Universidad de **Valladolid**

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

Interacción con el Campus Virtual mediante un asistente de voz

Autor:
D. Sergio Gómez Conde

Tutor:
Dr. Jesús María Vegas Hernández

A mi familia y amigos por su apoyo incondicional

Agradecimientos

A mis padres, por apoyarme siempre en mis decisiones y darme una educación basada en el trabajo y el esfuerzo.

A mis compañeros, porque sin su ayuda en los momentos más difíciles no habría sido posible llegar hasta aquí.

A mi tutor, el Dr. Jesús María Vegas Hernández, por haberme guiado en esta última etapa de mi carrera y al cual le tengo un especial cariño.

A mis profesores, porque todos me han enseñado algo que ha sido útil durante la carrera o me será útil en el resto de mi vida profesional y personal.

Resumen

Cada vez la domótica está más presente en nuestra sociedad, existiendo multitud de tecnologías, sistemas y dispositivos que hacen la vida mucho más sencilla. Uno de estos aparatos son los altavoces inteligentes, que cuentan con un asistente de voz o asistente virtual al cual podemos preguntar cualquier cosa que queramos saber o pedir que realice alguna acción concreta (p. ej., establecer una nueva alarma o crear un evento en nuestro calendario).

En el presente documento se expone el Trabajo de Fin de Grado en el que se utilizará un asistente de voz para que los alumnos de la Universidad de Valladolid puedan consultar información del Campus Virtual de la propia universidad, así como ejecutar diferentes tareas de forma automatizada para facilitar su día a día y la realización de sus estudios. Dicho trabajo será llevado a cabo mediante la creación de un script que obtenga la información necesaria del Campus Virtual de la Universidad de Valladolid y de diferentes *skills* para el asistente virtual *Mycroft* que ofrezcan dicha información al alumno.

Las funcionalidades que puede ofrecer el asistente son tantas como las que existen en el Campus Virtual, por lo que depende del ingenio del programador y de las necesidades de los alumnos el obtener más o menos información y desarrollar más o menos *skills*.

Abstract

Home automation is increasingly present in our society, with a multitude of technologies, systems and devices that make life much easier. One of these devices are smart speakers, which have a voice assistant or virtual assistant to which we can ask anything we want to know or ask to perform any specific action (e.g., set a new alarm or create an event in our calendar).

This document presents the Final Degree Project in which a voice assistant will be used so that the students of the University of Valladolid can consult information from the Virtual Campus of the university itself, as well as perform different tasks in an automated way to facilitate their daily life and the completion of their studies. This work will be carried out through the creation of a script that obtains the necessary information from the Virtual Campus of the University of Valladolid and different skills for the virtual assistant Mycroft that offer this information to the student.

The functionalities that the assistant can offer are as many as those that exist in the Virtual Campus, so it depends on the ingenuity of the programmer and the needs of the students to obtain more or less information and develop more or less skills.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la actualidad, la sociedad está sufriendo una digitalización a pasos agigantados en todos los ámbitos, desde la educación y el trabajo hasta las tareas del hogar, en la que no se concibe un mundo sin Internet y sin dispositivos como los modernos teléfonos móviles. Esto hace que nuestra vida sea más sencilla, pudiendo agilizar y realizar tareas y acceder a información en cualquier momento y desde cualquier lugar.

Concretamente, son las casas unos de los lugares donde se está produciendo un mayor aumento del número de dispositivos inteligentes, es decir, conectados a Internet. Con estos dispositivos se pueden realizar prácticamente todas las actividades del hogar como la limpieza con robots aspiradores o la automatización de la calefacción, el aire acondicionado y el encendido y apagado de luces.

Todas estas tareas pueden ser realizadas (y programadas) mediante un *smartphone* que cuente con la aplicación necesaria instalada, pero también a través de un asistente virtual alojado en un altavoz inteligente al que se le dictan las instrucciones. Por lo tanto, los altavoces inteligentes se han convertido en un elemento fundamental en las casas de aquellas personas que quieran realizar sus tareas cotidianas de una manera mucho más rápida y sencilla.

Pero además de para administrar y controlar toda la domótica de un hogar, los altavoces inteligentes con sus asistentes de voz pueden ser utilizados para buscar todo tipo de información con una simple pregunta formulada por el usuario como si estuviese introduciendo dicha consulta en un buscador web.

Teniendo todo esto en cuenta, y añadiendo la experiencia personal de haber cursado un grado universitario en Ingeniería Informática, se ha decidido desarrollar un prototipo de proyecto con el objetivo de analizar la viabilidad de utilizar un asistente de voz que permita a los estudiantes de la Universidad de Valladolid interactuar con el Campus Virtual de forma que puedan agilizar tareas de su día a día en esta plataforma. Para ello se ha utilizado como altavoz inteligente una Raspberry Pi 3 Model B+ [1] con Mycroft instalado como asistente virtual en la que se ha desarrollado un script Python que accede en

segundo plano a la página web del Campus Virtual, extrae la información y la almacena en un fichero JSON local. Cuando el usuario realiza una pregunta al asistente, este consulta dicho fichero y responde con los datos solicitados.

1.1. Objetivos

Los objetivos del proyecto, como bien se ha indicado en la introducción, son planteados para comprobar, mediante la realización de un prototipo, si es factible desarrollar por completo un sistema funcional utilizado para facilitar el día a día a los alumnos de la Universidad de Valladolid en su interacción con el Campus Virtual en algunas de sus gestiones y tareas, dotándoles de una herramienta con la que pueden conversar y a la que pueden pedir que realice por ellos ciertas acciones. Más concretamente, estos objetivos son:

- Extraer y almacenar información de la página web del Campus Virtual de la Universidad de Valladolid.
- Desarrollar *skills* para Mycroft con las que interactuar con el asistente.
- Permitir a los alumnos de la Universidad obtener información de sus asignaturas.
- Permitir a los alumnos de la Universidad obtener información personal.
- Permitir a los alumnos de la Universidad obtener información de sus mensajes, como el número de mensajes sin leer o el contenido de los propios mensajes junto con su autor.
- Permitir a los alumnos de la Universidad obtener información de sus eventos, como aquellos que tienen un cierto día o el siguiente en aparecer en el calendario.
- Permitir a los alumnos de la Universidad crear eventos en su calendario del Campus Virtual.
- Aprender los aspectos básicos del funcionamiento de un asistente de voz o asistente virtual.
- Mejorar las habilidades personales con Python.

1.2. Estructura de la memoria

El presente documento tiene el objetivo de explicar cómo se ha desarrollado todo el proyecto, por lo que se ha estructurado de la siguiente manera:

Primeramente, se ha introducido el trabajo, explicando el contexto que lo rodea y la utilidad que este proyecto puede tener en la sociedad tan digitalizada en la que vivimos.

A continuación, se describen los dos término más importantes para ayudar a entender mejor esta introducción y que es necesario conocer para entender cómo se ha planteado el proyecto y en qué se ha apoyado para un correcto desarrollo y funcionamiento. Esta información se completa con algunas referencias a otros trabajos similares a este que han sido útiles como documentación y apoyo.

En el siguiente capítulo se detalla todo el proceso que se ha seguido, incluyendo aspectos importantes como los tiempos de desarrollo o los riesgos que tiene el proyecto. En este capítulo se concreta el plan de trabajo realizado, en el que se indican las fases que han conformado el trabajo, desde aprendizaje y formación hasta el desarrollo en sí, con la correspondiente realización de pruebas finales. Además, se indica una aproximación de los costes que este proyecto habría tenido si hubiese sido realizado por una empresa real.

Posteriormente, se realiza un análisis del proyecto en el que se listan los requisitos funcionales, no funcionales y de información que necesita el software, parte fundamental a realizar en cualquier desarrollo para garantizar la calidad del servicio. También como parte de este análisis se muestran los casos de uso presentes en el trabajo.

En el capítulo relativo al diseño y la implementación se definen todos los elementos hardware y, sobre todo, software que se han utilizado durante todo el desarrollo del proyecto. Estas definiciones están acompañadas de diagramas que explican la lógica y el funcionamiento del sistema y sus *skills*, finalizando con una explicación de la solución planteada para llevar a cabo este trabajo. La primera sección, dedicada a las tecnologías utilizadas en este proyecto, es muy interesante, ya que sirve para comprender mejor el contexto de la solución planteada. Principalmente, estas tecnologías son Mycroft como asistente virtual y el web scraping como método para obtener la información de Internet. De Mycroft se explica su estructura, su funcionamiento y la parte más importante para este proyecto, sus *skills,* incluyendo sus mensajes de logs y cómo interactuar entre ellas. Por otro lado, respecto al web scraping se explica en qué consiste, qué maneras existen de realizarlo y unas nociones básicas sobre cómo realizarlo.

Con el objetivo de facilitar el uso a futuros usuarios y programadores se describen paso a paso, en el capítulo de despliegue, los procesos a seguir para poder instalar y configurar todo lo necesario para contar con el trabajo realizado en este proyecto en un sistema Linux. Esto incluye la instalación y configuración de Mycroft, la instalación de Selenium [2] junto con un webdriver, la planificación de tareas en Linux y la instalación de *skills*.

En el penúltimo capítulo y una vez introducido y detallado todo el proceso de desarrollo del proyecto se plasman en este documento las pruebas realizadas durante el mismo y al finalizarlo para dejar constancia de que funciona correctamente y de acuerdo a lo esperado.

Por último, y para finalizar la presente memoria, aparecen las conclusiones obtenidas en la realización del proyecto, indicando qué se ha aprendido y para qué son útiles los conocimientos adquiridos. Estas conclusiones son rematadas con una pequeña explicación de qué trabajo futuro sería posible realizar a partir de este proyecto y qué se podría mejorar del mismo.

Capítulo 2

Estado del arte

Con el objetivo de contextualizar el ámbito del presente trabajo se realiza este capítulo en el que se explican en profundidad diferentes conceptos importantes, cuyos detalles son muy útiles para entender por qué es necesario este proyecto.

Primeramente, conocer que el término de Internet de las cosas (IoT por sus siglas en inglés) hace referencia a la conexión a Internet de dispositivos comunes del día a día (p. ej., relojes, televisiones, coches, lavadoras, gafas, etc.) con el objetivo de ampliar sus funcionalidades y mejorar sus prestaciones. El hecho de que estos dispositivos estén conectados a Internet permite que puedan ser gestionados desde otro aparato también con conexión a la red como puede ser un teléfono móvil actual, por lo que pueden ser utilizados de forma remota sin tener que tenerlos físicamente presentes.

La incorporación del Internet de las cosas (y de otras tecnologías y herramientas) a un hogar con el fin de automatizarlo es lo que se entiende a día de hoy por domótica. Para conseguir esto, los dispositivos inteligentes que conforman la domótica de una casa realizan tareas propias del ser humano y son administrados por los usuarios a través de aplicaciones móviles o su propia voz. La domótica ofrece muchas ventajas a aquellas personas que deciden incluirla en su hogar, ya que, además de ayudar a agilizar tareas u ofrecer información, permiten gestionar de una manera mucho más eficiente el uso energético de la vivienda, lo que supone un gran ahorro para los usuarios. También se puede integrar en la seguridad mediante alarmas y cerrojos inteligentes.

El grueso de este proyecto está formado por un asistente de voz que lee información obtenida de un sistema de gestión de aprendizaje, por lo que es necesario ahondar en estos dos aspectos, indicando a continuación sus características y usos y explicando las diferencias existentes entre sus distintos tipos.

2.1. Asistentes de voz

También conocidos como asistentes virtuales, los asistentes de voz son software que permite al usuario consultar información en Internet de la misma manera que si lo hiciera por escrito utilizando un buscador de Internet. Además, estos asistentes también son capaces de realizar todo tipo de acciones como poner una alarma, encender las luces de la casa o reproducir una canción en un servicio de música en *streaming*. La principal ventaja de estos dispositivos es la gran facilidad de uso que tienen, ya que lo único que tiene que hacer el usuario es pedir lo que quiere buscar o realizar con su propia voz, como si hablase con otra persona. Por el contrario, el hecho de tener que estar en continua escucha para detectar la palabra o frase de activación, puede hacer que la privacidad del usuario se vea en entredicho, ya que nunca se sabe exactamente si todo lo que escucha se almacena en algún sitio o simplemente se ignora.

Para tener una idea más visual de cómo funcionan estos asistentes, en la Figura 2.1 se puede ver un ejemplo de una conversación entre una persona y un asistente de voz o virtual.

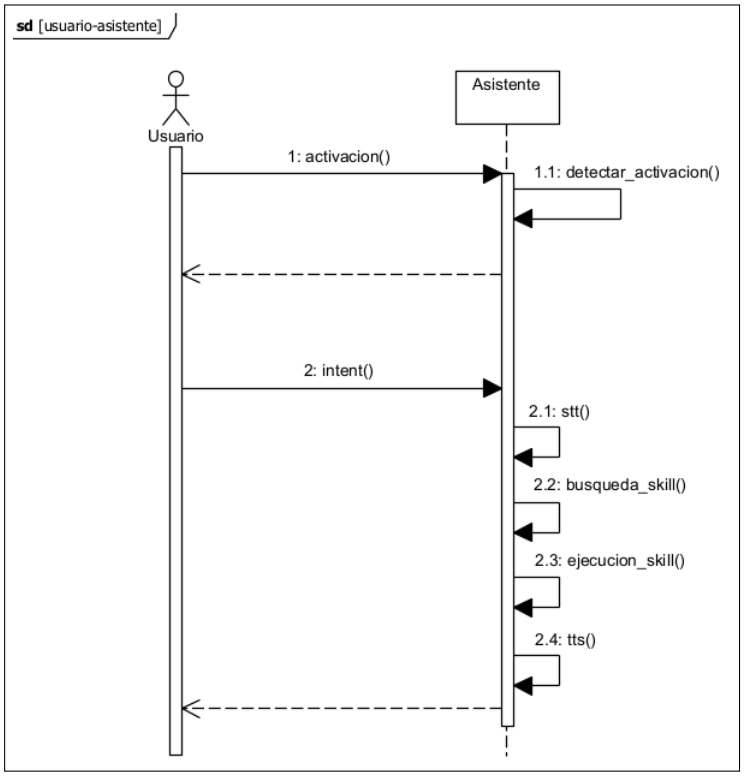

Figura 2.1: Diagrama de secuencia de una interacción usuario-asistente

Los tres asistentes de voz más utilizados a día de hoy en todo el mundo son Siri [3], Google Assistant [4] y Alexa [5]. Según la primera ola del EGM del 2019 [6], en España, Siri se situaba como el asistente virtual más utilizado, siendo la opción de un 86,2 % de las personas entrevistadas. Sin embargo, en cuanto a los altavoces inteligentes existentes en los hogares españoles, Google Home (con Google Assistant integrado) predomina con un 35 %, seguido muy de cerca por la familia de dispositivos Amazon Echo (que cuentan con Alexa) con una cuota del 33,6 %.

Para poner en contexto las diferencias entre los principales asistentes [7], cabe mencionar que Siri fue el primer gran asistente virtual disponible y es, en comparación con Google Assistant y Alexa, mucho más fiable en términos de privacidad, ya que utiliza un identificador anónimo a la hora de procesar las consultas de los usuarios. La desventaja de Siri respecto al resto de asistentes es el hecho de que solo está disponible para dispositivos iOS.

Por otro lado, Google Assistant es el asistente más completo, pudiendo responder con gran precisión peticiones complejas o aportando poca información en estas debido a la voluminosa cantidad de datos que almacena en sus bases de datos. Otra gran ventaja es que cuenta con una aplicación móvil muy completa para dispositivos iOS y Android, algo muy útil para usuarios que cuenten con ambos sistemas operativos en sus terminales. Además, el hecho de poder gestionar gran parte del ecosistema de Google (contactos, calendario, etc.) es un punto a favor de Google Assistant respecto al resto de asistentes. Este asistente fue la primera opción para realizar este prototipo, pero fue descartada debido la dificultad de desarrollar las *skills* necesarias, la cual era bastante mayor que en Mycroft, la alternativa finalmente elegida.

Por último, Alexa [5] es el asistente virtual presente en un mayor número de dispositivos (más de 28 000). Esto es, en parte, gracias a las facilidades que proporciona Amazon para que los fabricantes puedan producir dispositivos compatibles con Alexa simplemente utilizando el Alexa Connect Kit, el cual es un simple módulo con tecnología Bluetooth y Wi-Fi. Debido al hecho de ser una creación de Amazon, Alexa permite al usuario comprar mediante el uso de la voz, pero solo en la propia Amazon. Sin embargo, su principal desventaja es que tiene una menor capacidad para resolver consultas difíciles en comparación con Google Assistant, aunque este aspecto se encuentra en constante mejora y evolución, aprendiendo cada día y almacenando más información.

Para la realización de este proyecto no se ha optado por ninguna de estas opciones, sino que el asistente de voz elegido ha sido Mycroft. Esto ha sido así porque, a diferencia de los mencionados anteriormente, se trata de un asistente virtual de código abierto y de la opción con la que es más sencillo trabajar y desarrollar las *skills* necesarias para este trabajo, ya que prácticamente se necesitan únicamente conocimientos de programación con Python. Otra de las razones que se han tenido en cuenta para la elección de este asistente ha sido la computación local de las solicitudes del usuario, proporcionando una alta privacidad en comparación con las otras alternativas, las cuales realizan esa computación en la nube con la posibilidad de que las empresas utilicen esas consultas como información para otros fines de sus negocios. Mycroft cuenta con sus propios altavoces inteligentes, pero en el caso de este proyecto se ha utilizado una Raspberry Pi 3 Model B+ a la que se ha acoplado un micrófono y un altavoz, todo ello dentro de una carcasa impresa en 3D.

2.2. Sistemas de gestión de aprendizaje

El Campus Virtual de la Universidad de Valladolid se encuentra alojado en un sistema de gestión de aprendizaje (LMS por sus siglas en inglés). Como su propio nombre indica, se trata de una plataforma donde se puede realizar todo tipo de acciones relacionadas con la educación de manera digital, como almacenar apuntes, establecer calificaciones, entregar proyectos y prácticas, etc.

Estos sistemas cuentan con la gran ventaja de que pueden ser utilizados desde cualquier parte del mundo y a cualquier hora del día, lo que hace que los conocimientos sean mucho más accesibles, permitiendo al alumno desarrollar una mejor autogestión del aprendizaje. Además, también fomentan la participación de los usuarios, la interacción entre ellos y reducen los costes para la institución que gestiona el sistema. Por todo ello, son perfectos para ser usados tanto por grandes empresas como por pymes.

Cabe destacar que existen dos tipos de sistemas LMS: los comerciales, en los que se contrata el servicio y se cuenta con ventajas como la asistencia técnica propia de la plataforma; y los de código abierto, en los que se requiere de un desarrollo propio y conocimientos sobre tecnologías como HTML y sistemas de gestión de contenidos (CMS por sus siglas en inglés).

Debido a las diferentes alternativas existentes hay que tener en cuenta ciertos aspectos a la hora de elegir cuál es el sistema idóneo para la institución en la que se quiere aplicar [8]:

- Necesidades.

  Es clave identificar qué usuarios van a utilizar el sistema y, por lo tanto, que herramientas o funcionalidades (evaluaciones, comunicación, etc.) van a ser necesarias para una formación aprovechable y de calidad.

- Personalización.

  Con la finalidad de ofrecer al usuario una imagen familiar y acorde con la institución que ofrecer la formación, se debe buscar una opción que permita configurar todo lo relativo al diseño de forma que ese objetivo se cumpla de la mejor manera posible.

- Seguridad.

  Como no podría ser de otra manera, un sistema de gestión de aprendizaje debe asegurar la confidencialidad y privacidad de toda la información que en él se almacena y gestiona, desde los datos privados de los usuarios hasta los diferentes contenidos que se utilizan en la formación como documentos, vídeos o clases virtuales.

- Soporte.

  Cualquier sistema, LMS incluidos, puede tener errores de funcionamiento, los cuales deben ser solventados de la manera más rápida posible por un equipo de soporte técnico para que la ex-

periencia de uso sea cómoda y satisfactoria. Además, en ciertos momentos los usuarios pueden tener dudas o sugerencias de mejora que también deben ser atendidas por expertos de la plataforma encargados de esas tareas.

- Actualizaciones.

  En relación con el punto anterior, ningún sistema es perfecto y todos son mejorables, por lo que es altamente recomendable elegir una opción que obtenga regularmente correcciones de errores, mejoras y ampliaciones de características en forma de nuevas versiones para que no quede obsoleta, degradando su calidad.

- Precio.

  Debido a la existencia de opciones de código abierto o gratuitas, una decisión importante radica en valorar las ventajas y desventajas de una de estas opciones frente a otras de pago, ya que dependiendo de todo lo mencionado anteriormente puede no ser necesario realizar ningún desembolso económico para la instalación y utilización de un LMS.

A continuación, se listan algunas de las mejores opciones existentes actualmente en el mercado junto con algunas de sus características:

- Moodle. [9]

  El ejemplo más conocido y utilizado en el mundo de plataforma LMS de código abierto es Moodle, que es precisamente donde se encuentra el Campus Virtual de la Universidad de Valladolid. Al ser código abierto, Moodle se encuentra en constante actualización gracias a la gran comunidad que se encarga de revisar y mejorar las funcionalidades que ofrece. Además, puede ser usada en prácticamente cualquier idioma y personalizada totalmente según lo que el cliente o usuario necesite. Debido a estas y otras ventajas, como la seguridad y privacidad con las que cuenta, Moodle da soporte a grandes instituciones como la Escuela Londinense de Economía o la Universidad Estatal de Nueva York. Como principal aspecto negativo señalar que su uso es algo complejo para usuarios principiantes de esta plataforma.

- Canvas. [10]

  Al igual que Moodle, es un sistema de código abierto que, en este caso, destaca por su interfaz gráfica de usuario. En su caso, se puede implementar directamente descargando el software de su repositorio de GitHub para, posteriormente, ser implementado por programadores, o bien contratando la implementación a la empresa desarrolladora de la plataforma con el soporte técnico que ello conlleva.

  En cuanto a las características ofrecidas a los usuarios, cabe destacar que ofrece las funcionalidades mínimas de un LMS de forma gratuita, teniendo que realizar ciertos pagos para conseguir una experiencia de uso más completa. Además, utiliza jQuery, una tecnología menos avanzada que Vue.js. Un punto a favor es su integración con otras aplicaciones útiles para completar la formación y la experiencia como son Skype o Google Drive, así como su adaptabilidad a todo tipo de

dispositivos móviles, aunque con algunos pequeños problemas que disminuyen algo su calidad en este aspecto.

- Chamilo. [11]

  Otra de las grandes alternativas existentes de código abierto, pero que cuenta con una mala experiencia de usuario y una necesidad constante de ser actualizada que estropean su completa funcionalidad.

- Blackboard. [12]

  Se trata de una opción bastante costosa, pero que acumula años de experiencia en el mercado, por lo que no cuenta con prácticamente errores. Pese a ello, Blackboard cuenta con unos procesos de instalación y mantenimiento bastante complejos en comparación con Moodle, además de que consume una mayor cantidad de recursos. En cuanto a sus ventajas, una de ellas es la integración con otros productos de la compañía como Blackboard Collaborate, utilizado para la realización de clases virtuales participativas. Además, cuenta con una interfaz moderna y en la que la información aparece de forma clara, por lo que no es nada difícil de usar, sino todo lo contrario.

2.3. Proyectos similares

Con la ayuda del repositorio documental de la Universidad de Valladolid (UVaDOC) [13] y de la herramienta Google Scholar [14] se han localizado otros proyectos enfocados a la utilización de asistentes de voz o virtuales, incluso para lograr una interacción con un sistema de gestión de aprendizaje a través de ellos.

- Sincronización de información para un asistente virtual offline [15]

  El objetivo de este Trabajo de Fin de Grado radica en conseguir que la población dependiente que vive en zonas con mala conexión a Internet pueda consultar información, como la meteorología, a través de un asistente virtual. Para la realización de este proyecto se ha utilizado Mycroft, por lo que ha sido muy útil en aspectos como la instalación y configuración del dispositivo.

- Gestión de diálogos en asistentes virtuales para la difusión de información [16]

  Se trata de un proyecto muy parecido al presentado en la presente memoria, con la diferencia de que está más enfocado al ofrecimiento de información al usuario obtenida de diferentes fuentes sin que estos puedan realizar ninguna acción mediante el asistente virtual, que también es Mycroft.

- Moodle LMS Integration with Amazon Alexa: A Practical Experience [17]

  En el caso de este proyecto, el asistente a utilizar es Alexa, por lo que la estructura e implementación de la skill es totalmente diferente a la de una skill de Mycroft. Se ha tenido en cuenta como referencia debido a que uno de los principales objetivos que presenta es el hecho de ofrecer a los alumnos nuevas y mejores maneras de acceder al contenido las instituciones educativas a las que pertenecen. Cabe destacar el interés que tiene el hecho de aprender cómo desarrollar skills

para diferentes asistentes virtuales, algo que potenciaría en gran medida el alcance del presente proyecto.

- Alexa skill voice interface for the Moodle Learning Management System [18]

  Al igual que el anterior, este proyecto utiliza Alexa como asistente virtual, y su principal motivación es el hecho de que otros sistemas de gestión de aprendizaje como Blackboard Learn o Canvas cuentan con interfaces de voz para un uso mucho más cómodo, mientras que Moodle carece de esta característica. En este caso, el proyecto se centra en la difusión de información por parte de los profesores o del propio sistema hacia los alumnos y en la publicación de calificaciones.

Capítulo 3

Plan de desarrollo

3.1. Calendario de realización

La guía docente de la asignatura de Trabajo de Fin de Grado en la Mención de Tecnologías de la Información del grado en Ingeniería Informática de la Universidad de Valladolid indica que la duración de la misma es de 300 horas.

Este proyecto se inició en el mes de febrero de 2022, pero la dedicación ha sido mucho mayor a partir de mediados del mes de junio, ya que es la fecha de finalización del periodo de exámenes, teniendo mucho más tiempo disponible para avanzar.

Durante los primeros meses, el trabajo estuvo totalmente enfocado en la parte práctica del proyecto con el desarrollo del scraper y de las *skills* de Mycroft que se iban a implementar. Cuando todo este trabajo estuvo terminado y el asistente funcionaba perfectamente se procedió a redactar la presente memoria para concluir el proceso.

3.2. Plan de desarrollo

En la realización de este proyecto se ha seguido una metodología en cascada mediante la cual se ha diseñado, implementado y probado toda la funcionalidad antes de poder dar por concluido el prototipo.

Debido a que se trata de un proyecto pequeño, no ha sido difícil detectar los errores que han surgido tras la fase de codificación, ya que en dicha fase se han realizado pruebas unitarias y comprobaciones del fichero log generado al ejecutar el script y el fichero JSON donde se almacenan los datos recogidos por dicho script, archivos en los que se puede ver de manera sencilla dónde se ha producido el problema y por qué. Por tanto, en todo momento se sabía qué funcionalidad se estaba implementando

teniendo la certeza de que todo lo anterior no tiene ningún error. De esta forma también se es capaz de saber de una manera muy clara en qué punto del proceso nos encontramos comprobando qué funciones se ejecutan sin ningún error hasta el momento, que son todas aquellas anteriores a la que está en ese momento concreto en desarrollo.

Además, pese a no seguir una metodología incremental, una vez finalizado el proyecto es posible añadir cualquier nueva funcionalidad de una manera muy sencilla, ya que cada extracción de un tipo de información del Campus Virtual está perfectamente separada del resto y únicamente habría que añadir la función al script e implementar una nueva *skill* que lea esos datos.

En el caso concreto de este proyecto, la metodología en cascada se ha puesto en práctica de forma que la dificultad de la extracción fuese progresiva, consiguiendo obtener primero la información más básica (p. ej., email del alumno y nombre de las asignaturas en las que está matriculado en el curso actual) y posteriormente la más compleja (p. ej., eventos y mensajes). Una vez conseguida la información, el desarrollo de las diferentes *skills* se hizo de la misma manera, empezando por aquellas que simplemente muestran el contenido almacenado en el fichero JSON y terminando por las que requerían algo más de funcionalidad para construir una respuesta lo más completa posible.

A continuación, se listan las diferentes fases principales de las que se ha compuesto el desarrollo del proyecto:

N.º	Etapa	Fecha de inicio	Fecha de finalización
1	Aprendizaje sobre web scraping	01/03/2022	13/03/2022
2	Aprendizaje sobre Mycroft	14/03/2022	27/03/2022
3	Desarrollo del script	28/03/2022	19/06/2022
4	Implementación de *skills*	20/06/2022	03/07/2022
5	Desarrollo de la memoria	04/07/2022	31/08/2022

Tabla 3.1: Incrementos del proyecto

Como se puede apreciar, gracias a la elección de estas fases se puede realizar el proyecto de una forma mucho más sencilla, ya que primero se adquieren los conocimientos necesarios sobre extracción de información de Internet y asistentes virtuales para, posteriormente, aplicarlos al trabajo y poder desarrollarlo más fácilmente.

Cabe destacar que la etapa más larga, con una duración de 84 días, es la dedicada a la implementación del script encargado de acceder a la página web, obtener la información y almacenarla, así como de crear los eventos que el usuario desee. Estas tareas requieren bastante tiempo, ya que antes del propio desarrollo de la obtención de la información se debe realizar un análisis de la web para localizar los datos a extraer, además de realizarlo de forma eficiente para que el scraping se lleve a cabo rápida-

mente.

Estas fases son las principales del proyecto, pero, a su vez, algunas de ellas cuentan con diferentes tareas a realizar, lo cual se muestra de forma desglosada en las siguientes tablas.

En la Tabla 3.2 se ve como en la tarea de aprendizaje de conceptos sobre web scraping se decide utilizar Selenium como herramienta para obtener la información de la página web debido a las facilidades que aporta en sitios dinámicos como el del Campus Virtual.

Aprendizaje sobre web scraping			
N.º	Tarea	Fecha de inicio	Fecha de finalización
1	Aprendizaje de conceptos	01/03/2022	06/03/2022
2	Aprendizaje sobre Selenium	07/03/2022	13/03/2022

Tabla 3.2: Tareas del incremento 1

Las dos tareas de la Tabla 3.3, aunque se listan como dos actividades separadas, tienen bastante en común, ya que al realizar implementaciones de *skills* básicas se asientan los conocimientos adquiridos en la primera tarea de aprendizaje.

Aprendizaje sobre Mycroft			
N.º	Tarea	Fecha de inicio	Fecha de finalización
1	Aprendizaje de conceptos	14/03/2022	21/03/2022
2	Pruebas con *skills* básicas	22/03/2022	27/03/2022

Tabla 3.3: Tareas del incremento 2

Respecto al desarrollo del script, la Tabla 3.4 muestra como se ha aplicado un orden de dificultad creciente para poder llevar un proceso de aprendizaje continuo a la vez que se avanza en el proyecto.

Desarrollo del script			
N.º	Tarea	Fecha de inicio	Fecha de finalización
1	Obtención de información sobre el usuario	28/03/2022	03/04/2022
2	Obtención de información sobre asignaturas	04/04/2022	17/04/2022
3	Obtención de información sobre eventos	18/04/2022	08/05/2022
4	Creación de eventos	09/05/2022	29/05/2022
5	Obtención de información sobre mensajes	30/05/2022	19/06/2022

Tabla 3.4: Tareas del incremento 3

En cuanto a la información de los mensajes se aplica la misma idea que en la tabla anterior, y es que obtener el número de mensajes sin leer es mucho más sencillo que leer los propios mensajes, por lo que se realiza primero.

Obtención de información sobre mensajes			
N.º	Tarea	Fecha de inicio	Fecha de finalización
1	Obtención del número de mensajes	30/05/2022	07/06/2022
2	Lectura de mensajes	08/06/2022	19/06/2022

Tabla 3.5: Subtareas de la tarea 4.5

Por último, las *skills* son más sencillas de desarrollar debido a que la mayoría de ellas únicamente leen información de un fichero JSON y se la ofrecen al usuario a través del altavoz, por lo que prácticamente la totalidad del código es común (a excepción de la encargada de crear los eventos). Estas han sido implementadas siguiendo el mismo orden que el aplicado en el desarrollo del script anteriormente explicado, el cual se muestra en la Tabla 3.6.

Implementación de *skills*			
N.º	Tarea	Fecha de inicio	Fecha de finalización
1	ayuda-campus-skill	20/06/2022	25/06/2022
2	asignaturas-campus-skill	26/06/2022	26/06/2022
3	email-campus-skill	27/06/2022	27/06/2022
4	crear-evento-campus-skill	28/06/2022	28/06/2022
5	eventos-dia-campus-skill	29/06/2022	29/06/2022
6	eventos-hoy-campus-skill	30/06/2022	30/06/2022
7	siguiente-evento-campus-skill	01/07/2022	01/07/2022
8	mensajes-sin-leer-campus-skill	02/07/2022	02/07/2022
9	leer-mensajes-campus-skill	03/07/2022	03/07/2022

Tabla 3.6: Tareas del incremento 4

En la figura 3.1 de la siguiente página se muestra el diagrama de Gantt del proyecto, en el que se aprecia de manera mucho más visual la planificación del mismo, así como las diferentes tareas que forman cada una de las fases anteriormente indicados.

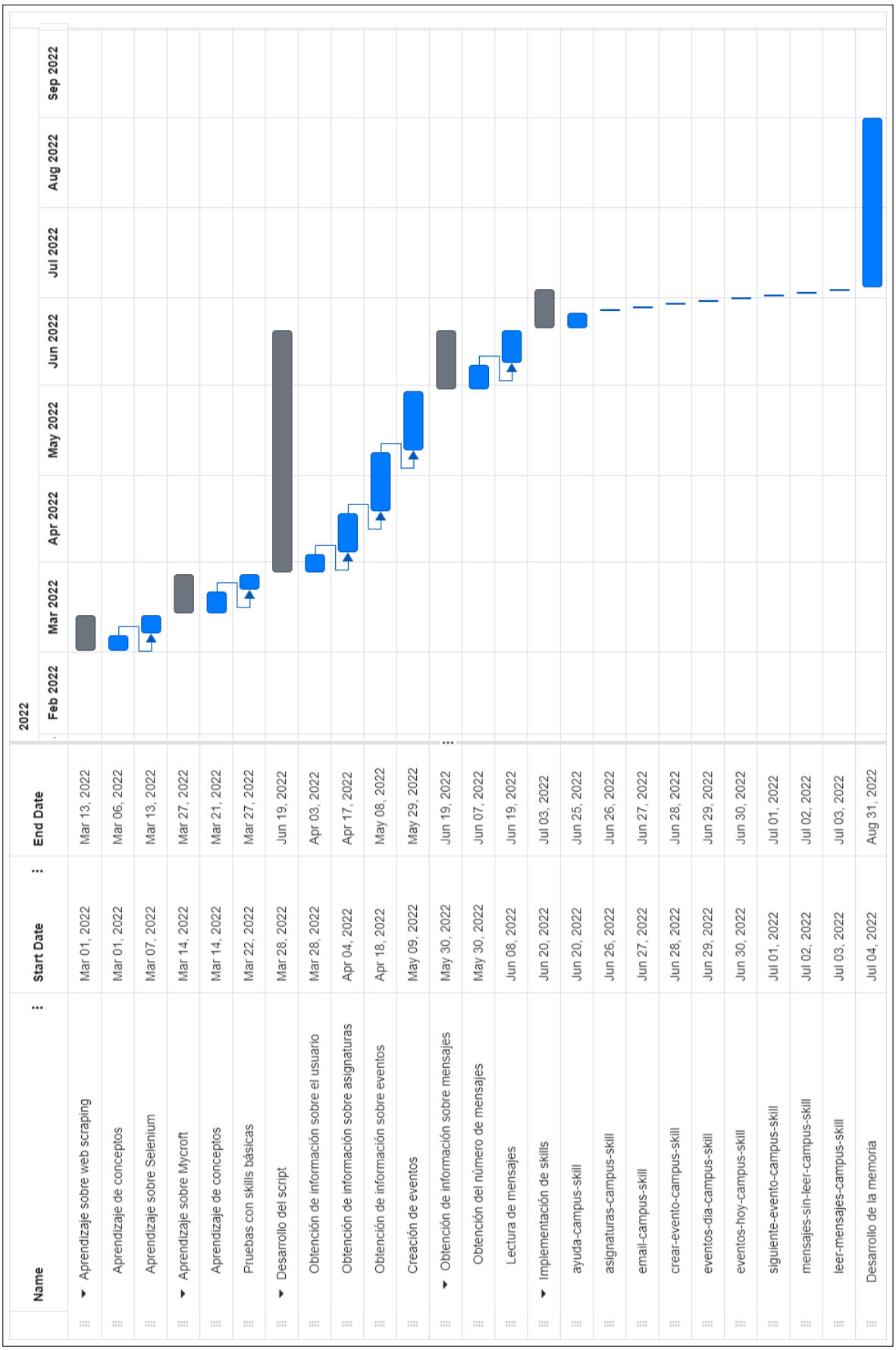

Figura 3.1: Diagrama de Gantt del proyecto

3.3. Riesgos

Como en todo proyecto software, existen ciertos riesgos que pueden tener una incidencia negativa en el desarrollo del mismo, generalmente provocando un incremento de los costes de realización o del tiempo necesario para poder finalizarlo con éxito. El grado en el que estos riesgos afectan a un proyecto depende, en gran medida, de las posibilidades de que este se materialice y del impacto que tendría en caso de suceder.

Teniendo todo esto en cuenta, en la tabla 3.7 se listan los riesgos que puede tener este proyecto, indicando sus probabilidades e impactos.

Referencia	Riesgo	Probabilidad	Impacto
R1	Baja laboral	Moderada	Bajo
R2	Cálculo erróneo de tiempos	Moderada	Moderado
R3	Cambios en los requisitos durante el desarrollo	Baja	Bajo
R4	Pérdida del código	Baja	Alto
R5	Avería en el hardware	Baja	Bajo
R6	Mayor complejidad de la esperada	Moderada	Bajo

Tabla 3.7: Riesgos del proyecto

Seguidamente, se muestran posibles acciones a llevar a cabo para mitigar (reducir probabilidad o impacto) cada uno de los riesgos anteriormente citados o los planes de contingencia a ejecutar en caso de que alguno se materialice:

- R1. Se trata de un riesgo del que es imposible reducir su probabilidad de ocurrencia, por lo que se llevaría a cabo un plan de contingencia consistente en una reducción del trabajo a realizar por el trabajador en caso de baja, lo que supondría el tener que establecer una fecha de finalización más tardía.
- R2. Se trata de otro riesgo difícil de mitigar. Por tanto, se ejecutaría un plan de contingencia en el que se establezca una fecha de finalización más tardía.
- R3. Se podría mitigar impidiendo al cliente realizar cambios en los requisitos que conlleven una gran carga de trabajo una vez se haya comenzado a desarrollar el proyecto, teniendo como plan de contingencia el establecer una fecha de finalización más tardía.
- R4. Se podría mitigar con el uso de un sistema de control de versiones que permita alojar el código fuente del proyecto en un repositorio en la nube, teniendo como plan de contingencia el volver a desarrollar todo el proyecto desde la última versión que se haya conseguido guardar.
- R5. La mejor forma de mitigar este riesgo es contar con hardware nuevo, moderno y de calidad para reducir las probabilidades de que sufra una avería. Aun así, es posible que se estropee o

falle, en cuyo caso habría que ejecutar un plan de contingencia consistente en comprar o reparar el hardware averiado, con lo que ello supone en cuanto a costes y retrasos temporales por la espera de la entrega de los productos.

- R6. Se podría mitigar mediante la realización de cursos online en los que se expliquen las tecnologías y lenguajes que se van a utilizar en el proyecto, teniendo como plan de contingencia el establecer una fecha de finalización más tardía debido al hecho de que el programador tiene que ir aprendiendo a medida que va desarrollando el proyecto.

La tabla 3.8 muestra de forma gráfica la importancia de cada uno de los riesgos en el proyecto y si alguno de ellos se encuentra por encima de la línea de tolerancia (zona gris de la matriz), lo cual indica que requiere de especial atención. Como se puede apreciar, la gran mayoría de los riesgos de este proyecto están alejados de dicha zona y ninguno se encuentra dentro de ella, lo cual es positivo.

Probabilidad / Impacto	Baja	Moderada	Significante	Alta
Alto	R4			
Significante				
Moderado		R2		
Bajo	R3, R5	R1, R6		

Tabla 3.8: Matriz de riesgos

Las tablas 3.7 y 3.8 han sido realizadas con la ayuda del libro de referencia sobre gestión de proyectos [19].

3.4. Plan de trabajo

3.4.1. Introducción al web scraping

El desarrollo del proyecto comenzó con una etapa de aprendizaje sobre web scraping debido a que no se conocía como obtener información almacenada en páginas web mediante esta técnica.

En esta fase se consultó información acerca de qué scrapers son los mejores y más comunes, como Beautiful Soup o Selenium, cómo funcionan y qué diferencias existen entre ellos. Con el objetivo de asimilar mejor estos conceptos, antes de empezar a desarrollar el código del proyecto se realizó algún programa de prueba con el que obtener información básica de páginas web simples.

En este trabajo se ha decidido utilizar Selenium por dos motivos principales: al tratarse de un prototipo y no contar con toda la funcionalidad implementada, como la autenticación, el inicio de sesión en el Campus Virtual es más sencillo de llevar a cabo porque Selenium introduce automáticamente

el usuario y contraseña como texto en los campos destinados a ello (previamente establecidos en el código del script), y porque la página web del Campus Virtual contiene muchos elementos dinámicos que se pueden manejar fácilmente con Selenium programando el flujo de acciones a realizar (hacer clic, escribir en un campo de texto, etc.).

3.4.2. Desarrollo de un script para web scraping

Un script o programa con el que realizar web scraping puede ser tan complejo o tan sencillo como se quiera, dependiendo de la cantidad de información que se desee o necesite obtener, de cómo esté almacenada o mostrada en la página web, de la complejidad de la misma (p. ej., si es dinámica es más laborioso trabajar con ella), etc. También es importante tener en cuenta qué se quiere hacer con esa información, ya que de eso va a depender la forma de almacenarla o de tratarla antes de escribirla en un fichero JSON como es el caso de este proyecto.

Para desarrollar estos scripts es muy importante el uso de las herramientas para desarrolladores con las que cuentan los navegadores web. Estas herramientas permiten obtener mucha información sobre los elementos que componen las páginas web, como sus clases CSS o sus rutas XPATH (XML Path Language), muy útiles para localizar a dichos elementos y que Selenium pueda interactuar con ellos.

Un aspecto negativo a tener en cuenta es que el tiempo de ejecución de estos programas depende principalmente de la velocidad de carga de la página web a la que acceden y en la que interactúan, ya que realmente lo que hacen es utilizar un navegador web, por lo que si se quiere obtener mucha información con un solo script puede llevar bastante tiempo.

3.4.3. Introducción a Mycroft

Una vez se adquirieron los conocimientos suficientes sobre web scraping se comenzó a investigar sobre cómo funciona el asistente virtual Mycroft. Más concretamente, el propósito era entender el código fuente de una *skill* para poder desarrollar más adelante las necesarias para el proyecto.

Durante este periodo, como en el anterior, y tras tener Mycroft instalado y configurado, se realizaron pruebas programando *skills* sencillas para comprobar que todo funcionaba correctamente, llegando a incluir en algunas el código probado en el aprendizaje de Selenium asegurando una correcta respuesta con la información obtenida de la web.

Crear una *skill* en Mycroft es un proceso sencillo, ya que únicamente debemos ejecutar el comando:

```
$ mycroft-msk create
```

y seguir la creación guiada que aparece en pantalla, en la que tendremos que establecer los siguientes aspectos:

1. Nombre.
2. Frases pronunciadas por el usuario que lanzarán la *skill*.
3. Frases que responderá Mycroft cuando la *skill* sea ejecutada.

   Estas frases son opcionales y, en nuestro caso, serán posteriormente escritas directamente en el propio código Python de la *skill*.
4. Descripción breve (una línea).
5. Descripción larga.
6. Autor.
7. Icono.
8. Color del icono.
9. Categoría principal y secundarias.

   Estas categorías son *Daily*, *Configuration*, *Entertainment*, *Information*, *IoT*, *Music & Audio*, *Media*, *Productivity* y *Transport*.
10. Etiquetas (opcionales).
11. Licencia.
12. Dependencias con paquetes Python u otras *skills*.
13. Creación o no de un repositorio en GitHub.

Al crearse la *skill* se genera una carpeta en el directorio `/home/user/mycroft-core/skills`, que es aquel en el que se encuentran las carpetas de todas las *skills* con las que cuenta el asistente.

3.4.4. Desarrollo de una *skill* de Mycroft

Una vez creada la *skill* es cuando empieza el desarrollo de su lógica escribiendo el código Python que le dará funcionalidad y que será ejecutado cuando el asistente detecte que el usuario ha pronunciado el intent correspondiente a dicha *skill*.

Las *skills* creadas para este proyecto, junto a una breve descripción y un ejemplo de diálogo, son las siguientes:

- `ayuda-campus-skill`

- Descripción: Responde ofreciendo las opciones sobre las que puede ser consultado (asignaturas, correo electrónico, mensajes y eventos) y el usuario puede contestar para obtener información más detallada sobre qué puede preguntar en cada opción.
- Ejemplo de diálogo:

  - Hey, Mycroft, ¿sobre qué te puedo preguntar?

  - Puedo crear eventos y te puedo ofrecer diferente información. ¿Sobre qué quieres hacerme una consulta? ¿Asignaturas, correo electrónico, mensajes o eventos?

  - Eventos.

  - Has seleccionado eventos. Puedes consultar los eventos de un día concreto, los eventos del día actual y el siguiente evento de tu calendario. Además, puedes pedirme que te cree un evento.

- `asignaturas-campus-skill`

  - Descripción: Responde citando el nombre completo de todas las asignaturas en las que el alumno está matriculado en el presente curso.
  - Ejemplo de diálogo:

    - Hey, Mycroft, dime mis asignaturas

    - En el curso actual estás matriculado en las siguientes asignaturas: Administración de Bases de Datos y Evaluación de Sistemas Informáticos.

- `email-campus-skill`

  - Descripción: Responde con la dirección de correo electrónico de la Universidad de Valladolid que pertenece al alumno.
  - Ejemplo de diálogo:

    - Hey, Mycroft, dime mi dirección de correo electrónico de la UVa

    - Su dirección de correo electrónico de la Universidad de Valladolid es:
    sergio.gomez.conde@alumnos.uva.es

- `crear-evento-campus-skill`

  - Descripción: Permite crear un evento en el calendario del alumno en el Campus Virtual gracias a una conversación real con el usuario en la que le el asistente le pregunta el nombre, el día y la hora del evento a crear.

    El asistente finaliza la *skill* citando un mensaje de confirmación con todos los datos del evento creado.
  - Ejemplo de diálogo:

    - Hey, Mycroft, quiero crear un evento

    - ¿Cuál es el nombre?

    - Cumpleaños

- ¿Qué día?
- El 23 de junio
- ¿Y a qué hora?
- A las 18:30

- `eventos-dia-campus-skill`

  - Descripción: Responde con todos los eventos que aparecen en el calendario del alumno en el Campus Virtual en el día indicado por el usuario al formular la petición, citando el nombre y la hora del evento, ya que el día ha sido mencionado en la consulta.
  - Ejemplo de diálogo:
    - Hey, Mycroft, ¿qué eventos tengo el día 23 de junio?
    - El 23 de junio de 2022 a las 18:30 tienes Cumpleaños

- `eventos-hoy-campus-skill`

  - Descripción: Responde con todos los eventos que aparecen en el calendario del alumno en el Campus Virtual en el día actual, citando el nombre y la hora del evento, ya que el día ha sido mencionado en la consulta.
  - Ejemplo de diálogo:
    - Hey, Mycroft, ¿qué eventos tengo hoy?
    - Hoy a las 22:00 tienes Cena de empresa.

- `siguiente-evento-campus-skill`

  - Descripción: Responde con el próximo evento que aparece en el calendario del alumno en el Campus Virtual, leyendo el nombre, la fecha y la hora del evento.
  - Ejemplo de diálogo:
    - Hey, Mycroft, ¿cuál es mi siguiente evento?
    - El 4 de octubre de 2022 a las 10:30 tienes Entrevista de trabajo.

- `mensajes-sin-leer-campus-skill`

  - Descripción: Responde con el número de mensajes sin leer que tiene el alumno en ese momento.
  - Ejemplo de diálogo:
    - Hey, Mycroft, ¿tengo algún mensaje sin leer?
    - Tienes dos mensajes sin leer

- `leer-mensajes-campus-skill`

  - Descripción: Responde leyendo todos los mensajes privados individuales que tiene el alumno en ese momento, leyendo el autor, la fecha y el contenido del mensaje.

- Ejemplo de diálogo:
  - Hey, Mycroft, lee mis mensajes
  - Tienes un mensaje.
  Mensaje de Jesús María Vegas Hernández el 12 de junio de 2022: Hola, Sergio.

3.5. Costes

Para el cálculo aproximado de los costes que habría supuesto la realización del proyecto en un entorno profesional se tienen en cuenta los siguientes datos en cuanto a los gastos relacionados con el empleado: el salario promedio para un desarrollador junior en España, que es de 19 183 € al año [20], el calendario laboral del año 2022 en Valladolid cuenta con 248 días laborables, la jornada laboral completa es de 8 horas al día y el tiempo empleado en el desarrollo del proyecto ha sido de 200 horas.

Respecto a los componentes hardware, para la realización del proyecto se ha utilizado una Raspberry Pi 3 Model B+ a la que se ha conectado una placa ReSpeaker 2-Mics Pi HAT para las funciones de micrófono, todo ello dentro de una carcasa impresa en 3D con un valor aproximado de 15 €.

Salario	1933.77 €
Raspberry Pi 3 Model B+	32.65 €
ReSpeaker 2-Mics Pi HAT	14.12 €
Carcasa	15 €
Total	1995.54 €

Capítulo 4

Análisis del proyecto

4.1. Identificación de requisitos

Una parte fundamental de cualquier proyecto de desarrollo software es el establecimiento de los requisitos que este debe cumplir para considerarlo finalizado con éxito. Esta fase es importante debido a que permite a las personas involucradas en el proyecto conocer en qué punto del mismo se encuentran y si lo implementado cumple con lo requerido.

Debido a que en el caso de este proyecto solo hay dos participantes, que son el autor y el tutor del mismo, este proceso tiene un objetivo más de seguimiento y control del proyecto que de establecer unas características que el sistema deba cumplir para satisfacer a un cliente. En un caso real, estos requisitos serían previamente discutidos entre todas las partes interesadas para dejar claro qué debe hacer el sistema y cómo debe conseguirlo en términos de usabilidad, seguridad, etc., pero sin indicar qué tecnologías o métodos utilizar, ya que es decisión del desarrollador o de su empresa.

En las siguientes secciones se explica qué tres tipos de requisitos se han establecido, su definición, y se listan los definidos para este trabajo junto a un código único y una breve descripción para su mejor comprensión y verificación de su cumplimiento.

4.1.1. Requisitos funcionales

Los requisitos funcionales son aquellos que indican qué debe hacer el software, es decir, sus funcionalidades. Estos deben aclarar explícitamente qué tiene que ofrecer el sistema, pero no deben contener explicaciones sobre cómo desarrollar dichas funciones, ya que eso es tarea del programador. Los requisitos funcionales establecidos para este proyecto son los presentes en la Tabla 4.1.

ID	Nombre	Descripción
RF1	Sonido	El sistema debe ofrecer una respuesta sonora
RF2	Escucha	El sistema debe ser capaz de escuchar las consultas del usuario
RF3	Internet	El sistema debe ser capaz de conectarse a Internet
RF4	Conversación	El sistema debe poder ofrecer una breve conversación al usuario
RF5	Asignaturas	El sistema debe poder dar a conocer al usuario sus asignaturas
RF6	Email	El sistema debe poder dar a conocer al usuario su email
RF7	Eventos	El sistema debe poder dar a conocer al usuario sus eventos (de un día y próximos)
RF8	Crear evento	El sistema debe permitir al usuario crear un evento
RF9	Mensajes sin leer	El sistema debe poder dar a conocer al usuario sus mensajes sin leer
RF10	Leer mensajes	El sistema deberá ser capaz de leer al usuario sus mensajes

Tabla 4.1: Requisitos funcionales del proyecto

4.1.2. Requisitos no funcionales

A diferencia de los requisitos funcionales, los no funcionales están relacionados con la calidad del software. En ellos se indican aspectos generales del sistema como la usabilidad, la seguridad o la disponibilidad. Los requisitos no funcionales establecidos para este proyecto son los presentes en la Tabla 4.2.

ID	Nombre	Descripción
RNF1	Disponibilidad	El sistema debe estar siempre disponible cuando se le pregunte
RNF2	Rapidez	El sistema deberá responder rápidamente a las consultas del usuario
RNF3	Portabilidad	El sistema debe funcionar en cualquier entorno Linux con Python
RNF4	Extensibilidad	El sistema debe ser capaz de poder ser ampliado en funcionalidades sin problema
RNF5	Usabilidad	El sistema debe ser fácil de aprender a usar
RNF6	Fiabilidad	El sistema debe responder siempre con la última información actualizada sin fallos
RNF7	Idioma	El sistema debe ser capaz de funcionar en castellano

Tabla 4.2: Requisitos no funcionales del proyecto

4.1.3. Requisitos de información

Por último, los requisitos de información hacen referencia a qué información debe guardar y administrar un sistema para funcionar correctamente y ofrecer los servicios indicados en los requisitos funcionales. Los requisitos de información establecidos para este proyecto son los presentes en la Tabla 4.3.

ID	Nombre	Descripción
RI1	Asignaturas	El sistema deberá almacenar los nombres de las asignaturas del alumno
RI2	Email	El sistema deberá almacenar la dirección de correo electrónico del alumno
RI3	Eventos	El sistema deberá almacenar nombre, fecha y hora de los eventos del alumno
RI4	Mensajes sin leer	El sistema deberá almacenar el número de mensajes sin leer
RI5	Mensajes	El sistema deberá almacenar autor, fecha y contenido de los mensajes del alumno

Tabla 4.3: Requisitos de información del proyecto

4.2. Casos de uso

El sistema únicamente va a ser utilizado por un solo usuario, que es el que activará las diferentes *skills* mediante sus peticiones verbales. Por lo tanto, el diagrama de casos de uso del sistema queda como el mostrado en la Figura 4.1.

Es importante destacar en este capítulo que, debido a que el proyecto se trata de un prototipo, hay cierta funcionalidad que no está disponible, siendo ese el motivo por el que tareas como la autenticación en Moodle no aparece ni en las tablas de requisitos ni en el diagrama de casos de uso.

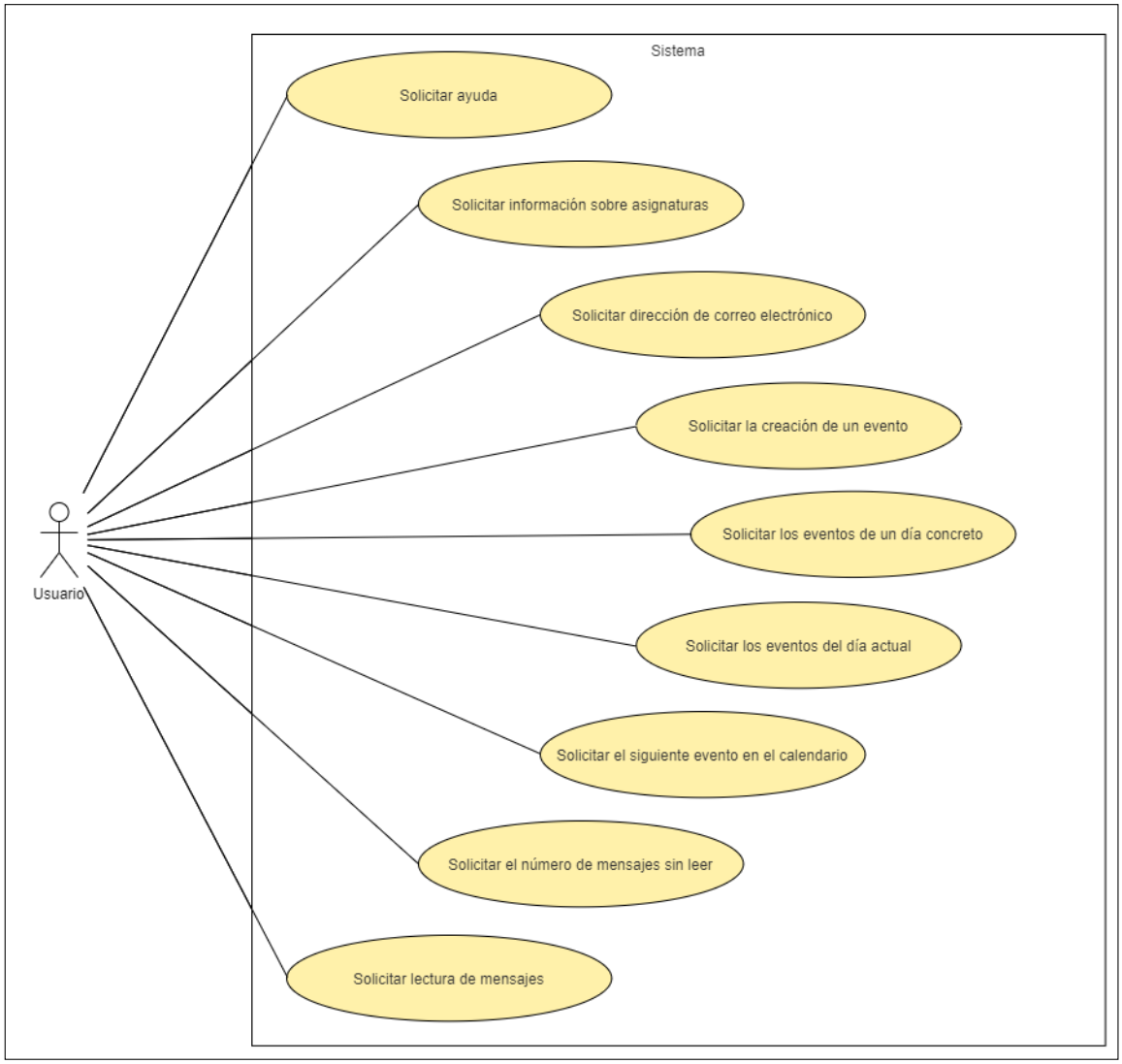

Figura 4.1: Diagrama de casos de uso del sistema

Capítulo 5

Diseño e implementación

5.1. Tecnologías utilizadas

5.1.1. Mycroft

Se trata del primer asistente virtual de código abierto del mundo, lo que le permite estar en crecimiento y mejora continuos, y enfocado en la privacidad [21] [22] [23]. Es un software ideal para poder ser ejecutado en cualquier entorno y dispositivo, desde un vehículo hasta una Raspberry Pi como en nuestro caso. Además, es útil para una gran cantidad de cosas: información, realización de tareas, reproducción multimedia, domótica...

Mycroft es modular, estando formado por Mycroft Core y ciertos componentes como la detección de la palabra de activación, el software de conversión de voz a texto (y viceversa) y el analizador de intents.

Para detectar la palabra o frase de activación, que es aquella que precede a una consulta indicando al asistente que se va a realizar una petición (por ejemplo, 'OK Google' en el caso de Google Assistant), Mycroft utiliza dos tecnologías diferentes: PocketSphinx y Precise. La primera está basada en Speech-to-Text y forma parte del paquete CMUSphinx, que es un motor de reconocimiento de voz ligero y orientado a pequeños dispositivos móviles, y únicamente reconoce el idioma inglés. En cambio, Precise se trata de una red neuronal de código abierto entrenada con patrones de sonidos, lo que facilita el uso con distintos idiomas o acentos y es la opción establecida por defecto en Mycroft para la detección de palabras de activación.

En cuanto a la conversión de texto a voz que requiere el asistente para interactuar con el usuario, actualmente Mycroft está desarrollando junto con Mozilla un software de código abierto llamado DeepSpeech desarrollado con el framework TensorFlow de Google y con base en la arquitectura Deep Speech del motor de búsqueda chino Baidu. Debido a que este sistema no está todavía listo para ser utilizado,

el usado por Mycroft es Google STT. En el sentido contrario, para poder devolver al usuario una respuesta sonora, Mycroft utiliza los motores Mimic (basado en Flite) y Mimic2 (basado en Tacotron), con una mayor calidad.

Un concepto muy importante en el funcionamiento de un asistente virtuak es el de intent, que no es más que la consulta que realiza el usuario después de pronunciar la palabra de activación. Por lo tanto, una vez el sistema ha reconocido la palabra de activación y ha convertido la petición de voz a texto, esta debe ser analizada con el objetivo de encontrar la *skill* más adecuada. Para este objetivo, Mycroft cuenta con dos alternativas: Adapt Intent Parser y Padatious. Adapt es la utilizada por defecto, mientras que Padatious se basa en redes neuronales para analizar los *intents* y se encuentra actualmente en desarrollo.

Respecto al middleware de Mycroft, este tiene dos componentes: Mycroft Core y Mycroft Home junto con Mycroft API. El núcleo del asistente es código escrito en Python que tiene el objetivo de permitir que los diferentes módulos interactúen entre sí pasándose información. Por otro lado, Mycroft Home es el lugar donde se guarda la información de los usuarios y los dispositivos emparejados, mientras que la API proporciona facilita funcionalidad para acceder a servicios externos que complementen las *skills*.

El núcleo de Mycroft cuenta con cuatro servicios más el *MessageBus*: *Enclosure*, *Voice Service*, *Audio Service* y *Skills Service*. Cada uno de estos servicios arranca en un proceso propio para, posteriormente, comunicarse con los demás a través del propio *MessageBus*. Además, todos ellos cuentan con una instancia de la clase *ProcessStatus* para poder conocer o establecer el estado del servicio en cualquier momento. Estos estados son: `NOT_STARTED`, `STARTED`, `ERROR`, `STOPPING`, `ALIVE` y `READY`.

Actualmente, Mycroft puede funcionar sobre sus propios dispositivos Mark (Mark 1 y Mark 2) utilizando una imagen dedicada y sobre cualquier Raspberry Pi 3 o 4 que tenga el sistema operativo Picroft instalado, como es el caso de este proyecto.

Mycroft Skills

Las *skills* de Mycroft son extensiones de funcionalidad que se pueden desarrollar para dotar al asistente de mayor capacidad para resolver diferentes situaciones. Para facilitar la creación, prueba y publicación de estas *skills* se encuentra disponible una utilidad basada en Python denominada *Mycroft Skills Kit* (MSK). MSK puede, además, crear una prueba para un intent o actualizar una *skill*. Para gestionar, añadir y eliminar *skills* existe otra herramienta de línea de comandos muy útil llamada *Mycroft Skills Manager* (MSM).

Las clases de todas las *skills* heredan de la clase `MycroftSkill`, la cual contiene diferentes métodos que se pueden usar en determinados momentos del ciclo de vida de la instancia de la clase:

- `__init__`. Es llamado al construirse la *skill* y normalmente es el lugar donde se declaran variables o se establece cierta configuración.
- `initialize`. Se trata del siguiente método en ser llamado y contiene la configuración final de la *skill*.
- `converse`. Este método es llamado cada vez que se escucha una consulta tras haber sido activada la *skill*. Tiene el objetivo de manejar los intents antes de ser pasados al manejador común en caso de no haber sido capaz de interpretarlos.
- `stop`. Únicamente es llamado cuando el usuario expresa el deseo de cancelar la ejecución de la *skill*.
- `shutdown`. Es llamado al finalizar la ejecución de la *skills*, asegurando que todo se detiene en orden y sin errores ni problemas.

Como en cualquier desarrollo, una parte muy importante es el establecimiento de logs con el objetivo de saber el motivo de los diferentes errores que pueden aparecer. Cada mensaje de log de Mycroft contiene la fecha y la hora, el nivel, el PID, el origen y el propio mensaje del evento registrado; y es almacenado en el fichero `/var/log/mycroft/skills.log` para su consulta. Los diferentes niveles de log son los siguientes:

- *DEBUG*. Proveen información sobre futuros problemas con el fin de solucionarlos fácilmente.
- *INFO*. Son utilizados para informar de la correcta ejecución de una *skill*.
- *WARNING*. Señalan pequeños errores existentes, pero que no impiden la ejecución de la *skill*.
- *ERROR*. Aparecen cuando un error imposibilita que una *skill* pueda ejecutarse.
- *EXCEPTION*. Extienden la información de los logs de error, incluyendo un seguimiento de la pila. Los logs de excepción deben ser llamados desde un manejador de excepciones.
- *CRITICAL*. Indican un error muy grave que interrumpe la ejecución de la *skill*.

Los ajustes de las *skills* brindan a los desarrolladores la opción de personalizar la funcionalidad o iniciar sesión en servicios externos. Por ejemplo, en el caso de este proyecto se podrían utilizar para iniciar sesión en el Campus Virtual o para establecer el usuario y la contraseña mediante texto, aunque no se haya decidido hacerlo de esta manera. Estos ajustes se deben indicar en el fichero `settingsmeta.json` (o `settingsmeta.yaml`) que tiene que estar ubicado en el directorio raíz de la *skill*. La estructura de estos ficheros está definida por la existencia de un objeto de tipo *skillMetadata* que contiene una o varias secciones, las cuales están formadas por ajustes similares o que tienen relación entre sí. Estas secciones, a su vez, están formadas por un nombre y una lista de campos con cuatro propiedades cada uno: nombre, tipo (`text`, `email`, `checkbox`, `number`, etc.), etiqueta y valor. Un ejemplo de fichero de configuración es el siguiente:

```
skillMetadata:
  sections:
    - name: Display
      fields:
        - name: show_time
          type: checkbox
          label: Show digital clock when idle
          value: "true"
```

Para poder configurar y utilizar estos ajustes se debe acceder a la página de nuestra cuenta de Mycroft. Posteriormente, para leer los valores desde el código de la *skill* se utiliza el método `self.settings.get('show_time', False)`, donde el primer argumento indica el ajuste a leer y el segundo establece el valor por defecto si no se consigue obtener de forma correcta mediante la función. Se debe tener en cuenta que la lectura de ajustes ha de ser realizada en el método `initialize` debido a que es llamado una vez la *skill* está completamente construida. Además, también es posible, desde el propio código de la *skills*, realizar acciones concretas cuando se detecta el cambio de algún ajuste, así como modificar los valores de un ajuste determinado.

Las *skills* de Mycroft requieren de paquetes externos o aplicaciones para poder ofrecer a los usuarios una experiencia de conversación realista con grandes funcionalidades. Estas dependencias pueden ser de paquetes Python, paquetes del sistema Linux e incluso de otras *skills* de Mycroft disponibles, y normalmente se encuentran establecidas en el fichero `manifest.yml`, de donde serán leídas por el MSM para comprobar si hace falta instalarlas, aunque también pueden ser instaladas manualmente por el desarrollador.

La *Skill API* facilita la interacción entre *skills* con el objetivo de complementar la funcionalidad con métodos externos ya desarrollados. Esta interacción tiene lugar gracias al *MessageBus* y es encapsulada en objetos Python. Para exportar un método y que esté disponible en otras *skills* simplemente se etiqueta con `@skill_api_method`. Es importante destacar que, debido a que se utiliza el *MessageBus* para la comunicación, los métodos deben devolver valores serializables en JSON. Para utilizar un método de otra *skill* se tiene que importar el API de dicha *skill*.

5.1.2. Web scraping

Se denomina web scraping a la obtención de información de páginas web mediante el análisis y el tratamiento de sus códigos fuente o a través de herramientas que automaticen las acciones que realiza un usuario al navegar por ellas. Gracias a esta información se consiguen duplicar contenidos de una web en otra, obtener datos que sean útiles para una empresa o para ofrecer un servicio como en el caso de este proyecto. Por otro lado, y pese a su utilidad, el gran peligro del web scraping es la legalidad, ya que se deben respetar los derechos a la propiedad intelectual de las páginas web. Por lo tanto, a la hora de realizar un script que obtenga información de un sitio de Internet hay que cumplir con la legalidad y asegurarse de que dichos datos pueden ser extraídos.

La extracción de información de páginas mediante web scraping puede realizarse con diferentes lenguajes de programación como Python, PHP o Ruby, y cada uno de ellos cuenta con diferentes librerías o herramientas para llevar a cabo la obtención de los datos. En este proyecto se ha decidido utilizar Python, por lo que se a continuación se explicarán qué alternativas existen, principalmente Beautiful Soup y Selenium, y las diferencias entre ellas.

Primeramente, hay que destacar que Beautiful Soup es una herramienta relativamente sencilla que examina el código HTML de la página objetivo, mientras que Selenium espera a que esta cargue para acceder a la información de manera automatizada y dinámica. Esta diferencia hace que la primera opción sea más recomendable para proyectos pequeños con un código bien organizado, a diferencia de la segunda, que cuenta con una mayor complejidad y coste computacional. Para el presente proyecto se ha decidido utilizar Selenium debido a la necesidad de extraer información de una página web dinámica como es la del Campus Virtual de la Universidad de Valladolid a través de la automatización de clics y entradas de texto.

Selenium puede automatizar la navegación tanto en Google Chrome como en Mozilla, así como mediante la interfaz gráfica o de forma transparente para el usuario (*headless*), como si se estuviese ejecutando en segundo plano.

Una vez elegida la herramienta se debe realizar un análisis de la página web para ubicar todos aquellos elementos que cuentan con la información que se requiere, además de para planificar cómo acceder a ellos de la forma más rápida y eficiente posible. Para extraer contenido de los elementos de una web con Selenium, la mejor manera es mediante la ruta XPATH de dichos componentes, la cual indica exactamente dónde se encuentran dentro de un código XML (Extensible Markup Language). Otras opciones son mediante el ID, nombre o clase del elemento. Un ejemplo de XPATH sería el siguiente:

```
/html/body/div[4]/div[2]/div/div/section/div/div/div/div[2]/div/div/div/div[1]
```

La obtención de estas rutas es muy sencilla, ya que simplemente hay que acceder a las herramientas para desarrolladores con las que cuentan los navegadores web. Con estas herramientas se puede observar el código fuente del sitio y copiar la ruta de los diferentes elementos haciendo clic derecho sobre su ubicación dentro del propio código, entre otras muchas funciones. Esto se puede ver en la Figura 5.1.

Por último, una vez obtenida la información debe ser almacenada en un fichero para su posterior tratamiento o lectura. Una manera sencilla de realizar este proceso en Python es guardar los datos en un diccionario (o en varios) de forma estructurada del que más adelante sean volcados a un fichero JSON, un formato fácil de leer y de manejar, mediante la función `json.dump()`:

```
ficheroJSON = '/home/serggom/scraping/datos.json'
contenidoJSON = {'asignaturas': [], 'usuario': [], 'eventos': [], 'numero_mensajes':
   [], 'mensajes': []}
```

```
# [...]

contenidoJSON['mensajes'].append({
        'autor': formatear_nombre(autor_mensaje),
        'contenido': contenido_mensaje,
        'fecha': fecha_mensaje
    })

with open(ficheroJSON, 'w') as ficheroDatosJSON:
   json.dump(contenidoJSON, ficheroDatosJSON, indent=4)
```

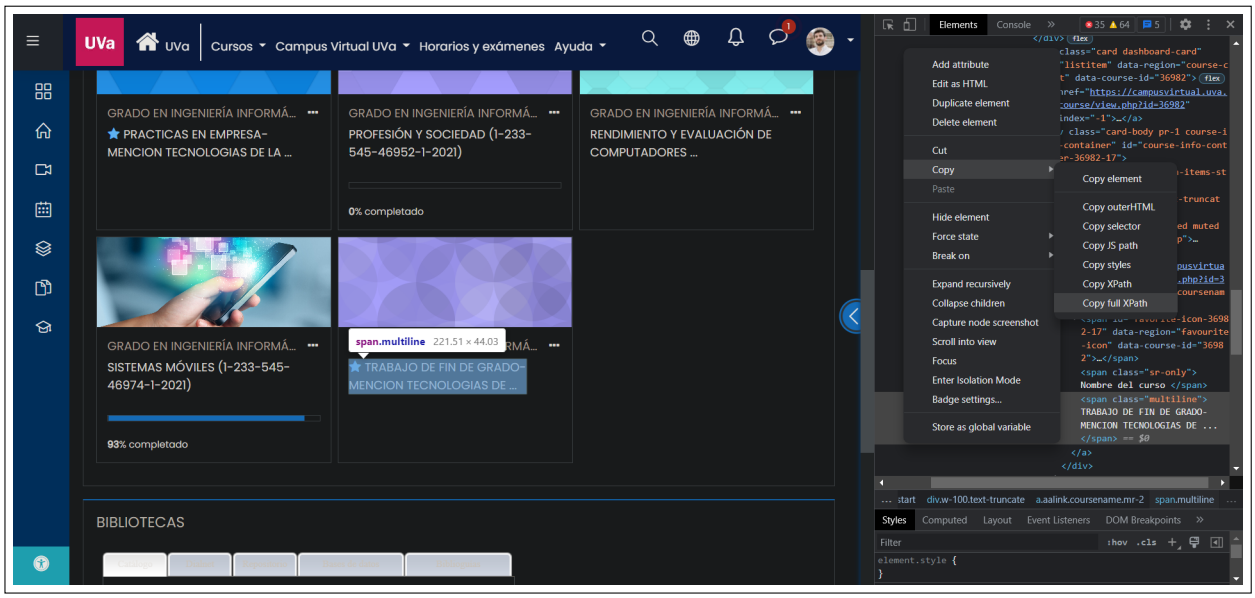

Figura 5.1: Obtención de la ruta XPATH de un elemento en Google Chrome

5.2. Recursos y herramientas

5.2.1. Raspberry Pi

Placa de un pequeño y sencillo ordenador pensado para acercar la informática y todo lo que esta puede ofrecer a los jóvenes para que estos potencien sus habilidades y conocimientos. Este dispositivo se puede usar conectándole un teclado y un monitor, aunque dependiendo del modelo puede tener conectados micrófonos, altavoces, cables Ethernet, etc. [24]

5.2.2. Picroft

Distribución basada en Raspbian (distribución Linux adaptada a la Raspberry Pi) que cuenta con Mycroft ya instalado y que se descarga como imagen para ser quemada en una tarjeta SD [25], pudiendo

solo ser ejecutada en los modelos 3, 3+ y 4 de Raspberry Pi.

5.2.3. Python

Lenguaje de programación de alto nivel muy potente, multiplataforma, de código abierto y fácil de entender o desarrollar [26], contando con la ventaja de ser interpretado, es decir, el código Python no necesita ser compilado para ejecutarse.

Es importante destacar respecto a Python las librerías que han sido indispensables para el desarrollo y correcto funcionamiento del proyecto. Estas son:

- `selenium`.

  Librería que realiza web scraping permitiendo automatizar las acciones que haría el usuario en una interacción real con la página web [27].

  Con esta librería se es capaz de acceder a la web del Campus Virtual y extraer toda la información que se le indica con los diferentes métodos con los que cuenta. Estos métodos son utilizados para localizar un elemento dentro de la página (p. ej. gracias a su ruta XPATH) y realizar acciones sobre él, como cliquear o rellenar un campo de texto.

  Cabe señalar que es posible ejecutar Selenium en un navegador sin interfaz de usuario, lo cual es útil en este proyecto para que el script Python que obtiene la información no abra una instancia gráfica de un navegador web cada vez que se ejecuta.

- `json`.

  Librería que permite trabajar con ficheros JSON, los cuales son archivos estructurados formados por colecciones de pares clave/valor para una fácil lectura y escritura en humanos y máquinas [28]. Las operaciones que hemos realizado en este proyecto son básicamente la escritura, la lectura e indirectamente la creación, ya que la primera vez que se ejecutó el código no existía el archivo que posteriormente iba a almacenar la información. Lo que permite esta librería, entre otras cosas, es pasar la información almacenada en un diccionario Python a un fichero JSON de una manera muy sencilla. También la lectura de su contenido es fácil de implementar.

- `datetime`.

  Librería con la que se puede obtener la fecha y la hora detalladas del momento actual [29], lo cual ha sido muy útil para poder comparar si ciertas fechas son anteriores al momento de la consulta o para saber si la consulta hace referencia al día actual o no cuando se pronuncia una fecha.

- `icalendar`.

  Librería utilizada para la realización de operaciones con archivos iCalendar (o iCal) [30] [31]. En este proyecto se ha sido capaz de descargar un fichero de este tipo con la información detallada

de todos los eventos y, gracias a esta librería, se ha podido "parsear" dicho fichero para poder almacenar y mostrar la información de una manera mucho más legible y entendible.

- `os`.

  Librería con un solo uso en el proyecto, que ha sido el de comprobar la existencia del fichero JSON antes de proceder a leer en él la información, para que si el fichero no existe, directamente se responde con un mensaje de error indicando que no se dispone de información a mostrar [32].

5.2.4. Pycharm

Entorno de programación muy completo que cuenta, entre otras cosas, con intérpretes y depurador además del editor de código. En él se pueden instalar diferentes plugins que ayuden al desarrollo de código Python [33].

Ha sido útil para la realización de este proyecto debido a su integración con Git para tener un control de versiones de todos los ficheros desarrollados.

5.2.5. Git

Sistema de control de versiones de código abierto con el que se ha sido capaz de guardar todos los cambios realizados en el código y volver a ellos en caso necesario [34].

5.2.6. GitHub

Plataforma de desarrollo de software donde crear repositorios en los que alojar, editar y mantener el código fuente de aplicaciones, programas, proyectos, etc. [35]

En el caso de este proyecto, se utilizará en conjunto con Git para lanzar los cambios desde la máquina local a los repositorios creados en GitHub.

5.2.7. Cron

Demonio Unix encargado de administrar regularmente procesos [36], ejecutando comandos cada cierto tiempo según se indique editando el fichero `crontab`.

5.2.8. Crontab

Fichero de texto en el que se indica qué comandos deben ser ejecutados y cada cuánto tiempo, haciéndolo en segundo plano [37].

5.2.9. Visual Paradigm

Herramienta para la creación de todo tipo de diagramas UML con los que realizar tareas como el diseño de software. En este proyecto se ha utilizado para elaborar los diagramas de esta memoria [38].

5.3. Lógica conversacional y diálogos

En esta sección se va a explicar cómo funciona el asistente de voz de Mycroft, el cual se basa en una interacción con el usuario que se desencadena cuando este pronuncia la frase de activación establecida en el asistente ("Hey, Mycroft" en el caso de este proyecto). Una vez iniciada la conversación, el asistente emite un sonido indicando al usuario que puede realizar la solicitud que desee.

Debido a la necesidad de reconocer rápidamente la voz de una forma muy precisa, Mycroft por defecto hace uso del motor Speech-to-Text (STT) de Google Cloud. Esto significa que lo que el usuario dice después del sonido emitido por el asistente es grabado y enviado a un servidor de Google en el que se transcribe y se devuelve a Mycroft. El asistente recibe el texto y busca entre sus *skills* cuál se debe ejecutar según lo percibido. Si encuentra una coincidencia, el código de esa *skill* es ejecutado generando un texto como respuesta, la cual es de nuevo enviada al servidor de Google para que se realice la operación contraria a la anterior, obteniendo el audio correspondiente a dicha respuesta. Por último, este audio es recibido por el asistente y reproducido a través del altavoz. Si la conversación no es de una sola consulta, el proceso se repite las veces necesarias hasta finalizar la interacción con la respuesta final.

En la figura 5.2 se muestra en forma de diagrama el proceso anteriormente descrito de una conversación sencilla entre un usuario y un asistente.

5.3.1. Estructura de una *skill* de Mycroft

Cada *skill* instalada en Mycroft cuenta con una carpeta en el directorio `/home/user/mycroft-core/skills`, dentro de la cual se encuentran los siguientes elementos y directorios:

- Fichero `__init__.py`. Se trata del archivo más importante y el que contiene el código que será ejecutado cuando el usuario pronuncie uno de los intent de la *skill*.

  Más concretamente, el código de la lógica de la *skill* será implementado dentro de la función `handle_nombreSkill(self, message)`, la cual es llamada al detectarse el intent correspondiente a dicha *skill*. Para realizar un desarrollo más cómodo y organizado es posible crear fuera de la

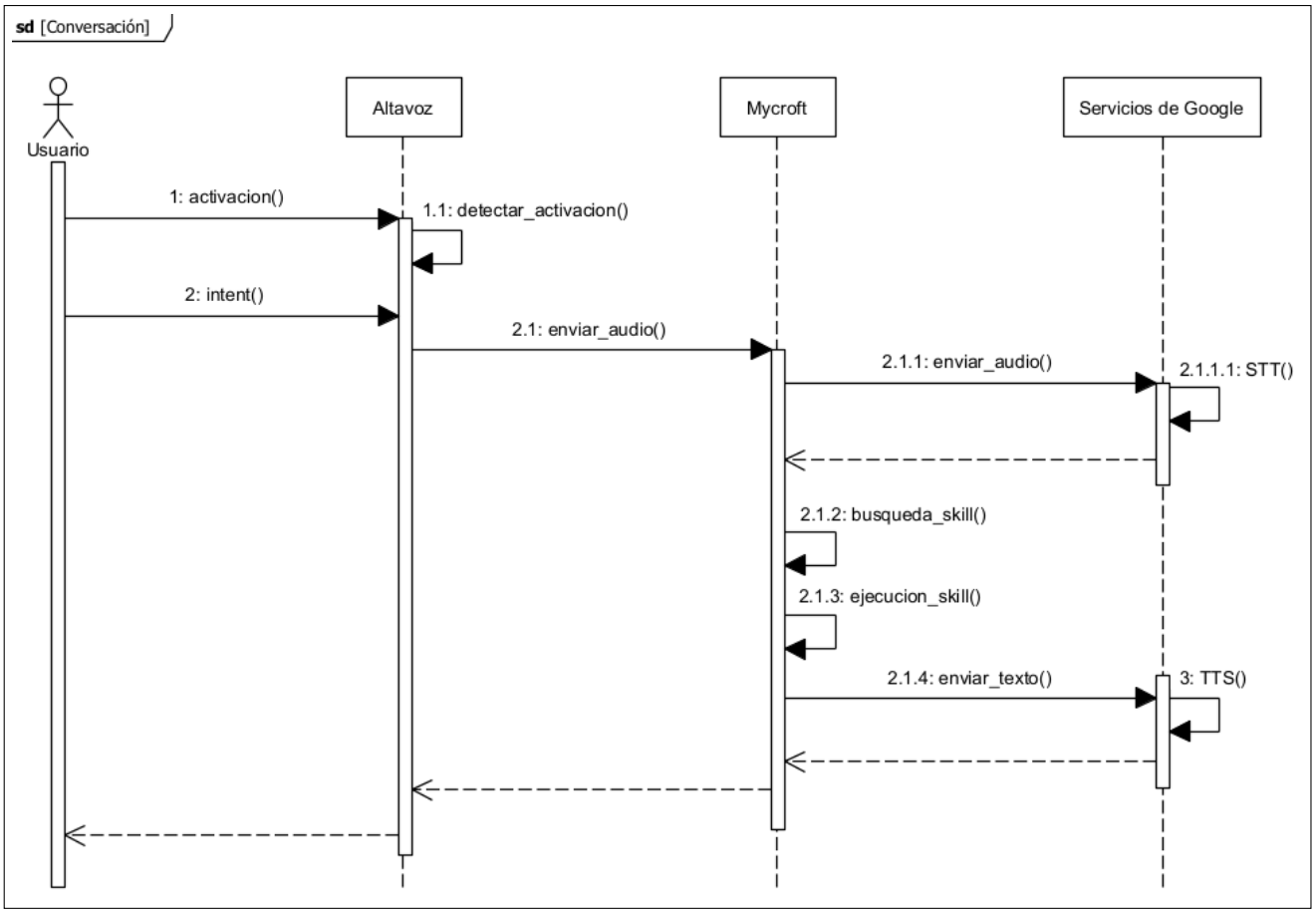

Figura 5.2: Diagrama de secuencia del funcionamiento de Mycroft

clase de la *skill* funciones complementarias que sean llamadas desde `handle_nombreSkill(self, message)`.

- Directorio `locale`. En esta carpeta se encuentran los subdirectorios correspondientes a cada uno de los idiomas en los que podría ser ejecutada la *skill*, siendo `en-us` el único que aparece por defecto.

  Este subdirectorio contiene dos tipos de ficheros:

  - Fichero `.intent` en el que están definidas las frases que, si son pronunciadas por el usuario, harán que se ejecute la *skill*.
  - Fichero `.dialog` en el que están definidas las diferentes respuestas que puede ofrecer el asistente cuando la *skill* es ejecutada. Si la *skill* ofrece una conversación con más de una interacción por parte del usuario, pueden existir varios ficheros `.dialog` con diferentes conjuntos de respuestas [39].

  Debido a que en este proyecto se va a utilizar el asistente en español, se debe crear un subdirectorio `es-es` en el directorio `locale` y mover a él el contenido de `en-us`, para ya en esa localización añadir las frases oportunas.

- Ficheros de configuración, `README` y ficheros necesarios para el uso de sistemas de control de versiones en caso si se decide usarlos.

5.4. Diseño de script de web scraping

Se ha decidido recoger la información del Campus Virtual de forma que los primeros datos son los más básicos y los últimos, los más complejos de obtener. De esta manera, y como al iniciar sesión en la página aparece nuestra área personal, es más sencillo guardar primero el email del alumno y las asignaturas en las que está matriculado con unas pocas acciones.

A continuación, se accede a la sección de mensajes directamente a través de su URL (`https://campusvirtual.uva.es/message/index.php`). En esta sección se pueden obtener fácilmente el número de mensajes sin leer mediante el icono de la notificación que muestra el Campus Virtual y el propio contenido de los mensajes junto con su autor y la fecha de envío.

Por último, se accede a la página de exportación del calendario, también mediante su URL (`https://campusvirtual.uva.es/calendar/export.php?`). En esta página se puede descargar el calendario en formato iCal para luego extraer de él la información de una manera mucho más cómoda gracias a la librería `icalendar` de Python.

5.5. Diseño de una *skill*

Todas las *skills* de este proyecto, excepto `crear-evento-campus-skill`, siguen la misma estructura de diseño e implementación, que es la mostrada en la figura 5.3.

Esta estructura se basa en la lectura de un fichero JSON, el cual se comprueba previamente si existe, con la que se extrae la información que se quiere ofrecer al usuario a través de una respuesta. En algunas *skills* se requieren unas operaciones previas a la lectura, como en el caso de `eventos-dia-campus-skill`, donde hay que obtener el día que el usuario ha pronunciado para buscar esa información en el fichero JSON.

El caso de `crear-evento-campus-skill` es diferente, ya que no lee ninguna información del fichero JSON, sino que accede al Campus Virtual mediante Selenium para crear un evento en el calendario del alumno.

5.6. Solución planteada

Para desplegar el sistema de la forma más sencilla posible, y teniendo en cuenta que el altavoz debe estar conectado a la corriente para funcionar, la solución que se ha planteado es ejecutar todo el código en la Raspberry Pi, como se puede ver en el siguiente diagrama:

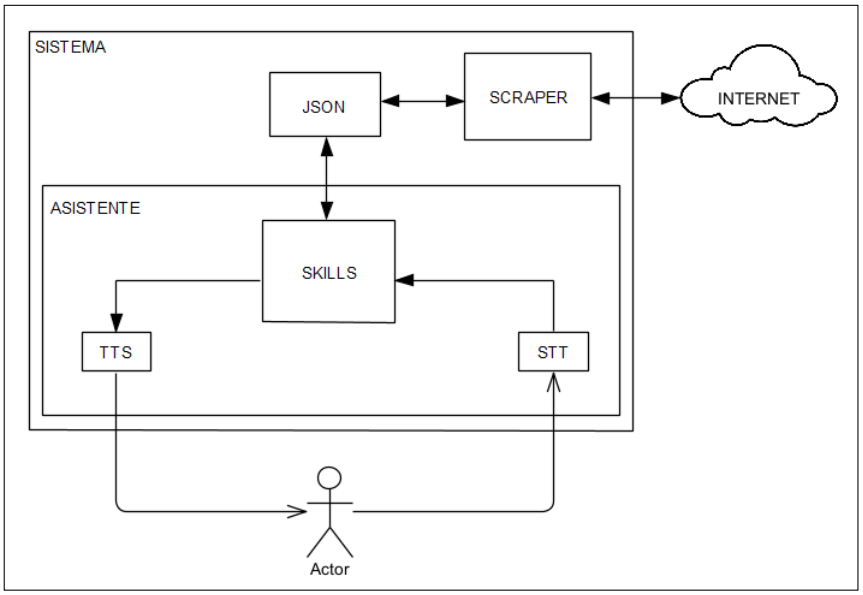

Figura 5.4: Diagrama de la solución planteada

El primer paso será desarrollar un script Python con el que realizar la tarea de web scraping y que almacenará la información obtenida de la red en un único fichero JSON. En este archivo se agruparán los datos en función de lo que representen (asignaturas, mensajes, eventos, etc.) y contando cada uno

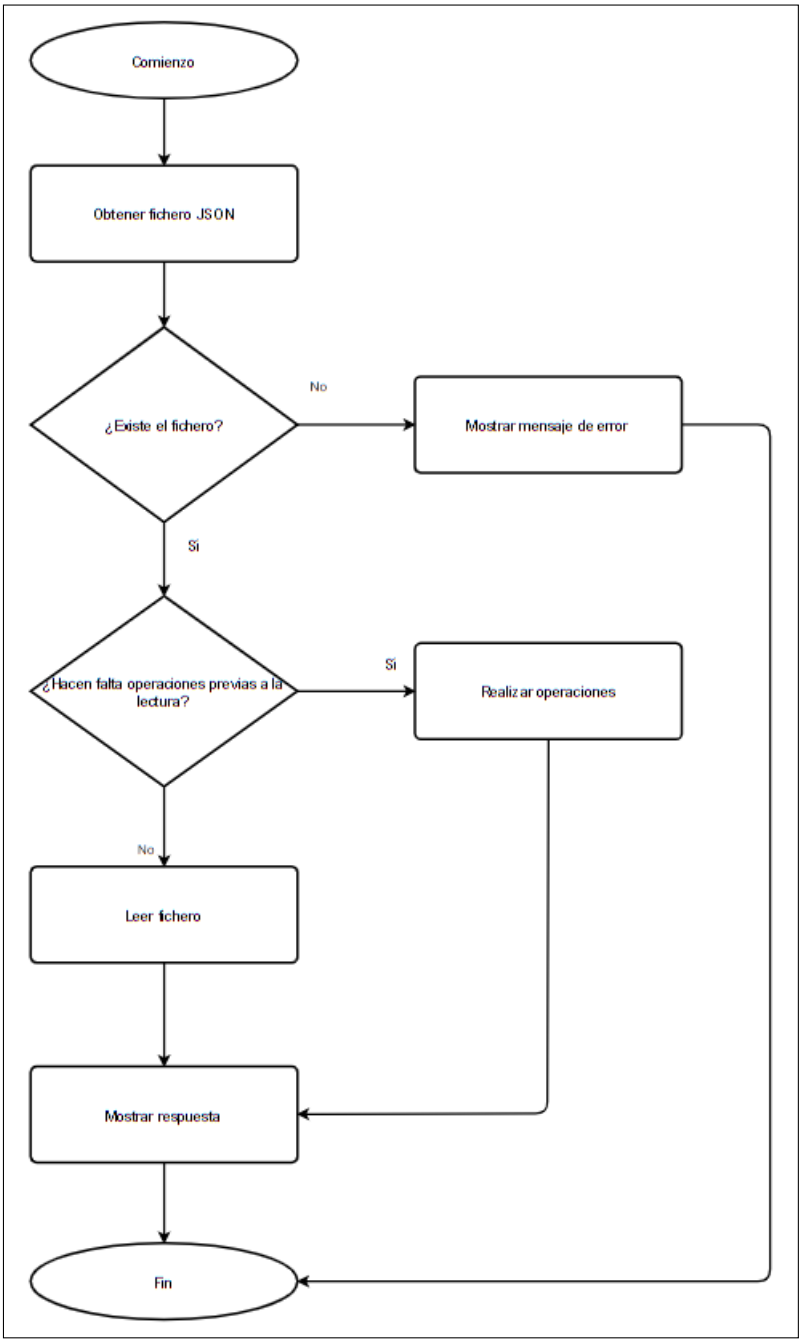

Figura 5.3: Diagrama de flujo del funcionamiento de una *skill* de nuestro proyecto

de estos “grupos” con diferentes pares clave-valor. Esta estructura organizada hace que sea muy sencillo escribir y leer información de ficheros JSON a través de scripts como el de este trabajo. Debido a que es importante que a la hora de realizar una consulta el asistente responda con información reciente, el script será ejecutado de forma automática cada tres minutos, por lo que el archivo JSON estará continuamente actualizándose.

A continuación, se instala y configura Mycroft de tal forma que reconozca y procese la voz humana en castellano y responda en el mismo idioma. Por último, se diseñan y construyen unas *skills* de Mycroft que sean capaces de acceder al fichero JSON anteriormente mencionado, leer la información que contiene y responder al usuario de forma clara lo que le ha requerido.

Capítulo 6

Despliegue

6.1. Instalación y configuración del sistema operativo

Debido a que la Raspberry Pi tenía ya instalado un sistema operativo diferente al que se iba a usar para este proyecto y, además, tenía almacenados en su memoria ficheros e información desconocida, lo correcto fue instalar el sistema operativo deseado desde cero. Este sistema operativo, en nuestro caso, es el anteriormente mencionado Picroft.

Para ello se accede a la página oficial de Mycroft para descargar la imagen que posteriormente será quemada en la Raspberry Pi. A continuación, se formatea la tarjeta SD que viene con la Raspberry Pi con el programa *SD Card Formatter* [40] y se quema en ella la imagen descargada con la ayuda del software *Raspberry Pi Imager* [41], en el cual podemos configurar algunos aspectos de la instalación como la conexión Wi-Fi o la distribución del teclado.

Una vez instalado el sistema operativo en la Raspberry Pi, la conectamos a un teclado, a un monitor y a la red para poder configurarla y empezar a trabajar con ella. Al iniciarse por primera vez, Picroft muestra la opción de realizar una configuración guiada, algo deseable para poder establecer los parámetros adecuados para trabajar correctamente. Entre estos parámetros se encuentran los altavoces y micrófono por defecto, la política de actualizaciones, cuestiones de seguridad como contraseñas, etc.

Tras estos pasos, solo faltaría vincular el dispositivo con una cuenta de Mycroft para poder empezar a usar las *skills* que vienen por defecto. Esto se realiza siguiendo los pasos que el propio altavoz indica mediante texto en el monitor. Después, y para que la Raspberry Pi hable por defecto a través del altavoz que está conectado debemos realizar una serie de ajustes que se indican y explican de forma clara seguidamente con el fin de que cualquier interesado los pueda replicar.

El primero de ellos radica en instalar los drivers, para lo cual se ejecutan los siguientes comandos:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt-get
$ git clone https://github.com/respeaker/seeed-voicecard.git
$ cd seeed-voicecard
$ sudo ./install.sh
$ sudo shutdown -r now
```

Tras ellos, Mycroft ya será capaz de responder a las consultas a través del altavoz (además de a través de texto si tenemos la terminal en modo CLI).

La instalación es realizada en inglés, siendo este el idioma por defecto para la ejecución de las *skills*, pero este proyecto está pensado para poder usar el asistente en castellano, por lo que debemos cambiar el lenguaje para las *skills* del dispositivo. Para ello ejecutamos los comandos:

```
$ mycroft-config set lang ‘‘es-es’’
$ mycroft-config set stt.mycroft.lang ‘‘es-es’’
```

También debemos editar la configuración de Mycroft del usuario con :

```
$ mycroft-config edit user
```

y escribir en el fichero lo siguiente:

```
{
"max_allowed_core_version": 20.8,
"lang": "es-es",
"tts": {
    "espeak": {
        "lang": "es",
        "voice": "es"
        },
    "module": "google"
    },
"stt": {
    "mycroft": {
        "lang": "es-es"
    }
}
```

Por último, reiniciamos el dispositivo para que los cambios se establezcan y podamos realizar consultas en castellano y que estas sean respondidas en el mismo idioma.

6.2. Instalación de Selenium y Chromium

En nuestro caso, como vamos a extraer información de una página web (el Campus Virtual de la Universidad de Valladolid) necesitamos hacer uso de una herramienta de web scraping como Selenium, la cual realiza acciones en un navegador web de manera automática incluso sin necesidad de una interfaz gráfica.

Para instalar Selenium y un navegador web como Chromium ejecutamos los siguientes comandos [42]:

```
$ sudo apt-get install python3-pip
$ pip install -U selenium
$ sudo apt install chromium-chromedriver
$ pip install requests
$ sudo shutdown -r now
```

6.3. Automatización de la obtención de información

En este punto ya contamos con todas las herramientas y utilidades necesarias para poder ejecutar el script Python que se encargará de acceder de forma autónoma al Campus Virtual, extraer toda la información necesaria y guardarla en un fichero JSON local del que posteriormente leerán las *skills* de Mycroft que se desarrollarán.

Este script será ejecutado automáticamente cada tres minutos gracias al demonio Linux `cron`. Esto se consigue editando el fichero `crontab` con el comando:

```
$ crontab -e
```

y añadiendo la siguiente línea:

```
*/3 * * * * cd /home/serggom/scraping && /usr/bin/python3 scraping.py >> registro.log
   2>&1
```

La última parte del comando redirige la salida al fichero `registro.log` para que se pueda comprobar qué ha fallado en caso de que algo salga mal.

6.4. Instalación de *skills*

Para instalar *skills* ya creadas como las desarrolladas para este proyecto, simplemente se debe descargar la carpeta acabada en "-skill" que se desee y guardarla en el directorio `/home/user/mycroft-core/skills`. De esta manera, a partir de ese momento el asistente ejecutará el código de dicha *skill* cuando el usuario pronuncie uno de los intent indicados en el fichero `locale/es-es/nombreSkill.intent`.

Capítulo 7

Pruebas

En este capítulo se muestran las distintas pruebas que se han realizado para comprobar que la funcionalidad implementada cuenta con las características que debe ofrecer el sistema a los usuarios. Estas pruebas son únicamente relativas a las consultas que pueden realizar los usuarios con el objetivo de comprobar y demostrar que se cumplen los requisitos funcionales indicados en el apartado correspondiente de esta memoria.

Estas pruebas incluyen la solicitud que debe realizar el usuario, el resultado que se espera que ofrezca el sistema y una valoración de la prueba, la cual puede ser positiva si se ha obtenido el resultado esperado o negativa en caso contrario. Como se puede ver en las siguientes tablas, la totalidad de las pruebas tiene un resultado positivo, lo que significa que el sistema funciona perfectamente y que no han existido problemas o errores a la hora de que el asistente responda correctamente a las consultas del usuario.

P1	Solicitar ayuda
Solicitud	“¿Qué eres capaz de hacer?”
Resultado esperado	El sistema responde con las funcionalidades que puede ofrecer al usuario
Valoración	Positiva

Tabla 7.1: Prueba de solicitud de ayuda

P2	Solicitar asignaturas
Solicitud	“Dime mis asignaturas”
Resultado esperado	El sistema responde con las asignaturas en las que está matriculado el usuario
Valoración	Positiva

Tabla 7.2: Prueba de solicitud de asignaturas

P3	Solicitar email
Solicitud	“Dime mi dirección de correo electrónico de la UVa”
Resultado esperado	El sistema responde con la dirección de correo electrónico del alumno en la Universidad de Valladolid
Valoración	Positiva

Tabla 7.3: Prueba de solicitud de email

P4	Crear evento
Solicitud	“Quiero crear un evento”
Resultado esperado	El sistema crea el evento indicado en el calendario del alumno en el Campus Virtual
Valoración	Positiva

Tabla 7.4: Prueba de creación de eventos

P5	Solicitar eventos de un día concreto
Solicitud	“¿Qué eventos tengo el día 23 de junio?”
Resultado esperado	El sistema responde con los eventos que tiene el alumno ese día
Valoración	Positiva

Tabla 7.5: Prueba de solicitud de los eventos de un día

P6	Solicitar eventos del día actual
Solicitud	“¿Qué eventos tengo hoy?”
Resultado esperado	El sistema responde con los eventos que tiene el alumno el día actual
Valoración	Positiva

Tabla 7.6: Prueba de solicitud de los eventos del día actual

P7	Solicitar el siguiente evento
Solicitud	“¿Cuál es mi siguiente evento?”
Resultado esperado	El sistema responde con el siguiente evento que tiene el alumno en el calendario
Valoración	Positiva

Tabla 7.7: Prueba de solicitud del siguiente evento

P8	Solicitar mensajes sin leer
Solicitud	“¿Tengo algún mensaje sin leer?”
Resultado esperado	El sistema responde con el número de mensajes sin leer que tiene el alumno
Valoración	Positiva

Tabla 7.8: Prueba de solicitud de mensajes sin leer

P9	Solicitar lectura de mensajes
Solicitud	“Lee mis mensajes”
Resultado esperado	El sistema lee todos los mensajes recibidos que tiene el alumno
Valoración	Positiva

Tabla 7.9: Prueba de solicitud de lectura de mensajes

Capítulo 8

Conclusiones

La realización de este proyecto ha sido muy útil para conocer a pequeña escala cómo funciona un asistente de voz integrado en un altavoz inteligente como el que puede tener cualquiera en su casa, siendo consciente de la gran cantidad de funcionalidades que pueden ofrecer, siendo prácticamente infinitas.

Con este trabajo se han mejorado las aptitudes y habilidades con el lenguaje de programación Python, comprendiendo lo potente que es para lo relativamente sencillo que resulta aprenderlo y trabajar con él.

Destacar también los conocimientos adquiridos sobre planificación y gestión de proyectos, necesarios para poder realizar este trabajo de una manera organizada y para conocer todo lo que hay detrás de un desarrollo software, como el diseño, los requisitos o los riesgos y su administración.

Otro aspecto a valorar es la importancia de la comunidad a la hora de realizar cualquier tipo de proyecto, ya que cualquier duda que pueda surgir sobre, por ejemplo, cómo implementar cierta funcionalidad, puede tener su respuesta en foros online como Stack Overflow. Si no se encuentra la solución al problema navegando por la red, siempre se puede iniciar una consulta en uno de estos foros donde otros desarrolladores del resto del mundo intentarán ayudar.

Más concretamente, y respecto a los objetivos marcados al inicio del proyecto, se concluye que:

- Se ha sido capaz de obtener información procedente del Campus Virtual de la Universidad de Valladolid y almacenarla para su posterior lectura por las *skills* de Mycroft.
- Se ha sido capaz de desarrollar *skills* de Mycroft a las que realizar consultas y de las que recibir sus respuestas de forma satisfactoria.
- Se ha sido capaz de ofrecer a los alumnos los nombres de las asignaturas en las que están

matriculados.

- Se ha sido capaz de ofrecer a los alumnos su dirección de correo electrónico de la Universidad de Valladolid.
- Se ha sido capaz de ofrecer a los alumnos información acerca de sus mensajes, como el número de mensajes sin leer o el contenido de los propios mensajes junto con su autor.
- Se ha sido capaz de ofrecer a los alumnos información acerca de sus eventos, como aquellos que tienen un cierto día o el siguiente en aparecer en el calendario.
- Se ha sido capaz de ofrecer a los alumnos la posibilidad de crear un evento en su calendario.
- Se han adquirido conocimientos básicos sobre el funcionamiento de un asistente de voz o virtual.
- Se han mejorado las habilidades personales con el lenguaje de programación Python.

Por último, indicar que gracias a la realización de este prototipo se ha podido confirmar la viabilidad y utilidad de mejorarlo y ampliar sus características haciendo de él un sistema completamente funcional.

8.1. Trabajo futuro

En cuanto al trabajo futuro relativo a este proyecto, indicar que se podría ampliar la funcionalidad del sistema tanto como se quisiera, convirtiendo el prototipo en una aplicación completa, teniendo como único límite el propio Campus Virtual, del que no se puede extraer más información que la que muestra. Sin embargo, con las funciones con las que cuenta actualmente es posible realizar mejoras en las *skills* desarrolladas en este proyecto, sobre todo implementando respuestas más completas o ampliando las conversaciones que se pueden tener con el asistente. Además, también es posible crear nuevas *skills* más complejas si se quiere permitir al alumno acceder a información mucho más concreta (p. ej., leer mensajes de los foros o conocer si hay novedades en una asignatura).

Una característica que sería interesante desarrollar en un futuro sería permitir a los alumnos iniciar sesión en el Campus Virtual a través de comandos de voz, para que así puedan usar el mismo altavoz y asistente diferentes personas, recibiendo cada una de ellas la información acorde a su cuenta personal.

Como en el momento de la publicación de la presente memoria el proyecto no es más que un prototipo, la implementación de todas estas características conseguiría que se tratase de una aplicación completamente funcional y lista para utilizar por cualquier alumno de la Universidad de Valladolid.

Debido a que Telegram es una herramienta de mensajería instantánea muy utilizada entre los estudiantes de Ingeniería Informática de la Universidad de Valladolid gracias a sus útiles funcionalidades como una sencilla compartición de archivos o la posibilidad de usarla en varios, podría desarrollarse

un bot de Telegram que implemente la funcionalidad de este proyecto para ofrecer a los alumnos otra forma de interactuar con el Campus Virtual más directa y sencilla.

Por último, y debido a que la información obtenida de la página web es almacenada en un fichero JSON, sería posible desarrollar estas *skills* en otros asistentes como Google Assistant o Amazon Alexa.

Bibliografía

[1] *Raspberry Pi 3 Model B+*, Raspberry Pi Foundation, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/

[2] *Installing selenium WebDriver for Linux*, Medium, [Consulta: 31 agosto 2022]. [Online]. Available: https://medium.com/@shreyas.techiespace/installing-selenium-webdriver-for-linux-7ea75d0b7df0

[3] *Siri*, Apple Inc., [Consulta: 31 agosto 2022]. [Online]. Available: https://www.apple.com/es/siri/

[4] *Asistente de Google: tu Google Personal*, Google LLC, [Consulta: 31 agosto 2022]. [Online]. Available: https://assistant.google.com/intl/es_es/

[5] *Amazon Alexa*, Amazon.com, Inc., [Consulta: 31 agosto 2022]. [Online]. Available: https://developer.amazon.com/es-ES/alexa

[6] *La 1ª Ola del EGM se estrena con un universo más amplio y novedades en su cuestionario*, AIMC, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.aimc.es/a1mc-c0nt3nt/uploads/2019/04/190404_NP_EGM_2019ola1.pdf

[7] *Google Assistant, Alexa o Siri: ¿cuál es el mejor asistente de voz?*, ADSLZone, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.adslzone.net/reportajes/domotica/google-assistant-alexa-siri

[8] *7 Puntos fundamentales a la hora de elegir una plataforma de e-learning (LMS)*, Plataforma SELF, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.plataformaself.com/recursos/7-puntos-fundamentales-a-la-hora-de-elegir-una-plataforma-de-e-learning-lms

[9] *Moodle*, Moodle, [Consulta: 31 agosto 2022]. [Online]. Available: https://moodle.org/

[10] *Canvas LMS*, Canvas, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.instructure.com/higher-education/products/canvas/canvas-lms

[11] *Chamilo*, Chamilo Association, [Consulta: 31 agosto 2022]. [Online]. Available: https://chamilo.org/

[12] *Blackboard*, Blackboard Inc., [Consulta: 31 agosto 2022]. [Online]. Available: https://www.blackboard.com/

[13] *UVaDOC*, Universidad de Valladolid, [Consulta: 31 agosto 2022]. [Online]. Available: https://uvadoc.uva.es/

[14] *Google Académico*, Google LLC, [Consulta: 31 agosto 2022]. [Online]. Available: https://scholar.google.es/

[15] S. Rabadán Fernández, *Sincronización de información para un asistente virtual offline*. Universidad de Valladolid, 2021, [Consulta: 31 agosto 2022]. [Online]. Available: https://uvadoc.uva.es/handle/10324/50433

[16] D. Díez Poza, *Gestión de diálogos en asistentes virtuales para la difusión de información*. Universidad de Valladolid, 2020, [Consulta: 31 agosto 2022]. [Online]. Available: https://uvadoc.uva.es/handle/10324/44150

[17] J. Ochoa-Orihuel, R. Marticorena-Sánchez, and M. C. Sáiz-Manzanares, *Moodle LMS Integration with Amazon Alexa: A Practical Experience*. Universidad de Burgos, 2020, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.mdpi.com/2076-3417/10/19/6859

[18] M. L. Melton and J. Fenwick Jr, *Alexa skill voice interface for the moodle learning management system*. Appalachian State University, 2019, [Consulta: 31 agosto 2022]. [Online]. Available: https://core.ac.uk/download/pdf/345088667.pdf

[19] B. Hughes and M. Cotterell, *Software Project Management*, 5th ed. McGraw-Hill Education, 2009, [Consulta: 31 agosto 2022].

[20] *Sueldos para el puesto de Programador Júnior en España*, glassdoor, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.glassdoor.es/Sueldos/programador-junior-sueldo-SRCH_KO0,18.htm

[21] *Mycroft AI Documentation. Get Started*, Mycroft AI, [Consulta: 31 agosto 2022]. [Online]. Available: https://mycroft.ai/get-started/

[22] *Mycroft – The Open Source Privacy-Focused Voice Assistant*, Mycroft AI, [Consulta: 31 agosto 2022]. [Online]. Available: https://mycroft.ai/

[23] *Who is Mycroft?*, Mycroft AI, [Consulta: 31 agosto 2022]. [Online]. Available: https://mycroft.ai/about-mycroft/

[24] *About us*, Raspberry Pi Foundation, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.raspberrypi.org/about/

[25] *Mycroft AI Documentation. Picroft*, Mycroft AI, [Consulta: 31 agosto 2022]. [Online]. Available: https://mycroft-ai.gitbook.io/docs/using-mycroft-ai/get-mycroft/picroft

[26] *Python: qué es y por qué deberías aprender a utilizarlo*, Santander Becas, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.becas-santander.com/es/blog/python-que-es.html

[27] *The Selenium Browser Automation Project*, Selenium, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.selenium.dev/documentation/

[28] *json — JSON encoder and decoder*, Python Software Foundation, [Consulta: 31 agosto 2022]. [Online]. Available: https://docs.python.org/3/library/json.html

[29] *datetime — Basic date and time types*, Python Software Foundation, [Consulta: 31 agosto 2022]. [Online]. Available: https://docs.python.org/3/library/datetime.html

[30] *iCalendar - Presentación del calendario digital*, IONOS, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/icalendar/

[31] *Parse dates with icalendar and compare to python datetime*, Stack Overflow, [Consulta: 31 agosto 2022]. [Online]. Available: https://stackoverflow.com/questions/26238835/parse-dates-with-icalendar-and-compare-to-python-datetime

[32] *os — Miscellaneous operating system interfaces*, Python Software Foundation, [Consulta: 31 agosto 2022]. [Online]. Available: https://docs.python.org/3/library/os.html

[33] *PyCharm. Get started*, JetBrains, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.jetbrains.com/help/pycharm/quick-start-guide.html

[34] *About*, Git, [Consulta: 31 agosto 2022]. [Online]. Available: https://git-scm.com/about/

[35] *About*, GitHub, [Consulta: 31 agosto 2022]. [Online]. Available: https://github.com/about

[36] *cron (8) - Linux manual page*, man7, [Consulta: 31 agosto 2022]. [Online]. Available: https://man7.org/linux/man-pages/man8/cron.8.html

[37] *IBM Docs. crontab Command*, IBM, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.ibm.com/docs/en/aix/7.2?topic=c-crontab-command

[38] *Documentations*, Visual Paradigm International Ltd., [Consulta: 31 agosto 2022]. [Online]. Available: https://www.visual-paradigm.com/support/documents/

[39] *Mycroft AI Documentation. Prompts*, Mycroft AI, [Consulta: 31 agosto 2022]. [Online]. Available: https://mycroft-ai.gitbook.io/docs/skill-development/user-interaction/prompts

[40] *SD Memory Card Formatter*, SD Association, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.sdcard.org/downloads/formatter/

[41] *Raspberry Pi OS*, Raspberry Pi Foundation, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.raspberrypi.com/software/

[42] *Introducción al WebScraping con Selenium(Py)*, Raspberry Pi Zaragoza, [Consulta: 31 agosto 2022]. [Online]. Available: https://www.raspberrypizaragoza.es/introduccion-al-webscraping-con-seleniumpy/