmit the relevant information of the faculty web page is through the conversation and, for this so complex work are fundamental techniques like the Processing Of natural language (NLP), Machine learning or Deep learning.

All the work done makes us understand the possibilities of all these technologies, and we are sure that in a near future, a lot of companies and institutions will make use of them to offer a lot of services to the users.

At the beginning our pretensions was bigger than the final result of the project, nevertheless, we leave open a door to increase the functionality and improve things like the way that the user interacts with the bot. Even to incorporate new functions provided by the regular Telegram Bot API updates.

Capítulo 9

Aportación de cada cada miembro al proyecto

En el presente capítulo detallaremos cada una de las tareas, llevadas a cabo por los miembros del equipo, que han hecho posible la realización de este proyecto.

9.1. Aportaciones de Eloy González Acedo

Al inicio del presente curso, me puse en contacto con Simon Pickin, el tutor, éste me propuso continuar con un proyecto de unos cursos atrás. Yo le propuse que me dirigiese en un proyecto que tenía en mente, realizar un Bot para Telegram. Al principio tenía varías ideas sobre que utilidad dar al proyecto, pensé en hacer una plataforma de generación de bots con funciones genéricas, que luego el usuario pudiese personalizar para un uso particular: como agenda personal, para promocionar productos en una tienda... Finalmente, después de estudiar la idea durante días, la descarté por suponer demasiado tiempo, necesitar un grupo más grande y la imposibilidad de crear bots de manera automática, entre otras cosas. Sin embargo, me surgió otra idea, más realista y con una finalidad social, crear un bot que ayudase a nuevos y veteranos alumnos en el día a día de la facultad. Se lo comenté a Simon y le pareció buena idea. Al mismo tiempo me puse en contacto con Pedro (mi compañero), con el cual ya había trabajado en otras ocasiones y además, casualmente, también estaba buscando un proyecto para su TFG, así fue como dio comienzo BotMentor.

Realicé el análisis del proyecto, en colaboración con mi compañero, definiendo la funcionalidad que iba a tener y como iba a estructurarse y dividirse el sistema en cada uno de los subsistemas/componentes. Una vez terminado el análisis, comencé con el desarrollo del servicio web, mientras lo hacía,

busqué una plataforma donde alojar el proyecto de LaTeX, encontré la web ShareLatex (ShareLaTex, 2017) y la propuse para ello. La idea fue bien recibida y comencé a estructurar el documento apoyándome en la plantilla TeXiS. En lo referido a la memoria, me he encargado de:

- 1.Introducción (gran parte de la versión en castellano)
- 2.Estado del arte (puntos 2.1 y 2.3)
- 3.Tecnología del BotMentor (3.1, 3.2, parte del 3.3)
- 4.Funcionalidad y requisitos (4.1, 4.2 y SRS del Bot)
- 5.Planificación, control y seguimiento (parte 5.2, 5.3)
- 6.Implementación (íntegro)
- 7.Verificación y validación (integro)
- 8.Conclusiones (parte)
- Diagramas UML

Para la elaboración de la memoria tuve que consultar mucha bibliografía, reflejada en la presente al final del documento.

En lo referido al desarrollo, como antes mencioné, me encargué del diseño y desarrollo del servicio web, de su testeo e implementación con el Bot. También me encargué de probar distintos frameworks para el Bot (Java, C++, Python), seleccionando finalmente Telepot (Lee, 2015a) para Python. Ya elegido, me dediqué a su desarrollo, a buscar documentación y ejemplos de otros bots desarrollados con el mismo framework. Una vez entendido el funcionamiento del bot, comencé a realizar prototipos con los distintos tipos de menús que pueden crearse para la interacción con el usuario. Finalmente por falta de tiempo decidí no llegar a implementarlos completamente y en su lugar realizar los comandos de texto, para más adelante poder ser incorporados.

En definitiva, me he encargado de la idea principal, el análisis de requisitos, de buscar plataformas y herramientas para la elaboración de los sistemas y la documentación, gran parte de la memoria y el desarrollo del Servicio web y del Bot.

9.2. Aportaciones de Pedro Sánchez Ramírez

Durante este proyecto mis aportaciones han sido varias, desde las ideas básicas que compondrían el bot, como información, análisis e implementación del prototipo.

En primer lugar he aportado junto con mi compañero algunas ideas para poder llevar acabo el desarrollo el BotMentor por ejemplo, durante los primero días surgió el problema de cómo íbamos a obtener la información de la página web, en ese momento se propusieron distintas técnicas, personalmente me interese por el web scraping, técnica capaz de extraer información relevante de cualquier sitio web.

Otra de mis aportaciones iniciales fue la búsqueda de un leguaje con el que implementar el bot, junto con mi compañero estudiamos varias posibilidades como utilizar java, leguaje muy conocido por ambos, pero tras mucho debatir decidimos optar por Python, ya que este nos aportaba una gran cantidad de posibilidades, aparte de los extensos frameworks y librerías de los cuales podríamos utilizar para el desarrollo tanto el bot, como el web scraping.

Por otro lado también empezamos a decidir que herramientas utilizaríamos para el desarrollo del proyecto, por ejemplo se decidió utilizar Google Drive, para compartir datos y ShareLaTex como editor de texto de la documentación. Además se propuso conjuntamente el uso de la herramienta GitHub para compartir el código que íbamos a desarrollar.

En lo que respecta a la planificación mi compañero y yo decidimos en un principio utilizar metodologías de planificación clásicas, pero por diversos problemas cambiamos a metodologías ágiles. En ese momento empecé a buscar herramientas con las que realizar una planificación ágil, tras considerar varias opciones propuse la herramienta Taiga, herramienta que nos ha aportado muy buenos resultados.

En cuanto al trabajo escrito y la documentación aportada a este proyecto que nos ocupa, mis aportaciones principalmente en los siguientes puntos:

- En el estado del arte hice una investigación exhaustiva sobre los bots, desde sus orígenes hasta su actualidad, las tecnologías que les rodean, las plataformas más relevantes donde desarrollarlos y sus aplicaciones hoy en día. En concreto me he centrado en la relevancia que tiene la inteligencia artificial (IA) en el mundo de los bots y sus aportaciones a estos.

- En el apartado de las tecnologías he documentado todo lo respectivo con la recolección de datos, desde las distintas propuestas a utilizar, hasta un análisis detallado del web scraping. Además en este apartado he hablado sobre el sistema del bot, desde los lenguajes utilizados, como plataformas de desarrollo, hasta las distintas librerías utilizadas.

- Para el apartado de la planificación documente parte de las metodolo-

gías que utilizamos a lo largo del proyecto, la planificación temporal, la línea base y las herramientas utilizadas, aportando varios ejemplos.

- En las conclusiones he hablado principalmente sobre la relevancia de la inteligencia artificial en nuestro proyecto y lo que esta puede aportar en el futuro. Además he comentado la posibilidad futura de mejorar los algoritmos utilizados.

En lo que respecta al desarrollo del bot mis aportes son los siguientes:

- BBDD: Desarrollo. Diagrama E/R.
- Web scraping: Documentación e implementación. En primer lugar me documente sobre las distintas librerías capaces de realizar esta tarea, una vez elegida la librería y aprendido su funcionamiento me propuse a realizar el scraping. Empecé realizando un exhaustivo análisis de las páginas web de la facultada de informática de la ucm, ejecute el scraping y estructure la información, la cual introduje en la BBDD diseñada.
- Machine learning: Documentación e implementación. Para empezar estudie las distintas maneras de implementar esta tecnología en nuestro bot, para ello he tenido que documentarme sobre las librerías que se enfocaban en machine learning y aprender a utilizarlas. Tras implementar esta tecnología el siguiente paso ha sido enseñar al chatbot una serie de interactuaciones similares a las que tendría con un usuario real. Con este aprendizaje inicial el BotMentor es capaz de responder a los usuarios de una forma más inteligente.

A modo de conclusión en este proyecto he aportado distintas ideas para el bot, como por ejemplo parte de sus requisitos, herramientas y plataformas para el desarrollo del proyecto, gran parte de la memoria y por último el desarrollo de la BBDD, Web Scraping y la implantación de tecnologías Machine Learning.

Apéndice A

Diagramas

A.1. Diagramas UML

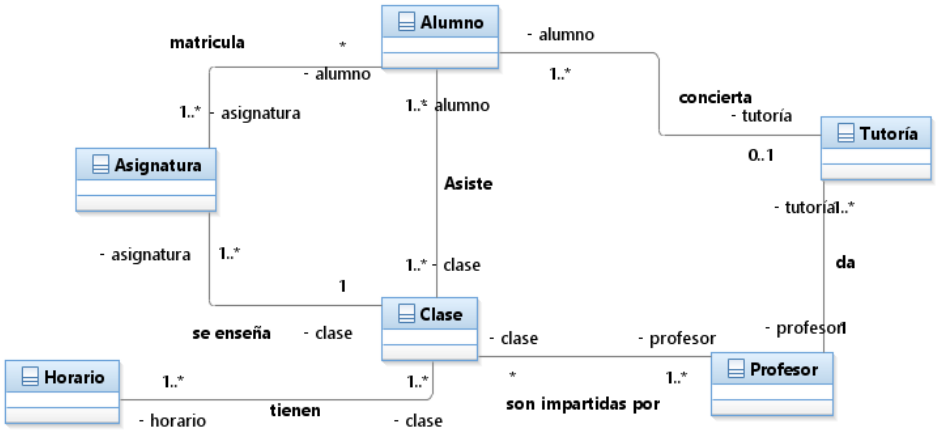


Figura A.1: Modelo del dominio

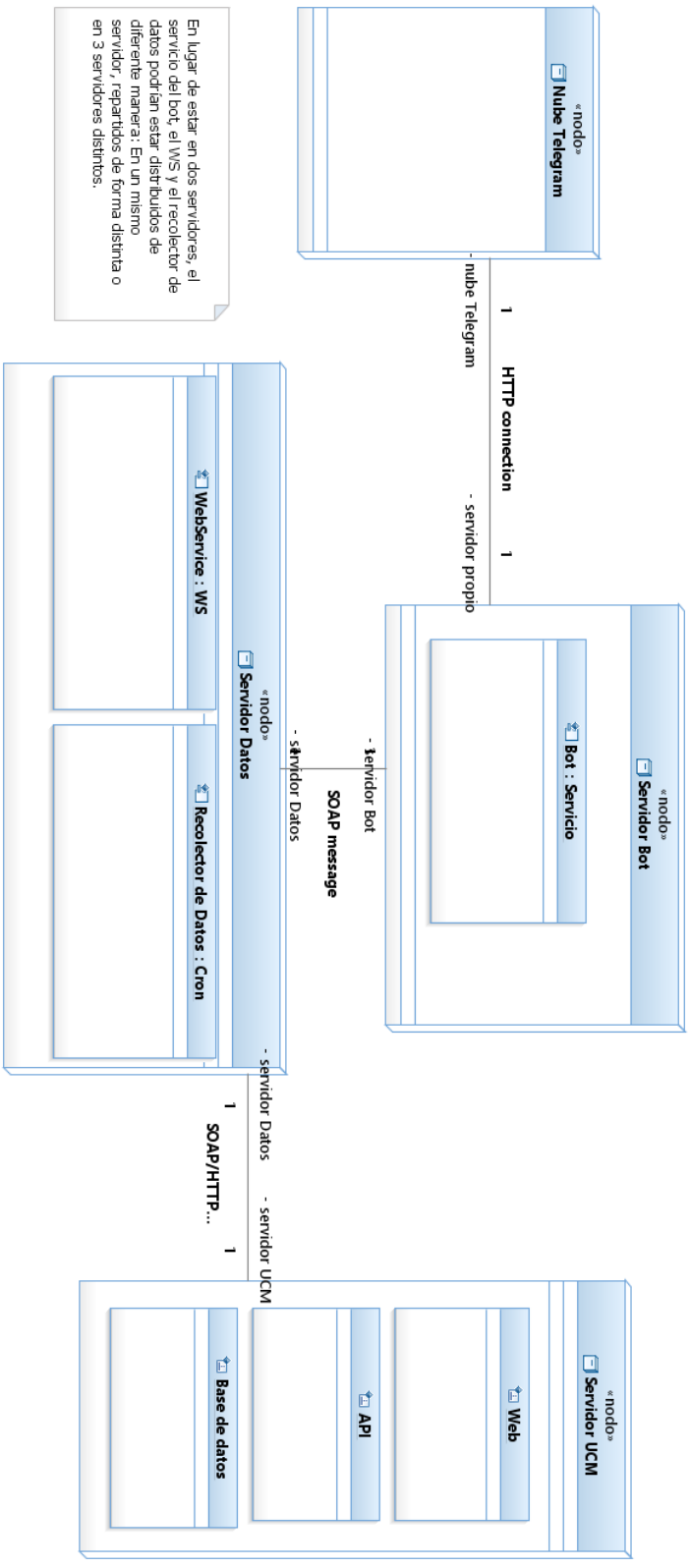


Figura A.2: Diagrama de despliegue

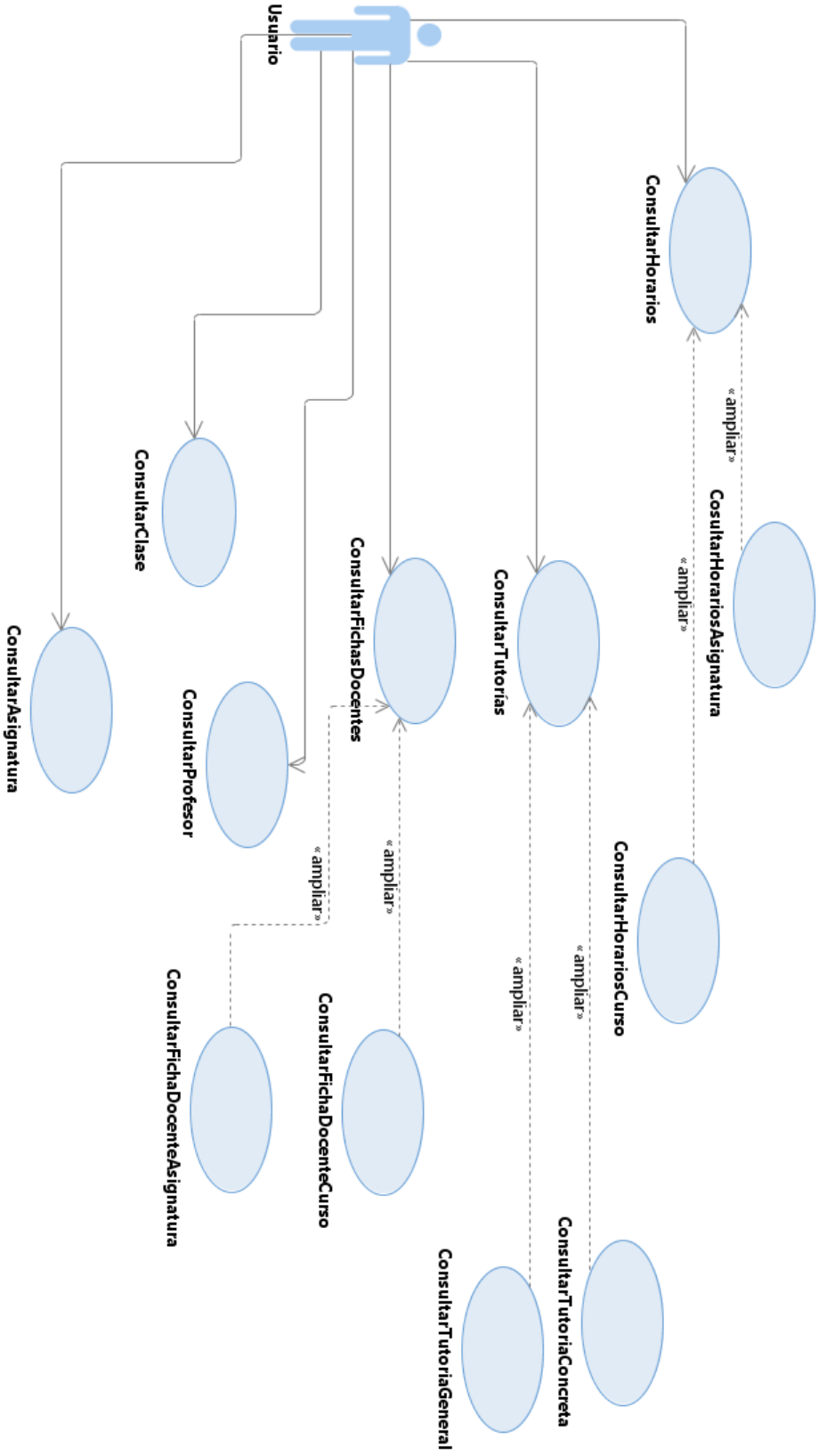


Figura A.3: Diagrama Principal de caso de uso Peticiones Usuario

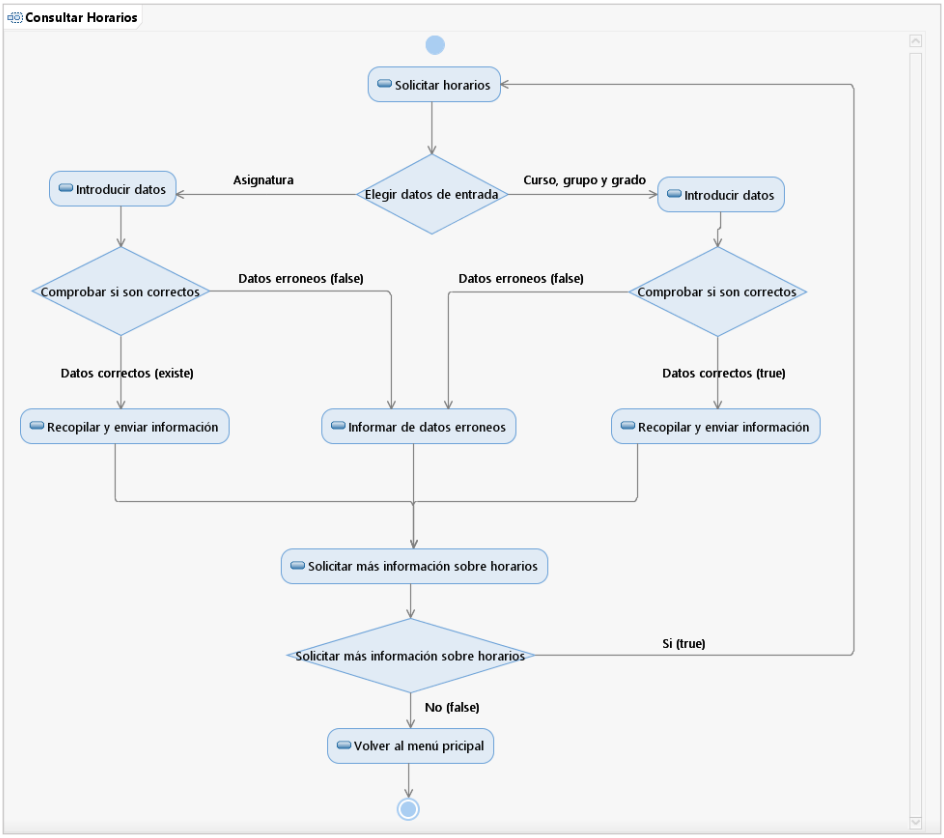


Figura A.4: Diagrama de actividad de requisito funcional “Consultar Horarios”:

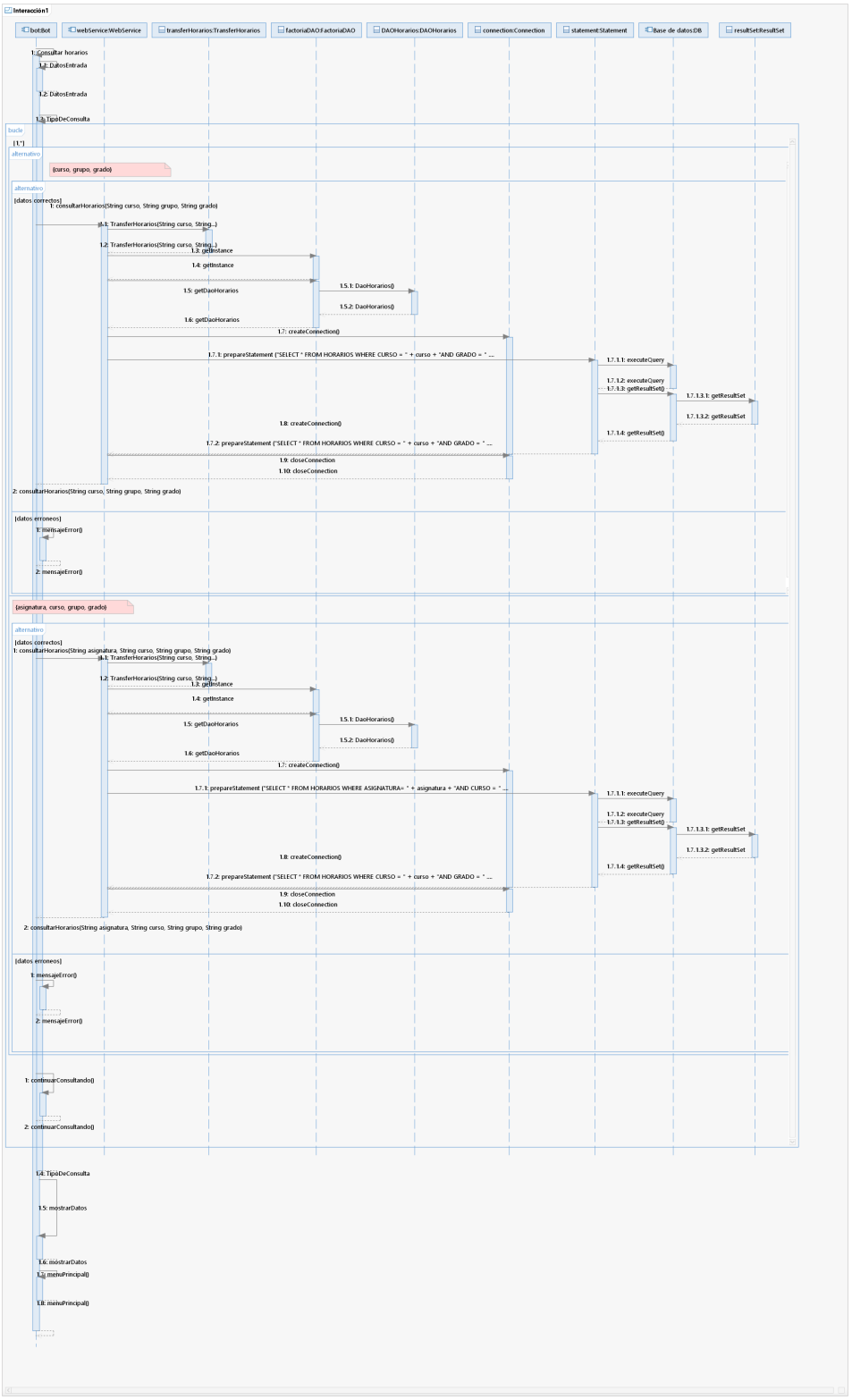


Figura A.5: Diagrama de secuencia de requisito funcional “Consultar Horarios”:

Bibliografía

Telegram. 2014. Disponible en `https://telegram.org/` (último acceso, Julio, 2017).

Telegram bot platform. 2015. Disponible en `https://telegram.org/blog/bot-revolution` (último acceso, Julio, 2017).

Meet the telegraph api for logins and stats. 2016. Disponible en `https://telegram.org/blog/telegraph` (último acceso, Julio, 2017).

Abdelmalik Moujahid, I. I. y. P. L. *Clasificadores K-NN*. Universidad del País Vasco, 2017. Disponible en `http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t9knn.pdf` (último acceso, agosto, 2017).

Alexander Menzinsky, J. P., Gertrudis López. *Scrum Manager - Guía de formación*. 2016. Disponible en `http://www.scrummanager.net/files/scrum_manager.pdf` (último acceso, Julio, 2017).

botsify. CREATE AI CHATBOT WITHOUT CODING. 2017. Disponible en `https://botsify.com` (último acceso, Julio, 2017).

Busacca, E. Eliza. 1998. Disponible en `https://www.letralia.com/40/ar01-040.htm` (último acceso, Febrero, 1998).

Cadwalladr, C. The great british brexit robbery: how our democracy was hijacked. 2017.

Castellanos, M. Patrón dao. 2013a. Disponible en `http://castellanosmiguel.blogspot.com.es/2013/07/dataaccess-object-dao-definicion-dao-es.html` (último acceso, Julio, 2017).

Castellanos, M. Patrón transfer. 2013b. Disponible en `http://castellanosmiguel.blogspot.com.es/2013/07/transfer-object.html` (último acceso, Julio, 2017).

chatfuel. Build a Facebook bot without coding. 2017. Disponible en `https://chatfuel.com` (último acceso, Julio, 2017).

CHUWIKI (DAO). Entrada: "Data Access Object". Disponible en http://chuwiki.chuidiang.org/index.php?title=Patrón_DAO (último acceso, Julio, 2017).

UNIVERSIDAD DE CORNELL. THE HELPFUL CHATBOT FOR HIGHER ED. 2017. Disponible en https://courseq.io (último acceso, Julio, 2017).

COX, G. ChatterBot: Machine learning, conversational dialog engine. 2017. Disponible en http://chatterbot.readthedocs.io/en/stable/index.html (último acceso, Agosto, 2017).

ERICH GAMMA, R. J., RICHARD HELM y VLISSIDES, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1994. ISBN 978-02016336100-201-63361-2. Disponible en http://wiki.c2.com/?DesignPatternsBook (último acceso, Julio, 2017).

ESCOM. Algoritmo de retropropagación. 2009. Disponible en https://es.slideshare.net/mentelibre/algoritmo-de-retropropagacin (último acceso, Septiembre, 2009).

FACEBOOK. Facebook messenger. 2017. Disponible en https://www.messenger.com (último acceso, septiembre, 2017).

FDI, D. Datos - informática ucm. 2017. Disponible en https://informatica.ucm.es/datos-y-cifras (último acceso, Julio, 2017).

FM, Y. Así era ELIZA, el primer bot conversacional de la historia. 2017. Disponible en https://www.xataka.com/historia-tecnologica/asi-era-eliza-el-primer-bot-conversacional-de-la-historia (último acceso, Mayo, 2017).

FOWLER, M. Plain Old Java Object. 2000. Disponible en https://www.martinfowler.com/bliki/POJO.html (último acceso, Julio, 2017).

GARCÍA, E. R. Los 7 mejores Chat Bots de Facebook. 2017. Disponible en http://omicrono.elespanol.com/2017/05/mejores-bots-telegram/ (último acceso, Mayo, 2017).

GARCÍA, J. J. M. Bots, ¿Profetas de una nueva era? 2016. Disponible en http://red.computerworld.es/actualidad/bots-profetas-de-una-nueva-era (último acceso, Diciembre, 2016).

UNIVERSIDAD DE GEORGIA. How Georgia State University supports every student with personalized text messaging. 2017. Disponible en http://blog.admithub.com/case-study-how-admithub-is-freezing-summer-melt-at-georgia-state-university (último acceso, Julio, 2017).

Gomez, M. Modelos Ocultos De Markov. 2009. Disponible en https://es.slideshare.net/slidemarcela/modelos-ocultos-de-markov-1395263 (último acceso, Mayo, 2009).

Gómez-Martín, M. A. y Gómez-Martín, P. P. Continuous integration in LaTeX. *The PracTEX Journal*, vol. *In Press., 2009.*

IEEE. *Std 1012-2004. 2004. Disponible en https://standards.ieee.org/findstds/standard/1012-2004.html (último acceso, Julio, 2017).*

IEEE. *Ieee std. 830-1998. 2008. Disponible en https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf (último acceso, Julio, 2017).*

Kohlhardt, C. *Gliffy. 2017. Disponible en https://www.gliffy.com/ (último acceso, Julio, 2017).*

Lee, N. *Telepot. 2015a. Disponible en http://telepot.readthedocs.io/en/latest/ (último acceso, Agosto, 2017).*

Lee, N. *Telepot. 2015b. Disponible en https://github.com/nickoala/telepot (último acceso, Agosto, 2017).*

Maderer, J. *Georgia institute of technology. 2016.*

Martí, M. *Web Scraping. 2016. Disponible en https://sitelabs.es/web-scraping-introduccion-y-herramientas/ (último acceso, Julio, 2017).*

Posts by Mazdak Rezvani, C. *Why chatbots need a big push from deep learning. 2017. Disponible en https://venturebeat.com/2017/07/23/why-chatbots-need-a-big-push-from-deep-learning/ (último acceso, Julio, 2017).*

of Minnesota Duluth, U. U. *Patrón singleton. 2017. Disponible en https://www.d.umn.edu/~gshute/softeng/presentations/patterns/singleton.xhtml (último acceso, Julio, 2017).*

de Nuevas Tecnologías, V. *Catálogo de servicios informática. 2003. Disponible en https://goo.gl/NUL9eL (último acceso, Mayo, 2017).*

Ortel, J. *Suds. 2015. Disponible en https://pypi.python.org/pypi/suds (último acceso, Agosto, 2017).*

Patil, M. *y* Patil, P. S. Review of Web Crawlers with Specification and Working. *IJARCCE, 2016. Disponible en http://www.ijarcce.com/upload/2016/january-16/IJARCCE%2052.pdf (último acceso, junio, 2016).*

Y PEDRO PABLO GÓMEZ MARTÍN, M. A. *Texis portal. 2009. Disponible en http://gaia.fdi.ucm.es/research/texis (último acceso, Julio, 2017).*

RODRIGUEZ, J. *Tipos y funcionamiento de bots maliciosos usados por los ciberdelincuentes. 2012. Disponible en https://www.xatakamovil.com/conectividad/tipos-y-funcionamiento-de-bots-maliciosos-usados-por-los-ciberdelincuentes (último acceso, Mayo, 2012).*

SERRANO, S. *Los 7 mejores Chat Bots de Facebook. 2017. Disponible en http://www.brandominus.com/7-mejores-chat-bots-facebook/ (último acceso, Semptiembre, 2016).*

SHARELATEX. *Sharelatex portal. 2017. Disponible en https://es.sharelatex.com/ (último acceso, Julio, 2017).*

SHEVAT, A. Designing bots : creating conversational experiences. *O'Reilly Media, Inc, 2017. ISBN 1491974826, 9781491974827.*

SOAPUI, S. S. *Soapui. 2017. Disponible en https://www.soapui.org/ (último acceso, Agosto, 2017).*

TELEGRAM. *Bots: An introduction for developers. 2017a. Disponible en https://core.telegram.org/bots (último acceso, Julio, 2017).*

TELEGRAM. *Telegram Bot API. 2017b. Disponible en https://core.telegram.org/api (último acceso, Julio, 2017).*

TELEGRAM. *Telegram Messengerr. 2017c. Disponible en https://telegram.org (último acceso, Julio, 2017).*

URBANAS, T. *Spambots. 2016. Disponible en http://todas-las-tribus-urbanas.blogspot.com.es/2015/06/spambots.html (último acceso, Marzo, 2016).*

W3C. *Servicio Web. 1999. Disponible en http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb (último acceso, Mayo, 2017).*

WELICKI, L. *Patrón singleton. 2017. Disponible en https://msdn.microsoft.com/es-es/library/bb972272.aspx (último acceso, Julio, 2017).*

WIKIPEDIA *(IA). Entrada: "Artificial intelligence". Disponible en https://en.wikipedia.org/wiki/Artificial_intelligence (último acceso, Junio, 2017).*

Wikipedia *(Game bots). Entrada: "Game bots". Disponible en* `https://en.wikipedia.org/wiki/Video_game_bot` *(último acceso, Junio, 2017).*

Wikipedia *(Procesamiento del lenguaje natural). Entrada: "Procesamiento del lenguaje natural". Disponible en* `https://en.wikipedia.org/wiki/Natural_language_processing` *(último acceso, Agosto, 2017).*

Worswick, S. *Welcome to the Mitsuku Website. 2016.*