UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

UBUVoiceAssistant 2.0 Documentación Técnica

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

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se detallan la planificación temporal que se ha seguido para realizar el proyecto, junto con la metodología utilizada, y la viabilidad del proyecto.

A.2. Planificación temporal

Para realizar el proyecto se ha usado una metodología ágil llamada SCRUM [8], pero reduciendo el número de personas a 1, y realizando reuniones semanales, para adaptarlo a las necesidades y a las características del trabajo.

Sprint 1

El primer sprint duró aproximadamente dos semanas (del 12 al 24 de febrero).

La reunión previa al sprint se dedicó para establecer los objetivos del proyecto. La mayor parte del sprint se dedicó a revisar el código inicial del proyecto, revisar errores, identificar posibilidades de mejora y oportunidades para incluir funcionalidades completamente nuevas. Principalmente, consultar cómo funciona el *Docker* proporcionado por *Docker* y comprobar cómo se puede integrar e iniciar la documentación.

Sprint 2

El segundo sprint dura dos semanas (del 25 de febrero al 10 de marzo).

En este sprint se creó la primera versión del instalador, se empezó a reescribir el código de la interfaz, se dieron los primeros pasos para incluir un sistema de localización y se hicieron algunas correcciones menores de errores.

Sprint 3

El tercer sprint dura dos semanas (del 11 al 24 de marzo).

En este sprint se añade una buena cantidad de documentación, se hacen una serie de mejoras de calidad al código existente de la versión anterior del proyecto y se sigue trabajando en la nueva interfaz, junto con arreglos de errores para intentar adaptarse al sistema de contenedores *Docker*.

Sprint 4

El cuarto sprint dura dos semanas (del 25 de marzo al 7 de abril).

A lo largo de este sprint se añadió el nuevo recuadro HTML para mostrar de forma más visual la interacción entre Mycroft y el usuario, se modificó el instalador para usar el proyecto de *Mycroft* para Linux en lugar de *Docker* y se hicieron varios arreglos al código para conseguir tener una primera versión con la nueva interfaz.

Sprint 5

El quinto sprint dura dos semanas (del 8 al 21 de abril).

En este sprint se arreglan una serie de errores que han aparecido al programar la nueva interfaz, se hacen algunas mejoras para aumentar la vivacidad de la aplicación y se añaden los primeros apartados de la memoria.

Sprint 6

El sexto sprint dura dos semanas (del 22 de abril al 5 de mayo).

A lo largo de este sprint se añaden múltiples apartados de la memoria del proyecto. Al mismo tiempo, se empieza a investigar el funcionamiento de Adapt como intent parser.

Sprint 7

El séptimo sprint dura una semana (del 5 al 12 de mayo).

En este sprint se realizan algunas pruebas con Adapt para poder empezar a crear una skill nueva que lo aproveche. También se arreglan algunos errores en los anteriores apartados de la memoria.

Sprint 8

El octavo sprint dura una semana (del 13 al 19 de mayo).

En este sprint se investigan algunos métodos disponibles en los *webservices* que pueden ser útiles para añadir nuevas funcionalidades al proyecto, configurando una plataforma de Moodle en un servidor propio. También se añaden los apartados restantes de la memoria.

Sprint 9

El noveno sprint dura una semana (del 20 al 26 de mayo).

Debido a problemas con el ordenador, se migra el proyecto de LaTeX a Overleaf. Desde esa plataforma se añaden el manual de programador y usuario a los anexos.

Sprint 10

El décimo sprint dura dos semanas (del 27 de mayo al 9 de junio).

Se añaden la mayoría de apartados restantes a los anexos y se adaptan el modelo y los *webservices* para dar soporte a las nuevas funcionalidades

Sprint 11

El undécimo sprint dura una semana (del 10 al 16 de junio).

Se arreglan algunos errores con el proceso de emparejamiento debido a cambios de *Mycroft* y se termina la *skill* que permite interactuar con los mensajes privados de Moodle. En el proceso, se han movido las *skills* a submódulos para poder instalar correctamente las dependencias.

Sprint 12

El duodécimo sprint dura una semana (del 17 al 23 de junio).

Se realizan varias correcciones de errores en varios puntos del programa y se realizan varias mejoras en la interfaz.

Sprint 13

El penúltimo sprint dura una semana (del 24 al 30 de junio).

Se realizan algunas correcciones de errores para conseguir soportar un mayor número de dispositivos que no funcionaban correctamente, se incluye un sistema de ayuda y se termina la documentación.

Sprint 14

El sprint final dura una semana (del 1 al 7 de julio).

A lo largo de este sprint se realizan las últimas correcciones de errores, se añade la documentación al código y se revisa su calidad, se crea la versión final y se prepara para su entrega.

A.3. Estudio de viabilidad

Viabilidad económica

Costes de personal

Este proyecto ha sido desarrollado por una persona y ha dedicado aproximadamente 300 horas de tiempo de trabajo. Según este estudio, el sueldo medio para un programador junior en España es de 19.787€ anuales. Si descontamos el IRPF (2.290,3€) y el impuesto que se paga a la Seguridad Social (1.256,5€), nos quedamos con un sueldo neto de 16.240€ anuales.

Si dividimos las 300 horas de trabajo en jornadas de 8 horas, nos quedan un total de 38 días que se han dedicado para este proyecto. Si tenemos en cuenta que en un mes se trabajan aproximadamente 21 días, y teniendo en cuenta las vacaciones correspondientes a ese tiempo de trabajo, sería necesario contratar al programador durante dos meses completos.

Durante estos dos meses, el programador cobraría aproximadamente 2.700€ pero, sumando los impuestos, el gasto que deberíamos asumir es de casi 3.300€.

Costes *software*

Todos los programas y servicios que se han usado para el desarrollo del proyecto disponen de una versión gratuita, por lo que no tenemos ningún tipo de coste *software*

Costes *hardware*

Para desarrollar el proyecto se ha usado un ordenador portátil de gama media. El equipo cuesta 550€ y se amortiza a 5 años. Como sólo lo hemos usado durante 2 meses, el coste amortizado del equipo es de algo menos de 20€.

Al ser código abierto, lo que se asume habitualmente es que debe ser gratuito siempre. Sin embargo, esto no es así para todos los proyectos. Por ejemplo, en el caso de Elementary OS, un sistema operativo basado en Linux, se da la opción de hacer una donación al descargar el sistema o algunas de sus aplicaciones. También existe Aseprite, cuya descarga es de pago, aunque también se puede obtener una versión de manera completamente legal si compilamos el código fuente.

En nuestro proyecto, conseguiríamos cubrir gastos si 665 personas nos donan los 5€ que se suelen pedir en muchos proyectos open source que están disponibles por Internet.

Viabilidad legal

Software

Este proyecto se ha licenciado pajo la licencia GPL-3.0 [7]. Esta licencia surge en 2007, como actualización de la GNU General Public License y sigue siendo de las más usadas en proyectos open source. Permite a todas las personas la posibilidad de usar, estudiar y redistribuir el software de manera libre, y además protege que ese código y modificaciones que se hagan del mismo puedan acabar siendo de código cerrado, ya que se debe seguir usando la misma licencia o una versión superior. Esto también afecta a programas que incluyen bibliotecas con licencia GPL3, como el nuestro. Esto se debe a que hemos usado las siguientes bibliotecas:

- Mycroft, con licencia Apache-2.0 [2]
- Python, que usa tanto la Python Software Fundation License y la Zero Clause BSD License [5]

- Las siguientes bibliotecas de Python:
  - PyQt5, distribuido mediante GPL-3.0 en su versión gratuita
  - Requests, con licencia Apache-2.0
  - Babel usa la Babel License [6]
  - Fuzzywuzzy se distribuye bajo la GPL-2.0 [4]

Documentación

La documentación de este proyecto se distribuye bajo la licencia Creative Commons Attribution ShareAlike 4.0 [3]. Es una licencia que se usa de manera habitual, y que otorga el permiso para redistribuir el contenido y modificarlo de cualquier manera, siempre que se mencione al autor original y se mantenga la misma licencia para las modificaciones que se realicen.

Imágenes

Los iconos que ya estaban incluidos en la anterior versión de la aplicación proceden de fontawesome y se distribuyen bajo la licencia Creative Commons 4.0

El icono de la aplicación ha sido creado por Smashicons y se puede descargar en www.flaticon.com. En su licencia especifican la obligatoriedad de atribuir la creación del contenido, aunque este haya sido modificado posteriormente.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este anexo se explican los objetivos del proyecto y los requisitos que se han definido para llevarlo a cabo.

B.2. Objetivos generales

El principal objetivo del proyecto es mejorar la versión anterior del asistente del UBUVoiceAssistant. Para cumplirlo, se proponen algunas mejoras, como una importante simplificación en el proceso de instalación y configuración inicial o en la interacción con el asistente. También se propone conseguir una mayor robustez en la aplicación, obtener más información de Moodle e incluso poder enviar datos a la plataforma.

B.3. Catalogo de requisitos

Los nuevos requisitos que se han añadido a la aplicación son los siguientes:

Requisitos funcionales

- RF-1 Obtener datos de Moodle: La aplicación debe poder obtener distintos datos de Moodle.
    - RF-1.1 Obtener los mensajes privados: La aplicación debe poder leer los últimos mensajes que ha recibido el usuario de la misma.

    - RF-1.2 Buscar en los foros: La aplicación debe poder buscar en los foros los hilos relacionados con una frase que diga el usuario.

- RF-2 Enviar datos a Moodle: La aplicación debe poder enviar algunos datos a nuestra plataforma de Moodle.

    - RF-2.1 Enviar mensajes privados: La aplicación debe poder enviar mensajes privados a otros usuarios con los que se tiene una conversación abierta o que comparten un curso con el usuario de la aplicación.

- RF-3 Mostrar un sistema de ayuda: La aplicación debe poder indicar al usuario cómo interactuar, sugiriéndole algunas frases que puede decir.

- RF-4 Mejorar la instalación: Se deben mejorar varios aspectos para hacer más cómodo el proceso de puesta en marcha.

    - RF-4.1 Crear un instalador: Se debe crear un programa que prepare todas las dependencias necesarias y coloque todos los archivos en las ubicaciones necesarias para el correcto funcionamiento de la aplicación.
    - RF-4.2 Crear un asistente de emparejamiento: La aplicación debe guiar al usuario por el proceso de emparejamiento del cliente con el servidor de Mycroft.

Requisitos no funcionales

- RNF-1 Soporte para Windows: La aplicación debe poder ser ejecutada en Windows 10.

- RNF-2 Naturalidad: La interacción de la aplicación con el usuario debe ser lo más natural posible.

- RNF-3 Robustez: Se debe mejorar la robustez de la aplicación, haciéndola más tolerante a fallos.

- RNF-4 Internacionalización: La aplicación tiene que permitir añadir idiomas nuevos fácilmente.

- RNF-5 Calidad de código: La calidad del código debe ser buena, y tener una documentación completa.

B.4. Especificación de requisitos

En esta sección se detallan los casos de uso que se han añadido a los ya existentes en la versión anterior [1].

CU-7: Consultar foros		
Descripción	El usuario consulta los foros de una asignatura concreta	
Precondiciones	El usuario se ha autenticado y puede acceder a la asignatura	
Secuencia normal	Paso	Acción
	1	Se descarga la información de los cursos
	2	Se elige el curso más parecido al que ha dicho el usuario
	3	Se descargan los foros de ese curso
	4	Se lee el primer post disponible
	5	Se pregunta al usuario si quiere seguir leyendo. Si responde sí, se vuelve al paso 4. Si no, termina.
Excepciones	Paso	Acción
	2	Si no hay ningún curso parecido, se emite un error.

Tabla B.1: Caso de uso 7: Consultar foros

CU-8: Consultar eventos		
Descripción	El usuario consulta los eventos futuros	
Precondiciones	El usuario se ha autenticado y tiene al menos un evento futuro	
Secuencia normal	Paso	Acción
	1	Se descarga la información del calendario
	2	Se leen los próximos eventos del calendario

Tabla B.2: Caso de uso 8: Consultar eventos

CU-9: Consultar cambios		
Descripción	El usuario consulta las modificaciones recientes de una asignatura concreta	
Precondiciones	El usuario se ha autenticado y puede acceder a la asignatura	
Secuencia normal	Paso	Acción
	1	Se descarga la información de los cursos
	2	Se elige el curso más parecido al que ha dicho el usuario
	3	Se descargan los cambios de ese curso
	4	Se lee la información obtenida
Excepciones	Paso	Acción
	2	Si no hay ningún curso parecido, se emite un error.

Tabla B.3: Caso de uso 9: Consultar cambios

CU-10: Consultar calificaciones		
Descripción	El usuario consulta las notas de una asignatura concreta	
Precondiciones	El usuario se ha autenticado y puede acceder a la asignatura	
Secuencia normal	Paso	Acción
	1	Se descarga la información de los cursos
	2	Se elige el curso más parecido al que ha dicho el usuario
	3	Se descargan as notas de ese curso
	4	Se leen las notas obtenidas en voz alta
Excepciones	Paso	Acción
	2	Si no hay ningún curso parecido, se emite un error.

Tabla B.4: Caso de uso 10: Consultar calificaciones

CU-11: Consultar mensajes privados		
Descripción	El usuario consulta los últimos mensajes privados	
Precondiciones	El usuario se ha autenticado	
Secuencia normal	Paso	Acción
	1	Se descarga la lista de conversaciones del usuario
	2	Se obtienen más mensajes en cada una de las conversaciones
	3	Se filtran los mensajes para mostrar solo los recibidos
	4	Se ordenan para obtener los más nuevos primero
	5	Se leen los cinco últimos mensajes recibidos al usuario

Tabla B.5: Caso de uso 11: Consultar mensajes privados

CU-12: Enviar mensajes privados		
Descripción	El usuario envía un mensaje a otra persona	
Precondiciones	El usuario se ha autenticado y tiene una conversación abierta con la otra persona, o comparten un curso y puede ver su perfil	
Secuencia normal	Paso	Acción
	1	Se descarga la lista de conversaciones
	2	Se busca un usuario con nombre parecido al solicitado. Si no hay ninguno, se salta al paso 5
	3	Se pregunta al usuario si ese es el usuario deseado. En caso de serlo, se salta al paso 6
	4	De no serlo, se pregunta un curso que tengan en común.
	5	Se repite desde el paso 2 buscando en los participantes de ese curso
	6	Se pregunta el mensaje deseado al usuario
	7	Se lee el mensaje de nuevo y se espera confirmación del usuario
Postcondiciones	El nuevo mensaje aparece reflejado en Moodle	
Excepciones	Paso	Acción
	4	Si no hay ningún curso parecido, se emite un error.

Tabla B.6: Caso de uso 12: Enviar mensajes privados

Apéndice C

Especificación de diseño

C.1. Introducción

En este apartado se detalla el diseño del proyecto, indicando claramente cuáles han sido los cambios más importantes con respecto a la versión anterior.

C.2. Diseño de datos

El modelo de datos que se usa dentro de la aplicación está fuertemente ligado con la estructura de datos que se usa dentro de Moodle. Sólo se ha creado el modelado necesario para almacenar los datos que se usan en las diferentes skills. Todos los archivos fuente que componen el modelo de datos se encuentran en la carpeta src/UBUVoiceAssistant/model. Las clases que lo forman son las siguientes:

- **Course** guarda los datos relacionados con los cursos.
- **Forum** almacena los foros de un curso.
- **Discussion** representa cada uno de los hilos de un foro.
- **GradeItem** guarda las notas.
- **Event** representa los eventos del calendario.
- **User** almacena todos los datos relacionados con el usuario.

- Se ha añadido la clase **Conversation** que almacena las conversaciones por mensaje privado.

- La nueva clase **Message** representa cada mensaje de una conversación.

Atributo	Tipo de dato	Descripción
conversation_id	int	Identificador de la conversación
isread	bool	Indica si hay mensajes sin leer
unreadcount	int	Número de mensajes pendientes
name	str	Nombre de la conversación
subname	str	Subtítulo de la conversación
members	dict	Diccionario de miembros Son objetos de tipo User La clave es el id del usuario
messages	dict	Diccionario de mensajes Son objetos de tipo Message La clave es el id del mensaje

Tabla C.1: Diccionario de datos de Conversation

Atributo	Tipo de dato	Descripción
message_id	int	Identificador del mensaje
useridfrom	int	Identificador del usuario que ha mandado este mensaje
text	str	Contenido del mensaje Contiene algunas etiquetas HTML
timecreated	int	Momento de envío del mensaje Timestamp en formato Unix

Tabla C.2: Diccionario de datos de Message

Atributo	Tipo de dato	Descripción
user_id	int	Identificador del usuario
courses	dict	Diccionario de los cursos del usuario Son objetos de tipo Course La clave es el id del curso Está vacío si no representa al usuario de la aplicación
fullname	str	Nombre completo del usuario

Tabla C.3: Diccionario de datos de User

Atributo	Tipo de dato	Descripción
course_id	str	Identificador del curso
name	str	Nombre del curso
grades	list	Lista que contiene las calificaciones del usuario de ese curso
events	list	Lista que contiene los eventos del curso
forums	list	Lista que contiene los foros de ese curso
participants	dict	Diccionario de los participantes Son objetos de tipo User La clave es el id del usuario

Tabla C.4: Diccionario de datos de Course

Las clases “Forum”, “Discussion”, “GradeItem” y “Event” no han sufrido cambios con respecto a la versión anterior [1].

C.3. Diseño procedimental

Las peticiones que se habían creado en la anterior versión del proyecto [1] no han sufrido cambios y, por tanto, no se van a dar muchos detalles.

En primer lugar, al introducir los credenciales en la aplicación, se realiza una primera petición al *webservice* de Moodle, para validar el usuario y la contraseña que hemos introducido, y obtener un token que nos identifique durante el resto de la aplicación. También se realiza una petición

para recuperar algunos valores que son importantes más adelante, como la información de la plataforma, el identificador del usuario y también los cursos en los que se está matriculado.

Para las interacciones del usuario con el asistente, en caso de realizarlas por voz, el componente *Voice* graba unos segundos de sonido y lo envía al servidor de Speech-To-Text que tengamos configurado. Al recibir la respuesta del servidor con la frase del usuario (*utterance*) ya transformada a texto, lo envía al componente *Skills* a través del MessageBus, quien invoca a la *skill* correspondiente y devuelve la respuesta, también a través del bus. Aquí, se realiza una petición al servidor de Text-To-Speech, quien devuelve la frase en forma de sonido. En el caso de que los servicios TTS y STT sean locales, todo ese proceso se realiza en la propia máquina. También, si el usuario introduce la entrada por texto, no se realiza el primer paso, sino que el mensaje se envía directamente de la interfaz gráfica al bus.

Entre los procedimientos nuevos, tenemos la posibilidad de obtener los últimos mensajes privados que se han recibido a través de la plataforma. En este, una vez se activa la *skill*, lo primero que se hace es enviar una petición al *webservice* para obtener la lista de conversaciones del usuario. Después, debido a que a través de esa petición únicamente hemos obtenido el último mensaje de cada conversación, tenemos que realizar una petición por cada conversación, para obtener más mensajes. Una vez se ha recuperado la información de los mensajes, se van enviando de uno en uno a través del bus de Mycroft para que aparezcan en la interfaz y se lean en voz alta.

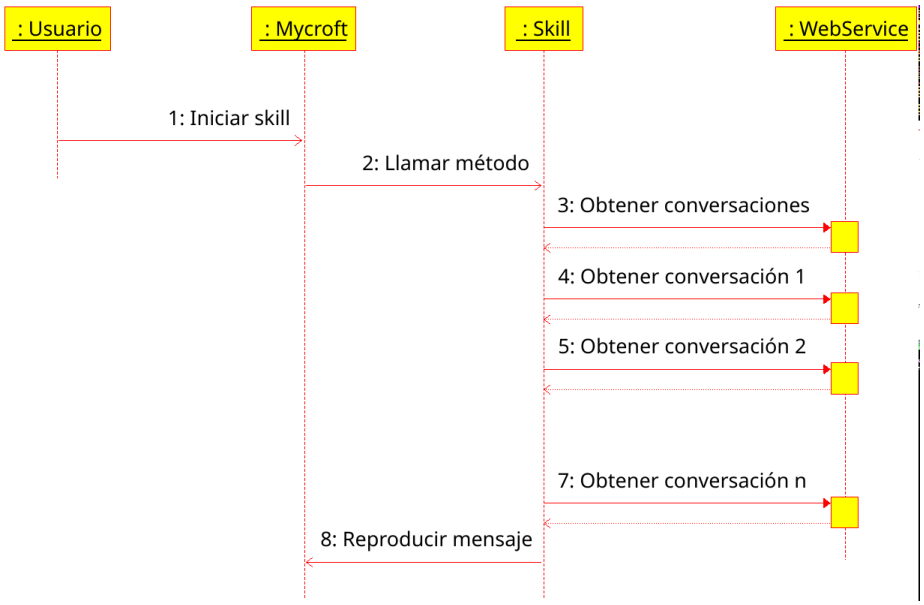

Figura C.1: Diagrama de secuencia al consultar mensajes

También tenemos la posibilidad de enviar mensajes a otros usuarios de Moodle. Para esta funcionalidad, en primer lugar se descargará la lista de conversaciones y se preguntará al usuario cuál de las opciones quiere elegir, en caso de que haya varias parecidas. En caso de que ninguna de las opciones sea la que desea el usuario, se le preguntará un curso, del que se descargarán los participantes y se volverá a preguntar al usuario si una de las opciones es la que deseaba. Una vez seleccionada la persona, se preguntará al usuario el texto del mensaje, y tras una confirmación para que el usuario compruebe si el texto se ha entendido correctamente, se enviará el mensaje al destinatario.

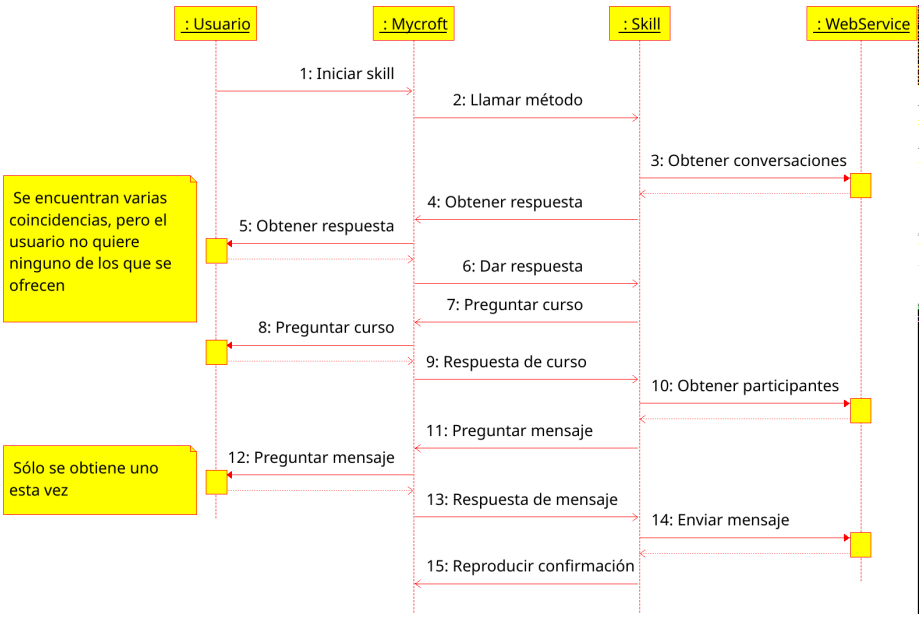

Figura C.2: Diagrama de secuencia al enviar mensajes

C.4. Diseño arquitectónico

El diseño arquitectónico de la aplicación no ha sufrido cambios con respecto a la anterior versión del programa [1].

La aplicación está dividida en dos partes, siendo una de ellas la interfaz gráfica y la otra, Mycroft. En varias partes de la aplicación se usa una arquitectura cliente-servidor, donde el cliente realiza peticiones HTTP al servidor, y este envía la información solicitada en la respuesta.

Al iniciar el programa, una vez se han introducido los credenciales en la interfaz, estos se transfieren a cada una de las *skills* instaladas en Mycroft mediante un socket, siendo la interfaz gráfica el servidor y las *skills* los clientes.

A partir de este momento, también se crea un MessageBus entre la interfaz gráfica y Mycroft. Este componente se basa en un WebSocket que permite intercambiar mensajes de manera simple entre todos los componentes que forman Mycroft y también con programas de terceros.

También, para hacer uso del Speech-To-Text o el Text-To-Speech, en caso de que estemos usando servicios que requieren conexión a internet, Mycroft realizará las peticiones correspondientes a los servicios necesarios.

Finalmente, en el caso de las skills que interaccionan con Moodle, estas se comunican mediante una API REST con los *web services* del servidor Moodle al que nos estemos conectando.

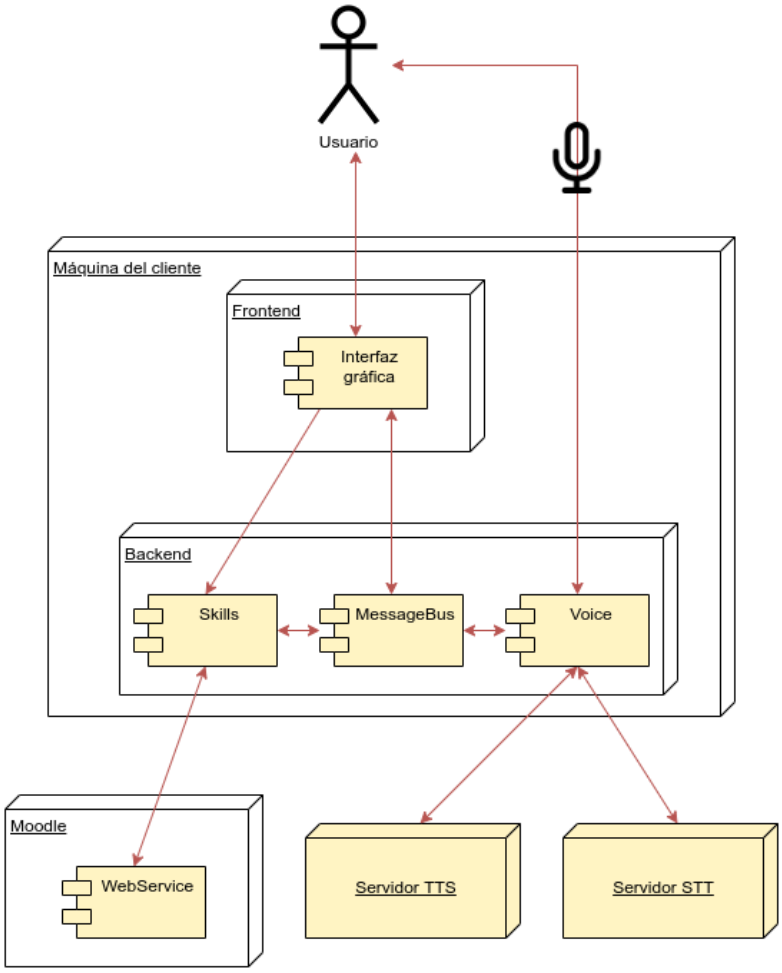

Figura C.3: Diagrama de despliegue

Apéndice D

Documentación técnica de programación

Nota importante a partir de este punto: Muchos lectores de PDF introducen saltos de línea que hacen que los comandos más largos dejen de funcionar. Cada comando debe introducirse por completo en una línea y luego pulsar Enter para ejecutarlo. Además, dependiendo del lector de PDF, puede que algunos comandos no contengan los caracteres correctos. Por ello, he dejado los comandos en Github Gists

D.1. Introducción

En este capítulo se van a mencionar todos los aspectos relevantes que es necesario conocer para poder seguir trabajando en una versión futura de la aplicación.

D.2. Estructura de directorios

Para organizar el proyecto, se ha hecho de la siguiente manera:

- /docs/: carpeta que contiene todos los documentos e imágenes para generar los archivos PDF con la memoria y anexos.
- /scripts/: varios scripts que se han usado para automatizar algunas acciones frecuentes, como la ejecución del programa desde la versión

en desarrollo, la generación de la plantilla para las traducciones, o para realizar las pruebas de calidad de código.

- /testing/: algunos archivos de prueba que he ido generando a lo largo del proyecto para probar algunas funcionalidades sin tener que lanzar el programa completo.
- /src/UBUVoiceAssistant/: la carpeta base del código fuente del proyecto. Tiene las siguientes carpetas en su interior:
  - GUI: contiene los archivos necesarios para la interfaz gráfica
  - GUI/old_files: archivos que hacían funcionar la anterior interfaz gráfica. No se usan, pero se han dejado como referencia.
  - GUI/forms: en esta carpeta se encuentran los formularios generados con QtDesigner.
  - GUI/forms/chat_window_html: los documentos html, javascript y CSS necesarios para mostrar los bocadillos en la ventana de chat con el asistente.
  - imgs: contiene las imágenes que se usan en la interfaz gráfica.
  - lang: los archivos para traducir la interfaz gráfica. Tiene la plantilla llamada "translation_template.pot" y las carpetas para cada uno de los idiomas. Siguen la estructura *languageCode_countryCode*/LC_MESSAGES/UBUVoiceAssistant
  - model: archivos que representan los datos de Moodle
  - prototipo: archivos para un prototipo de skill para Alexa que se usaron en la anterior versión del proyecto.
  - skills: contiene las *Skills* del proyecto.
  - util: archivos que ofrecen varias funcionalidades para el proyecto.
  - webservice: código encargado de realizar las peticiones web al servidor de Moodle

D.3. Manual del programador

Como entorno de programación se recomienda usar como sistema operativo la última versión estable de Ubuntu, aunque también es posible usar la última LTS o cualquiera de las distribuciones que derivan de ellas. Es necesario instalar las dependencias de python3 que hemos usado en el proyecto, Mycroft, QtDesigner, Poedit y un editor de código como Visual Studio Code con algunas extensiones recomendadas para facilitar el trabajo.

Python3

Python3 ya viene preinstalado en Ubuntu, pero para que la aplicación funcione correctamente necesitamos instalar las bibliotecas usadas en el proyecto. Esto se puede hacer escribiendo los siguientes comandos en una terminal:

- `sudo apt-get install python3-pip python3-pyqt5 python3-pyqt5.qtwebengine git gettext mypy pylint -y`
- `sudo pip3 install mycroft-messagebus-client babel bs4`

  En el caso de que se vayan a modificar o crear skills nuevas, es recomendable ejecutar los siguientes comandos para tener autocompletado dentro del IDE:
- `sudo apt-get install libfann-dev python3-dev swig -y`
- `sudo pip3 install adapt-parser padatious`

Mycroft

Para poder usar Mycroft es necesario tener una cuenta en su página web. También será necesario descargar el código de Mycroft desde su repositorio. Lo podemos hacer ejecutando los siguientes comandos, donde sustituimos your_user por tu usuario de Ubuntu:

- `sudo mkdir -p /usr/lib/mycroft-core`
- `sudo chown your_user /usr/lib/mycroft-core`
- `git clone https://github.com/MycroftAI/mycroft-core.git /usr/lib/mycroft-core`
- `/usr/lib/mycroft-core/dev_setup.sh -sm`

Tras introducir el último comando se abrirá una instalación interactiva de Mycroft. En todas las preguntas deberemos responder escribiendo la letra Y en la terminal.

Cuando termine, podemos emparejarlo abriendo la consola de Mycroft escribiendo los siguientes comandos:

- `cd /usr/lib/mycroft-core`

- `./start-mycroft.sh debug`

Tras esto, se mostrará en la terminal los registros de las operaciones que realiza Mycroft en la parte superior, mientras que en la parte inferior aparecen los registros de los mensajes intercambiados entre el usuario y Mycroft. Pasados unos segundos o unos minutos, dependiendo de la velocidad tanto del equipo como de la conexión a internet, se iniciará el proceso de emparejamiento. En caso de no realizarse de manera automática, podemos escribir "pair my device" para iniciarlo manualmente. Mycroft recitará y escribirá lo que tenemos que hacer en este punto. Tendremos que ir a la página web de Mycroft, y en la parte superior derecha, hacer click en **Add device**. Deberemos introducir el código de 6 caracteres en el campo que pone **Pairing Code**. En la parte inferior deberemos seleccionar **Google Voice** como motor de voz. Una vez hecho, Mycroft dirá que se ha emparejado correctamente y puede empezar a funcionar. Podemos cerrar la interfaz de Mycroft pulsando *ctrl + c* o escribiendo `:exit`.

QtDesigner

Para instalar QtDesigner deberemos escribir el siguiente comando en una terminal:

- `sudo apt-get install qttools5-dev-tools`

Una vez hecho esto, podremos acceder al programa desde el menú de aplicaciones del sistema.

Aquí podemos crear una ventana de la lista de opciones que se nos ofrece al abrir el programa. Una vez lo hagamos, tendremos tres columnas. En la parte central se nos mostrará el diseño de lo que estamos creando y podremos reorganizar los elementos arrastrándolos. En la parte izquierda tendremos una serie de *widgets* que podemos añadir a la ventana que hemos creado simplemente con arrastrarlos y soltarlos dentro de ella. Finalmente, en la parte derecha podremos configurar una serie de parámetros acerca del elemento que hayamos seleccionado (como el texto de una etiqueta o de un campo de texto)

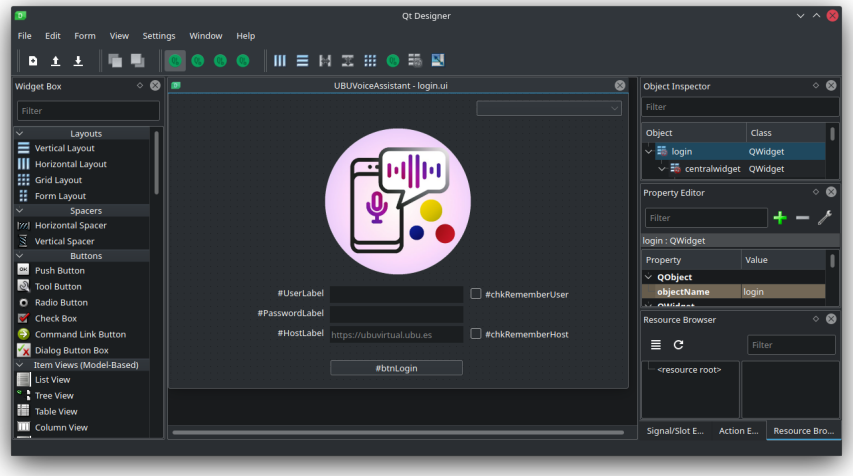

Figura D.1: Captura de pantalla de Qt Designer

Poedit

Para editar los archivos de traducción de la interfaz es recomendable usar un programa que las muestre de forma visual y ofrezca algunas herramientas, como Poedit.

Para instalarlo, es tan sencillo como hacer:

- `sudo apt install poedit`

En la pantalla principal de la aplicación se nos mostrará la opción de editar una traducción existente o empezar una nueva a partir de una plantilla. Elegimos la que nos corresponda

Visual Studio Code

Si bien es posible usar cualquier editor de texto, voy a describir el proceso para instalar Visual Studio Code, ya que es uno de los editores más completos que existen a día de hoy.

Para descargarlo tendremos que ir a su página web, descargamos el archivo .deb, y hacemos doble click sobre él para instalarlo. En caso de que no funcione, podemos abrir una terminal, navegar hasta la carpeta donde se encuentra el archivo usando el comando `cd` y realizar la instalación usando `sudo apt install ./nombreDelArchivo.deb`.

Una vez instalado, lo abrimos desde el menú de aplicaciones del sistema, hacemos click en el icono de las extensiones de la barra lateral e instalamos las siguientes:

- `Python`, de Microsoft. Nos permite usar funciones de autocompletado en el IDE.
- `Python Docstring Generator`, de Nils Werner. Nos permite generar la documentación de los métodos, a partir de los argumentos definidos en sus cabeceras.
- `Pylance`, de Microsoft. Mejora el autocompletado, permitiendo tener en cuenta el tipado.
- `Code Spell Checker`, de Street Side Software. Añade un corrector con las palabras inglesas.
- `Spanish - Code Spell Checker`, de Street Side Software. Añade el español como paquete de idioma a la anterior extensión.

D.4. Compilación, instalación y ejecución del proyecto

Descargamos el repositorio del proyecto desde la web usando el comando `git clone https://github.com/rogama25/UBUVoiceAssistant`.

Posteriormente, debemos usar los siguientes comandos para crear las carpetas necesarias y colocar los archivos en las rutas correctas:

- `cd UBUVoiceAssistant`
- `mkdir -p ~/.config/UBUVoiceAssistant`
- `mkdir -p ~/.config/mycroft`
- `cp -r ./src/UBUVoiceAssistant/skills/. /opt/mycroft/skills`
- `sudo mkdir -p /usr/lib/UBUVoiceAssistant`
- `sudo chown your_user /usr/lib/UBUVoiceAssistant`
- `cp -r ./src/UBUVoiceAssistant/ /usr/lib/UBUVoiceAssistant`

Para ejecutarlo desde el entorno de desarrollo, escribimos:

- `cd src`
- `python3 -m UBUVoiceAssistant.GUI.main`

D.5. Pruebas del sistema

Para realizar las pruebas de calidad de código lo podemos hacer con las herramientas Pylint y mypy. Sin embargo, debido a que en las carpetas de las skills de Mycroft se suelen usar guiones y que no son caracteres válidos en Python, es necesario renombrar temporalmente las carpetas para que tengan un guión bajo.

Después podremos hacer:

- `pylint src -ry`
- `mypy src`

Para las pruebas de funcionalidad lo podemos realizar conectándonos a un servidor de Moodle cualquiera. Durante el desarrollo se ha usado tanto UBUVirtual, como Mount Orange School como un servidor Moodle instalado en la propia máquina, usando la documentación oficial.

Apéndice E

Documentación de usuario

Nota importante a partir de este punto: Muchos lectores de PDF introducen saltos de línea que hacen que los comandos más largos dejen de funcionar. Cada comando debe introducirse por completo en una línea y luego pulsar Enter para ejecutarlo. Además, dependiendo del lector de PDF, puede que algunos comandos no contengan los caracteres correctos. Por ello, he dejado los comandos en Github Gists

E.1. Introducción

En este apartado se explica cómo se puede instalar y usar el proyecto.

E.2. Requisitos de usuarios

- Sistema operativo Windows 10 o Ubuntu 20.04 o superior.
- Conexión a internet
- 4GB de memoria RAM para Ubuntu, 6GB para Windows
- 5GB de espacio en disco libres

E.3. Instalación

Para instalar el proyecto se puede hacer o bien en Windows o bien en Ubuntu. Por el momento, se recomienda optar por la segunda opción ya que

es un proceso considerablemente más sencillo debido a que la opción para Windows todavía no es completamente estable.

En el caso de estar usando una máquina virtual dentro de Windows, es recomendable desactivar Hyper-V. Esto se debe a que impide que el flag "AVX" se pase a la máquina virtual y no funcione correctamente el sonido. Para comprobar si está activo, hay que ejecutar el siguiente comando en una terminal de la máquina virtual:

```
cat /proc/cpuinfo | grep avx
```

En el caso de que este no devuelva salida, tendremos que desactivarlo. Esto se consigue ejecutando los siguientes comandos en Windows y después reiniciando la máquina anfitriona.

- `bcdedit /set hypervisorlaunchtype off`
- `DISM /Online /Disable-Feature:Microsoft-Hyper-V`

En algunas versiones de Windows, el segundo comando devolverá un error informando que la característica no existe. En tal caso, ejecutaremos este comando:
`Disable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V-Hypervisor`

Después de ejecutar estos comandos, puede que haya funciones que dependan de la virtualización nativa de Windows que dejen de funcionar, como el WSL. Tenemos que tener en cuenta que no es posible tener la virtualización nativa de Windows y una máquina virtual que pase correctamente el flag "AVX", por lo que tendremos que pensar si queremos estar usando una máquina virtual o, por el contrario, WSL. Para volver a activarlos, hay que ejecutar lo siguiente y después reiniciar:

- `bcdedit /set hypervisorlaunchtype auto`
- `DISM /Online /Enable-Feature:Microsoft-Hyper-V`
- `Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V-Hypervisor`

Windows Susbsystem for Linux

En el caso de que hayamos optado por usar Windows 10 como sistema operativo, es necesario que instalemos este componente para poder ejecutar la aplicación. Si se va a usar Ubuntu, saltar a la siguiente sección.

Instalación de WSL

Para comenzar, deberemos conocer la versión de Windows 10 que estamos ejecutando en nuestro equipo. Para ello, abrimos el menú de configuración de Windows, hacemos click en Sistema" y luego en Acerca de". En esa pantalla deberemos fijarnos que tengamos un sistema operativo de 64 bits y tengamos la versión 1607 o superior.

Ahora, tenemos que buscar Powershell" en el menú de inicio y la ejecutaremos como administrador. En esa ventana, escribimos el siguiente comando:

```
dism.exe /online /enable-feature
/featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

En el caso de que además tengamos la versión 1903 o superior, con la compilación 18362 o superior, es recomendable instalar WSL2 para obtener un mayor rendimiento. **En el caso de no tenerlo, reiniciamos el equipo y saltamos al apartado de la instalación de Ubuntu para WSL.**

Lo siguiente es activar la virtualización, usando este comando en la Powershell que habíamos abierto:

```
dism.exe /online /enable-feature
/featurename:VirtualMachinePlatform /all /norestart
```

En este punto, reiniciamos el ordenador. Después del reinicio, instalamos el paquete de actualización del kernel de Linux, desde este enlace.

Abrimos otra Powershell como administrador y establecemos el uso de WSL2 por defecto:

```
wsl --set-default-version 2
```

Instalación de Ubuntu para WSL

Para instalar Ubuntu en el Windows Subsystem for Linux, tenemos que descargarlo desde la Microsoft Store, accesible en este enlace. Una vez instalado, es probable que se abra de manera automática una terminal en la que se nos pedirá elegir un nombre de usuario y una contraseña. En el caso de que no se abra automáticamente, podremos abrirla desde el menú de inicio, en el acceso directo a Ubuntu.

En el caso de que no tengamos disponible la Microsoft Store en el ordenador, podemos descargar el paquete de Ubuntu desde este enlace. Una vez se haya descargado el archivo, abrimos una Powershell y escribimos el siguiente

comando: `Add-AppxPackage .\app_name.appx`, donde app_name.appx es el nombre del archivo descargado.

Instalación de Pulseaudio y servidor X11

Para que el susbsistema pueda recibir sonido del micrófono y mostrar interfaces gráficas debemos instalar lo siguiente en Windows 10:

VcXserv, descargable desde este enlace. Usando nano o cualquier otro editor de texto, deberemos añadir al ~/.bashrc del WSL las siguientes líneas:

`export DISPLAY=:0` (únicamente si no usamos WSL2)

`export DISPLAY=$(awk '/nameserver / {print $2; exit}' /etc/resolv.conf 2>/dev/null):0` (si usamos WSL2)

`export LIBGL_ALWAYS_INDIRECT=1`

Abriremos VcXserv desde el menú de inicio en Windows, que aparecerá con el nombre "XLaunch". Se nos abrirá una ventana de configuración. En las dos primeras pantallas dejaremos las opciones que vienen seleccionadas por defecto, y en la tercera marcaremos la casilla de "Disable access control"

Pulseaudio, descargable desde este enlace, haciendo click donde pone "zipfile containing preview binaries". Tendremos que editar los siguientes archivos:

`etc/pulse/default.pa`, donde tendremos que añadir:

`load-module module-native-protocol-tcp auth-ip-acl=127.0.0.1 auth-anonymous=1`

`load-module module-waveout sink_name=output source_name=input record=1`

`etc/pulse/daemon.conf`, donde introducimos lo siguiente:

`exit-idle-time = -1`

Añadimos lo siguiente al ~/.bashrc del WSL:

`export PULSE_SERVER=tcp:127.0.0.1` (WSL1)

`export PULSE_SERVER=tcp:$(awk '/nameserver / {print $2; exit}' /etc/resolv.conf 2>/dev/null)` (si usamos WSL2)

Ejecutamos el programa bin/pulseaudio.exe dentro de una terminal. Es probable que aparezcan varios errores o advertencias similares a las que se ven en la imagen, pero el programa funciona con normalidad.

```
C:\Users\user\Downloads\pulseaudio-1.1\bin>pulseaudio.exe
W: [(null)] pulsecore/core-util.c: Secure directory creation not supported on Win32.
W: [(null)] pulsecore/core-util.c: Secure directory creation not supported on Win32.
W: [(null)] pulsecore/core-util.c: Secure directory creation not supported on Win32.
W: [(null)] pulsecore/core.c: failed to allocate shared memory pool. Falling back to a normal memory pool.
W: [(null)] pulsecore/core-util.c: Secure directory creation not supported on Win32.
W: [(null)] pulsecore/core-util.c: Secure directory creation not supported on Win32.
W: [(null)] pulsecore/core-util.c: Secure directory creation not supported on Win32.
E: [(null)] daemon/main.c: Failed to load directory.
W: [(null)] pulsecore/core-util.c: Secure directory creation not supported on Win32.
```

Figura E.1: Errores que aparecen en la terminal al ejecutar Pulseaudio

Para continuar, es necesario cerrar y volver a abrir la ventana de Ubuntu

Descarga del repositorio

Para descargar el código de la aplicación se puede hacer usando un navegador o usando la terminal. En el caso de que estemos usando Windows Subsystem for Linux, es recomendable usar la terminal.

Usando un navegador web

Desde cualquier navegador de internet, hay que ir a la página web del repositorio.

Allí, en la parte derecha, haz click en la sección "Releases" y, en el desplegable de "Assets", descarga el "Source code (.zip)" que aparezca en la versión más reciente.

Una vez se complete la descarga, primero hay que descomprimirlo y luego abrir una terminal donde nos moveremos hasta la carpeta que se ha creado con el nombre de la versión usando el comando `cd`.

Allí hay que ejecutar el comando `sudo ./install.sh install` y esperar a que termine. Puede que tarde varios minutos, dependiendo de la velocidad de internet y del equipo, ya que descarga y configura todos los componentes necesarios para ejecutar el programa.

En el caso de que el instalador se quede atascado en una sección interactiva, lee el último párrafo de la siguiente sección.

Usando la terminal

Desde una terminal, ejecutamos los siguientes comandos:

- `sudo apt update`

- `sudo apt install git`
- `git clone https://github.com/rogama25/UBUVoiceAssistant.git`
- `cd UBUVoiceAssistant`
- `sudo ./install.sh install`

Este proceso puede tardar varios minutos, dependiendo de la velocidad del equipo y de la conexión a internet. En el caso de estar usando WSL, el tiempo es considerablemente más alto, ya que necesita instalar una serie de paquetes extra para que funcione correctamente.

Bajo algunas circunstancias (habitualmente máquinas virtuales, debido a su menor velocidad), el instalador se puede quedar atascado en la parte de configuración de Mycroft, que hace una serie de preguntas de sí o no. En el caso de que esto ocurra, debemos pulsar Control + C para salir del instalador, ejecutar `sudo ./install.sh uninstall` y después `sudo ./install.sh install --manual`. Usando este modo, se reiniciará la instalación, informando al usuario cómo responder las preguntas.

E.4. Manual del usuario

Después de haber completado la instalación, en el caso de que estemos usando Ubuntu como sistema operativo, tendremos un icono en su lanzador con el que podremos abrir nuestro programa. Si estamos usando el Windows Susbsystem for Linux o si no aparece el icono en el lanzador de aplicaciones, deberemos escribir `UBUVoiceAssistant` para ejecutar el programa.

Si lo estamos ejecutando en WSL, en algunos casos puede aparecer un error que dice "ImportError: libQt5Core.so.5: cannot open shared object file: No such file or directory". Este es un error conocido de WSL y se soluciona ejecutando el siguiente comando:

```
sudo strip --remove-section=.note.ABI-tag
/usr/lib/x86_64-linux-gnu/libQt5Core.so.5
```

La primera ventana que veremos es la de inicio de sesión. En la esquina superior derecha podemos elegir el idioma de la aplicación y en la parte inferior tendremos los campos donde debemos introducir los credenciales de Moodle. Deberemos especificar también la dirección web del servidor, siendo en nuestro caso `https://ubuvirtual.ubu.es`. Las direcciones deben incluir el protocolo http o https.

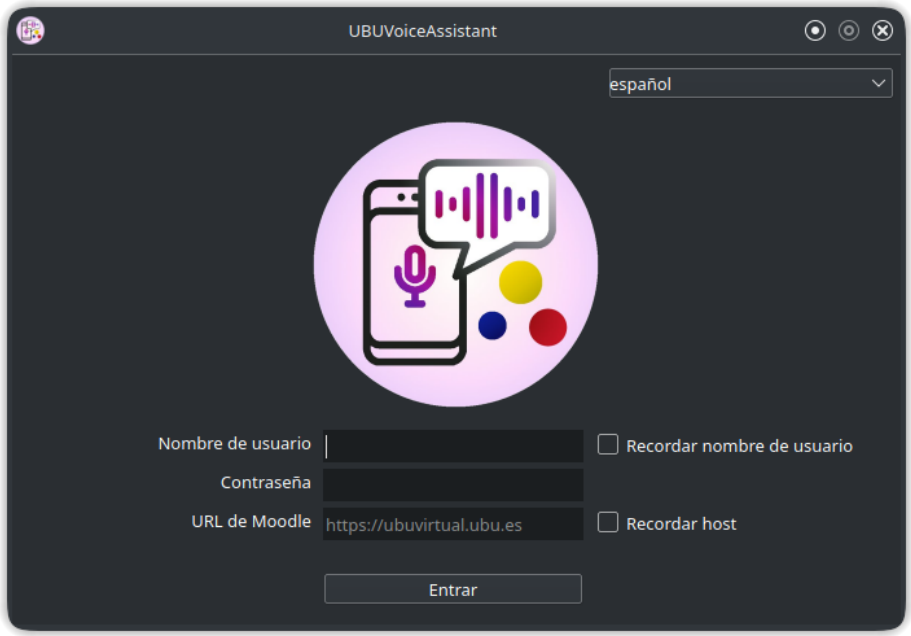

Figura E.2: Pantalla de inicio de sesión

Una vez hayamos iniciado sesión por primera vez (puede tardar varios minutos mientras Mycroft instala algunos de sus componentes), se nos mostrará la ventana de emparejamiento. En ella aparecen los pasos que debemos realizar para vincular el cliente de Mycroft que se ha instalado en el equipo con sus servicios web. El procedimiento es el siguiente:

- Abrir en un navegador de Internet la página web de Mycroft.
- Registrarnos o iniciar sesión en una cuenta que ya tengamos.
- En la parte superior derecha, donde aparece el icono del perfil, hacer click en “Add Device”.
- Escribir el código dicho por Mycroft en el campo que pone “Pairing Code”.
- Indicar un nombre para el equipo, a nuestra elección, en el campo “Name”
- En la parte inferior, seleccionar “Google Voice” y guardar los cambios.

Figura E.3: Ventana de emparejamiento de UBUVoiceAssistant

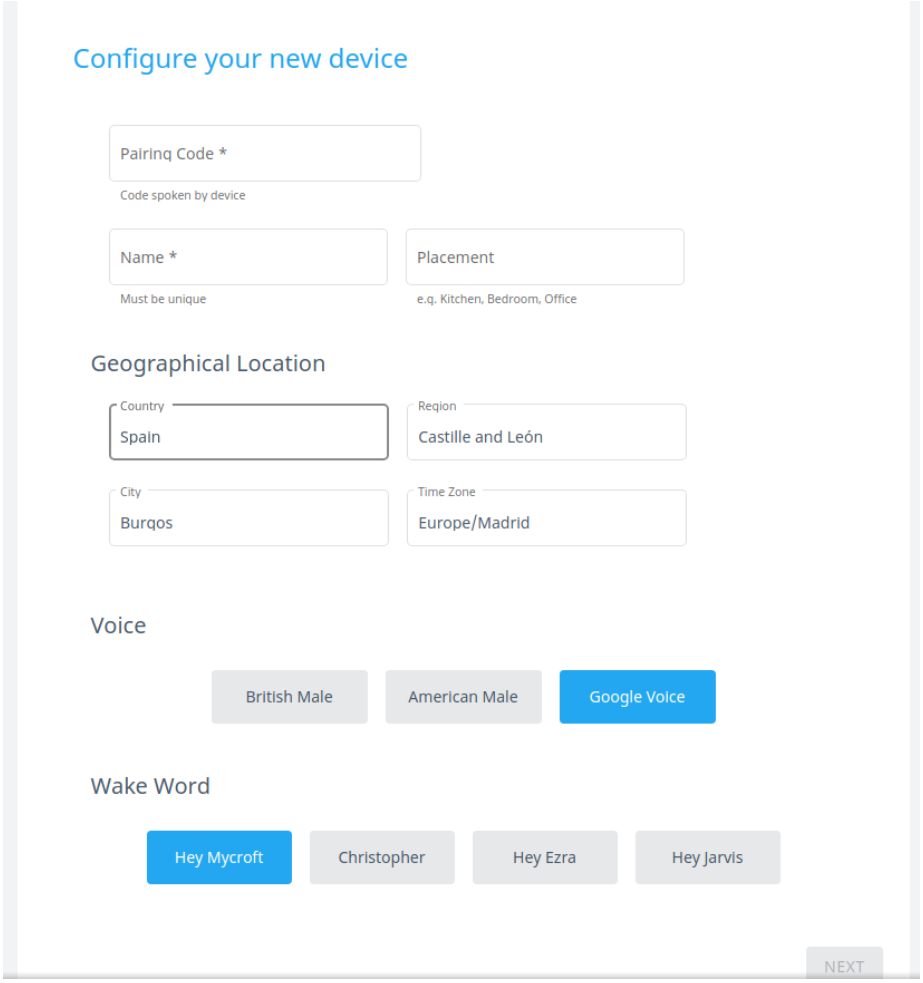

Figura E.4: Formulario de emparejamiento

Pasados unos segundos, se debería abrir la ventana con la interfaz de chat. En ella se muestra la conversación que tengamos con el asistente de voz. A partir de este punto podremos decir nuestras órdenes a través del micrófono. Al decir "Hey Mycroft", deberíamos oír un sonido que significa que ha detectado la wake word y está grabando la siguiente frase que digamos. También podemos silenciar el micrófono mediante el botón que hay en la interfaz, o escribir nuestras órdenes por teclado en el campo inferior. Haciendo click en el botón de la parte superior derecha de la ventana, podemos activar y desactivar las skills que queramos.

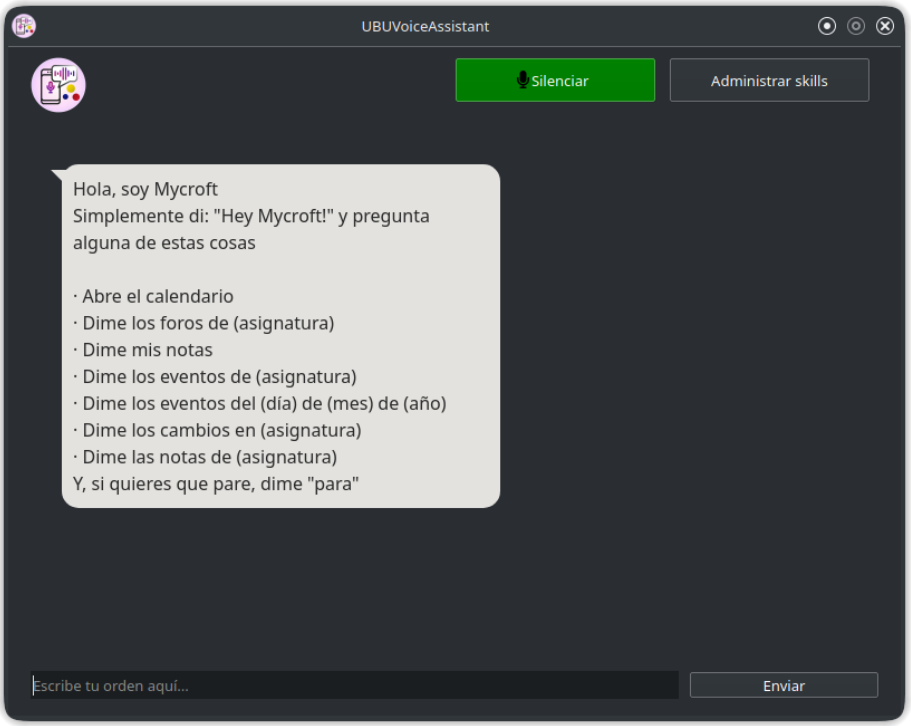

Figura E.5: Ventana de chat

Bibliografía

[1] adp1002. *UBUVoiceAssistant*. [Online; accessed 29. Jun. 2021]. Jun. de 2021. URL: https://github.com/adp1002/UBUVoiceAssistant/tree/master/latex.

[2] *Apache License, Version 2.0*. [Online; accessed 29. Jun. 2021]. Jun. de 2021. URL: https://www.apache.org/licenses/LICENSE-2.0.

[3] *Creative Commons — Attribution-ShareAlike 4.0 International — CC BY-SA 4.0*. [Online; accessed 2. Jul. 2021]. Jul. de 2021. URL: https://creativecommons.org/licenses/by-sa/4.0.

[4] *GNU General Public License v2.0 - GNU Project - Free Software Foundation*. [Online; accessed 29. Jun. 2021]. Jun. de 2021. URL: https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html.

[5] *History and License — Python 3.9.6 documentation*. [Online; accessed 29. Jun. 2021]. Jun. de 2021. URL: https://docs.python.org/3/license.html#terms-and-conditions-for-accessing-or-otherwise-using-python.

[6] *License — Babel 2.7.0 documentation*. [Online; accessed 29. Jun. 2021]. Ene. de 2021. URL: http://babel.pocoo.org/en/latest/license.html#babel-license.

[7] *The GNU General Public License v3.0 - GNU Project - Free Software Foundation*. [Online; accessed 29. Jun. 2021]. Jun. de 2021. URL: https://www.gnu.org/licenses/gpl-3.0.en.html.

[8] *What is Scrum?* [Online; accessed 16. Jun. 2021]. Jun. de 2021. URL: https://www.scrum.org/resources/what-is-scrum.