UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

título del TFG
Documentación Técnica

Presentado por nombre alumno
en Universidad de Burgos — 8 de junio de 2020
Tutor: nombre tutor

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se va a comentar la planificación temporal del proyecto así como la metodología que se ha seguido para su desarrollo y la viabilidad del proyecto en el marco económico y legal.

A.2. Planificación temporal

En el desarrollo se ha empleado SCRUM, que es un marco de trabajo para desarrollo ágil de software, pero en una versión adaptada a un equipo de 1 persona, a diferencia de los equipos normales que suelen ser de 5 a 9 personas. También se ha cambiado la frecuencia de las reuniones de seguimiento, siendo semanales en vez de diarias.

Estas reuniones se han utilizado para exponer el trabajo realizado, los problemas que se han encontrado durante su realización, mostrar el avance del proyecto y fijar nuevas tareas para continuar con el proyecto.

Como es normal utilizando metodología ágil el desarrollo ha sido iterativo organizado por *sprints*. La duración de los *sprints* ha sido de una, dos y tres semanas.

Sprint 1

El primer *sprint* duró tres semanas (del 17 de febrero al 2 de marzo). La primera reunión de seguimiento se centró en explicar detalladamente los objetivos del trabajo. Gran parte del trabajo realizado en este *sprint* se

centró en investigar y aprender como funcionan los asistentes virtuales, en concreto Alexa, así como centrarme en leer trabajos relacionados con lo que se pretendía en el proyecto para analizar que cosas están bien hechas, que se puede cambiar y la viabilidad de algunas otras cosas.

También aprendí cómo funciona Moodle, que son los *web services* y como utilizarlos. Finalmente, intentando hacer el prototipado de una primera versión de *Skill* de Alexa, se decidió cambiar el asistente con el que se iba a desarrollar el trabajo.

Sprint 2

Este *sprint* tuvo una duración de tres semanas (del 2 de marzo al 12 de marzo). En este *sprint* me dediqué a plantear si seguir con este proyecto, cambiar totalmente o hacer alguna modificación. Finalmente me decanté por continuar con el asistente de voz pero necesitaba encontrar un asistente de voz que no me fuera a dar problemas. Se contempló el uso de Google Assistant pero no estaba seguro de que no me fuera a dar problemas similares que los que tuve con Alexa.

Continuando la búsqueda de un asistente me encontré con Mycroft, que era open source y no tenía ningún tipo de limitación, además de estar bastante extendido para ser un asistente de este tipo. Así que decidí usar Mycroft para continuar el proyecto. Tras comentarle la decisión al tutor, empleé el resto del *sprint* para prototipar una primera version de la *Skill* y contemplar nuevas opciones que se habían abierto al usar Mycroft en vez de Alexa, como un cliente gráfico.

Sprint 3

El tercer *sprint* duró una semana (del 18 al 25 de marzo) y se desarrolló una versión inicial de la interfaz gráfica. También se implementó el acceso al calendario mediante los *web services* de Moodle. Finalmente estuve haciendo tests e investigando si podía haber algún tipo de problema en cuanto al número de accesos, y descubrí un problema de comunicación entre la aplicación y Mycroft.

Sprint 4

En las dos semanas (del 2 al 16 de abril) que duró este *sprint* seguí pensando sobre que funcionalidad podía implementar en la *skill* y se integró la opción de preguntar por los eventos de un día concreto, así como la

posibilidad de realizar las preguntar vía texto en la aplicación. En este *sprint* también se solucionó el problema de comunicación entre el proceso de la aplicación y los procesos de Mycroft mediante comunicación por *sockets*. Finalmente se inició el manual de usuario utilizando Sphinx con ReadTheDocs en un repositorio de GitHub aparte.

Sprint 5

El quinto *sprint* fue de una semana (del 23 al 30 de abril) y se centró en aumentar las interacciones con la *skill*, implementando las posibilidades de preguntar por actividad reciente dentro de un curso, eventos de un curso, notas finales del usuario y notas de un curso, que en esta primera implementación se utilizó *web scraping* a falta de un *web service*. Por último se mejoró la GUI permitiendo mostrar las respuestas también mediante texto y no solo audio.

Sprint 6

Este *sprint* tuvo una duración de dos semanas (del 30 de abril al 14 de mayo) y se centró en mejoras para la interacción del usuario con la aplicación. Entre estas mejoras están la recolección y muestra de logs, tanto de Mycroft como de la app, comprobar si el micrófono está activo, y se decidió separar la *skill* en tres para poder deshabilitar módulos. También se aumentaron las interacciones de la *skill* permitiendo leer los foros de un curso.

Sprint 7

El séptimo *sprint* también duró dos semanas (del 14 al 28 de mayo) y se dedicó para mejorar la interfaz gráfica y crear el manual de instalación para que el tutor pudiera probarlo.

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

B.2. Objetivos generales

B.3. Catalogo de requisitos

B.4. Especificación de requisitos

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

Los datos están correlacionados con el formato con el que se guardan en Moodle, aunque solo están modelados aquellos datos que me son útiles. En el paquete /src/model/ están las clases que modelan estos datos y son:

- **User** para guardar datos relativos al usuario.
- **Course** para guardar los datos de los cursos.
- **Forum** para los foros de un curso.
- **Discussion** para las discusiones de un foro.
- **GradeItem** para las notas.
- **Event** para los eventos del calendario.

C.3. Diseño procedimental

C.4. Diseño arquitectónico

El proyecto realmente consiste en dos aplicaciones, una la aplicación con la interfaz gráfica y la otra sería Mycroft. Ambas aplicaciones utilizan una

arquitectura cliente-servidor, que es una arquitectura en la que el cliente realiza peticiones a través de un medio y el servidor envía una respuesta en función de la petición a través del mismo medio. La aplicación gráfica actúa como cliente y el servidor es la API REST de la plataforma de Moodle, es decir, los *web services*. Mycroft tiene como cliente sus *skills* que se comunican con dos servidores, uno para las funciones de STT y otro para TTS. Y entre las dos aplicaciones también existe una arquitectura cliente-servidor, siendo la aplicación gráfica el servidor y los clientes las diferentes *skills*.

Apéndice D

Documentación técnica de programación

D.1. Introducción

A continuación se va a detallar la estructura del proyecto, como montar el entorno de desarrollo, como ejecutar la aplicación y como realizar pruebas.

D.2. Estructura de directorios

El proyecto está dividido de esta forma:

- /latex/: contiene los ficheros que se necesitan para generar le PDF con la memoria y los anexos, así como las imágenes utilizadas y la bibliografía.
- /src/: contiene los ficheros fuentes del proyecto..
- /src/GUI/: ficheros encargados del cliente gráfico
- /src/model/: ficheros para representar los datos de Moodle.
- /src/prototipo/: ficheros del prototipo para Alexa
- /src/webservice/: contiene los ficheros encargados de hacer las peticiones al servidor de Moodle para recuperar los datos.
- /src/util/: utilidades para los distintos módulos de la aplicación.
- /src/skills/: contiene las 3 *Skills* del proyecto.

D.3. Manual del programador

Para el entorno de desarrollo se necesita Python3 con sus dependencias, Mycroft y un editor de código fuente.

Python 3

Python 3 viene instalado por defecto en Ubuntu, pero necesitamos instalar las dependencias para la aplicación. Estas dependencias son PyQt5 y MycroftMessageBus y se instalan mediante los comandos **sudo apt-get install python3-pyqt5** y **pip3 install mycroft-messagebus-client**

Mycroft

Una vez instalado, ir a la sección **Running Mycroft** para ejecutarlo, mediante el los comandos **cd ~/mycroft-core** para ir a la carpeta y **./start-mycroft.sh debug** para ejecutarlo. Tras iniciar Mycroft tenemos que decir por voz cualquier cosa para que se inicie el proceso de enlazamiento. La aplicación nos dirá un código (si se ha iniciado con la opción de **debug** también aparecerá en pantalla). Este código hay que ponerlo al añadir un nuevo dispositivo. En la página de Mycroft en la esquina superior derecha, en la sección de **Devices**, **Add Device** en el apartado de **Pairing code**. Al añadir el nuevo dispositivo es importante que en el apartado **Voice** se seleccione **Google Voice**.

Para para Mycroft si se ha iniciado con la opción de *debug* se puede presionar la combinación de teclas **ctrl + c** o escribir en la interfaz **:exit**

Una vez instalado, ir a la sección **Running Mycroft** para ejecutarlo. Tras iniciar Mycroft tenemos que decir por voz cualquier cosa para que se inicie el proceso de enlazamiento. La aplicación nos dirá un código (si se ha iniciado con la opción de **debug** también aparecerá en pantalla). Este código hay que ponerlo al añadir un nuevo dispositivo. En la página de Mycroft en la esquina superior derecha, en la sección de **Devices**, **Add Device** en el apartado de **Pairing code**. Al añadir el nuevo dispositivo es importante que en el apartado **Voice** se seleccione **Google Voice**.

Editor

Se puede utilizar el editor de código fuente con el que más cómodo se sienta cada uno, en esta sección voy a explicar como configurar el editor

Atom que es el que he estado usando y el que más cómodo me parece, aunque hay alternativas como VisualStudio Code que son muy similares.

Para instalar Atom nos vamos a su página, lo descargamos y lo ejecutamos. Una vez instalado vamos a instalar dos paquetes para que sea mucho más cómodo programar con Python. En la pestaña **Edit** hacemos click en **Preferences**.

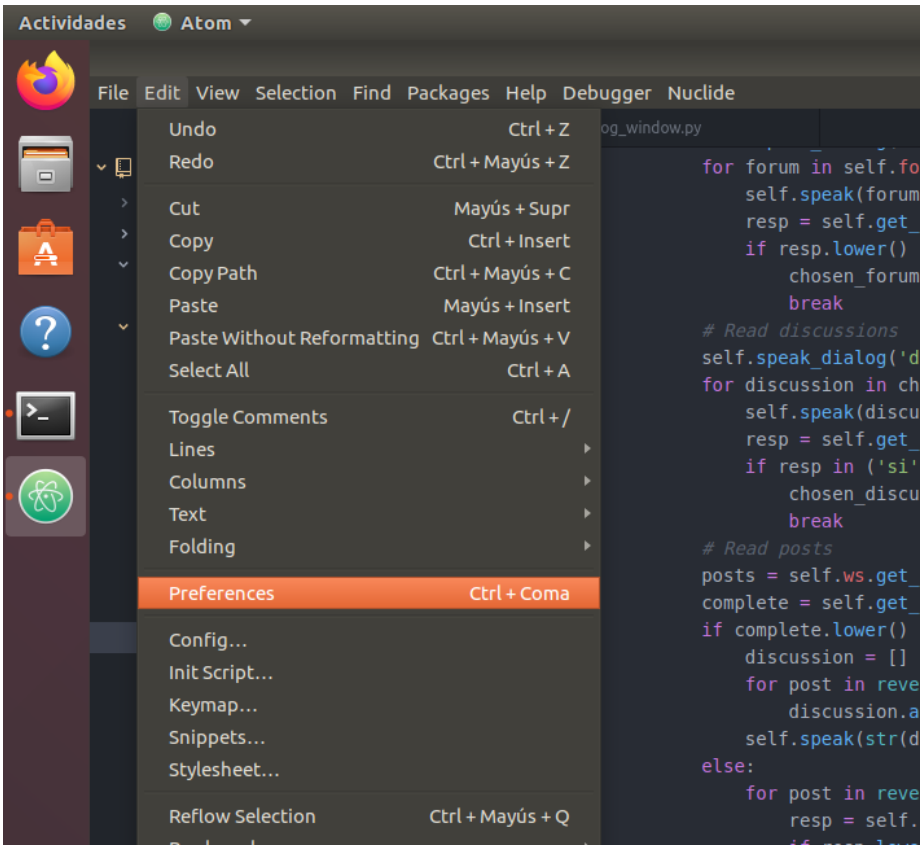

Figura D.1: Preferencias Atom

En esta nueva ventana que se nos ha abierto hacemos click en **Install** y en el buscador escribimos **atom-ide-ui** e instalamos el paquete. Hacemos lo mismo buscando **ide-python**.

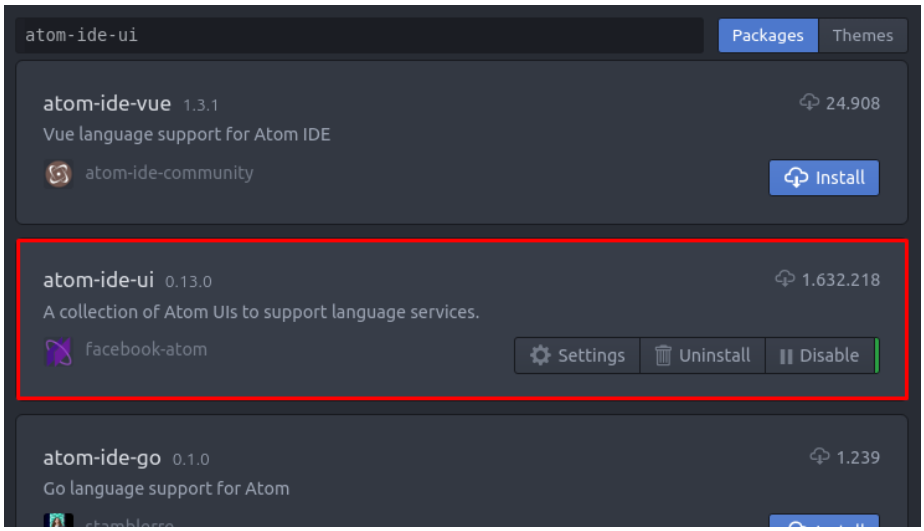

Figura D.2: Instalar atom-ide

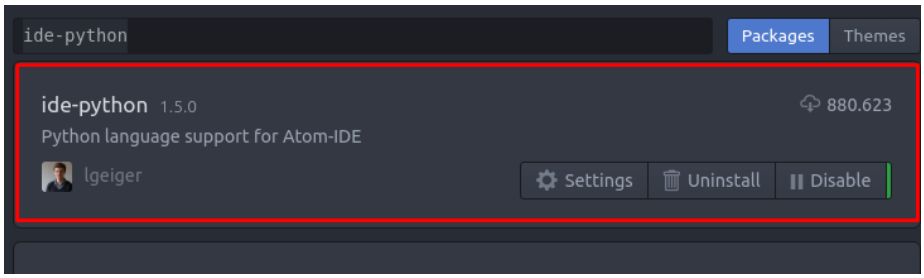

Figura D.3: Instalar ide-python

Para que funcione el paquete **ide-python** tenemos que instalar **python-language-server** con el comando **python3 -m pip install python-language-server[all]** en la consola. Ahora vamos a configurar el paquete, para ello hacemos click en la pestaña **Packages**, después en **Settings** de **ide-python**. En la nueva ventana que se nos ha abierto se puede configurar el comportamiento de todos los componentes de **python-language-server**, pero para que funcione de momento solo nos interesa cambiar la primera línea en la que pone python y poner **python3**

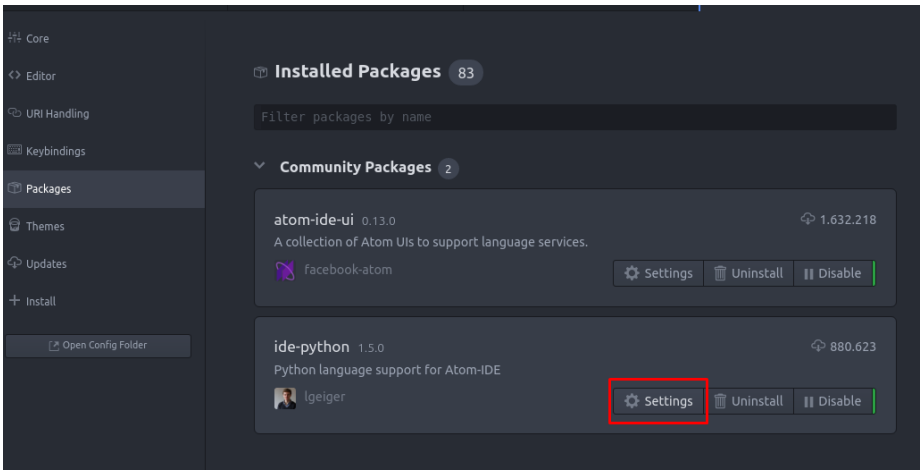

Figura D.4: Configuración del paquete ide-python

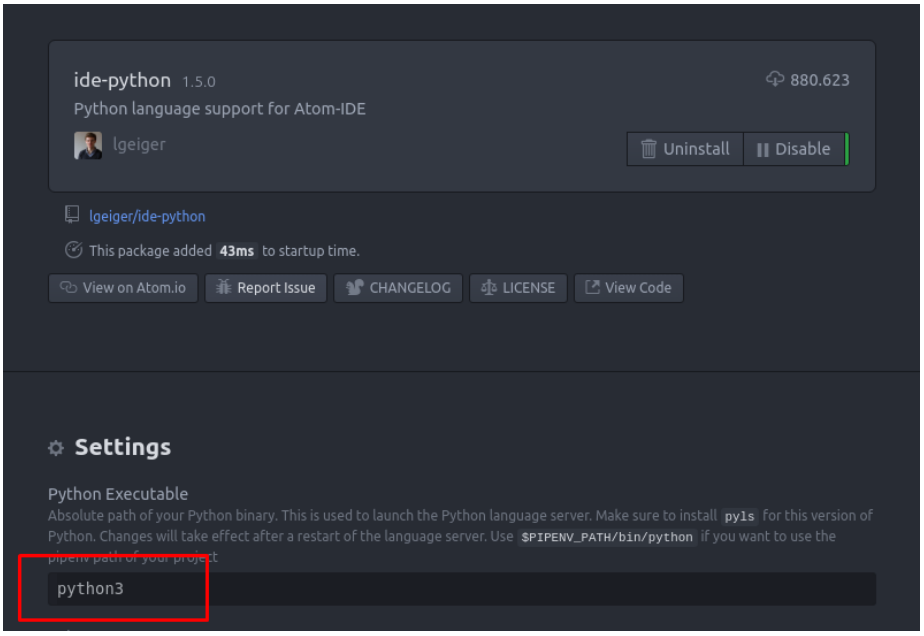

Figura D.5: Configuración del paquete ide-python

D.4. Compilación, instalación y ejecución del proyecto

Clonamos el repositorio en nuestra carpeta personal. Si no estamos en la carpeta personal ejecutar el comando **cd ~**. Para clonar el repositorio ejecutar el comando **git clone https://github.com/adp1002/UBUAssistant**.

Para iniciar la aplicación vamos al directorio /src/ con el comando **cd ~/UBUAssistant/src/** y ejecutamos con **python3 -m ./GUI/main.py »logs.log**

D.5. Pruebas del sistema

Para probar nueva funcionalidad introducida se puede utilizar la plataforma de pruebas de Moodle Mount Orage School o la plataforma con la que se vaya a utilizar la aplicación, que en mi caso es UBUVirtual y es con lo que he realizado pruebas principalmente.

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado se explica como puede un usuario prepara el entorno para ejecutar la aplicación y cómo utilizar la aplicación.

E.2. Requisitos de usuarios

Sistema operativo GNU/Linux Python3 Conexión a Internet

E.3. Instalación

Ubuntu

Si ya tienes Ubuntu u otra distribución Linux vete al siguiente apartado.

Para utilizar Ubuntu se va a necesitar algún software de virtualización. En esta guía se va a utilizar VirtualBox, que se puede descargar desde su página.

Figura E.1: Descarga de VirtualBox

Desde su página elegimos la opción de **Windows hosts** (u otra en función del sistema operativo que se esté utilizando). Esperamos a que se inicie la descarga y ejecutamos el archivo .exe que hemos descargado. Se nos abrirá el instalador y le damos a **Next, Next, Next, Yes, Install**.

Una vez instalado VirtualBox vamos a descargar Ubuntu. Vamos a su página y hacemos click en el enlace que pone **64-bit PC (AMD64) desktop image**. Una vez haya finalizado la descarga abrimos VirtualBox. En la ventana de VirtualBox, hacemos click en **Nueva** y la configuramos para **Ubuntu (64-bit)**. A continuación elegimos la cantidad de memoria y creamos un disco virtual de, al menos, 30 GB.

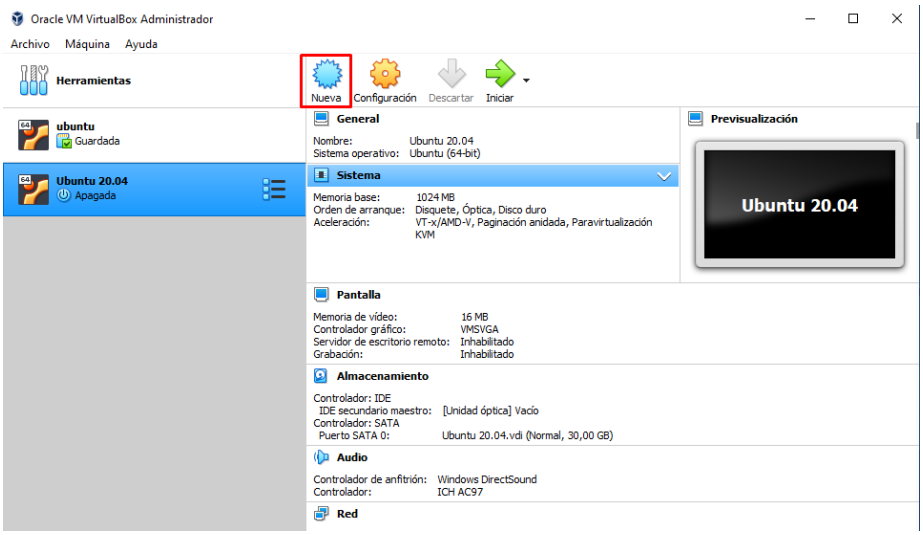

Figura E.2: Nueva máquina virtual

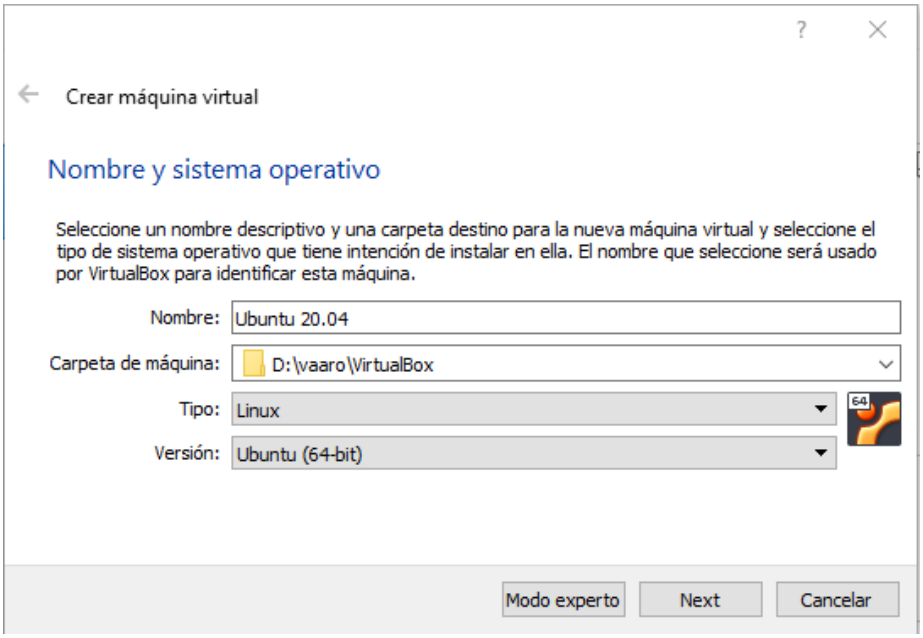

Figura E.3: Tipo de sistema operativo

Después hacemos click en el apartado de configuración en la ventana inicial de VirtualBox. En la pestaña **Almacenamiento** hacemos click en **Añadir una unidad óptica**, que es el botón a la derecha de **Controlador: IDE**. En la nueva ventana hacemos click en **Añadir** y seleccionamos el

archivo .iso de Ubuntu. En la ventana de configuración también se puede ajustar el número de procesadores y la cantidad de memoria gráfica desde las pestañas **Sistema** y **Pantalla** respectivamente. Recomiendo utilizar, como mínimo, 2048MB de memoria RAM, 2 núcleos de procesador y 128MB de memoria de vídeo.

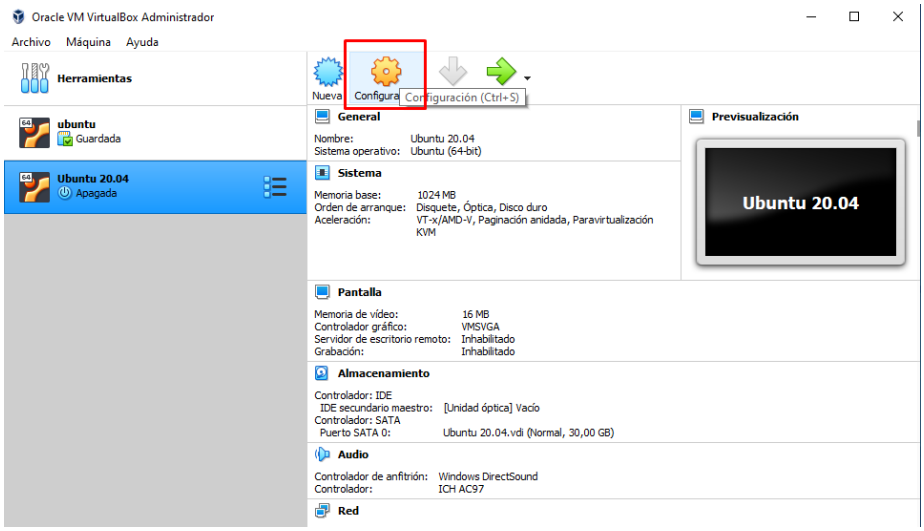

Figura E.4: Configuración de máquina virtual

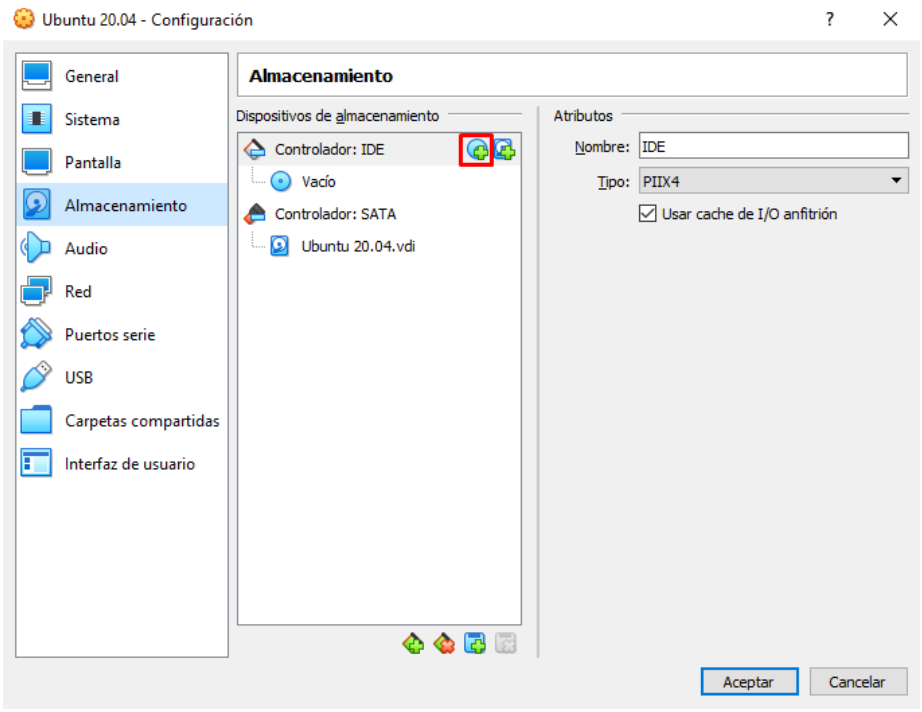

Figura E.5: Parámetros de configuración

Una vez terminada la configuración, iniciamos Ubuntu desde la ventana principal de VirtualBox con el botón **Iniciar**. En la primera pantalla de Ubuntu seleccionamos el idioma (español) y le damos a instalar Ubuntu. Después seleccionamos el idioma del teclado y la instalación de Ubuntu, en nuestro caso es suficiente con la **instalación mínima**. A continuación borramos el disco virtual, seleccionamos la franja horaria y completamos los campos que nos pide. Una vez termine de instalar nos pedirá reiniciar y después darle al **Enter**. Cuando se haya iniciado Ubuntu solo queda instalar las **huest additions**, que es totalmente opcional pero aumenta el rendimiento. Para instalarlas hacer click en la pestaña **Dispositivos** y **Insertar imagen de CD de las «Guest Additions»...** y reiniciar el sistema operativo.

Figura E.6: Instalación de Ubuntu

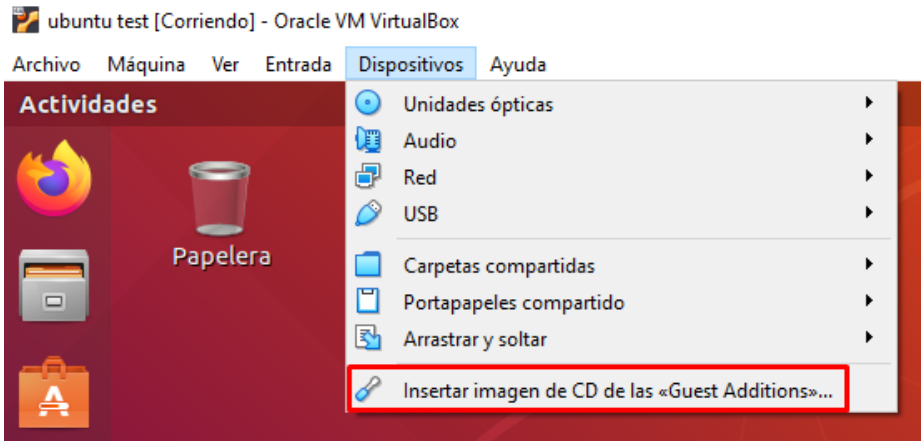

Figura E.7: Instalación de Ubuntu

Python3 y sus dependencias

Python3 viene instalado por defecto en Ubuntu 18.04. Para instalar las dependencias necesitamos pip3, que se instala con el comando **sudo apt install python3-pip**. Ahora las dependencias que necesitamos se instalan mediante los comandos **sudo apt-get install python3-pyqt5** y **pip3 install mycroft-messagebus-client**

Mycroft

Para clonar el repositorio necesitamos git, se instala mediante el comando **sudo apt install git**.

Para instalar Mycroft hay que seguir los pasos en la sección **Getting Started** de la guía que hay en su repositorio, que son básicamente ir a la carpeta personal con el comando **cd** ~, clonar su repositorio mediante el comando **git clone https://github.com/MycroftAI/mycroft-core.git**, ir a la carpeta /mycroft-core/ con el comando **cd mycroft-core** e iniciar la instalación con el comando **bash dev_setup.sh**

Una vez instalado, ir a la sección **Running Mycroft** para ejecutarlo, mediante el los comandos **cd ~/mycroft-core** para ir a la carpeta y **./start-mycroft.sh debug** para ejecutarlo. Tras iniciar Mycroft tenemos que decir por voz cualquier cosa para que se inicie el proceso de enlazamiento. La aplicación nos dirá un código (si se ha iniciado con la opción de **debug** también aparecerá en pantalla). Este código hay que ponerlo al añadir un nuevo dispositivo. En la página de Mycroft en la esquina superior derecha, en la sección de **Devices**, **Add Device** en el apartado de **Pairing code**. Al añadir el nuevo dispositivo es importante que en el apartado **Voice** se seleccione **Google Voice**.

Para para Mycroft si se ha iniciado con la opción de *debug* se puede presionar la combinación de teclas **ctrl + c** o escribir en la interfaz **:exit**

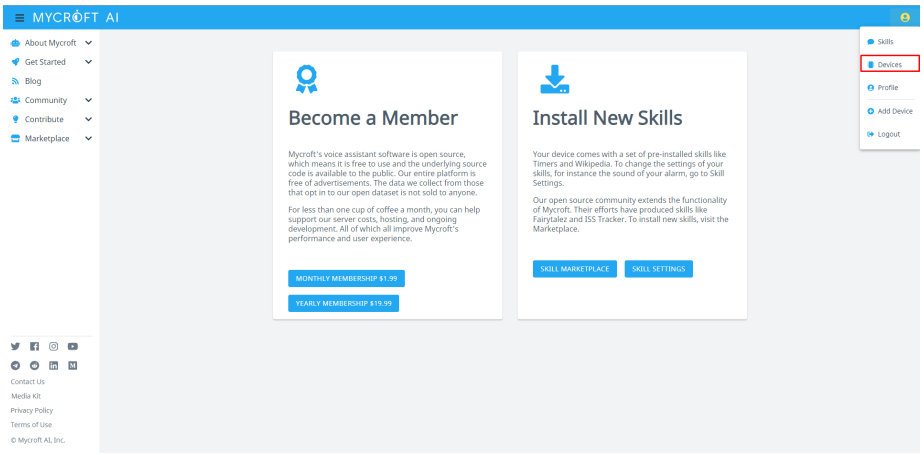

Figura E.8: Crear dispositivo en Mycroft

E.4. Manual del usuario

En Ubuntu, para ejecutar la aplicación descargar el proyecto de GitHub mediante los comandos **cd ~** (para descargarlo en la carpeta personal) y **git clone https://github.com/adp1002/UBUAssistant**. Cuando tengamos el proyecto descargado, mover el contenido de la carpeta **~/src/skills/** a **/opt/mycroft/skills/** (se puede hacer ejecutando el comando **sudo mv ~/UBUAssistant/src/skills/* /opt/mycroft/skills/**).

Nos vamos a la carpeta **/src/** del proyecto (comando **cd ~/UBUAssistant/src**) y ejecutar con el comando **python3 -m ./GUI/main.py »logs.log &**. Se nos abrirá la ventana de inicio de sesión. En esta ventana se introducirán los credenciales del usuario para la plataforma de Moodle a la que se quiera conectar, así como la URL de esta plataforma. Es importante que en el campo **host** se introduzca la URL exactamente con el formato que viene indicado. Por ejemplo, para conectarse a UBUVirtual la URL que habrá que introducir es, literalmente, **https://ubuvirtual.ubu.es**.

Para usar la aplicación mediante voz, hay que decir la *wake word* (por defecto es "Hey Mycroft") y se escuchará un sonido si ha detectado la *wake word*. Después de escuchar el sonido se puede preguntar lo que queramos. Para utilizar la conversación mediante texto hay que escribir la pregunta, sin la *wake word*, y darle a enviar. Toda la conversación, ya sea mediante texto o voz, será transcrita en la ventana.

Se pueden tener activas las *skills* que queramos, haciendo click en el botón de **Administrar Skills**. Las skills que están marcadas son las activas y las no marcadas no están activas.

Con el botón de **Abrir Logs** podemos ver los distintos mensajes de información que se van guardando durante la ejecución de la aplicación.

Bibliografía