



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**Asistente de voz para Moodle  
UBUVoiceAssistant 2.0**



Presentado por Rodrigo Garcia Martin  
en Universidad de Burgos — Julio 2021  
Tutor: Raúl Marticorena Sánchez







UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



D. Raúl Marticorena Sánchez, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Rodrigo Garcia Martin, con DNI 71709733B, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado UBUVoiceAssistant 2.0.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 6 de julio de 2021

Vº. Bº. del Tutor:

D. Raúl Marticorena Sánchez





## Resumen

Los asistentes de voz son una tecnología de reciente aparición que se puede ver cada vez en más aspectos de la vida cotidiana, como en nuestros móviles, algunas páginas web, e incluso en las cajas registradoras automáticas que hay en algunos supermercados. Nos permiten realizar una enorme cantidad de funciones, como buscar información, enviar mensajes o navegar por diferentes páginas web, intentando que el funcionamiento sea lo más natural posible para el usuario, como si estuviese interactuando con otra persona.

En el campo de la educación se han utilizado muchas tecnologías diferentes hasta la fecha, siendo Moodle una de las más importantes, ya que permite ser un punto de unión entre alumnos y profesores, donde poder consultar todos los apuntes de las asignaturas, subir las tareas, realizar exámenes, comunicarse mediante foros y mensajería instantánea, y muchas otras funciones. También es importante destacar la facilidad para crear aplicaciones de terceros.

Debido a esto, surgió la idea de crear un asistente de voz para Moodle, para intentar hacer la experiencia más agradable. UBUVoice-Assistant está programado en Python, y usa el asistente de código abierto *Mycroft*. Se parte de una versión inicial y se buscan incluir varias mejoras en diferentes partes de la aplicación. En concreto, la instalación y configuración inicial se han simplificado en gran medida, se han realizado varios cambios en la interfaz para conseguir un aspecto más moderno, se han corregido una serie de errores y se han añadido algunas funcionalidades nuevas, como la posibilidad de interactuar con los mensajes privados de Moodle.

## Descriptores

asistente de voz, Mycroft, TTS, STT, Moodle, skill, Python

## Abstract

Voice assistants are a recently emerging technology that each day is being incorporated into more aspects of everyday life, such as cellphones, websites or even in the automatic cash registers in some supermarkets. They allow us to perform a huge number of actions, like looking for information, sending text messages or browsing the web, while trying to be as natural to the user as talking to another person.

In education, there are a lot of different technologies, staying Moodle as one of the most important ones, because students and teachers can use it as a nexus. There you can get the notes for the different subjects, upload assignments, take tests, chat with other people using forums and instant messaging, and many other things. Is worth noting the ease of creating third-party apps.

Because of that, we had the idea of creating a voice assistant for Moodle, trying to get a better user experience on the platform. I coded *UBUVoiceAssistant* in Python, and it uses the open-source assistant Mycroft. I started from an initial version and I'm looking to include several improvements in different parts of the application. The installation and initial configuration are now way easier than before, and I redesigned the interface to get a more modern appearance. I also fixed a bunch of errors and added some additional features, such as the interaction with private messaging in Moodle.

## Keywords

voice assistant, Mycroft, TTS, STT, Moodle, skill, Python



---

# Índice general

---

<b>Índice general</b>	<b>III</b>
<b>Índice de figuras</b>	<b>V</b>
<b>Índice de tablas</b>	<b>VI</b>
<b>Introducción</b>	<b>1</b>
<b>Objetivos del proyecto</b>	<b>3</b>
<b>Conceptos teóricos</b>	<b>5</b>
3.1. Asistente de voz . . . . .	5
3.2. Wake word . . . . .	5
3.3. Skill . . . . .	6
3.4. Fuzzy matching . . . . .	7
3.5. Speech-to-text . . . . .	7
3.6. Text-to-speech . . . . .	8
<b>Técnicas y herramientas</b>	<b>9</b>
4.1. API REST . . . . .	9
4.2. Moodle . . . . .	9
4.3. Mycroft . . . . .	10
4.4. Qt5 . . . . .	12
4.5. Poedit . . . . .	12
4.6. Python . . . . .	13
4.7. Bash . . . . .	14
4.8. LaTeX . . . . .	14

4.9. Metodología de desarrollo . . . . .	14
4.10. Git . . . . .	15
4.11. Visual Studio Code . . . . .	15
4.12. Análisis del software . . . . .	16
<b>Aspectos relevantes del desarrollo del proyecto</b>	<b>19</b>
5.1. Uso de contenedores Docker . . . . .	19
5.2. Creación del instalador . . . . .	20
5.3. Mejoras de la interfaz . . . . .	20
5.4. Localización . . . . .	20
5.5. Uso de contexto . . . . .	21
5.6. Instalación en Windows . . . . .	21
<b>Trabajos relacionados</b>	<b>23</b>
<b>Conclusiones y Líneas de trabajo futuras</b>	<b>25</b>
7.1. Conclusiones . . . . .	25
7.2. Líneas de trabajo futuras . . . . .	26
<b>Bibliografía</b>	<b>27</b>

---

## Índice de figuras

---

---

# Índice de tablas

---

6.1. Comparación de características . . . . .	24
-----------------------------------------------	----

---

# Introducción

---

Los asistentes de voz son una tecnología nueva que no se ha usado apenas hasta el momento para la educación. Sin embargo, se ha visto que puede ser muy interesante usarlas, debido a que es mucho más rápido y cómodo buscar determinadas informaciones con un comando de voz en vez de navegar por diferentes menús que pueden ser bastante complejos.

Este proyecto se basa en mejorar la primera versión de un asistente de voz que permitía conectarte a una plataforma *Moodle* (como UBUVirtual). Con respecto a la anterior versión, se busca simplificar el proceso de instalación y configuración inicial, conseguir una apariencia más moderna, incluir una funcionalidad que te permita consultar y enviar los mensajes privados, ampliar el sistema de ayuda y hacer que la conversación sea más natural, haciendo que el programa vaya almacenando el contexto.

En esta memoria se irán explicando los conceptos necesarios para su comprensión, así como las tecnologías usadas para realizar el proyecto, los problemas que se han encontrado a lo largo del desarrollo, las decisiones tomadas y las soluciones para intentar solventarlos.



---

# Objetivos del proyecto

---

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

## Objetivos generales

- Mejorar la instalación del proyecto.
- Facilitar la internacionalización.
- Añadir sistemas de *logging*.
- Mejorar el estilo visual.
- Conseguir mayor robustez y estabilidad en la aplicación.
- Mejorar la experiencia de usuario con una interacción más natural.
- Ampliar funcionalidad de interacción con Moodle.

## Objetivos técnicos

- Crear un script de instalación, que sea capaz de descargar e instalar todos los componentes necesarios, desinstalar el programa, o poder actualizar a una nueva versión.
- Construir nuevas interfaces gráficas.

- Usar HTML embebido en la aplicación para hacer que sea más visual que con los *widgets* del sistema operativo.
- Incorporar una biblioteca de internacionalización para incluir nuevos idiomas de manera más sencilla, y sin tener que modificar el código.
- Crear una *skill* de *Mycroft* que aproveche el contexto para recordar las anteriores interacciones.
- Aprovechar el *fuzzy matching* para poder entender lo que el usuario quiere decir aunque el reconocimiento de voz no reconozca correctamente las palabras.



---

## Conceptos teóricos

---

En este apartado se desarrollan algunos de los principales conceptos teóricos que son necesarios para comprender el proyecto, como las definiciones de los componentes usados, o su funcionamiento.

### 3.1. Asistente de voz

Los asistentes de voz [30] son programas que permiten a un usuario interactuar con una máquina, intentando que la comunicación entre ellos sea lo más natural posible, como si estuviésemos hablando con otra persona. Habitualmente para esto se usa directamente la voz, aunque también se suelen poder introducir órdenes escritas que realizan las mismas funciones. El asistente procesa las órdenes y responde de una forma similar.

### 3.2. Wake word

Para poder distinguir si una persona está hablando a un asistente de voz o lo está haciendo por cualquier otro motivo, se suelen usar una serie de palabras que el usuario debe pronunciar para que el asistente comience a escuchar y procese la orden que se le da. Estas palabras son conocidas habitualmente como *Wake word* [2]. Cada asistente usa una *wake word* diferente. En Mycroft es configurable, y para este en particular, se suele usar “*Hey Mycroft*”

### 3.3. Skill

Las *skills* [16] son programas o aplicaciones que están diseñadas para un asistente de voz. Normalmente cada una de las *skills* se encarga de una función diferente, por ejemplo, una podría encargarse de mostrarte los próximos eventos del calendario mientras que otra se encarga de leerte las últimas noticias. Cada *skill* está compuesta por varios elementos, como pueden ser las *utterances*, los *intents*, los *dialogs*, los *prompts* o el contexto, que se describen más adelante.

#### Utterance

Una *utterance* [4] es la frase que dice el usuario que sirve para activar una *skill* concreta y que en algunos casos causará que el asistente realice una acción determinada. Un posible ejemplo de *utterance* relacionada con UBUVirtual sería: “Dime los próximos eventos de Sistemas Distribuidos”

#### Intent

Un *intent* [4] son unas palabras clave del *utterance* que permiten al asistente determinar cuál es la acción que el usuario quiere realizar. Una *skill* puede tener asociados varios *intents*. Varios de ellos pueden lanzar una misma acción, consiguiendo una interacción más natural, ya que el usuario puede pedir lo mismo de diferentes formas, por ejemplo: “Abrir el calendario” o “Consultar los eventos”.

#### Dialog

Este término es específico de *Mycroft*, aunque muchos otros asistentes usan otras herramientas para desempeñar la misma función. Los *dialogs* son las frases con las que responde el asistente a las peticiones que hace el usuario. También es posible que en vez de un *dialog*, la respuesta sea completamente dinámica y no se usen. En el ejemplo anterior, el *dialog* podría ser “Los próximos eventos son: Entrega de la práctica 2 para el viernes 30 de Abril”.

#### Contexto

El contexto es una herramienta que se usa cada vez más dentro de los asistentes de voz ya que sirve para guardar parte de la información que se ha intercambiado en las anteriores preguntas y hacer que en las próximas interacciones los resultados ofrecidos por el programa estén relacionados. Por

ejemplo, podríamos decir primero “Dime los foros de Sistemas Distribuidos” y si luego decimos “Crear un hilo en los foros de esa asignatura”, el programa asociará “esa asignatura” con “Sistemas Distribuidos”.

## Prompt

Los *prompts* son preguntas que el asistente hace al usuario si el asistente necesita más información para completar la orden del usuario. Por ejemplo, si queremos crear un hilo en los foros de una asignatura, el asistente podría repetir el texto que ha entendido, y finalmente, usando un *prompt*, preguntar al usuario si el texto que ha entendido es correcto o no.

## 3.4. Fuzzy matching

Debido a que los sistemas de reconocimiento de voz a día de hoy distan bastante de ser perfectos, en ocasiones no detectan correctamente las palabras que el usuario dijo. Esto suele ocurrir de manera más habitual en el caso de palabras que no están recogidas en un diccionario o son usadas de manera frecuente, como por ejemplo, los nombres propios. Si se tiene una lista de opciones entre las que el usuario tiene que elegir una, se puede usar *fuzzy matching* [6] para estimar cuál es la palabra más parecida de entre las opciones a lo que el usuario quiso decir. En nuestra aplicación, se pueden obtener los nombres de los contactos del usuario, estimar el parecido de la frase identificada por el reconocimiento de voz con cada uno de ellos y elegir el más similar.

## 3.5. Speech-to-text

El *Speech-to-text*, también conocido como STT o reconocimiento de habla [28] es una disciplina que desarrolla una serie de tecnologías para interpretar y traducir el lenguaje hablado en texto. Habitualmente este tipo de tecnologías se conocen como “Reconocimiento de voz”, pero este término no es correcto, ya que se usa para diferenciar la voz de una persona de la de otra.

El reconocimiento de voz puede ser o bien ajustado para una sola persona, o bien independiente del hablante. En el primero de los casos hay que realizar un entrenamiento previo, en el que la persona tiene que decir unas palabras concretas para que el modelo se ajuste a su voz concreta.

Esta tecnología se está usando en la actualidad en numerosos interfaces de voz diferentes, como en sistemas de marcación telefónica, en domótica, en sistemas preparados para personas discapacitadas o bien en dictado de textos.

### 3.6. Text-to-speech

El *Text-to-speech*, también conocido como TTS o síntesis de voz [29] es una tecnología que permite generar habla a partir de textos. En un primer momento esto se realizaba en equipos específicamente diseñados para ello, aunque en la actualidad se hace por software en multitud de dispositivos de ámbito general.

Esto se puede conseguir bien a través de sonidos pregrabados que se unen en el momento de generar las frases, o bien se crea un modelo de las características del habla humana, permitiendo generar una voz completamente sintética.

También es importante destacar que suele estar formado por dos partes. La primera de ellas convierte abreviaciones o números que pueda haber en el texto a palabras completas, tal y como las diríamos, y les asigna los fonemas que le corresponden a cada una de las letras. La segunda parte se encarga de la generación de los sonidos a partir de los fonemas anteriormente generados.

---

## Técnicas y herramientas

---

### 4.1. API REST

Una API (*Application Program Interface*) es un conjunto de definiciones y protocolos usados para desarrollar e integrar diferentes aplicaciones entre sí. En una API REST [19] se establecen una serie de posibles peticiones HTTP que se pueden usar para intercambiar datos o realizar operaciones entre los diferentes componentes del sistema final.

### 4.2. Moodle

Moodle [8] es una plataforma educativa gratuita y de código abierta que permite poner de manera sencilla a profesores y estudiantes en contacto, y ofrece multitud de herramientas para impartir y gestionar cursos a distancia. Se puede descargar desde su página oficial e instalar en prácticamente cualquier máquina en cuestión de minutos, o bien usar una versión de prueba como la **Mount Orange School**. Moodle se usa en un gran número de colegios y universidades en todas partes del mundo, aunque en muchas de ellas se han implementado funciones específicas que son necesarias para esa institución concreta u ofrecer un diseño alternativo de la plataforma, pero manteniendo las funcionalidades básicas y las ventajas que ofrece, como la API REST, entre otras. Este es el caso de la Universidad de Burgos, cuya plataforma se llama UBUVirtual.

### WebServices

Web Services [18] es el nombre que da Moodle a su API REST. Estos dan muchas facilidades a la hora de interactuar con la plataforma desde

aplicaciones de terceros debido a que son peticiones muy simples y concretas. Son una gran ventaja con respecto a otras plataformas que no tienen este tipo de tecnologías ya que en esos otros casos habría que recurrir a técnicas bastante más complejas. Como en la mayoría de APIs, es necesario primero obtener un código que identifica la sesión del usuario, habitualmente conocido como token, mediante las APIs que nos ofrecen para identificarnos como si estuviésemos usando la aplicación oficial de Moodle. A partir de ese punto, enviando nuestro token en cada petición a la API podemos interactuar con el resto de la plataforma y realizar las diferentes acciones, como consultar el calendario, los foros o los mensajes.

### 4.3. Mycroft

Mycroft [13] es un asistente de voz gratuito y de código abierto. Actualmente está disponible tanto para sistemas basados en Linux como para Android, Raspberry Pi y dos dispositivos fabricados por Mycroft, el Mark 1 y el Mark II. Es una aplicación que se ejecuta directamente en la máquina cliente, a diferencia de otros como Google Assistant que se ejecutan completamente en el servidor. También es diferente con respecto a otros muchos asistentes ya que Mycroft es modular y permite activar, desactivar y sustituir componentes[17], dependiendo de las necesidades.

Además, una de las principales ventajas de ser código abierto es que se pueden ver los componentes que lo forman de manera más sencilla y poder separar responsabilidades en el caso de tener que reimplementar alguno de sus componentes. Por defecto trae implementados seis componentes, aunque sólo nos importan cuatro de ellos para este proyecto, que se describen a continuación.

#### Audio

Este componente se encarga tanto de transformar la voz del usuario al texto que se usa como entrada (llamado *speech-to-text* o STT) como de transformar el texto de salida a voz (*text-to-speech* o TTS). Para STT actualmente se usa por defecto el motor de análisis de voz de Google, aunque también es importante destacar que están trabajando en un motor completamente *open-source* en colaboración con Mozilla, llamado DeepSpeech. A pesar de usar estos por defecto, también se pueden usar otros como el IBM Watson o el servicio de wit.ai. Debido a que para usar la mayoría de estos motores se necesita un modelo entrenado previamente, que puede llegar a ser muy pesado y habitualmente es imposible descargar al ser software

privativo, en la mayoría de estas opciones es necesario recurrir a un servidor en Internet que analice los *clips* de sonido que se han grabado en el cliente. Con respecto a TTS, algunas de las configuraciones más básicas se pueden hacer desde su página web. Ahí te permiten seleccionar tres motores, Mimic 1 si seleccionas British Male, Mimic 2 si seleccionas American Male o Google Voice para usar la API de Google TTS. Se pueden configurar algunos otros como eSpeak, Microsoft Azure o Amazon Polly editando la configuración de forma local. Ya que algunos de estos motores son más ligeros e incluso hay opciones de código abierto, es posible generar y reproducir la respuesta sin necesidad de conexión a Internet.

## Voice

Este módulo es el que se encarga de detectar la *Wake Word* dicha por el usuario. Por defecto se usa Precise, que es una red neuronal entrenada con sonidos y que permite usar cualquier frase especificando los fonemas usados. Sin embargo, en caso de que este motor no esté disponible, ya sea porque es incompatible o porque el equipo no tiene tanta potencia de cálculo, es posible usar PocketSphinx que es más ligero pero sólo funciona correctamente en inglés.

## Skills

Esta parte de Mycroft es un servicio que se encarga de relacionar el *utterance* entendido por el módulo de audio con alguno de los *intents* definidos en las *skills* que haya instaladas. También se encarga de enlazar alguna de las palabras de la *utterance* con las variables que hay en los *intents* (por ejemplo, en “Dime los próximos eventos de Sistemas Distribuidos”, asigna “Sistemas Distribuidos” a la variable de curso). Esto se consigue gracias a un *intent parser*. Actualmente hay dos, Adapt y Padatious. Padatious está basado en redes neuronales y es bastante sencillo de usar, ya que tiene una buena documentación, se pueden declarar los *intents* de manera muy sencilla y gran parte del trabajo lo hace la red neuronal de manera transparente al programador. Adapt, en cambio, es un sistema bastante más ligero que está pensado para dispositivos con menos capacidad de cómputo ya que la mayor parte del trabajo se hace en el momento de programar la *skill*, lo que junto a su mala documentación, hace que sea más complejo de usar. Sin embargo, en el primero no se ha implementado la posibilidad de usar contexto, ya que es un sistema que necesita bastante más tiempo de desarrollo

## MessageBus

El MessageBus es un módulo que permite que los componentes anteriormente mencionados se comuniquen entre sí. Se basa en un *websocket* que se usa para transferir mensajes entre las diferentes partes de Mycroft e incluso permite integrar aplicaciones de terceros. Por ejemplo, cuando el usuario dice una frase, esta se interpreta en el módulo de Audio y se transfiere el texto al módulo de Skills mediante el bus, y este le devuelve la respuesta a través del bus, que se transforma a voz y se reproduce.

## 4.4. Qt5

Qt [26] es un *framework* orientado a objetos que permite desarrollar interfaces gráficas, que puede usarse tanto en Windows como en Mac o Linux, entre otros, sin tener que realizar grandes cambios en la aplicación. Gran parte del código está hecho y pensado para C++, aunque hay un gran número de *bindings* que permiten usar sus ventajas en otros lenguajes de programación. Este *framework* está disponible bajo dos tipos de licencias, una de código abierto y otra comercial.

## Qt Designer

Qt Designer [27] es una aplicación que permite desarrollar de manera muy rápida e intuitiva prototipos para interfaces que usen la biblioteca Qt. Permite usar un gran número de *Widgets* que trae la biblioteca, como campos de texto, imágenes, casillas, o desplegables. Sin embargo, para funcionalidades más avanzadas es necesario hacerlo mediante código. La aplicación permite convertir los prototipos a código fuente o exportarlo a un formato que se puede cargar en muchos de los lenguajes soportados.

## 4.5. Poedit

Poedit [10] es un programa multiplataforma que está pensado para ayudar a traducir multitud de programas de una manera sencilla. Tiene una versión gratuita y de código abierto que permite editar archivos basados en la biblioteca GNU *gettext*. También ofrece una versión comercial que ofrece algunas funciones adicionales, como es la posibilidad de unirse a la comunidad de usuarios para compartir y descargar traducciones, lo que agiliza el trabajo.



## 4.6. Python

Python [25] es un lenguaje de programación interpretado y con tipado dinámico. Es multiplataforma ya que se puede usar en Windows, Mac y Linux y también multiparadigma, porque se puede usar para hacer programación imperativa, orientada a objetos y en las últimas versiones también para programación funcional. En los últimos años se ha vuelto extremadamente popular debido a que es un lenguaje sencillo de entender, tiene una gran cantidad de bibliotecas para realizar todo tipo de aplicaciones y además permite crear programas en un menor tiempo que otros lenguajes. Algunas de las bibliotecas reseñables que he usado son las siguientes:

### PyQt5

PyQt5 [12] es un *binding* no oficial para usar la biblioteca de interfaces gráficas que hemos elegido, Qt. Sin embargo, pese a no ser oficial, es el más completo disponible y además se puede usar sin muchos problemas siguiendo la documentación oficial de Qt.

### requests

Requests [11] nos ofrece la posibilidad de hacer peticiones HTTP de manera más sencilla y más elegante que usando las bibliotecas por defecto de Python. Se ha usado para interactuar con la API de Moodle de una manera eficaz.

### gettext

Gettext [3] es una biblioteca que viene en el paquete por defecto de Python que sirve para hacer bastante más sencilla la internacionalización de un programa. Esta biblioteca permite cargar las frases de texto a partir de archivos codificados en un estándar usado habitualmente en sistemas GNU.

### babel

Babel [1] trae una serie de herramientas relacionadas con el idioma. En concreto, permite obtener de manera automática el nombre del idioma sin tener que añadirlo explícitamente en cada idioma nuevo que añadamos al programa.

## fuzzywuzzy

Fuzzywuzzy [14] permite simplificar el proceso del *fuzzy matching* al tener que comparar el nombre dicho por el usuario con los elementos que encontramos en la plataforma, especialmente útil para tratar con nombres propios.

## 4.7. Bash

Bash [21] es un lenguaje de órdenes usado ampliamente en los sistemas Linux. Este lenguaje permite al usuario escribir órdenes en forma de texto desde una terminal. Nos ofrece numerosas herramientas, siendo una de las más importantes la posibilidad de ejecutar comandos de manera automatizada desde un archivo de texto, conocido habitualmente como *script*. En el proyecto se ha usado para crear la herramienta de instalación.

## 4.8. LaTeX

LaTeX [24] es un sistema de composición de textos, usado habitualmente para artículos y libros científicos ya que permite conseguir alta calidad tipográfica. Una de las principales características es que procesa archivos de texto plano en los que se incluyen marcas que indican el formato que tiene que usarse en el documento final. Hay múltiples paquetes que permiten crear documentos usando este sistema, siendo Tex Live una de las más habituales y completas.

## 4.9. Metodología de desarrollo

Para desarrollar el proyecto se ha hecho uso de una de las metodologías ágiles más populares, SCRUM[20], aplicándole algunos cambios que son necesarios. A lo largo del proyecto se han ido realizando una serie de sprints, siendo los primeros de dos semanas y los finales de sólo una para arreglar los últimos errores. También se han sustituido las reuniones diarias por una reunión al principio y al final de cada sprint. Se han seguido también algunas prácticas como la integración continua, en la que se realizan pruebas del código de forma habitual a lo largo del desarrollo.

## 4.10. Git

Git [22] es un programa de control de versiones ampliamente utilizado para gestionar el código fuente durante el desarrollo de aplicaciones. Este sistema se puede usar de forma local o también se puede sincronizar el contenido de los proyectos en un repositorio, permitiendo a múltiples personas colaborar. Una de las funcionalidades clave es el sistema de ramas, que permiten trabajar en diferentes funciones independientes entre sí al mismo tiempo.

### Github

Github [23] es una de los principales servidores que usan Git para el control de versiones. Esta plataforma aloja gran cantidad de repositorios de código abierto de forma gratuita. Además, permite crear Issues o tareas en las que ir organizando el trabajo a realizar. Ofrece una herramienta muy potente para integración continua llamada Github Actions. Además, Github tiene una API que permite a otros desarrolladores añadir funcionalidades extra para adaptarlo a tus necesidades.

### Zenhub

Zenhub [32] es una de las múltiples aplicaciones integradas con Github que existen para poder crear un tablero Kanban para organizar las tareas y usar otra serie de funcionalidades útiles relacionadas con SCRUM y otras metodologías ágiles, como la gestión de sprints o la generación de gráficas que permiten conocer la evolución del proyecto a lo largo del tiempo.

## 4.11. Visual Studio Code

VSCoide [31] es un entorno de desarrollo integrado, o IDE, que permite editar archivos de texto plano, como el código fuente, y además cuenta con una serie de herramientas que permiten hacer más cómodo y rápido el trabajo y detectar algunos errores antes de llegar a probar el código. Está desarrollado por Microsoft y tiene una gran comunidad que ha desarrollado numerosas extensiones que añaden algunas funcionalidades muy útiles. Las más relevantes son:

## Code Spell Checker

Esta extensión es un corrector que analiza los ficheros que tengas abiertos y marca las palabras con errores tipográficos. Soporta varios idiomas, entre ellos el inglés y el español, ambos usados en el proyecto.

## GitLens

GitLens añade una serie de funcionalidades adicionales a la potente integración con Git ya incluida en el IDE. En concreto, mejora la gestión de ramas y la visualización de cambios en los *commits* anteriores.

## LaTeX Workshop

Esta extensión permite trabajar de forma sencilla con los documentos LaTeX, añadiendo algunas plantillas de autocompletado para las marcas de capítulo, sección o tablas, y añadiendo la posibilidad de generar los archivos PDF finales de la memoria una vez acabada la redacción.

## Pylance y Python

Estas dos extensiones ofrecen funcionalidades imprescindibles para trabajar con Python dentro de Visual Studio Code, siendo algunas de las más importantes el autocompletado, las ayudas de tipos o la búsqueda de errores y *debugging*.

## Python Docstring Generator

Esta última extensión permite generar parte de la documentación incluida en cada método a partir del código que hemos programado anteriormente. Es especialmente importante porque dejar una buena documentación mejora la calidad del código.

## 4.12. Análisis del software

Uno de los procesos que se debe hacer en el desarrollo de software es su análisis. En este proceso se comprueban algunos aspectos del mismo, como la calidad, mantenibilidad y seguridad. Debido a que es un proceso muy importante, han surgido multitud de herramientas, entre las cuales destacaré:

## SonarCloud

Es un servicio gratuito en la nube que permite analizar y visualizar mediante una página web algunas de las características más importantes del análisis del software.

## Pylint

Pylint [7] es una herramienta gratuita y de código abierto que se ejecuta de forma local e informa al desarrollador acerca de algunos posibles errores en el código que podrían pasar inadvertidos y podrían causar problemas graves en un futuro si no se corrigen.

## Mypy

Mypy [9] es una herramienta de código abierto que analiza los archivos fuente para avisar de posibles incoherencias de tipos que podrían causar errores graves más adelante.



---

# Aspectos relevantes del desarrollo del proyecto

---

En este apartado se recogen los aspectos más importantes del desarrollo del proyecto, como los problemas que han ido surgiendo y las diferentes decisiones que se han ido tomando.

## 5.1. Uso de contenedores Docker

En un primer momento se consideró la posibilidad de usar contenedores Docker tanto para la distribución de Mycroft como para la interfaz gráfica para facilitar el proceso de instalación. Mycroft ofrece un contenedor ya creado con su aplicación, junto a las instrucciones de uso. Al instalarlo y ejecutarlo durante las primeras semanas del proyecto, todo funcionaba correctamente. Sin embargo, al probar a realizar la instalación en otra máquina, el contenedor no captaba sonido del micrófono y en ocasiones fallaba al reproducir las respuestas de Mycroft. Al reinstalar el contenedor de Mycroft en la máquina en la que anteriormente funcionaba correctamente, también dejó de captar el sonido y no vi manera de solucionarlo. Por esto, se optó por usar la aplicación nativa de Mycroft para Linux. Con respecto a la interfaz gráfica, al ser más complicado lanzar de manera automatizada Mycroft, ubicado en el equipo host, desde un contenedor en el que se ubicase la interfaz, se desestimó completamente el uso de Docker.

## 5.2. Creación del instalador

Debido a la compleja instalación que había en la versión anterior del proyecto, y al no poder usar contenedores ya listos, se opta por crear un instalador que automatice la obtención e instalación de las dependencias, la copia de los archivos a las ubicaciones adecuadas y la creación de accesos directos a la aplicación. Para realizarlo se opta por crear un *script* de línea de comandos para la primera versión, eligiendo Bash debido a que se usa habitualmente para esta finalidad y a que está disponible en la gran mayoría de sistemas. Debido a la diferente arquitectura y popularidad de las diferentes distribuciones Linux, se opta por Ubuntu como la principal distribución a soportar, aunque el funcionamiento será también correcto en otras que derivan de ella.

## 5.3. Mejoras de la interfaz

En la versión anterior del proyecto se creó una interfaz gráfica, pero los archivos que permitían editarla mediante QtDesigner no se subieron al repositorio. Al ser conveniente realizar algunas mejoras de estabilidad y de diseño, se opta por crear una segunda versión de la interfaz desde cero, pero manteniendo un diseño lo más simple posible. En la nueva interfaz se consigue que sea más estable que la anterior, evitando algunos de los cuelgues que antes se producían en algunas partes de la ejecución. También se crea un nuevo apartado de la interfaz que te guía en el proceso del emparejamiento con los servicios de Mycroft, paso necesario para poder usar el reconocimiento de voz. Finalmente, se opta por hacer un rediseño de la interfaz de chat, pasando de usar en algunos apartados los *widgets* nativos que ofrece Qt a usar HTML embebido que ofrece mayor flexibilidad a la hora de crear los bocadillos de conversación.

## 5.4. Localización

Anteriormente se podía seleccionar entre inglés o español como idioma para usar la aplicación, que estaban incluidos en el código fuente de la aplicación. Debido a que esto es una mala práctica, se opta por usar una biblioteca de localización que cargue cada uno de los textos a mostrar desde archivos externos al código, lo que también hace más fácil añadir nuevos idiomas posteriormente sin tener que modificar nada de código. Se elige *gettext* como biblioteca a usar debido a la existencia de software como Poedit que hace sencilla la generación de nuevos archivos de idioma.



## 5.5. Uso de contexto

También se busca mejorar la interacción con el asistente, y se considera empezar a trabajar con el contexto para que el asistente recuerde algunos elementos a lo largo de la conversación. En las anteriores skills que se habían realizado para el proyecto, se había usado Padatious como intent parser pero al revisar la documentación, todavía no es posible usar el contexto. Al ver este problema, se opta por realizar una nueva skill usando Adapt. Si bien es cierto que ambos *intent parsers* permiten acceder a un gran número de funciones comunes proporcionadas por Mycroft, la manera de crear *skills* es completamente diferente. Destacar que es más complicado crearlas usando Adapt debido a que la documentación es considerablemente más escasa y que es imprescindible usar expresiones regulares para definir cada uno de los *intents*, donde además hay que indicar también los posibles signos de puntuación.

## 5.6. Instalación en Windows

Previamente, uno de los principales puntos débiles de este proyecto era la necesidad de tener un sistema operativo Linux para poder usar el asistente. Sin embargo, en las últimas versiones de Windows 10 se ha incorporado una funcionalidad (Windows Subsystem for Linux, o WSL) con la que se puede ejecutar una versión ligera de Linux dentro de ese sistema operativo, pudiendo ejecutar nuestra aplicación sin necesidad de instalar una máquina virtual completa. Por el momento, es necesario instalar en Windows un servidor X11 para mostrar interfaces gráficas y Pulseaudio para grabar y reproducir sonido, pero Microsoft está trabajando para que en versiones futuras del sistema, esto ya venga incluido.



---

## Trabajos relacionados

---

Si bien es cierto que este campo estaba prácticamente inexplorado hace unos pocos años, cada poco tiempo aparece algún proyecto que pone a prueba las capacidades de los asistentes en el campo de la educación. Algunos de los más interesantes son los siguientes:

En primer lugar tenemos los predecesores del proyecto actual, realizados por la Universidad de Burgos. Podemos encontrar tanto [la versión anterior de este mismo proyecto](#) como [una aplicación para Android](#) que permite hacer preguntas sobre la información de la página web de la universidad.

En segundo lugar podemos encontrar algunos *chatbots* que permiten interactuar con Moodle de manera bastante similar al UBUVoiceAssistant. Entre los más destacables se encuentra [una integración con Telegram](#), o [el plugin para Microsoft Teams](#), donde además del bot, encontramos la posibilidad de ver los cursos de manera visual en el cliente de mensajería. También podemos encontrar una [skill para Alexa](#), que tiene un propósito y una funcionalidad parecida a nuestro proyecto.

También podemos encontrar una serie de asistentes que permiten interactuar con otras plataformas universitarias. Por ejemplo, tenemos [Pounce](#) para interactuar con la *Georgia State University*, o [Genie](#) en la *Deakin University*. Es necesario mencionar el caso de *Georgia Tech* [5], en el que, durante uno de sus cursos online se usaron dos *chatbots* que sustituyeron a profesores de apoyo para responder algunas de las preguntas más habituales.

Finalmente, cabe destacar el proyecto de la *Sant Louis University* [15], donde instalaron un total de 2300 dispositivos *Amazon Echo* con los que la comunidad universitaria puede hacer preguntas sobre la universidad a Alexa, sin necesidad de usar un dispositivo propio.

Nombre del proyecto	Lenguaje de programación	Tipo de información	Dispositivos compatibles	Complejidad de instalación	Complejidad de uso
UBUVoiceAssistant 2	Python	Dinámica (Moodle)	Windows y Linux	Baja	Moderada
UBUVoiceAssistant 1	Python	Dinámica (Moodle)	Linux	Alta	Moderada
UBUAssistant	Java	Estática	Android	Baja	Moderada
Bot de Telegram	Java	Dinámica (Moodle)	PC y móvil	Alta	Alta
MS Teams	PHP	Dinámica (Moodle)	PC y móvil	Alta	Baja
Skill de Alexa	Java	Dinámica (Moodle)	Móvil	Moderada	Moderada
Georgia State University	Desconocido	Dinámica	Móvil	Baja	Baja
Deakin University	Desconocido	Estática	Móvil	Baja	Moderada
Georgia Tech	Desconocido	Estática	Aula virtual	Ninguna	Baja
Sant Louis University	Java	Estática	Amazon Echo/Dot	Ninguna	Moderada

Tabla 6.1: Comparación de características

Se puede ver una comparación de las características en la tabla [6.1](#)

---

# Conclusiones y Líneas de trabajo futuras

---

## 7.1. Conclusiones

La principal conclusión que he sacado de este proyecto es la gran dificultad que tiene trabajar con un código con una documentación pobre. Incluir comentarios a lo largo del código y, especialmente, dejar documentado lo que hacen todos los métodos que no son triviales es fundamental para que otra persona pueda retomar el trabajo más adelante. Esto también es imprescindible en el caso de usar APIs de terceros (como las de Mycroft o Moodle), cosa que no siempre se cumple.

Relacionado con lo anterior, otra conclusión que se ha sacado de este proyecto es que en muchas ocasiones es considerablemente más lento y difícil de lo que parece en un primer momento conseguir cumplir las líneas de trabajo futuras de la versión anterior. Especialmente cuando en muchos casos tienes que probar varias alternativas para hacer una tarea hasta acertar con una que funcione exactamente como esperas.

También es muy importante dejar en algún sitio accesible todos los archivos que se han usado en el proyecto, ya que puede que en el futuro haya que reescribir gran parte del código para conseguir algo similar pero con algunas modificaciones, como ha ocurrido en el caso de la interfaz.

También me ha quedado claro que depender de algunas características de un programa de terceros que no están pensadas para que interactúes con ellas puede dar muchos problemas. Por ejemplo, en el caso de la obtención del código de emparejamiento a través de los registros, que puede romperse cuando cambien el formato.

A lo largo del trabajo se han aplicado algunos de los conocimientos de la carrera, como es la interacción hombre-máquina, para intentar conseguir unos diálogos bastante naturales con el asistente o algunas buenas prácticas, como la creación de la documentación, seguir un estilo uniforme a lo largo de todo el código o el uso de tipado.

Entre los puntos positivos de haber realizado el proyecto se encuentra el haber trabajado con varias tecnologías diferentes, integradas en un mismo programa, y la satisfacción al final de que todo va cobrando forma hasta convertirse en un proyecto que puede ser muy útil. Sin embargo, entre los puntos negativos se encuentra la frustración al ver la falta de documentación y tener que haber rehecho una parte del proyecto, lo que ha impedido poder trabajar en otras ideas interesantes.

## **7.2. Líneas de trabajo futuras**

En este apartado se incluyen algunas de las ideas que surgieron en la anterior versión del proyecto que no se han podido cumplir, junto a otras nuevas que han ido surgiendo a lo largo del desarrollo.

Por el momento sólo es posible usar la aplicación en Linux o en Windows 10 mediante el WSL, pero no se puede usar en MacOS ni en móviles, donde quizá es más útil. Por el momento es necesario esperar a que avance el desarrollo de Mycroft para que sea viable usarlo en esas plataformas, ya que todavía no se ofrecen unas versiones estables. También se podría optar por intentar conseguir una funcionalidad similar mediante otras tecnologías que sí sean multiplataforma, o crear una aplicación cliente ligero que se conecte a un servidor de la universidad donde se procesen todas las peticiones de los diferentes usuarios.

También podría ser interesante la posibilidad de trabajar sin conexión. Sería necesario descargar los datos obtenidos de Moodle a una serie de archivos o una base de datos local que se podrían consultar más adelante cuando no haya una conexión disponible.

Finalmente, se podría plantear en un futuro ayudar a eliminar las limitaciones que trae Mycroft, como el mal funcionamiento que tiene la entrada por teclado cuando se pregunta al usuario para obtener datos adicionales, la imposibilidad de indicarle a Mycroft que pare a través de la voz, o la obligación de que el instalador que traen sea interactivo.

---

## Bibliografía

---

- [1] *Babel — Babel 2.7.0 documentation*. [Online; accessed 16. Jun. 2021]. Ene. de 2021. URL: <http://babel.pocoo.org/en/latest>.
- [2] *Wake Word — English Cambridge Dictionary*. en-US. URL: <https://dictionary.cambridge.org/us/dictionary/english/wake-word> (visitado 16-06-2021).
- [3] *gettext — Multilingual internationalization services — Python 3.9.5 documentation*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://docs.python.org/3/library/gettext.html>.
- [4] *Glossary of terms*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://mycroft-ai.gitbook.io/docs/about-mycroft-ai/glossary>.
- [5] *Jill Watson, Round Three*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://www.news.gatech.edu/2017/01/09/jill-watson-round-three>.
- [6] Varghese P. Kuruvilla. “Fuzzy Matching or Fuzzy Logic Algorithms Explained”. En: *AI & Machine Learning Blog* (mayo de 2021). URL: <https://nanonets.com/blog/fuzzy-matching-fuzzy-logic>.
- [7] Logilab. *Pylint - code analysis for Python* | [www.pylint.org](http://www.pylint.org). [Online; accessed 16. Jun. 2021]. Ago. de 2020. URL: <https://pylint.org>.
- [8] Moodle - Open-source learning platform | [Moodle.org](http://Moodle.org). [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://moodle.org>.
- [9] *mypy - Optional Static Typing for Python*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <http://mypy-lang.org>.
- [10] *Poedit Translation Editor — Poedit*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://poedit.net>.

- [11] *Requests: HTTP for Humans™ — Requests 2.25.1 documentation*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://docs.python-requests.org/en/master>.
- [12] *Riverbank Computing | Introduction*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://www.riverbankcomputing.com/software/pyqt>.
- [13] Derick Schweppe. “Mycroft – The Open Source Privacy-Focused Voice Assistant - Mycroft”. En: *Mycroft* (mayo de 2021). URL: <https://mycroft.ai>.
- [14] seatgeek. *fuzzywuzzy*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://github.com/seatgeek/fuzzywuzzy>.
- [15] *SLU Alexa Project*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://www.slu.edu/news/2018/august/slu-alexaproject.php>.
- [16] David Smehlik. “Add new skills - Mycroft”. En: *Mycroft* (mayo de 2021). URL: <https://mycroft.ai/get-started/#4-add-new-skills>.
- [17] *Technology Overview*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://mycroft-ai.gitbook.io/docs/mycroft-technologies/overview>.
- [18] *Web services - MoodleDocs*. [Online; accessed 16. Jun. 2021]. Mayo de 2021. URL: [https://docs.moodle.org/311/en/Web\\_services](https://docs.moodle.org/311/en/Web_services).
- [19] *What is REST*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://restfulapi.net>.
- [20] *What is Scrum?* [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://www.scrum.org/resources/what-is-scrum>.
- [21] Wikipedia contributors. *Bash (Unix shell) — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-June-2021]. 2021. URL: [https://en.wikipedia.org/w/index.php?title=Bash\\_\(Unix\\_shell\)&oldid=1027745992](https://en.wikipedia.org/w/index.php?title=Bash_(Unix_shell)&oldid=1027745992).
- [22] Wikipedia contributors. *Git — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-June-2021]. 2021. URL: <https://en.wikipedia.org/w/index.php?title=Git&oldid=1028031915>.
- [23] Wikipedia contributors. *GitHub — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-June-2021]. 2021. URL: <https://en.wikipedia.org/w/index.php?title=GitHub&oldid=1028816468>.



- [24] Wikipedia contributors. *LaTeX* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 16-June-2021]. 2021. URL: <https://en.wikipedia.org/w/index.php?title=LaTeX&oldid=1023081178>.
- [25] Wikipedia contributors. *Python (programming language)* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 16-June-2021]. 2021. URL: [https://en.wikipedia.org/w/index.php?title=Python\\_\(programming\\_language\)&oldid=1028384838](https://en.wikipedia.org/w/index.php?title=Python_(programming_language)&oldid=1028384838).
- [26] Wikipedia contributors. *Qt (software)* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 16-June-2021]. 2021. URL: [https://en.wikipedia.org/w/index.php?title=Qt\\_\(software\)&oldid=1028221487](https://en.wikipedia.org/w/index.php?title=Qt_(software)&oldid=1028221487).
- [27] Wikipedia contributors. *Qt Creator* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 16-June-2021]. 2021. URL: [https://en.wikipedia.org/w/index.php?title=Qt\\_Creator&oldid=1021600453](https://en.wikipedia.org/w/index.php?title=Qt_Creator&oldid=1021600453).
- [28] Wikipedia contributors. *Speech recognition* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 29-June-2021]. 2021. URL: [https://en.wikipedia.org/w/index.php?title=Speech\\_recognition&oldid=1030555901](https://en.wikipedia.org/w/index.php?title=Speech_recognition&oldid=1030555901).
- [29] Wikipedia contributors. *Speech synthesis* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 29-June-2021]. 2021. URL: [https://en.wikipedia.org/w/index.php?title=Speech\\_synthesis&oldid=1030992952](https://en.wikipedia.org/w/index.php?title=Speech_synthesis&oldid=1030992952).
- [30] Wikipedia contributors. *Virtual assistant* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 16-June-2021]. 2021. URL: [https://en.wikipedia.org/w/index.php?title=Virtual\\_assistant&oldid=1026987190](https://en.wikipedia.org/w/index.php?title=Virtual_assistant&oldid=1026987190).
- [31] Wikipedia contributors. *Visual Studio Code* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 16-June-2021]. 2021. URL: [https://en.wikipedia.org/w/index.php?title=Visual\\_Studio\\_Code&oldid=1027143164](https://en.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=1027143164).
- [32] *ZenHub - Agile Project Management for GitHub*. [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: <https://www.zenhub.com>.