UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

ANÁLISIS, DESARROLLO E IMPLEMENTACIÓN DE LA INFRASTRUCTURA PARA UN ASISTENTE VIRTUAL CON ESCUCHA ACTIVA DESTINADO A DETECTAR SOLEDAD NO DESEADA EN PERSONAS MAYORES

Trabajo Fin de Máster

Máster Universitario en Ingeniería Biomédica

AUTOR/A: Cilleruelo Méndez, Nerea

Tutor/a: Traver Salcedo, Vicente

Cotutor/a: Lull Noguera, Juan José

Cotutor/a externo: FIDES VALERO, ALVARO

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

ANÁLISIS, DESARROLLO E IMPLEMENTACIÓN DE LA INFRASTRUCTURA PARA UN ASISTENTE VIRTUAL CON ESCUCHA ACTIVA DESTINADO A DETECTAR SOLEDAD NO DESEADA EN PERSONAS MAYORES

Trabajo Fin de Máster

Máster Universitario en Ingeniería Biomédica

AUTOR/A: Cilleruelo Méndez, Nerea

Tutor/a: Traver Salcedo, Vicente

Cotutor/a: Lull Noguera, Juan José

Cotutor/a externo: FIDES VALERO, ALVARO

CURSO ACADÉMICO: 2021/2022

“La soledad es un buen sitio para ir de visita,

pero un mal lugar para quedarse.”

~Anónimo

Resumen

El proyecto DIAL desarrolla una herramienta para combatir la soledad no deseada en las personas mayores. Esta herramienta basada en tecnologías asistivas de voz multiplataforma, pretende aportar una solución para el envejecimiento activo y saludable de la población. En este proyecto final de máster se ha realizado el diseño del hardware necesario para la infraestructura del despliegue de los pilotos, que incluye los router SIM a utilizar y las alternativas comerciales de conectividad IoT por tarjetas SIM M2M, así como también se ha desarrollado la compilación y despliegue de la infraestructura de backend del asistente de voz utilizando Mycroft como asistente virtual de software libre y código abierto. En concreto, se ha desplegado el servidor Mycroft Selene, implementando la base de datos y las APIs definidas. Para ello se ha tenido en cuenta la seguridad de los datos, ya que en este tipo de aplicaciones la información privada de pacientes es crítica. Por otra parte, también se ha configurado la interfaz gráfica de usuario así como la administración remota de los dispositivos. Por último, se han analizado los resultados de las sesiones de prueba con usuarios reales.

Palabras clave: Asistente de voz, Salud mental, Soledad, Tecnología asistiva, Lenguaje natural, Escucha activa, Raspberry Pi 4, PostgreSQL, Redis.

Abstract

The DIAL project develops a tool to combat unwanted loneliness in elderly people. This tool, based on multiplatform voice assistive technologies, aims to provide a solution for the active and healthy aging of the population. In this final master's project, the design of the required hardware for the deployment infrastructure of the pilots has been carried out, which includes the SIM router to be used and the commercial alternatives of IoT connectivity by M2M SIM cards, as well as the compilation and deployment of voice assistant backend infrastructure using Mycroft as a free and open source virtual assistant. Specifically, the Mycroft Selene server has been deployed, implementing the database and the defined APIs. For this, data security has been taken into account, since in this type of application, private patient information is critical. On the other hand, the graphical user interface has also been configured as well as the remote administration of the devices. Finally, the results of the test sessions with real users have been analyzed.

Keywords: Voice assistant, Mental health, Loneliness, Assistive technology, Natural language, Active listening, Raspberry Pi 4, PostgreSQL, Redis.

Agradecimientos

Este apartado está dedicado a todas las personas que de alguna manera han colaborado en el desarrollo de este Trabajo Final de Máster ayudándome así a terminar mi máster en Ingeniería Biomédica.

Quisiera agradecer a mis tutores, a mis compañeros y a mi familia, puesto que son parte del desarrollo de este trabajo.

En primer lugar, quiero mostrar mi agradecimiento a mi cotutor Álvaro Fides por su tiempo, disponibilidad e implicación y a mi tutor Vicente Traver por darme la posibilidad de realizar este desarrollo dentro del proyecto DIAL. Gracias también a Juan José Lull por su breve pero no por ello menos importante contribución.

En segundo lugar, quiero darles las gracias a mis compañeros y amigos. A todos aquellos que de algún modo me ayudaron con este trabajo y que me sugirieron ideas o puntos de vista diferentes.

Por último, pero no menos importante, quisiera agradecer a mi familia. Gracias por escucharme y apoyarme en todas las decisiones que tomo. En especial a mi hermana María, por acompañarme todos los días y por ser un referente para mí.

Índice General

Resumen **3**

Lista de Tablas **7**

Lista de Figuras **7**

Lista de Acrónimos **9**

I. Memoria **10**

1. Introducción **11**

1.1. Motivación ……...……... 12
1.2. Objetivos……..……..... 12
1.3. Planificación del trabajo……... 13
1.4. Estructura de la memoria……. 14

2. Contexto social **16**

2.1. Envejecimiento saludable…… 16
2.2. Soledad no deseada ……..……..….. 19
2.2.1. Causas y factores de riesgo en personas mayores …..... 19
2.2.2. Diagnóstico…….…….. 20
2.2.3. Consecuencias……...…… 24

3. Tecnologías asistivas de voz multiplataforma **25**

3.1. Clasificación y evolución de las tecnologías asistivas…….. 25
3.2. Asistentes de voz…….. 27
3.3. Asistente virtual Mycroft……. 28
3.3.1. Arquitectura modular …… 28
3.3.2. Licencias…… …….. 30
3.3.3. Picroft…….…….. 31

4. Prototipo **32**

4.1. Hardware ……..…..... 32
4.2. Router SIM y plan de datos…….…… 34
4.3. Monitorización…….……... 35

5. Servidor Mycroft **40**

5.1. Servidor central……..... 40

5.2. Selene Backend ……..... 41

5.2.1. Base de datos: PostgreSQL ……...……. 42
5.2.2. Base de datos: Redis…….. …… 47
5.2.3. Configuración general de las APIs ……..… 55
5.2.4. Configuración de puertos de las APIs..…… 58
5.2.5. API de inicio de sesión único (SSO)...…….…….. 59
5.2.5.1. JSON Web Tokens…… 59
5.2.5.2. Salt…… 60
5.2.5.3. Acceso SSO mediante Github…….. 61
5.2.5.4. Sengrid……. 68
5.2.6. Servicios para las APIs……... 68
5.2.6.1. Servicio API SSO…… 68
5.2.6.2. Servicio API de inicio de cuenta……. 69
5.2.6.3. Servicio API de “market”…… …… 70
5.2.6.4. Servicio API de dispositivo (Device)…... 72

5.3. Interfaz gráfica de usuario…..... 73

5.4. Servidor Apache……..…..... 74

5.5. Docker……...……...……...… 78
5.5.1. Arquitectura Docker..……..…… 78
5.5.2. Archivos Dcoker de configuración……..…… 79
5.5.3. Instalación de Selene..……..…….. 80
5.5.4. Puesta en marcha…...……..……..… 80

6. Resultados y pruebas **83**

6.1. Sesión de pruebas iniciales con usuarios reales…… 83
6.2. Funcionamiento del prototipo…….…….. 87
6.3. Conclusiones……. 91
6.5. Líneas futuras…….…….. 92

II. Presupuesto **94**

ANEXO I. 97

Referencias **105**

Lista de Tablas

Tabla 1. Indicadores de salud……...21
Tabla 2. Consumo de datos para diferentes comandos ……..35
Tabla 3. Correspondencia de puertos con cada Raspberry……..36
Tabla 4. Funcionalidades evaluadas y resultados con usuarios reales…….85
Tabla 5. Enrutamiento de URLs…… 88
Tabla 6. Presupuesto de material para el proyecto y para el piloto…….94

Lista de Figuras

Fig. 1. Diagrama de Gantt…….…….....14
Fig. 2. Población mayor de 65 y de 80 años durante el periodo 1900-2068 en España ……..…… 17
Fig. 3. Media de edad de la población durante el periodo 2001-2020 en España…....17
Fig. 4. Evolución de la esperanza de vida al nacer entre hombres y mujeres durante el periodo 1991-2020 en España…….……..18
Fig. 5. Arquitectura modular del asistente de voz utilizando Mycroft……. 30
Fig. 6. Raspberry Pi 4 Model B..……..... 32
Fig. 7. Altavoz portátil EMEET OfficeCore Luna …… 33
Fig. 8. Tarjeta microSD de 32GB…… 33
Fig.9. Cobertura en España ofrecida por Things Mobile ...…….. 34
Fig.10. Conexión equipo remoto-Raspberry sin certificados añadidos…….37
Fig.11. Conexión equipo remoto-Raspberry con certificados añadidos……38
Fig.12. Script para el mantenimiento del túnel……..39
Fig. 13 Dispositivos de bloque……...41
Fig. 14 Disco rápido……...41
Fig. 15. Disco lento……41
Fig. 16. Versión de Python instalada……42
Fig. 17. Estado de PostgreSQL……..42
Fig. 18. Creación del entorno virtual para la base de datos……...44
Fig. 19. Archivo countryInfo.txt……44
Fig. 20. Archivo timeZones.txt……..44
Fig. 21. Archivo admin1CodesASCII.txt……...45
Fig. 22. Archivo cities500.zip……45
Fig. 23. Variables de entorno para la base de datos……...45
Fig. 24. Contraseñas seguras……..46
Fig. 25. Archivo ‘pg_hba.conf’……..46
Fig. 26. Archivo ‘postgresql.conf’……47
Fig. 27. Archivo ‘postgresql.conf’……47
Fig. 28. Perfiles de aplicaciones disponibles……48
Fig. 29. Estado de UFW……49

Fig. 30. Directiva “supervised”…….....50
Fig. 31. Servicio redis……50
Fig. 32. Comando ping……..50
Fig. 33. Comando set……....51
Fig. 34. Comando get……....51
Fig. 35. Persistencia de los datos en Redis……....51
Fig. 36. Enlace a localhost en Redis…….52
Fig. 37. Conexiones de red Redis……..52
Fig. 38. Token de usuario en Redis…….. 53
Fig. 39. Comprobación del token de usuario…….54
Fig. 40. Configuración de comandos susceptibles…….54
Fig. 41. Comandos renombrados……55
Fig. 42. Entorno virtual para la API de SSO……..56
Fig. 43. Entorno virtual para la API de cuenta……...56
Fig. 44. Entorno virtual para la API de “market”……...57
Fig. 45. Entorno virtual para la API de dispositivo……57
Fig. 46. Entorno virtual para la API “precise”……58
Fig. 47. Puertos en los que se ejecuta cada API……..58
Fig. 48. Claves RSA……59
9Fig. 49. Archivos finales que contienen las claves ..……59
Fig. 50. JWT de acceso……...60
Fig. 51. Clave “salt”……61
Fig. 52. Archivo Gemfile en root ……...62
Fig. 53. ID de cliente para la app ……63
Fig. 54. Configuración de la app SSO ……64
Fig. 55. Variables de entorno que contienen las claves ……..64
Fig. 56. Test de funcionamiento de la API ……65
Fig. 57. Primer paso. Log in en cuenta Github ……...65
Fig. 58. Verificación de email ……66
Fig. 59. Autenticación persistente ……..66
Fig. 60. API Sengrid ……...68
Fig. 61. Servicio “systmd” para la API SSO ……..69
Fig. 62. Proceso de implementación de skills ……71
Fig. 63. Versión Angular instalada ……74
Fig. 64. Perfiles por defecto Apache en UWF ……74
Fig. 65. Perfil “Apache” ……75
Fig. 66. Comprobación del servicio Apache en el navegador ……75
Fig. 67. Archivo “environment.test.ts” ……...76
Fig. 68. Archivo de configuración para cada uno de los “sites” ……77
Fig. 69. Vistas “account” y “login” con Apache ……78
Fig. 70. Autenticación por GitHub ……78
Fig. 71. Arquitectura Backend y Frontend con despliegue Docker ……79
Fig. 72. Perfil de usuario ……84
Fig. 73. Muestra representativa tanto de hombres como de mujeres ……85

Fig. 74. Reinicio remoto de Raspberry ……88
Fig. 75. Vistas /login, /new account, política de privacidad, términos de uso y /new ..89
Fig. 76. Vistas “/skills” y “/skills/*……90
Fig. 77. Vistas “/dashboard” y “/profile”……...90
Fig. 78. Vistas “/devices” y “/devices/add”…….. 90

LISTA DE ACRONIMOS

API	Application Programming Interfaces
GUI	Graphical User Interface
IoT	Internet Of Things
JWT	JSON Web Token
M2M	Machine To Machine
OMS	Organización Mundial de la Salud
SIM	Subscriber Identity Module
SSO	Single Sign On
UI	User Interface
UPV	Universidad Politécnica de Valéncia

I. MEMORIA

Capítulo 1

Introducción

Los avances tecnológicos en materia de las ciencias de la salud han modificado nuestra sociedad y forma de vida. Continuamente aparecen nuevas tecnologías destinadas a diagnosticar y tratar problemas de salud.

En este marco, el proyecto se centrará en la soledad no deseada como uno de los problemas que más preocupan entre las personas mayores. La población mundial está envejeciendo por lo que hay una necesidad creciente de tener a nuestra disposición herramientas que nos permitan asistir y cuidar de nuestros familiares ancianos o con necesidades de asistencia.

La soledad no deseada nos afecta a todas las personas en algún momento de nuestras vidas, a cualquier edad, desde la infancia hasta la vejez. Puede aparecer a raíz de experiencias vitales concretas y se trata como un problema cuando genera aislamiento social y sentimiento de soledad. Hablamos de soledad no deseada cuando esta situación no se escoge, sino que se impone a pesar de nuestra voluntad y perdura en el tiempo, pudiendo afectar a nuestro bienestar y estado de salud.

Con el objetivo de paliar la soledad no deseada que sufren muchas personas mayores se pone en marcha el proyecto DIAL. En este proyecto, la solución tecnológica que se va a desarrollar está inspirada en la radio, un medio con el que las personas mayores se sienten muy familiarizadas puesto que las personas que actualmente tienen más de 70 años han tenido relación a lo largo de su vida con la idea del “altavoz que les habla y les informa”. En concreto, se desarrollará una solución amigable e innovadora que está basada en Tecnologías Asistivas de Voz Multiplataforma y que facilitará la vida independiente de las personas mayores. Mediante conversaciones con la persona mayor, será capaz de detectar cuándo ésta está sufriendo una situación de soledad no deseada o está en riesgo de padecerla. En tal caso, el sistema alertará al o la profesional de la atención sociosanitaria correspondiente.

1.1. Motivación

En los últimos años, el análisis de las condiciones de vida de las personas mayores se ha convertido en un tema prioritario. Debido a los cambios demográficos que vienen sucediendo en nuestras sociedades, tales como una mayor esperanza de vida y longevidad, cada vez se va teniendo más en cuenta la salud mental en este periodo de la vida.

Una de las características principales de la Cuarta Edad es la pérdida de capacidades que posibilitan vivir una vida plena. Esta nueva etapa suele estar asociada a enfermedades crónicas y, en algunos casos, hasta discapacidades, lo que supone que esta población se convierte mayoritariamente en dependiente de atenciones y cuidados especiales.

Estos datos indican que existe y va a existir de manera creciente, un necesidad clara y real de tener una atención especial que ayude a estas personas en su día a día.

Ante este escenario, nos planteamos cómo poner nuestros conocimientos sobre asistentes de voz al servicio de la gerontecnología.

Además, la lucha contra la soledad no deseada no es lo único que se pretende, también se quiere reducir la brecha digital entre los mayores y facilitar el deseo que muchas de estas personas tienen de seguir viviendo en sus propias casas durante su vejez.

1.2. Objetivos

Este trabajo tiene el propósito de introducir la problemática de la soledad no deseada, los distintos actores que pueden estar implicados en ella y pretende desarrollar una herramienta para su mitigación.

Este desarrollo dentro del proyecto DIAL está orientado a implementar una herramienta para combatir la soledad no deseada en las personas mayores. Esta herramienta basada en tecnologías asistivas de voz multiplataforma, pretende aportar una solución para el envejecimiento activo y saludable de la población.

El trabajo presentado tiene como objetivos principales el diseño del hardware necesario para llevar a cabo el prototipo de asistente virtual y el despliegue de la infraestructura de red en servidores propios de la Universidad Politécnica de Valencia (UPV) para implementar el backend con el que se administran los usuarios y dispositivos. Los asistentes de voz comerciales no son compatibles con el requisito de DIAL de ejecutar los servicios en servidores propios por motivos de seguridad y privacidad, de ahí el despliegue del backend en servidores propios, así como la utilización de herramientas de código abierto.

Se ha elegido Mycroft como asistente virtual de software libre y código abierto por su arquitectura modular que nos permite un mayor grado de libertad al ser más personalizable que otras opciones encontradas de código abierto. El trabajo se centrará particularmente en los módulos de servidor de Mycroft, que en la arquitectura DIAL se definen como “Configuration Backend (Mycroft Home & API)".

Con el fin de desarrollar esta herramienta, se han establecido los siguiente objetivos:

- **Objetivo 1**: Análisis de la soledad no deseada y el impacto en la sociedad.
- **Objetivo 2:** Estudio de las tecnologías asistivas y los diferentes asistentes de voz de libre distribución que podemos encontrar.
- **Objetivo 3**: Diseño del prototipo para el despliegue de los pilotos, realizando el correspondiente estudio de mercado, que incluye los router SIM a utilizar y las alternativas comerciales de conectividad IoT por tarjetas SIM M2M. Debido a la imposibilidad de implementar de manera real el prototipo final para el desarrollo de los pilotos que se realizará fuera de plazo de este trabajo, el prototipo desarrollado deberá ser funcional con la debida arquitectura funcionando correctamente.
- **Objetivo 4**: Compilación y despliegue de la infraestructura de backend del asistente de voz utilizando Mycroft en servidores propios de la UPV, implementando la base de datos y las APIs definidas.
- **Objetivo 5:** Garantizar la seguridad de los datos, ya que en este tipo de aplicaciones la información privada de pacientes es crítica.
- **Objetivo 6:** Implementación y configuración de la interfaz gráfica de usuario
- **Objetivo 7:** Administración remota de los dispositivos.
- **Objetivo 8:** Análisis de los resultados de las sesiones de prueba con usuarios reales y pruebas de funcionamiento. Identificar las necesidades y barreras de los usuarios en la interacción con la tecnología asistiva de voz.

1.3. Planificación del trabajo

Este proyecto sigue la metodología del ***double diamond***, que identifica cuatro áreas principales del proceso de diseño y nos permite:

1. Realizar una investigación estable para apoyar las decisiones del proyecto, esencial en usuarios con necesidades tan especificas
2. Concretar objetivos
3. Realizar un prototipo acorde a las necesidades reales de los usuarios

Se busca entender los determinantes de salud, causas y factores de la soledad no deseada en personas mayores para poder desarrollar una herramienta de ayuda adaptada. Dentro del proyecto DIAL, la parte del desarrollo que corresponde en este trabajo se validará en un entorno de laboratorio. También se realizará un estudio inicial con usuarios reales con el objetivo de obtener información sobre la experiencia con los asistentes de voz Alexa y Google Assistant. Las pruebas de cara a la realización del piloto se realizarán sobre la versión final del prototipo, donde se validará la solución con usuarios reales.

Para ver un desglose detallado de las tareas llevadas a cabo, se incluye el siguiente diagrama de Gantt.

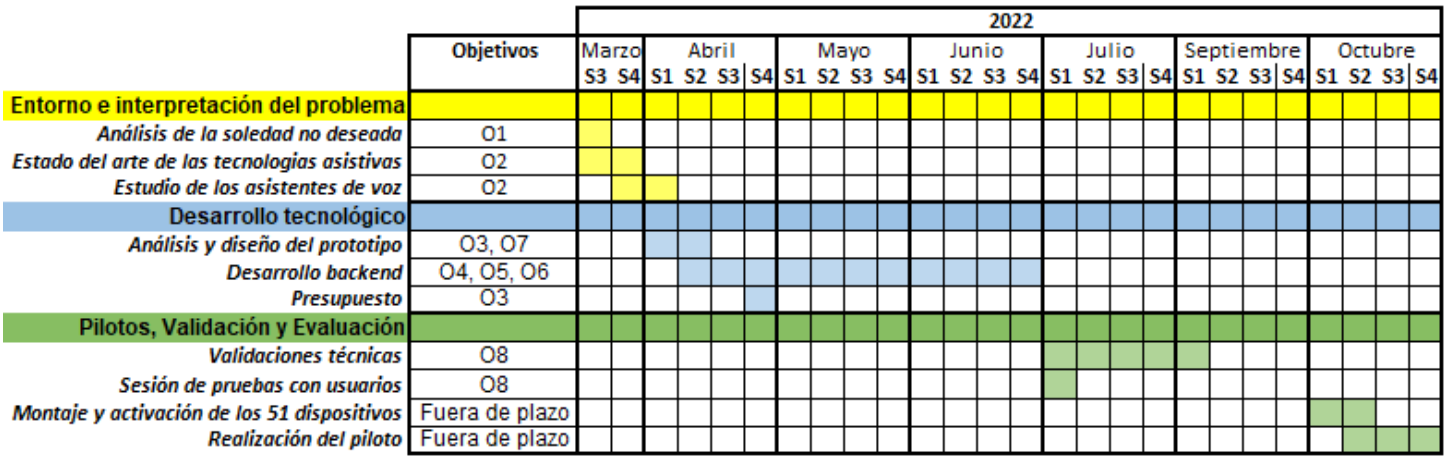

Fig. 1. Diagrama de Gantt

La ejecución del piloto comienza en octubre del 2022 terminando en septiembre de 2023. Los desarrollos o mejoras correspondientes a subsiguientes prototipos se encuentran fuera de plazo de este trabajo fin de máster.

1.4. Estructura de la memoria

A lo largo del documento se hace una recopilación de información proveniente de informes, estudios, papers, publicaciones y otra tipología de recursos que son referenciados al final del mismo.

La secciones del documento están estructuradas de la siguiente forma:

Capítulo 2. Contexto social.
Este capítulo describe la tendencia del envejecimiento poblacional en España e introduce la problemática de la soledad no deseada y los distintos factores implicados en ella. Analiza las consecuencias en las personas mayores y trata de abordar el problema para ofrecer el tratamiento adecuado.

Capítulo 3: Tecnologías asistivas de voz multiplataforma.
A lo largo de este capítulo se realiza una revisión del estado del arte de las tecnologías asistivas. Se introducen conceptos como escucha activa, asistente de voz o “skills”. Además, se describen distintas soluciones, entre las cuales se ha seleccionado Mycroft como asistente a implementar en este proyecto.

Capítulo 4: Diseño del prototipo.
A lo largo de este capítulo, se realiza el diseño hardware del dispositivo con el correspondiente estudio, que incluye los router SIM a utilizar y las alternativas comerciales de conectividad IoT por tarjetas SIM M2M.

Capítulo 5: Servidor Mycroft.
El despliegue backend para la administración de los dispositivos con su respectiva interfaz de usuario es necesario realizarlo en servidores propios de la UPV, sin que pasen por terceros. Este capítulo contiene el desarrollo de la infraestructura de red backend, que incluye entre otros, la base de datos, las APIs y la interfaz de usuario.

Capítulo 6: Resultados y pruebas.
Este capítulo presenta los resultados obtenidos de las primeras sesiones de prueba con usuarios reales, así como el correcto funcionamiento del prototipo y de las funcionalidades implementadas en el backend.

Capítulo 7: Conclusiones.
Esta sección final describe los detalles de implementación del piloto, el resumen del presupuesto final, conclusiones y futuras líneas de trabajo.

Capítulo 2

Contexto social

2.1. Envejecimiento saludable

El envejecimiento de la población está a punto de convertirse en una de las transformaciones sociales más significativas del siglo XXI, con consecuencias en la mayoría de países del mundo, incluido España. La población está envejeciendo y este envejecimiento demográfico es una tendencia mundial.

En nuestro país, la población mayor de 65 años supone 9 millones de personas, es decir, el 19.3% de la población total.

Esta tendencia se puede ver a través de diferentes indicadores estadísticos como la evolución de la *proporción de la población mayor* y la *media de edad* en la población.

En primer lugar, la Organización Mundial de la Salud (OMS) define al adulto mayor a partir de los 65 años en los países en vías de desarrollo, y al grupo que llega a los 80 años como la cuarta edad. Si observamos la evolución de la proporción de personas mayores en la población: en 1991, cerca de 5 millones y medio de personas tenía 65 años o más, frente a 9 millones en 2019, lo que supone un aumento considerable. Si nos fijamos más concretamente en el grupo de 80 años o más, su cuota era de casi el 1 millón y medio de personas en 1991, mientras que en 2019 era del 3 millones de personas, lo que significa que se ha prácticamente duplicado durante este periodo. En el período de 1900 a 2018 los datos son reales, sin embargo, observando las tendencias mundiales, de 2028 a 2068 se han realizado proyecciones. Según la proyección del INE (2018-2068), en 2068 podría haber más de 14 millones de personas mayores, aproximadamente el 30% del total de una población [1].

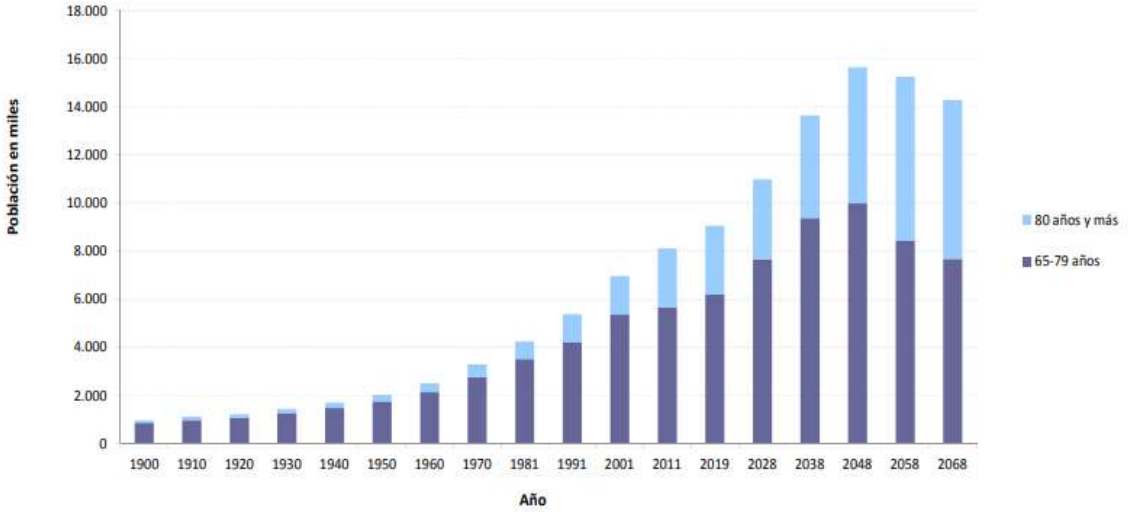

Fig. 2. Población mayor de 65 y de 80 años durante el periodo 1900-2068 en España

Otra forma de analizar el envejecimiento de la sociedad en la UE es observar la edad media de la población. La media de edad ha aumentado en el periodo 2001-2020: era de 37,6 años en 2001, 39,9 años en 2010 y 44,3 años en 2020 [2]. Esto supone un aumento de 6,7 años en la media durante este periodo en España.

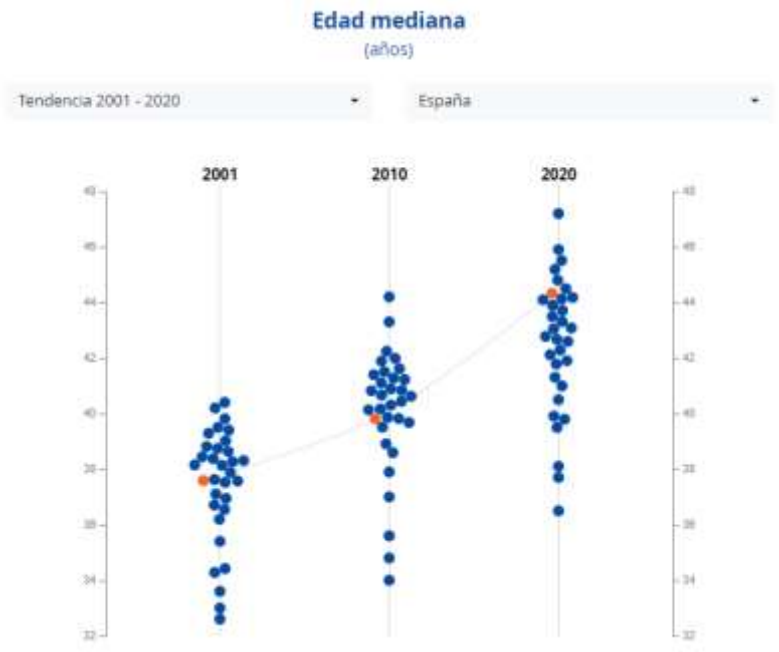

Fig. 3. Media de edad de la población durante el periodo 2001-2020 en España

Este envejecimiento demográfico en España está causado por diversos factores, entre ellos la disminución de la mortalidad y morbilidad, el aumento de la esperanza de vida y el descenso de la natalidad. Las mejores condiciones de vida, la mejora de la sanidad pública y los avances de la tecnología médica entre otros factores, han producido un fuerte incremento en la esperanza de vida de los españoles, elevándose hasta los 82,34 años de media en 2021 [3].

Sin embargo, observamos una esperanza de vida diferente entre hombres y mujeres de hasta 6 años superior en el caso de las mujeres, diferencia que se mantiene prácticamente constante durante todo el periodo de 1991 a 2020.

Fig. 4. Evolución de la esperanza de vida al nacer entre hombres y mujeres durante el periodo 1991-2020 en España

Por todo lo anterior, las condiciones de vida de las personas mayores se han convertido en un tema prioritario. Debido a los cambios demográficos que vienen sucediendo en nuestras sociedades, cada vez se va teniendo más en cuenta la salud en este periodo de la vida.

Adicionalmente, también es interesante resaltar que estos "años ganados" deben ser lo más saludables posibles, teniendo un envejecimiento saludable en la manera de lo posible. La población femenina es más longeva que la masculina por lo que tiene una tendencia a padecer un "envejecimiento no tan saludable". Una de las características principales llegada esta etapa es la pérdida de capacidades que posibilitan vivir una vida plena. Esta nueva situación suele estar asociada a enfermedades crónicas y, en algunos casos, hasta discapacidades, lo que supone que esta población se convierte mayoritariamente en dependiente de atenciones y cuidados especiales. Todo indica que existe y va a existir una influencia clara y creciente en la salud mental de los mayores.

Por último, si observamos los porcentajes sobre hogares unipersonales, según los datos del Instituto Nacional de Estadística (INE) publicados en 2018, vemos que en España había 4.732.400 personas viviendo solas en dicho año. De esta cifra, 2.037.700 (un 43,1%) tenían 65 o más años. Y, de ellas, 1.465.600 (un 71,9%) eran mujeres. De ellos, más de 850.000 personas tienen 80 o más años. Unos 112.000 mayores de 80 sin compañía más que en 2013 [4]. Por lo tanto, más viejos y más solos. Y aunque existe un creciente interés en estudiar la prevalencia y los efectos perjudiciales de la soledad en la vejez, la comprensión total del fenómeno está lejos de ser completa. Parece que la soledad afecta los resultados de salud a través de vías conductuales, psicológicas y biológicas. Por todo lo anterior, observamos que es necesario potenciar y proveer de una atención especial que ayude a estas personas en su día a día.

2.2. Soledad no deseada

La Organización Mundial de la Salud considera la soledad no deseada una cuestión de salud pública, porque puede tener efectos importantes en la salud de las personas: deterioro cognitivo, depresión, pérdida de movilidad, enfermedades cardiovasculares y mortalidad temprana, entre otros.

Las relaciones sociales "amortiguan el estrés", ya que proporcionan recursos (informativos, emocionales o materiales), que promueven respuestas neuroendocrinas adecuadas, a factores estresantes agudos o crónicos (enfermedad, eventos de la vida, transiciones de la vida). La soledad es un problema cuando genera aislamiento social y sentimiento de soledad. Hablamos de soledad no deseada cuando esta situación no se escoge, sino que se impone a pesar de nuestra voluntad y perdura en el tiempo, pudiendo afectar a nuestro bienestar y estado de salud [5].

La soledad tiene que ver con lo social y emocional, y se puede dar en jóvenes, adultos y mayores. La soledad no deseada nos afecta a todas las personas en algún momento de nuestras vidas, a cualquier edad, desde la infancia hasta la vejez. Puede aparecer a raíz de experiencias vitales concretas, habitualmente, relacionadas con pérdidas como la muerte de un ser querido, desempleo o rupturas. También, es frecuente en situaciones de dependencia (infancia, vejez, procesos de enfermedad…) o en situaciones vitales estresantes. A veces, aparece de forma más insidiosa, gradual e invisible.

Ya denominada como "la epidemia del siglo XXI", la soledad no deseada afecta a la salud y a la mortalidad. Durante la pandemia se registró un aumento de noticias sobre este problema de salud pública, pero ¿cómo se ataja el problema?

Gracias a la solución diseñada en este trabajo, podemos mitigar este problema. El asistente que aquí proponemos se engloba dentro de un proyecto más amplio cuyo objetivo es revertir esta realidad y facilitar el diagnostico de esta situación que sufren millones de personas mayores. Además, ofrece la posibilidad de facilitar acceso a otras actuaciones o apoyo personalizado en las situaciones que así lo requieran.

El análisis completo de la soledad no deseada se realiza en los siguientes apartados.

2.2.1 Causas y factores de riesgo en personas mayores

Las causas de la creciente soledad no deseada vienen derivadas principalmente del aumento de personas viviendo solas sin elección, el paro y la precariedad en el empleo, la baja autoestima, familias despreocupadas, la frenética vida en las grandes ciudades y la tendencia a relaciones personales menos duraderas, y en el caso de las personas mayores tienen una tendencia mayor a desarrollar esta soledad debido a sus características específicas. Podemos diferenciar varios tipos de soledad [17]:

- *La soledad emocional* es la ausencia de relaciones de apego, es decir, relaciones especialmente significativas para la persona y que proporcionan una base segura. Asociada con sensaciones de vacío y el deseo de alguien especial con quien compartir la vida.

- *La soledad social* es una carencia de relaciones interpersonales sanas que produce un sentimiento de marginalidad, la sensación de no ser aceptado por los otros, aislamiento y aburrimiento. Se desea fervientemente tener un lugar dentro de un grupo de personas con las que se puedan compartir intereses y preocupaciones.
- *Soledad Familiar* es cuando tenemos falta de apoyo familiar y no cuidamos a la familia como valor necesario.
- *Soledad conyugal* hace referencia a la ausencia de sentimientos de amor en la pareja.
- *Soledad Existencial* elude a aquella soledad cuando hemos perdido el sentido de vida, y no tenemos un para qué.

Con respecto a los factores de riesgo que pueden llevar a generar situaciones de soledad, son muchos y muy variados, entre ellos, podemos mencionar los siguientes:

- Fallecimiento de la pareja u otros familiares.
- Disminución y/o pérdida de las relaciones de amistad.
- Jubilación.
- Problemas de salud.
- Cambios en las rutinas y estilo de vida.
- Falta de apoyo afectivo por parte de los familiares o pareja.
- Traslados de vivienda.
- Estigma.
- Insatisfacción con las relaciones sociales.
- Pobreza.
- Situaciones de dependencia.
- Déficit de habilidades sociales.
- Paro y precariedad en el empleo.
- Inseguridad y vulnerabilidad.
- Baja autoestima.
- Limitaciones físicas.
- Pérdida de un rol en la vida.
- Negación al afecto, ocio y compañía.
- Limitaciones por la edad.

Los Sistemas públicos de Servicios Sociales tienen una labor muy importante a la hora de prevenir la soledad, aislamiento y exclusión, por lo que deberían de ser la primera fuente para identificar este problema.

2.2.2 Diagnóstico

A la hora de diagnosticar este problema, contamos con ciertos indicadores de salud que ayudan a identificar la soledad no deseada. Además, existen varios marcos establecidos para medir este problema.

<u>Indicadores de salud:</u>

Red social	
	Tamaño
	Diversidad
	Frecuencia de interacción
	Asistencia a eventos grupales
	Cantidad de tiempo dedicado a socializar
	Número de llamadas de teléfono realizadas
Características del usuario	
	Tipo de personalidad
	Debilidades
	Experiencias
	Aficiones
	Recursos
	Intereses
	Medicación
	Cuidado de mascotas
Estilo de vida	
	Rutinas
	Nivel de actividad física
	Alimentación
	Autocuidado
	Higiene
	Sueño
	Residencia
Vida familiar	
	Pareja
	Familia cercana
	Hijos

Tabla 1. Indicadores de salud.

Escalas y métricas:

- *Escala de soledad de UCLA revisada*: consta de 10 preguntas, puntuables entre 1 y 4 puntos, lo que permite obtener una puntuación mínima de 10 y máxima de 40. De esta forma podemos clasificar el nivel de soledad según la puntuación obtenida.

- *Escala de soledad de De Jong Gierveld*: se fundamenta en que la soledad es la discrepancia entre las relaciones sociales deseadas y las que efectivamente se tienen, es decir, en el modelo de la discrepancia cognitiva, donde a mayor diferencia entre lo esperado y la realidad, mayor es la soledad experimentada.
- *Métricas de una sola pregunta* ("single question metrics").

El tratamiento o procedimientos posteriores a la detección de la soledad no deseada queda fuera de este trabajo final de máster ya que, dentro del proyecto DIAL, un equipo de psicólogos está desarrollando las técnicas efectivas para tratar este problema una vez detectado. El asistente virtual que se desarrolla, interactúa con el usuario mediante diálogos donde diariamente va recolectando información. Esta información, en base a los identificadores de salud, va generando un perfil del individuo para obtener un diagnóstico. El asistente virtual no solo detecta, sino que también propone al usuario cambios en su rutina, métodos o actividades acorde a mejorar sus relaciones sociales.

Esta posterior intervención tiene el objetivo de reducir el aislamiento social y sentimiento de soledad mediante actividades que fomenten rutinas saludables, la creación de nuevos vínculos o fomentando que las personas mantengan las relaciones existentes. Se utilizarán enfoques psicológicos para cambiar las percepciones de las personas que sufren dichas condiciones. También fomentará las intervenciones grupales, con actividades de apoyo y educativas, mejorando las habilidades sociales y desarrollando una buena autoestima.

A continuación, se incluyen algunos diálogos de muestra en proceso de desarrollo, donde se tratan aspectos de la vida diaria de estas personas para obtener datos clave para su diagnóstico:

Diálogo 1 "Buenos Días":

USUARIO: Buenos días DIAL

DIAL: Buenos días {nombre}, / Hoy es..... / San ..... / El tiempo.....

D: ¿Cómo has dormido? / Me gustaría saber cómo has dormido

U: Bien

D: Me alegro.....

U: Como siempre

U: Mal

D: Por alguna razón has dormido mal/como siempre

U: Por malestar emocional / Por malestar físico / Por otra cosa

D: Si tienes malestar físico (Registramos si se repite mucho el malestar y es continuo, pautas)

D: Es posible que tengas una forma de aliviarlo

Si tienes ……

U: Si. Voy a preguntar

U: No. No hace falta

D: (Si dice que No tiene: pautas)

D: (Si dice algo nuevo, seguir con:) ¿Contemplas pedir cita al médico?

U: Si

U: No hace falta

D: Recuerda tener la tarjeta del SIP a mano.

D: (Determinar si Dial va a llamar o te recuerda el número de teléfono)

D: Es importante que el malestar emocional que sentimos, podamos compartirlo.

¿Querrías hablarlo con alguien?

U: Sí

U: No. No es necesario

D: Con alguna amistad? Con un familiar? Con Laura (psicóloga)?

D: ¿Quieres que te pongamos en contacto nosotros?

…….

Diálogo 2: "Higiene"

D: Sabes que la ducha es buena ¿Cuándo la tienes prevista?

U: Luego

D: Entonces lo tienes en cuenta. ¿Quieres que te lo recuerde más tarde, por ejemplo en una hora?

U: Si / No hace falta / Mañana

D: Hoy no lo tienes previsto, mañana hablamos de ello.

(Al día siguiente)

D: Me dijiste ayer que hoy te ducharías.

U: Ya lo he hecho

D: Me alegro que ya lo has tenido en cuenta porque la ducha es buena para nuestro bienestar.

U: Todavía No

D: Lo tienes en cuenta

U: Ya veremos

D: Por alguna razón tienes en duda ducharte hoy?

U: Me da pereza / Tengo miedo a caerme / Con un día a la semana es suficiente

D: Me dices que tienes dificultades.

D: Sería conveniente hablar de ello. Laura contactará contigo.

…….

2.2.3. Consecuencias

La soledad no deseada puede afectar a nuestra salud, ya que con cierta frecuencia afecta a nuestras rutinas de autocuidado (sedentarismo, tabaquismo, dieta poco sana, horarios desajustados…).

Desde la depresión, hasta el suicidio, enfermedades cardiovasculares, deterioro cognitivo, obesidad, hipertensión, cáncer, artritis, alzheimer, caídas y roturas de huesos. La soledad afecta a la salud y a la calidad de vida de las personas mayores, y por lo tanto, es un factor de riesgo a la depresión y mortalidad temprana. También va a conllevar problemas económicos y repercutir en la alimentación y su higiene [17].

La soledad es un importante y serio problema de salud pública porque las relaciones sociales son centrales en el bienestar de las personas y son necesarias para el mantenimiento de una buena salud.

Capítulo 3

Tecnologías asistivas de voz multiplataforma

Las tecnologías asistivas son productos o servicios utilizados por personas con dificultades para realizar tareas con el objetivo de mejorar su calidad de vida y su integración en la comunidad. Estas tecnologías de apoyo pueden o no ser de índole tecnológico.

El proyecto DIAL se llevará a cabo implementando como tecnología asisitva de voz, un asistente virtual de código abierto, el cual se modificará para adaptarlo a los objetivos de la soledad no deseada. La solución tecnológica está inspirada en la radio, un medio con el que las personas mayores se sienten muy familiarizadas puesto que las personas que actualmente tienen más de 70 años han tenido relación a lo largo de su vida con la idea del "altavoz que les habla y les informa". De ahí que el proyecto no solo tome el nombre de DIAL por su referencia a la radio sino porque la herramienta a desarrollar estará inspirada en ella.

3.1. Clasificación y evolución de las tecnologías asistivas

En los últimos años encontramos que el desarrollo de la industria de dispositivos biomédicos se caracteriza por los grandes avances en el hardware, mayor número de prestaciones al incorporar elementos innovadores, y sobre todo la capacidad de conexión de unos sistemas con otros. Paralelamente a la evolución del hardware, también se ha ido mejorando en prestaciones y posibilidades el software. El incremento de desarrollos y aplicaciones de software se ha ido extendiendo a todos los sectores, incluyendo el sector de la salud.

Surge entonces la necesidad de adaptar soluciones tecnológicas que inicialmente se habían dirigido a un sector de población, a personas con dificultades en el manejo del ordenador, bien por limitaciones cognitivas, sensoriales o físicas, o incluso por actitud negativa.

Así, se ha ido incrementando de este modo la investigación y el desarrollo de productos dirigidos hacia las personas con discapacidad y frente a las necesidades del envejecimiento de la población. Estas tecnologías asistivas, de forma general, se pueden clasificar en base a diferentes criterios.

El nivel tecnológico:

- No tecnología (usos especiales de métodos y objetos comunes).
- Baja tecnología (adaptación de herramientas ya existentes).
- Media tecnología (productos y equipos de cierta complejidad tecnológica).
- Alta tecnología (productos y equipos de gran complejidad tecnológica).

Las características de su fabricación:

- Desarrollo específico.
- Adaptación de un producto ya existente.

Las características de los usuarios, dependiendo de si su dificultad es:

- Física o motora.
- Psico-cognitiva.
- Sensorial.
- Declives propios de la edad.

La lógica de operación:

- Alternativas, que sustituyen una metodología o herramienta por métodos alternativos que si puede utilizar el individuo.
- Aumentativas, que complementan la falta de recursos de un individuo para realizar determinadas tareas.
- Sustitutivas, que permiten sustituir el uso de una funcionalidad ausente o dañada.

El área de la vida diaria a asistir:

- Sistemas de habilitación, aprendizaje y entrenamiento.
- Sistemas alternativos y aumentativos de acceso a la información del entorno.
- Tecnologías de acceso al ordenador, contempla los sistemas (hardware y software).
- Sistemas alternativos y aumentativos de comunicación.
- Tecnologías para la movilidad.
- Tecnologías para la manipulación y el control del entorno.
- Tecnologías de la rehabilitación y fisioterapia.
- Tecnologías asistenciales.
- Tecnologías para el deporte, ocio y tiempo libre.
- Tiflotecnologías, o tecnologías para la lectura de personas con disminución visual.

3.2. Asistentes de voz

Existe una amplia variedad de asistentes virtuales en el mercado, pero ninguno orientado a mejorar la salud de las personas mayores.

Los asistentes de voz que se pueden adquirir por el público en general son comercializados por las típicas grandes compañías tecnológicas. Por lo general, los asistentes de voz son accesibles en dos formatos: 1) A través del móvil, como una aplicación integrada en el propio sistema operativo o 2) a través de un "altavoz inteligente", que es un dispositivo que incorpora el software del asistente además del hardware mínimo para su ejecución, incluyendo un micrófono y un altavoz (y en algunos casos recientes, una pantalla).

En ambos casos, el dispositivo del usuario actúa como mero interfaz de audio mientras que el procesamiento de voz y los servicios que se prestan se ejecutan en los servidores de la empresa. Por este motivo, los asistentes de voz comerciales no son compatibles con el requisito de DIAL de ejecutar los servicios en servidores propios por motivos de seguridad y privacidad. Por ello, en DIAL se usarán herramientas de código abierto y se descartarán las soluciones comerciales, por su invasión en la privacidad de los usuarios.

A continuación, en este contexto, se detalla una clasificación de los asistentes de voz no comerciales más relevantes que cubren distintas necesidades. Destacar que, según lo investigado, no existen muchas opciones de libre distribución para implementarse fácilmente. En general, son relativamente recientes por lo que no tienen las mismas funcionalidades que los comerciales. La ventaja principal es la posibilidad de desplegar toda su infraestructura en servidores locales, sin necesidad de pasar por servidores de terceros.

- **Mycroft**: Uno de los primeros y más populares hasta el punto de comercializar su propio hardware de asistente, aunque puede instalarse en hardware genérico, como la Raspberry Pi. Una de las ventajas de Mycroft es su composición modular, que permite conectarlo con otros módulos de código abierto para personalizarlo, incluyendo soporte para varios idiomas o incluso conectarlo con soluciones comerciales. Mycroft tiene una estructura similar a los asistentes comerciales y ofrece servicios similares, también organizados en Skills. Distintos módulos tienen distintas licencias de código abierto, pero por lo general son permisivas [6].
- **Leon**: Una de las características principales de Leon es que está orientado principalmente a servidores. Es decir, el sistema se instala "en la nube" o "backend" y los clientes se conectan a sus servicios mediante APIs. Leon no ofrece estos clientes (únicamente una interfaz web de ejemplo), todo se ejecuta en el servidor. Correspondería a DIAL desarrollar esta parte. Aun así, Leon permite la ejecución en modo offline, sin interacción con servidores externos. En cuanto a idiomas, Leon acepta inglés y francés para activarlo, mientras que la interacción hablada se basa en Mozilla Deepspeech para el soporte de otros idiomas. La licencia de código abierto es MIT por lo que es permisiva [7].

- Rhasspy: De manera similar a Mycroft, Rhasspy se basa en una colección modular de servicios, por lo que también da la posibilidad de personalizarlo y conectarlo a servicios de lenguaje compatibles con varios idiomas. Sin embargo, Rhasspy no se basa en Skills y los servicios que se ofrecen al usuario deben conectarse a través de APIs. Rhasspi también ofrece el software de la parte cliente para instalarse en dispositivos Raspberry Pi. Se ofrece bajo licencia MIT permisiva [8].
- LinTO: Financiado por el gobierno de Francia, LinTO soporta inglés y francés, y se espera compatibilidad con español pronto. La arquitectura cubre tanto los módulos desplegados en el servidor como varias opciones para clientes. La licencia es AGPL, por lo que puede imponer ciertas limitaciones. [9].
- Jasper: Otro proyecto popular aunque hace años que no se actualiza, es Jasper que se ejecuta completamente dentro de una Raspberry Pi, sin conectarse a servidores (excepto para servicios/Skills que lo requieran) por lo que es más limitado. Ofrece compatibilidad con otros idiomas, incluido español, a través de CMUSphinx. La licencia es la permisiva MIT [10].

3.3. Asistente virtual Mycroft

De los anteriores asistentes mencionados, se ha elegido Mycroft por ser el que mejor puede adaptarse a las necesidades de DIAL. No solo es uno de los primeros y más populares, también es uno de las más activos y constantes además de tener un plan de negocio para sostenerse. Esto hace prever que garantizará la existencia de soporte durante la duración total del proyecto. Además, nació para ser usado con GNU/Linux y puede funcionar en Raspberry Pi de forma sencilla.

3.3.1. Arquitectura modular

Su arquitectura es la más parecida a la de los asistentes comerciales, además de ser completamente modular, por lo que las opciones de personalización deberían ser suficientemente amplias para los objetivos del proyecto. Tiene además la mayor compatibilidad de opciones en los sistemas de reconocimiento y síntesis de texto. Esto facilitará a DIAL implementar y ofrecer soporte en castellano y valenciano. Sus distintas funcionalidades están asignadas a módulos independientes que pueden sustituirse por diferentes implementaciones. En concreto, Mycroft se divide en los siguientes módulos:

-Mycroft Core: Es el "middleware" que se encarga de coordinar y comunicar los diferentes módulos del sistema. Al formar el núcleo del mismo sistema, no puede sustituirse por otra alternativa, es el corazón del sistema Mycroft.

-Mycroft Home/API: Es el "backend" con el que se administran los usuarios y dispositivos. Mycroft le da el nombre de "Selene". Idealmente, se trata de una aplicación web para la administración de los asistentes Mycroft, y por defecto está desplegada en los servidores de Mycroft. En nuestro caso, este módulo debe desplegarse en servidores propios- Está compuesto por Mycroft Home (la aplicación web) y Mycroft API (interfaz para la comunicación con los asistentes).

-Wake Word: Este es el sistema de detección de la palabra de activación. De manera similar a como se debe invocar a "Alexa" o "Hey Google", Mycroft escucha continuamente intentando detectar su palabra de activación ("Hey Mycroft" por defecto). Se utilizará Precise, el motor por defecto de Mycroft, entrenado con una palabra de activación propia de DIAL.

-Speech-To-Text: Abreviado como STT, es el sistema que se encarga de traducir la voz grabada del usuario a texto que pueda ser analizado por el sistema. Este módulo no está grabando continuamente, sino que sólo se activa durante un corto tiempo tras detectarse la palabra de activación para poder interpretar las órdenes del usuario. Se utilizará Mozilla Deepspeech, que es compatible con Mycroft, entrenado para entender castellano y catalán con los datasets de Mozilla Common Voice de estos idiomas, potencialmente añadiendo datasets propios grabados por DIAL.

-Text-To-Speech: Abreviado como TTS, es el sistema complementario a STT. Se encarga de traducir las respuestas generadas por Mycroft de texto a voz sintética. Traduce el texto de la respuesta de la Skill a audio. Se utilizará MaryTTS. Al no disponer de síntesis en castellano o catalán de manera nativa, se desarrollará una en DIAL usando como base los datasets de Mozilla Common Voice de estos idiomas, potencialmente añadiendo datasets propios grabados por DIAL.

-Intent Parser: O módulo de procesamiento de lenguaje. Se encarga de analizar el texto proveniente de STT (pronunciado por el usuario) para entender qué orden se le ha dado, y enlazarla a la "Skill" adecuada. Se utilizará Adapt, el motor por defecto de Mycroft.

-Skills: Estas son las "habilidades" del asistente, de manera similar a las Skills de Alexa o Google Assistant. Se trata de los distintos servicios percibidos por los usuarios. Hay dos módulos integrados del sistema: El Skill Marketplace de Mycroft, que es una aplicación web centralizada y administrada por Mycroft donde terceros pueden desarrollar sus propias Skills y ponerlas a disposición de cualquier usuario de Mycroft. Y por otro lado, el Skill Manager, que es el módulo local de cada asistente Mycroft que se encarga de instalar/desinstalar Skills. Los Skill Manager locales de cada Raspberry se pueden configurar para bajarse Skills de cualquier repositorio Github, por lo que la mejor solución es desplegar las Skills personalizadas para el proyecto en repositorios privados.

La arquitectura básica de DIAL se compone, pues, de un dispositivo "local" (su despliegue se realiza en el hogar o entorno del usuario) que se comunica con un servidor "en la nube" (en la práctica, un servidor desplegado en la UPV). El dispositivo local consiste en una Raspberry Pi con un micrófono y un altavoz, y ejecuta distintos módulos software que se encargan de varias tareas del proceso de interacción con el asistente, mientras otras se delegan al software en el servidor.

Siguiendo una interacción típica con el asistente en la que el usuario genera una orden y recibe una respuesta auditiva a esa orden, (1) el usuario invoca al asistente con la palabra de activación, (2) que es detectada por el módulo Wake Word. (3) Se inicia una grabación durante la cual el usuario pronuncia su orden y ésta se envía al reconocimiento de voz del

módulo Speech-To-Text. (4) El texto resultante se devuelve al Intent Parser, que lo analiza para determinar (5) qué Skill activar y con qué parámetros. (6) La Skill puede necesitar conectar con su respectivo backend (en el caso de DIAL, presente en el mismo servidor) para completar su función (7). (8) La respuesta se entrega al módulo Text-To-Speech que lo traduce a audio, (9) reproducido por el altavoz al usuario (10).

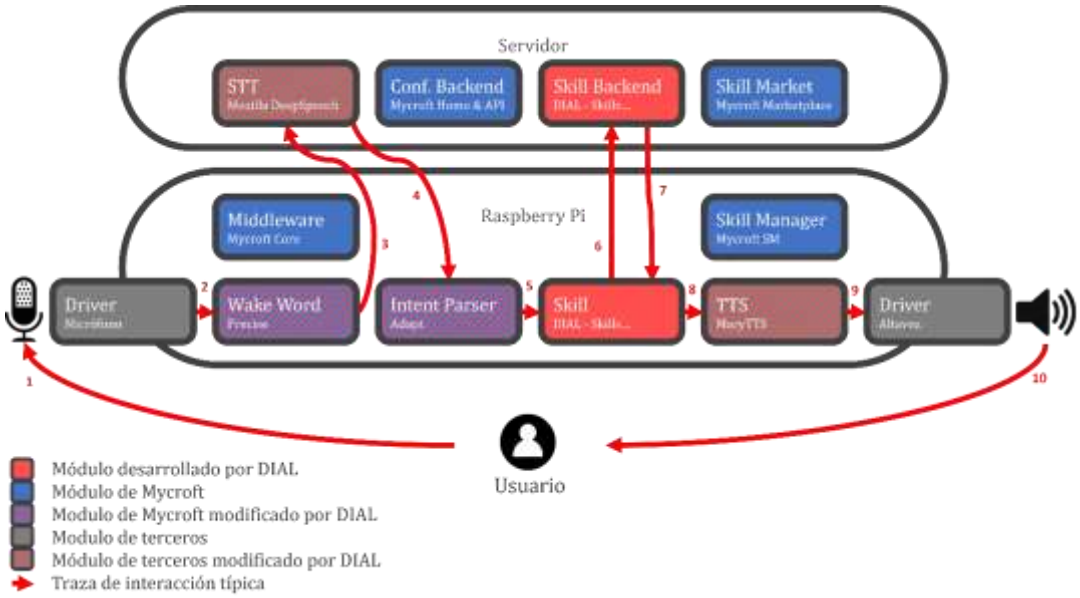

Fig. 5. Arquitectura modular del asistente de voz utilizando Mycroft

3.3.2. Licencias

En cuanto a licencias, cada módulo tiene licencias distintas por lo que habrá que tener cuidado con aquellos que usen licencias menos permisivas, como GPL. No obstante, al ser modular, estos podrían ser reemplazados por alternativas con licencias más permisivas.

Dentro de los diferentes módulos, tenemos Mycroft Core que está disponible bajo licencia Apache, por lo que no supone ningún problema su utilización, modificación o comercialización. Esta es una licencia de software libre permisiva creada por la Apache Software Foundation (ASF). Esta licencia requiere la conservación del aviso de derecho de autor y el descargo de responsabilidad, pero la licencia permite al usuario la libertad de usar el software para cualquier propósito, para distribuirlo, modificarlo y distribuir versiones modificadas del software, bajo los términos de la licencia, sin preocuparse de las regalías.

Por otra parte, el módulo Mycroft Home/API para el backend está licenciado bajo AGPL, por lo que podría imponer ciertas limitaciones (como la obligación de publicar el código). La licencia pública general de Affero (en inglés, Affero General Public License, AGPL) es una licencia copyleft derivada de la Licencia Pública General de GNU diseñada específicamente para asegurar la cooperación con la comunidad en el caso de software que corra en servidores de red. Esta licencia obliga a que si realizas modificaciones en el código fuente debes publicar tu modificación como código abierto también bajo GPL.

Dado que no realizamos cambios en el código de Selene (tan solo en archivos de configuración) no nos vemos afectados.

3.3.3. Picroft

Picroft es una distribución ya preparada para ejecutar Mycroft en una Raspberry Pi 3, 3B+ o 4 y se proporciona como una imagen de disco que se puede grabar en una tarjeta Micro SD. Picroft está basado en Raspbian Buster Lite y es completamente de código abierto.

Una vez que lo hemos descargado y grabado en una tarjeta de microSD, sólo debemos introducir el nombre de usuario y la contraseña de IP, que puede ser cambiada, y listo. Después lo configuramos para conectarse a internet, el tipo de teclado, audio, lenguaje, tipo de voz…

Capítulo 4

Diseño del prototipo

A lo largo de este apartado se van a detallar los elementos hardware que conforman el prototipo. Además, se realiza de especifica el rputer SIM M2M así como el plan de datos a utilizar en base al consumo esperado de datos del asistente. Para finalizar el capítulo, se escogerá la herramienta adecuada para la monitorización del prototipo.

4.1. Hardware

Para desplegar Mycroft es necesario un mínimo de dos dispositivos físicos: Un dispositivo local que haga de interfaz con el usuario allí donde se encuentra, y un servidor remoto donde se ejecuten los procesos más "pesados" como la síntesis de voz o la ejecución de Skills.

Se ha seleccionado el modelo de placa Raspberry Pi Model B de 4GB RAM (Mycroft recomienda un mínimo de 2GB) como el elemento fundamental para el desarrollo del prototipo por su compatibilidad con la imagen Picroft ya creada y lista para funcionar. Además, se requiere que la solución sea portable y de bajo coste para poder implementarla en los 45 hogares en los que se espera probar el asistente para el piloto.

Fig. 6. Raspberry Pi 4 Model B

Para poder cumplir con su cometido, la Raspberry Pi necesita de hardware adicional que debe adquirirse aparte: micrófono/altavoz y, opcionalmente, pantalla, además de una carcasa adecuada.

El micrófono debe ser de alta calidad para reducir al máximo los errores e interferencias a la hora de ejecutar el reconocimiento de voz y de la palabra de activación. Debe ser omnidireccional para captar al usuario independientemente de dónde esté. Hay un gran rango de micrófonos en el mercado, con características similares. El altavoz también debe ser omnidireccional para que el usuario lo oiga en cualquier lugar. Debe tener la potencia y la calidad suficiente para que el texto sintetizado sea entendible, en especial teniendo en cuenta los posibles problemas de audición de los usuarios.

Finalmente, se ha seleccionado el altavoz con micrófono incorporado EMEET OfficeCore Luna por ser un altavoz portátil de alta calidad a un precio asequible. El micrófono con cancelación de ruido mejora las voces humanas reduciendo el ruido ambiental y cancela el eco, lo que garantiza que se escuche con claridad.

Fig. 7. Altavoz portátil EMEET OfficeCore Luna

Por último, es necesaria la utilización de una tarjeta microSD para grabar la imagen Picroft e introducirla en la Raspberry Pi. Se ha elegido SanDisk Extreme Pro de 32GB por tener la suficiente capacidad para quemar la imagen.

Fig. 8. Tarjeta microSD de 32GB

Además de los elementos mencionados anteriormente, los servicios software que deben desplegarse “en la nube”, donde los servidores de la UPV serán el “backend” propietario de DIAL, tendrán una serie de requisitos hardware específicos. En este trabajo solo se tendrán en cuenta los requisitos necesarios para desplegar el backend, pero se deben añadir los requisitos correspondientes al uso de Machine Learning (por ejemplo, debido

al procesamiento de audio). Sin embargo, es de esperar que el hardware usado en los servidores empleados por la UPV sea suficiente.

4.2. Router SIM y plan de datos

El dispositivo local necesita una conexión a internet para poder transmitir y recibir datos del servidor. Una forma de tener acceso a internet es a través del módulo WiFi o mediante la conexión Ethernet de la Raspberry Pi, con la que podemos conectarnos a un router con una SIM M2M que incluye un plan de datos. Una tarjeta SIM Máquina a Máquina (M2M) difiere de una SIM estándar en varias características, básicamente su finalidad es el intercambio de información a través de un formato de datos entre dos máquinas.

Se diferencia de una SIM convencional porque una línea M2M/IoT tiene todo limitado, al contrario de una línea normal. En un línea M2M/IoT, en principio, solo se permite el tráfico de datos. Se tienen que levantar de manera explícita las restricciones sobre el tráfico SMS y de voz para su uso. De esta forma se evita el uso fraudulento de tarjetas. Además, una tarjeta M2M/IoT suele tener un coste individual mientras que el precio de una SIM normal está incluido en la mensualidad de la línea. Al cursar poco tráfico, una línea M2M/IoT tiene también una mensualidad mucho más reducida que una línea normal.

Por otra parte, una de las mayores diferencias entre una SIM normal y una SIM M2M/IoT reside en que las líneas M2M/IoT se gestionan a través de una plataforma (interfaz web) mientras que las SIM normales no tienen esta posibilidad. Las SIM M2M/IoT se activan/desactivan según las necesidades del usuario (no hace falta llamar al operador, se accede a la plataforma de gestión). Estas plataformas ofrecen también otras funciones como por ejemplo el diagnostico remoto, el control de consumo y la puesta en marcha de alarmas para ciertos comportamientos de las tarjetas SIM, como por ejemplo puede ser un consumo excesivo. Existen varios operadores para dispositivos IoT M2M con cobertura global y red multioperador GSM/2G/3G/4G LTE que operan en España. Para asegurarnos de que vamos a tener cobertura, en la página web de cada operador móvil podemos comprobar que España está dentro de los países en los que se ofrece cobertura. Concretamente, tendremos cobertura GPRS/3G/4G de Vodafone y Movistar, cobertura suficiente para nuestra aplicación.

Fig.9. Cobertura en España ofrecida por Things Mobile

Al elegir una tarjeta SIM M2M/IoT existen básicamente tres alternativas en las tarifas que se ofrecen:

1. Tarifa de consumo: es una tarifa individual para cada tarjeta en la que se define una capacidad máxima de datos, de SMS y de voz para cada una. Si se excede el límite se paga por uso de datos consumidos extra.

2. Tarifa de tráfico compartido: también se define una capacidad máxima pero se comparte entre todas las líneas. De esta forma balanceamos el consumo de datos entre las tarjetas que sobrepasan el límite de consumo de datos y las que no.

3. Tarifa escalonada: se definen varias tarifas con distintas capacidades. De esta forma, las líneas van saltando de una tarifa a otra cuando llegan al límite de la tarifa que se les ha asignado. Empezando en la tarifa con menor capacidad, cambiando progresivamente a mayores capacidades según la necesidad de consumo.

Por último, es necesario conocer el volumen de datos que va a consumir Mycroft para poder elegir un plan de datos u otro. Podemos basarnos en el consumo de DIAL con comandos básicos. La tabla inferior muestra el consumo de datos para distintos comandos de Alexa [11], lo que se espera como una buena aproximación a lo que consumiría el asistente.

Alexa (Echo Dot)	**Comando**	**Consumo de datos**
	Configurar alarma	36.7KB
	Preguntas	150-250 KB / pregunta
	30 minutos de música	38 MB
	1h interaccionando + apagado/encendido de dispositivos	2MB
	Inactivo con conversaciones cercanas sin decir la palabra de activación	5MB/24 H
	Búsqueda online	94.72KB
	Dictar texto (p.e. un email)	72.5KB

Tabla 2. Consumo de datos para diferentes comandos

En este proyecto la interacción con DIAL se basa en diálogos. El asistente puede puntualmente sugerir escuchar música o proponer alarmas o recordatorios de eventos para las rutinas del usuario, pero en general, serán las conversaciones lo que mayor tiempo ocupe. En base a lo anterior, podemos estimar que, con un uso de 30 minutos al día de media y una estimación de 60 interacciones entre el usuario y el asistente, nuestro sistema consumirá un total de 12MB al día por usuario. Siendo el cómputo total al mes de 2.88GB aproximadamente.

Con respecto al módem SIM M2M utilizado, se ha elegido el router TP-Link TL-MR100 Wi-Fi N 4G LTE, con velocidades de hasta 300Mbps, suficientes para la aplicación del dispositivo. Además, es compatible con Linux.

4.3. Monitorización

Para poder administrar los dispositivos en los domicilios de los usuarios, se necesita una herramienta que nos permita acceder a ellos remotamente. Las Raspberrys no tienen IP

fija, por lo que resulta más complicado acceder. Además, todos los servicios deben iniciarse correctamente tras el reinicio del dispositivo en remoto para poder volver a acceder de nuevo.

Una solución fácil y práctica que nos podría ayudar en este caso es configurar un túnel SSH inverso en Linux. El túnel SSH inverso permite crear una conexión entre la Raspberry remota y el equipo desde el que se quiere realizar la conexión, con la característica de que la petición principal la realiza la Raspberry remota y no el equipo remoto desde donde accedemos. También, al añadir certificados entre la Raspberry y el servidor, aumentamos el nivel de seguridad de acceso.

La Raspberry es la que genera el túnel inverso, de manera que abre una puerta en el servidor para poder conectarnos a ella. Es decir, el servidor va a recibir las conexiones entrantes de las 45 Raspberrys, abriendo 45 puertos diferentes, y las va a redirigir por el puerto 22. Al acceder al servidor desde nuestro equipo remoto, nos redirecciona a cada una de ellas. Debido a que las Raspberrys no tienen IP fija es una forma efectiva de poder acceder a ellas pasando por el servidor como elemento central entre ambos dispositivos remotos.

Gracias a esto, se puede comprobar el estado de los dispositivos en los domicilios de los usuarios, mandar comandos y reiniciar las Raspberrys desde el equipo remoto desde el que queramos acceder.

-Como primer paso, necesitamos abrir puertos en el servidor para el acceso en localhost. Vamos a utilizar el puerto 10001 para la primera Raspberry que conectemos, el puerto 10002 para la siguiente y así sucesivamente para seguir una nomenclatura acorde a las 45 Raspberrys. Así, empezamos con el puerto 10001 hasta el 10045.

Rpi	Puerto que abre en el servidor
Rpi1	10001
Rpi2	10002
Rpi3	10003
Rpi4	10004
…	…
Rpi45	10045

Tabla 3. Correspondencia de puertos con cada Raspberry

-Abrimos el puerto 10001 en el firewall del servidor:

```
Sudo ufw allow 10001
Sudo ufw status verbose
```

-Comenzamos comprobando la conexión entre la Raspberry y el servidor mediante SSH de forma común con el siguiente comando, con el que nos pedirá ingresar la contraseña del servidor:

```
Ssh dial@158.42.167.31
```

-Una vez comprobado que tenemos conexión, salimos de las sesión y realizamos la conexión por SSH inverso, creando el túnel:

```
Ssh -N -R 10001:localhost:22 dial@158.42.167.31
```

-10001: el puerto utilizado para el túnel SSH inverso.
-22 – puerto ssh predeterminado de Raspberry.
-usuario@xxx.xxx.xxx.xxx: usuario y dirección IP del servidor.

-Después realizamos la conexión del servidor a la Raspberry mediante el siguiente comando, pidiendo la contraseña del usuario de Raspeberry:

```
Ssh -l pi -p 10001 localhost
```

-Ahora, para acceder a la Raspberry desde otro equipo remoto, es decir, abrir el túnel SSH directamente a Raspberry desde otra máquina sin conectarse primero al servidor por SSH (en realidad, la conexión también pasará por el servidor, pero ya no veremos el inicio de sesión del servidor), debemos agregar/modificar una variable en / etc/ssh/sshd-config en el servidor:

```
nano /etc/ssh/sshd-config
```

-En el fichero, buscamos la entrada “GatewayPorts” y lo descomentamos y cambiamos a “GatewayPorts yes”. Después de cambiar el archivo, tenemos que volver a cargar el servicio sshd:

```
systemctl recargar sshd.servicio
```

-Ahora podemos iniciar una conexión directamente a Raspberry desde una máquina con Windows usando este comando en CLI o usando Putty:

```
ssh -p 10001 pi@158.42.167.31
```

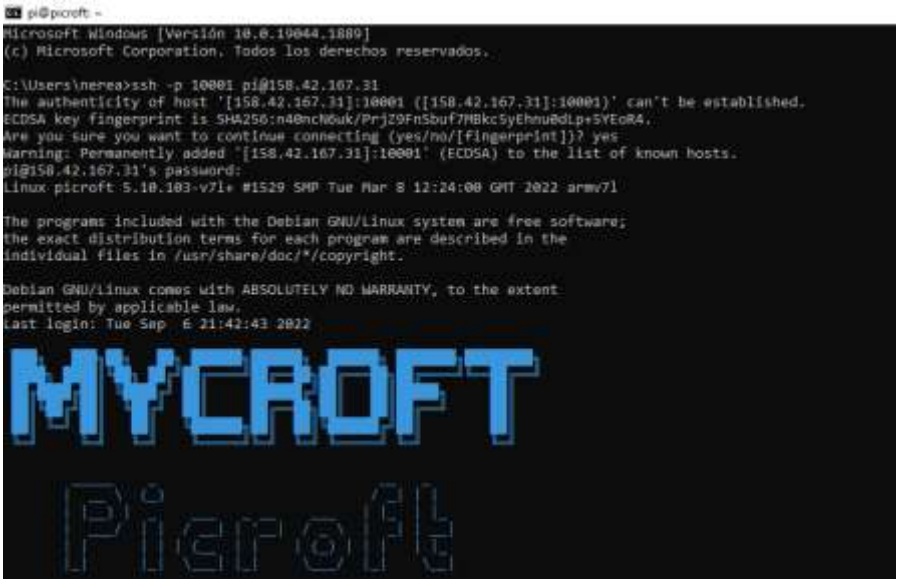

Fig.10. Conexión equipo remoto-Raspberry sin certificados añadidos

-En este punto, observamos que aunque tengamos conexión, la autenticación no puede ser establecida porque debemos añadir antes los certificados de seguridad mediante claves público/privadas. De esta forma, como no tendremos acceso a las Raspberrys para iniciar la conexión manualmente, es necesario que se conecte sin necesitar introducir la contraseña. Una clave SSH es una credencial de acceso en el Protocolo SSH. Las claves SSH ofrecen una forma más segura de iniciar sesión en un servidor con SSH que usar el usuario y la contraseña. Además, el uso de una clave SSH brinda la posibilidad de automatizar el proceso de inicio de sesión, que es lo que buscamos.

-En la Raspberry generamos el par de claves público-privadas con el comando incluido a continuación. Este comando da la opción de cambiar la ubicación y el nombre del par de claves SSH generado o dejar el valor predeterminado, que es el directorio de inicio del usuario y el nombre predeterminado: id_rsa. Lo dejaremos por defecto en este caso. Las claves quedan guardadas en /home/pi/.ssh/id_rsa y /home/pi/.ssh/id_rsa.pub, respectivamente para la clave privada y la pública. Además, pedirá una frase de contraseña. Esta vez no estableceremos una frase de contraseña porque necesitamos crear un inicio de sesión SSH automático. Tenemos que copiar ahora la clave pública en el servidor:

```
Ssh-keygen
ssh-copy-id dial@158.42.167.31
```

-De nuevo, probamos a conectarnos desde la Raspberry al servidor para comprobar que con la creación de estas claves ya no es necesario ingresar la contraseña del servidor y que funciona correctamente:

```
Ssh dial@158..42.167.31
```

-Además, puedes generar las claves público-privadas para Windows como se menciona aquí [16] y acceder desde cualquier equipo remoto a la Raspberry directamente:

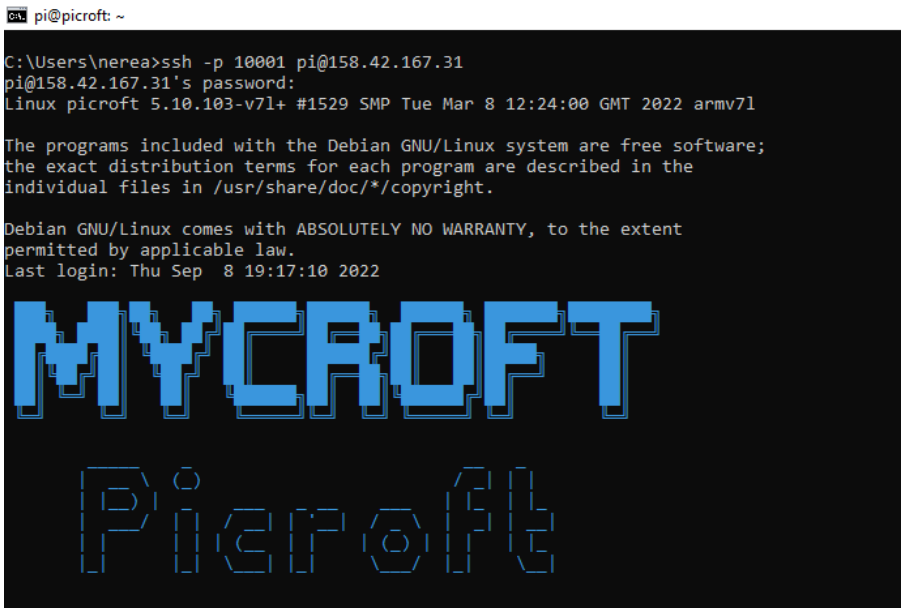

Fig.11. Conexión equipo remoto-Raspberry con certificados añadidos

-A continuación, hacemos el túnel que sea persistente. Necesitamos algún servicio o script que realice la tarea de vigilar que el túnel está funcionando y en el caso de que no sea así, volver a abrirlo automáticamente. El túnel creado hasta ahora no será persistente y se eliminará si la conexión de la Raspberry cae. Como solución, creamos un script que se iniciará automáticamente en el arranque y creará el túnel SSH.

-Este fichero comprueba si hay un proceso SSH inverso ejecutándose, y si no es así abre el túnel. El fichero creado está basado en el encontrado aquí [15]. Creamos el fichero en la Raspberry bajo el nombre "~/create_ssh_tunnel.sh" con el siguiente contenido:

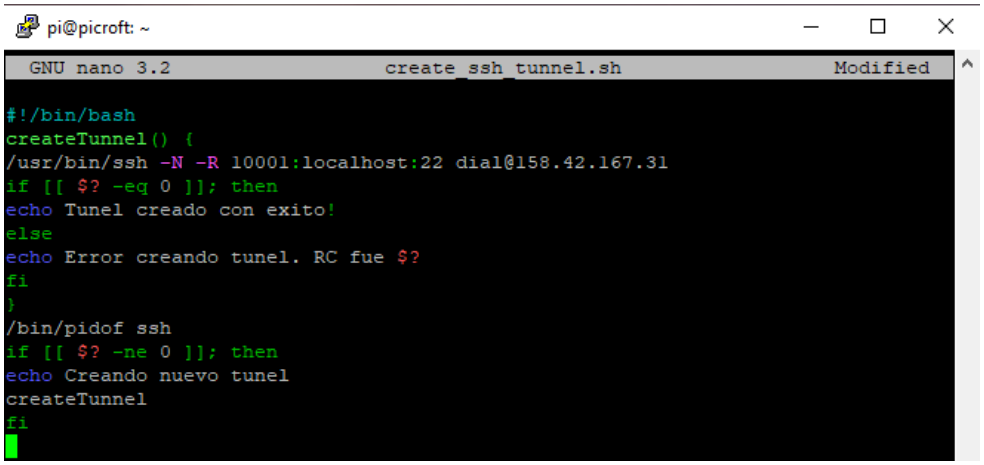

Fig.12. Script para el mantenimiento del túnel

-Hacemos ejecutable el script:

```
chmod 700 ~/create_ssh_tunnel.sh
```

-Modificamos crontab para que se ejecute cada minuto:

```
crontab -e
```

-Y añadimos al final del fichero lo siguiente para hacer que se ejecute cada minuto:

```
*/1 * * * * ~/create_ssh_tunnel.sh > tunnel.log 2>&1
```

Una vez hecho el diseño del prototipo hardware, su administración remota y la definición del sistema con de acceso a internet, vamos a desarrollar la parte backend y frontend para la administración de los asistentes.

Como se ha mencionado anteriormente, vamos a desplegar el servidor de Mycroft en servidores propios de la UPV en vez de que esto se desarrolle en servidores de Mycroft. El siguiente apartado explica el desarrollo seguido.

Capítulo 5

Servidor Mycroft

A continuación, se detalla el despliegue del “backend” con el que se administran los usuarios y dispositivos. Idealmente, se trata de una aplicación web para la administración de dichos elementos, y por defecto está desplegada en los servidores de Mycroft. En nuestro caso, este módulo debe desplegarse en servidores propios de la UPV para mayor seguridad de los datos de los usuarios. De esta forma, el servicio no pasa por terceros, garantizando privacidad y cumplimiento del RGPD.

5.1. Servidor central

El desarrollo se realiza en un servidor situado en la UPV, que funcionará como “nube”. Este servidor es una máquina virtual Hyper V con Linux Ubuntu 20.04 (que es lo que recomienda Mycroft) desplegada en uno de los servidor físicos (Windows) de ITACA-SABIEN, con 22GB RAM y CPU de 32 núcleos. Los dispositivos recibirán y entregarán datos al servidor, mediante la solución de conectividad mencionada en capítulos anteriores.

Para conectarnos al servidor remotamente, se ha usado PuTTY [12]. En general, PuTTY no es más que un terminal de simulación open source desarrollado para actuar como cliente de conexiones seguras a través de protocolos raw TCP, Telnet, rlogin y portal serial. Por lo tanto, este software es suficiente para establecer conexiones seguras de acceso remoto al servidore a través de Shell Seguro (SSH).

Una vez realizada la conexión, vamos a analizar cómo está construido el servidor.

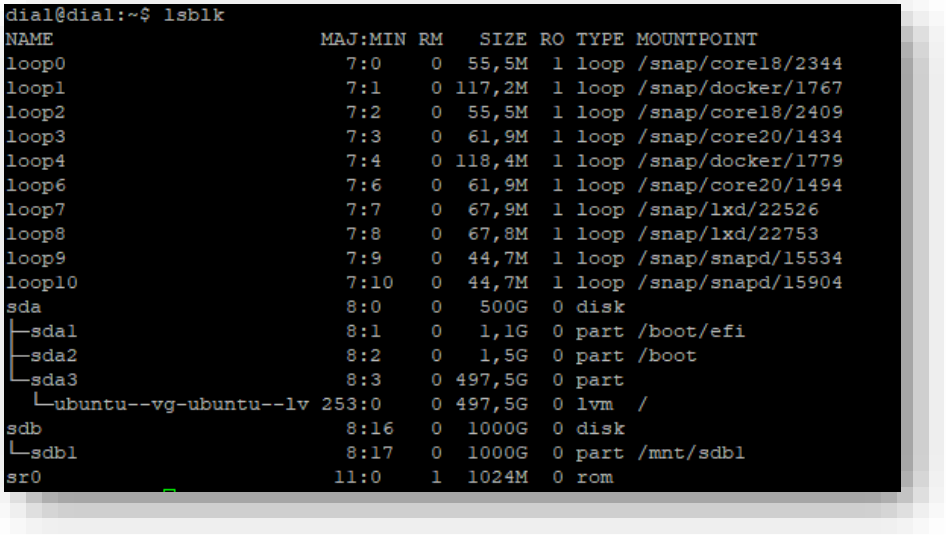

Fig. 13 Dispositivos de bloque

Observamos que está compuesto por un disco rápido y otro lento. Dentro del disco rápido (sda) vemos que el volumen donde se encuentra root comprende 497.5GB de memoria, de los cuales se han usado 109GB.

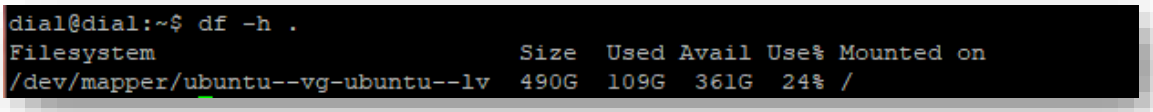

Fig. 14. Disco rápido

Dentro del disco lento (sdb) tenemos la partición ‘sdb1’ con 984GB de los cuales 136GB están en uso.

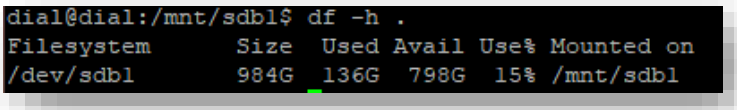

Fig. 15. Disco lento

Vamos a desplegar el backend sobre el disco rápido debido a que la cantidad de usuarios es reducida por lo que es más sencillo acceder por ejemplo a la base de datos de esta forma más rápida.

5.2. Selene Backend

Selene proporciona los servicios utilizados por Mycroft Core para administrar dispositivos, Skills y configuraciones.

En la documentación oficial encontramos que consta de dos repositorios:

1. Selene Backend: contiene Python y SQL que representan la definición de la base de datos, la capa de acceso a datos, las API y los scripts y será el que vamos a utilizar a lo largo de esta sección [13].
2. Selene UI: contiene aplicaciones web Angular que usan las API definidas en este repositorio [14]. Esto se desarrollará en el subcapítulo 5.3.

Hay cuatro API definidas en el repositorio Selene Backend: administración de cuentas (account), inicio de sesión único (SSO), skills market (market) y dispositivo (device o también se menciona como "public"). Los tres primeros admiten account.mycroft.ai (también conocido como home.mycroft.ai), sso.mycroft.ai y market.mycroft.ai, respectivamente. Estos nombres son los subdominios que Mycroft le asigna a cada API en su servidor. En nuestro caso no tenemos ni subdominios ni dominio, así que usaremos IPs directamente. La API "dispositivo" es la forma en que los dispositivos que ejecutan Mycroft Core se comunican con el servidor. También se incluye en este repositorio un paquete que contiene scripts para el mantenimiento y la definición del esquema de la base de datos.

Cada API está diseñada para ejecutarse independientemente de las demás. El código común a cada una de las API, como la capa de acceso a datos, se puede encontrar en el directorio "shared". El código dentro de "shared" es un paquete de Python independiente requerido por cada una de las API. Cada API es un programa Python independiente en el que se utiliza Pipenv para empaquetar aplicaciones. Pipenv es una herramienta recomendada para empaquetar aplicaciones complejas y sus dependencias cuando utilizamos Python, se utiliza para el entorno virtual y la gestión de estos paquetes. Vemos que cada API tiene su propio Pipfile por lo que podemos ejecutarlas en su propio entorno virtual.

Con respecto a los requisitos para su utilización, las APIs no van a dar servicio a una gran cantidad de dispositivos, el número máximo de usuarios serían 45 para la realización del piloto por lo que no es necesario que cada una se ejecute en su propio servidor o máquina virtual. Utilizando un único servidor, los requisitos recomendados son 4 CPU, 8 GB de RAM y 100 GB de disco, requisitos que el servidor de la UPV cumple con creces.

Por otra parte, el usuario creado para la aplicación lo hemos llamado 'mycroft'. Este es el usuario para cada API en sus respectivos sistemas. Conviene mencionar que todos los valores de contraseñas, keys y tokens que han utilizado en el desarrollo del trabajo, son los que se han utilizado en las pruebas pero en producción se generarán unos nuevos.

Además de las cuatro APIs mencionadas, Selene requiere de dos bases de datos que se explicarán en las siguientes secciones: PostgreSQL para la administración de datos y Redis para el almacenamiento en caché.

5.2.1. Base de datos: PostgreSQL

En este apartado vamos a instalar la base de datos PostgreSQL y la vamos a configurar en base a los repositorios que encontramos en la documentación oficial. De nuevo,

encontramos una configuración recomendada: Ubuntu 18.04 LTS, 2 CPU, 4 GB de RAM y disco de 50 GB, requisitos que cumple el servidor.

1. Comenzamos utilizando el sistema de administración de paquetes para instalar Python 3.7, Python 3 pip y PostgreSQL 10:

```
sudo apt-get install postgresql python3.7 python python3-pip
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt-get update
sudo apt-get install python3.7
sudo apt install python3.7-distutils
python3.7 -m pip install SomePackage
```

La versión instalada de Python es 3.7.13.

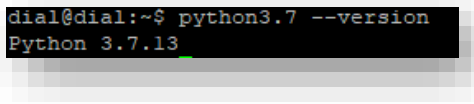

Fig. 16. Versión de Python instalada

2. Configuramos Postgres para que comience en el arranque y comprobamos su correcto funcionamiento:

```
sudo systemctl enable postgresql
service postgresql status
```

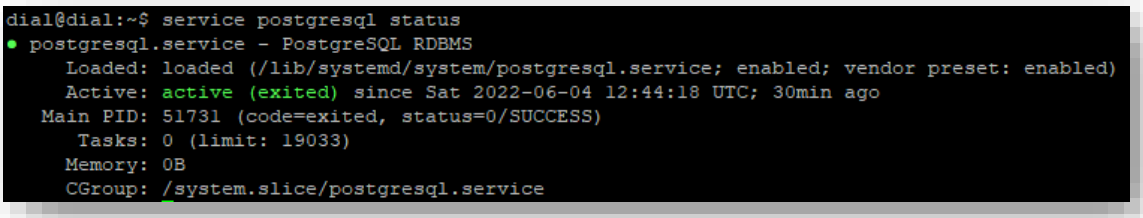

Fig. 17. Estado de PostgreSQL

3. Clonamos los repositorios de documentación y backend de Selene:

```
sudo mkdir -p /opt/selene
sudo chown -R dial:users /opt/selene
cd /opt/selene
git clone https://github.com/MycroftAI/selene-backend.git
```

4. Creamos el entorno virtual para el código de la base de datos:

```
    sudo python3.7 -m pip install pipenv
cd /opt/selene/selene-backend/db
pipenv install
```

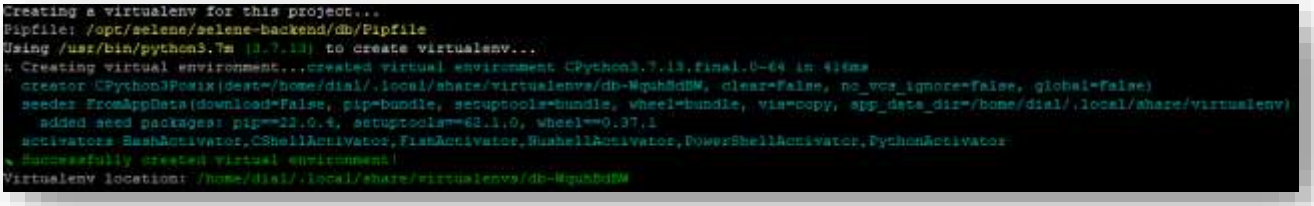

Fig. 18. Creación del entorno virtual para la base de datos

5. Descargamos los archivos de geonames.org utilizados para completar las tablas del esquema geográfico. El archivo de texto de información de países de geonames countryInfo.txt contiene los nombres de los países y una variedad de códigos junto con información vinculada a la ciudad capital. El archivo timeZones.txt lista la hora Greenwich Mean Time (GMT) con los respectivos offset para los países. Admin1Codes ASCII.txt incluye la información en formato ASCII y por último, cities500.zip que contiene todas las ciudades con una población superior a 500 habitantes. Estos datos se usan para rellenar las opciones disponibles en los desplegables a la hora de configurar las cuentas de usuario y la ubicación de los dispositivos Mycroft.

```
mkdir -p /opt/selene/data

cd /opt/selene/data

wget http://download.geonames.org/export/dump/countryInfo.txt
```

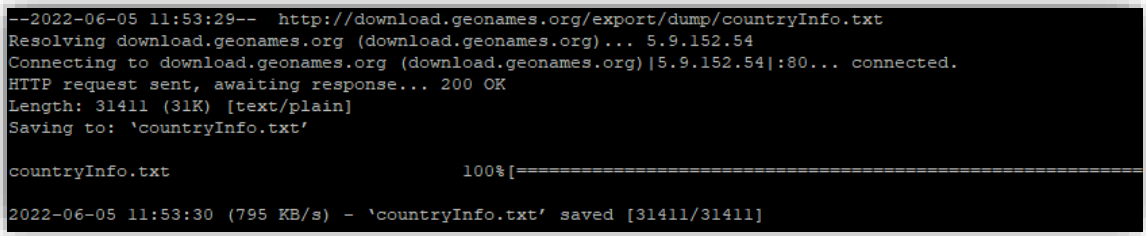

Fig. 19. Archivo countryInfo.txt

```
wget http://download.geonames.org/export/dump/timeZones.txt
```

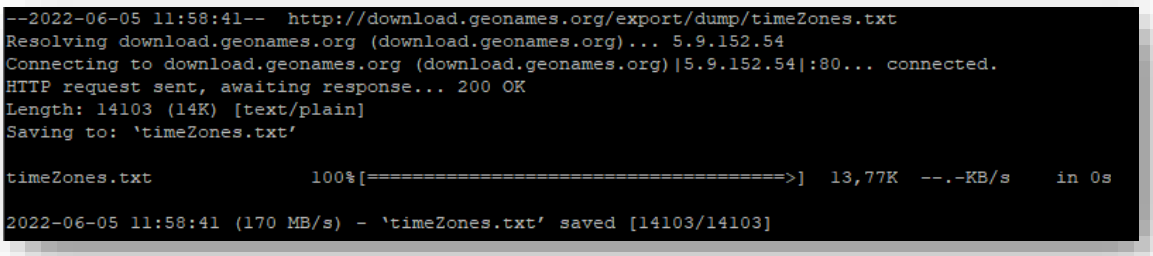

Fig. 20. Archivo timeZones.txt

```
wget http://download.geonames.org/export/dump/admin1CodesASCII.txt
```

```
--2022-06-05 12:00:20--  http://download.geonames.org/export/dump/admin1CodesASCII.txt
Resolving download.geonames.org (download.geonames.org)... 5.9.152.54
Connecting to download.geonames.org (download.geonames.org)|5.9.152.54|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 135356 (132K) [text/plain]
Saving to: 'admin1CodesASCII.txt'

admin1CodesASCII.txt       100%[=====================================>] 132,18K  --.-KB/s    in 0,1s

2022-06-05 12:00:20 (1,07 MB/s) - 'admin1CodesASCII.txt' saved [135356/135356]
```

Fig. 21. Archivo admin1CodesASCII.txt

```
wget http://download.geonames.org/export/dump/cities500.zip
```

```
--2022-06-05 12:00:45--  http://download.geonames.org/export/dump/cities500.zip
Resolving download.geonames.org (download.geonames.org)... 5.9.152.54
Connecting to download.geonames.org (download.geonames.org)|5.9.152.54|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10516925 (10M) [application/zip]
Saving to: 'cities500.zip'

cities500.zip              100%[=====================================>]  10,03M  25,0MB/s    in 0,4s

2022-06-05 12:00:46 (25,0 MB/s) - 'cities500.zip' saved [10516925/10516925]
```

Fig. 22. Archivo cities500.zip

6. Agregamos variables de entorno que contengan las contraseñas definidas por el usuario para la administración de la base de datos:

```
dial@dial:/opt/selene/data$ export DB_PASSWORD=mycroft
dial@dial:/opt/selene/data$ echo $DB_PASSWORD
mycroft
dial@dial:/opt/selene/data$ export POSTGRES_PASSWORD=mycroft
dial@dial:/opt/selene/data$ echo $POSTGRES_PASSWORD
mycroft
```

Fig. 23. Variables de entorno para la base de datos

7. A continuación, generamos contraseñas seguras para el usuario de postgres y el usuario de Selene en la base de datos con las anteriormente definidas:

```
sudo -u postgres psql -c "ALTER USER postgres PASSWORD
'$POSTGRES_PASSWORD'"

sudo -u postgres psql -c "CREATE ROLE selene WITH LOGIN ENCRYPTED
PASSWORD '$DB_PASSWORD'"
```

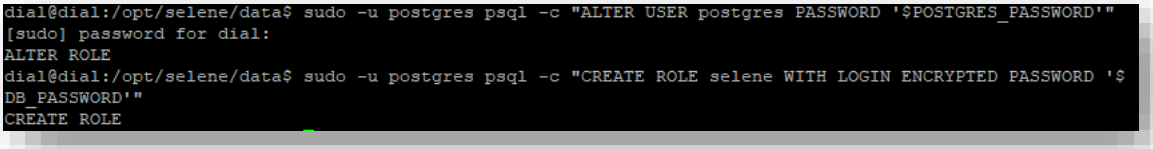

Fig. 24. Contraseñas seguras

8. Ejecutamos el script de arranque:

```
cd /opt/selene/selene-backend/db/scripts
pipenv run python bootstrap_mycroft_db.py
```

*En este paso, obtenemos un error de autentificación ya que el usuario Selene no está correctamente definido y configurado para permitir conexiones. Por defecto, Postgres solo permite conexiones desde localhost. Por lo que agregamos una entrada en el archivo 'pg_hba.conf' para cada ip que necesite acceder a esta base de datos. Tenemos que editar el archivo permitiendo las conexiones de la ip local:

```
sudo -u postgres vi /etc/postgres/10/main/pg_hba.conf
```

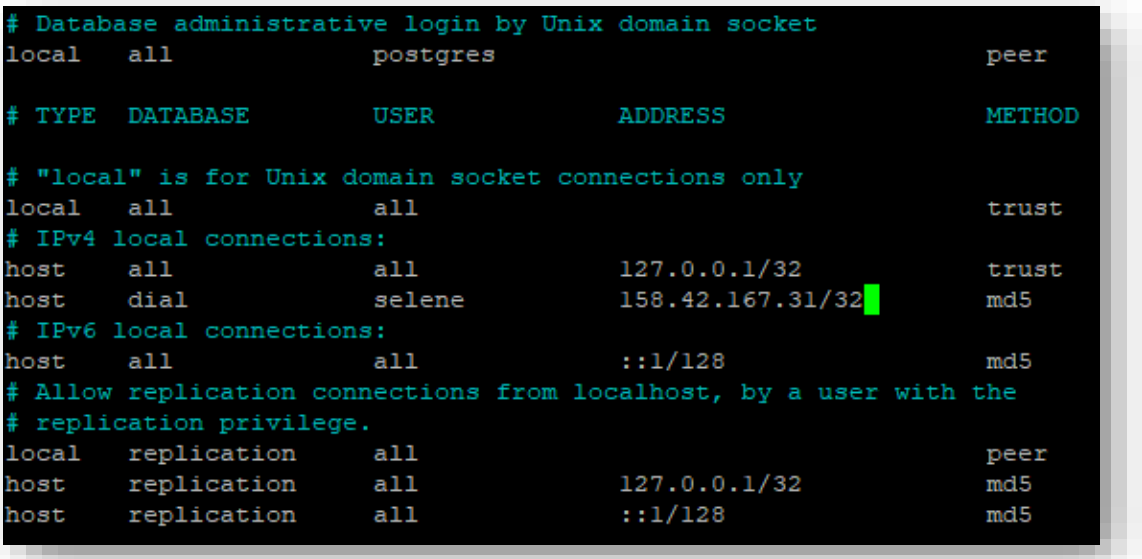

Fig. 25. Archivo 'pg_hba.conf'

Además, debemos cambiar la ip que aparece en 'listen_addresses' en el archivo 'posgresql.conf' a la IP privada del servidor de la base de datos.

```
sudo -u postgres vi /etc/postgres/10/main/postgresql.conf
apt install net-tools
ifconfig
```

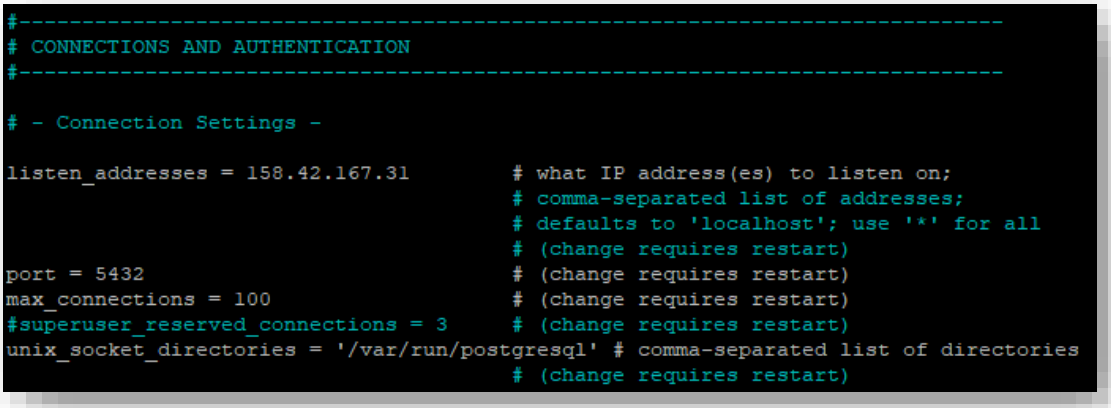

Fig. 26. Archivo 'postgresql.conf'

9. Por último, reiniciamos Postgres para que los cambios se actualicen y ejecutamos de nuevo el script de arranque con los cambios realizados:

```
sudo systemctl restart postgresql
cd /opt/selene/selene-backend/db/scripts
pipenv run python bootstrap_mycroft_db.py
```

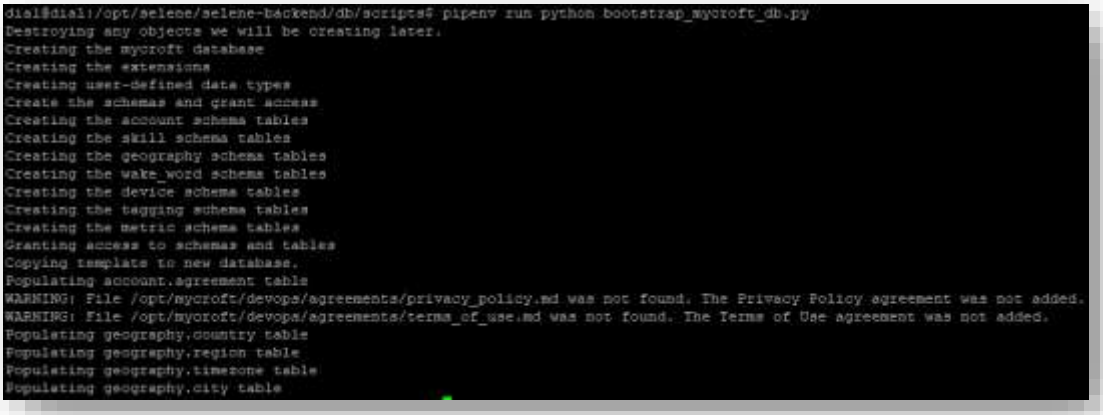

Fig. 27. Archivo 'postgresql.conf'

Finalmente, la base de datos se construye con éxito, creando todas las tablas, roles y esquemas necesarios.

5.2.2. Base de datos: Redis

De nuevo, tenemos una configuración de servidor recomendado que cumplimos: Ubuntu 18.04 LTS, 1 CPU, 1 GB de RAM y disco de 5 GB.

Redis es un "almacén" rápido de datos clave-valor que se guardan en memoria, es decir, es una base de datos con estructura de tablas hash que gracias a que los datos residen en

memoria, proporciona muy buenos tiempos de respuesta en la recuperación de la información. De código abierto, está diseñado para que puedan acceder a él usuarios de confianza dentro de entornos confiables. Es decir, va a proporcionar un control de accesos seguro para nuestra aplicación web. Funcionará de forma que dentro del front-end de la aplicación, los usuarios (en este caso, solo accede el administrador del sistema, la única conexión segura) consultarán a Redis para gestionar páginas o realizar operaciones solicitadas en la aplicación web.

Antes de nada, como pre-requisitos tenemos que configurar un cortafuegos básico para aumentar la seguridad. Vamos a configurar el firewall predeterminado UFW (Uncomplicated Firewall) para asegurarnos de que solo se permiten conexiones a ciertos servicios del servidor. UFW es una herramienta de configuración de firewall predeterminada que viene con los servidores Ubuntu y por defecto estádesactivado.

OpenSSH es el servicio que le permite las conexiones al servidor y tiene un perfil registrado dentro de UFW. Ejecutamos el siguiente comando para obtener una lista de todos los perfiles disponibles actualmente:

```
ufw app list
```

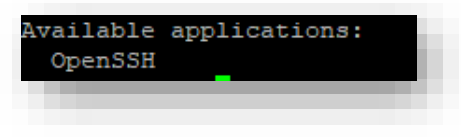

Fig. 28. Perfiles de aplicaciones disponibles

Nos aseguramos de que el firewall permite conexiones SSH para poder volver a iniciar sesión la próxima vez, permitimos estas conexiones y habilitamos el firewall escribiendo:

```
ufw allow OpenSSH

ufw enable
```

Podemos ver que las conexiones SSH todavía están permitidas y el estado actual del firewall está en activo:

```
ufw status

sudo ufw status verbose
```

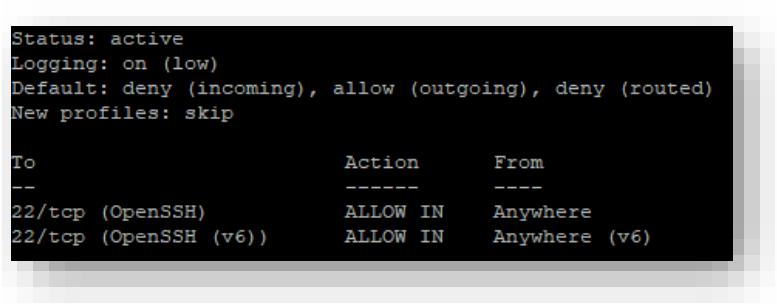

Fig. 29. Estado de UFW

Como el cortafuegos actualmente está bloqueando todas las conexiones excepto SSH, si instalamos y configuramos servicios adicionales, deberemos ajustar la configuración del cortafuegos para permitir la entrada de tráfico.

Los puertos de las web frontend Selene UI son: 8081, 8082, 8083 (que en el desarrollo implementado con Docker será solo el puerto 443 (https)), por lo que tendremos que abrir estos puertos para más adelante. Además, el puerto de la API "dispositivo" es el 5003 (aunque cambiará al 5000 con el despliegue en Docker).

A continuación, vamos a obtener la última versión de Redis desde los repositorios oficiales de Ubuntu e instalamos Redis:

```
sudo apt update

sudo apt install redis-server
```

Esto descargará e instalará Redis y sus dependencias. Después de esto, realizamos un cambio de configuración importante en el archivo de configuración de Redis que se generó automáticamente durante la instalación.

Abrimos el archivo redis.conf, donde buscamos la directiva "supervised" Esta directiva permite declarar el sistema de inicio para administrar Redis como un servicio. La directiva está establecida en "no" por defecto.

Como estamos ejecutando Ubuntu, que usa el sistema de inicio systemd, cambiamos esto a systemd y reiniciamos el servicio Redis para reflejar los cambios:

```
sudo nano /etc/redis/redis.conf

sudo systemctl restart redis.service
```

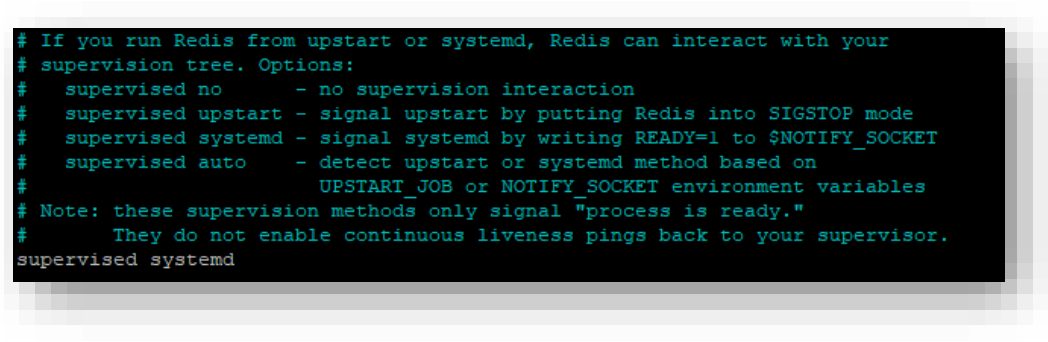

Fig. 30. Directiva “supervised”

Con eso, ya tenemos instalado y configurado Redis y se está ejecutando correctamente. Sin embargo, antes de comenzar a usarlo, vamos a verificar que funciona correctamente. Comenzamos verificando que el servicio Redis se esté ejecutando y que está configurado para iniciarse cada vez que se inicia el servidor:

```
sudo systemctl status redis
```

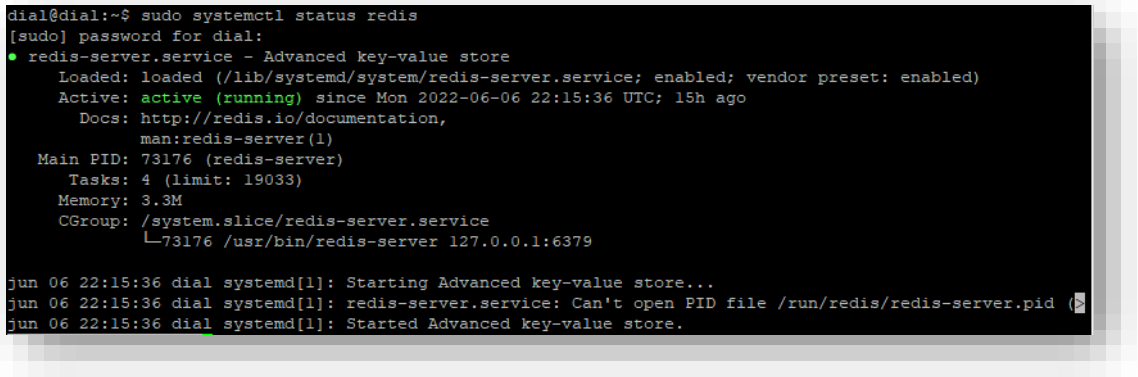

Fig. 31. Servicio redis

Para probar que Redis funciona correctamente, nos conectamos mediante el cliente de línea de comandos: redis-cli, y probamos la conectividad con el comando ping:

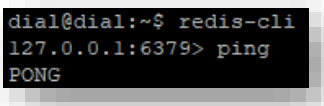

Fig. 32. Comando ping

Este resultado confirma que la conexión del servidor aún está activa. A continuación, comprobamos que podemos establecer claves ejecutando el siguiente comando y que podemos recuperar su valor:

```
set test "It's working!"
get test
```

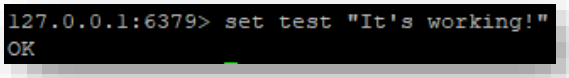

Fig. 33. Comando set

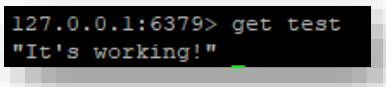

Fig. 34. Comando get

Como prueba final, verificaremos si Redis puede conservar los datos incluso después de que se haya detenido o reiniciado. Para hacer esto, primero reiniciamos la instancia de Redis. Luego, nos conectamos con el cliente de línea de comandos para confirmar que la prueba anterior está todavía disponible:

```
sudo systemctl restart redis
```

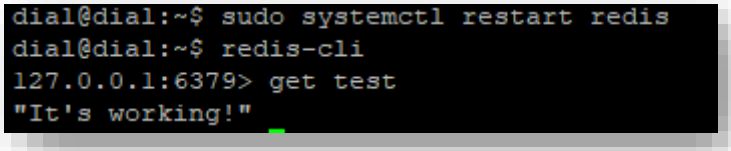

Fig. 35. Persistencia de los datos en Redis

Con esto, la instalación de Redis está completamente operativa y lista. Comenzaremos realizando ciertas configuraciones para mitigar vulnerabilidades.

De forma predeterminada, Redis solo escucha en el host local. En nuestro caso podemos cambiar la variable "bind" en el archivo /etc/redis/redis.conf para que sea la IP privada del host Redis, pero no es tan seguro como enlazar a localhost. Vamos a forzar que sea localhost la ip enlazada. Eliminamos el comentario en la línea correcta y reiniciamos el servicio para asegurarnos de que systemd lee los cambios:

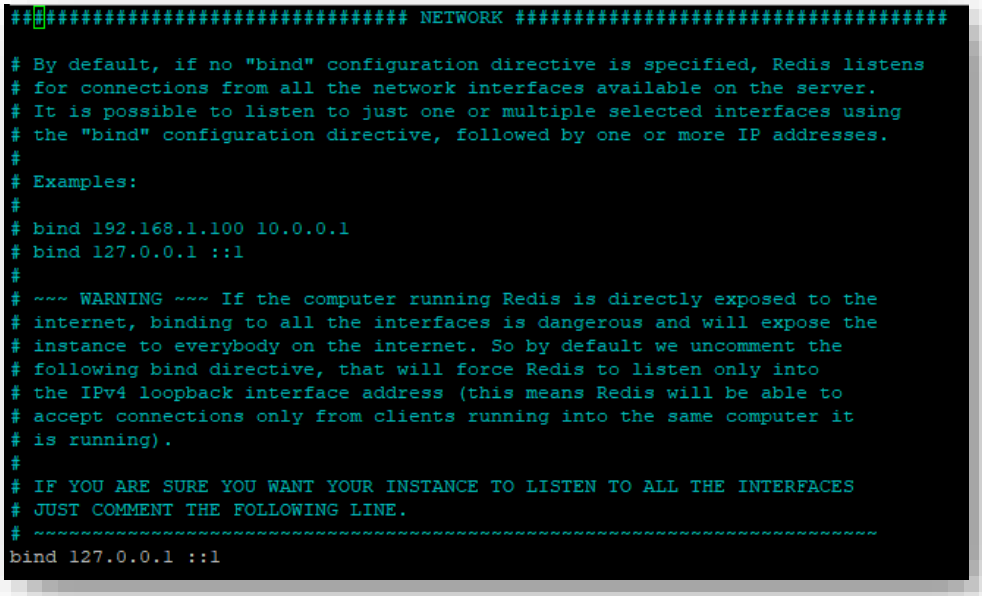

Fig. 36. Enlace a localhost en Redis

```
sudo systemctl restart redis
```

Para comprobar que este cambio ha entrado en vigor:

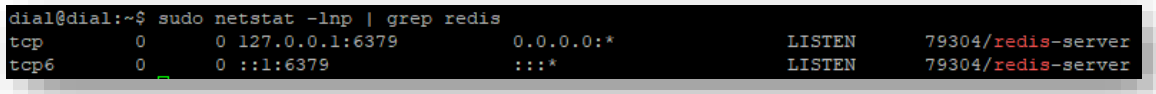

Fig. 37. Conexiones de red Redis

Este resultado muestra que el redis-server está vinculado a localhost (127.0.0.1), lo que refleja la configuración “bind” en el archivo de configuración de Redis. Ahora que Redis solo escucha en localhost, el servidor está más protegido siendo más complicado obtener acceso o realizar solicitudes por un usuario malicioso externo.

Para terminar este apartado, vamos a configurar la autenticación de los usuarios. Redis, actualmente, no está configurado para requerir que los usuarios se autentiquen antes de realizar cambios en su configuración o en los datos que contiene. Para remediar esto, vamos a configurar Redis para que solicite a los usuarios que se autentiquen con un token de autenticación o contraseña antes de realizar cambios a través del cliente de Redis. De esta forma mejoramos la seguridad de los datos.

La configuración de una contraseña de Redis habilita una de sus dos funciones de seguridad integradas: el comando AUTH, que requiere la autenticación del usuario para acceder a la base de datos.

La contraseña se configura directamente en el archivo de configuración de Redis /etc/redis/redis.conf:

```
sudo nano /etc/redis/redis.conf
```

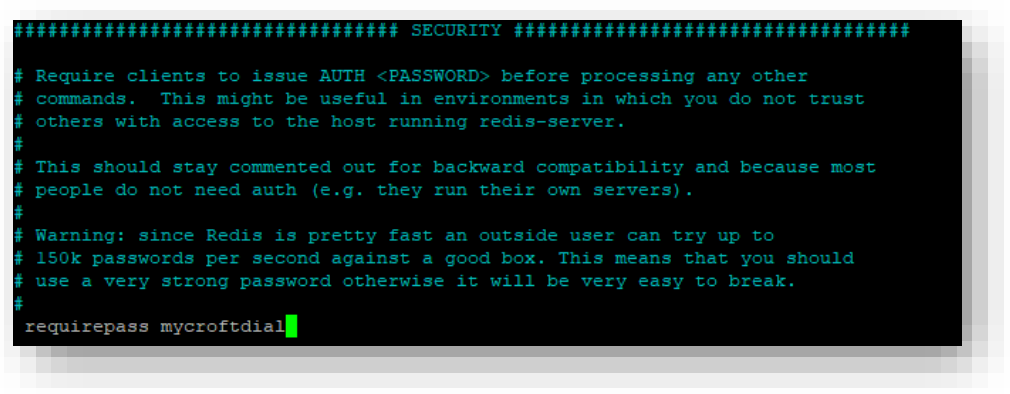

Fig. 38. Token de usuario en Redis

Lo cambiaremos a una contraseña segura. Si nos fijamos en el 'warning' que aparece, nos advierte que, al ser Redis tan rápido, un usuario podría probar hasta 150k contraseñas por fuerza bruta por segundo. Debido a esto, el número de intentos es tan elevado, que nuestra contraseña tiene que ser muy fuerte. Por lo tanto, es necesario definir un valor muy fuerte y largo como contraseña. Podemos usar el comando 'openssl' para generar una contraseña aleatoria. Para generar una contraseña más fuerte, utilizamos la salida del primer 'openssl' para la generación de la contraseña final.

```
openssl rand 60 | openssl base64 -A
```

Su salida debe ser algo como:
RBOJ9cCNoGCKhlEBwQLHri1g+atWgn4Xn4HwNUbtzoVxAYxkiYBi7aufl4MILv1n
xBqR4L6NNzI0X6cE
Después de copiar y pegar la salida de ese comando como el nuevo valor de requirepass, guardamos y reiniciamos Redis.

```
sudo systemctl restart redis.service
```

Para probar que la contraseña funciona, accedemos a la línea de comandos de Redis e intentamos realizar un cambio (p.e establecer un valor en una clave: 'set key1 10'). Al no haber realizado la autenticación esto no funciona y devuelve un error. Una vez que nos autentiquemos con la contraseña que hemos generado anteriormente y está guardada en el archivo de configuración ('auth mycroftdial'), lo volvemos a realizar y el comando funcionará.

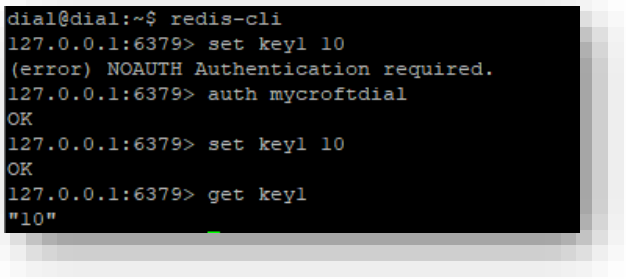

Fig. 39. Comprobación del token de usuario

La otra característica de seguridad integrada en Redis implica cambiar el nombre o deshabilitar por completo ciertos comandos que se consideran peligrosos. Cuando los ejecutan usuarios no autorizados, dichos comandos se pueden usar para reconfigurar, destruir o borrar los datos.

Al igual que la contraseña de autenticación, los comandos de cambio de nombre o deshabilitación se configuran en la misma sección de seguridad del 'redis.conf' archivo. Algunos de los comandos que se consideran peligrosos incluyen: FLUSHDB, FLUSHALL, KEYS , PEXPIRE , DEL , CONFIG , SHUTDOWN , BGREWRITEAOF , BGSAVE , SAVE , SPOP , SREM , RENAME y DEBUG . Esta no es una lista completa, pero cambiar el nombre o deshabilitar todos los comandos en esa lista es un buen punto de partida para mejorar la seguridad del servidor Redis.

Para deshabilitar comandos, cambiamos el nombre a una cadena vacía (representada por un par de comillas sin caracteres entre ellas). Al seguir las siguientes directivas con cadenas vacías, 'FLUSHDB', 'FLUSHALL' y 'DEBUG' se desactivarán en esta configuración de Redis. Para cambiar el nombre de un comando, asignamos otro nombre. Los comandos renombrados deberían ser difíciles de adivinar para otros, pero fáciles de recordar para nosotros. Aquí hemos cambiado el nombre de los comandos SHUTDOWN y CONFIG.

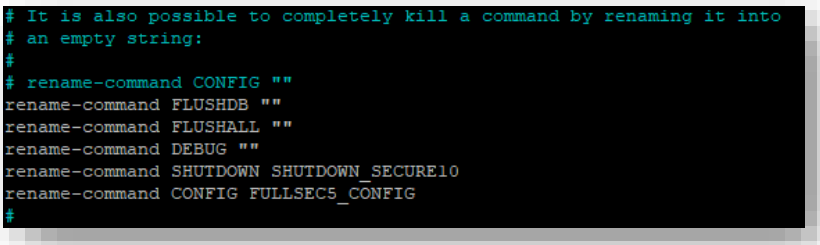

Fig. 40. Configuración de comandos susceptibles

De nuevo, reiniciamos Redis y probamos que se han cambiado correctamente. Supongamos que hemos cambiado el nombre del comando 'CONFIG' a 'FULLSEC5_CONFIG'. Este comando permite que quien lo ejecute interactúe con el archivo de configuración de Redis, donde tenemos las contraseñas guardadas. Primero,

probamos con el comando CONFIG original, y obtenemos un error. Si probamos con el comando renombrado, funciona correctamente.

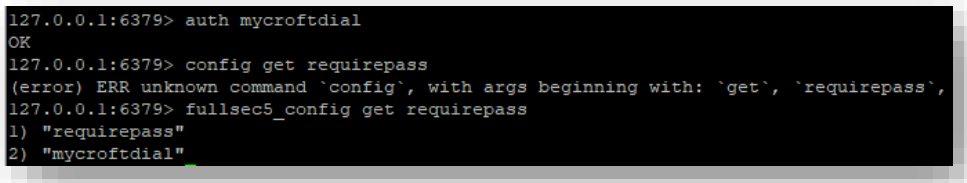

Fig. 41. Comandos renombrados

En este punto ya lo tenemos todo configurado correctamente, validado y sus funciones de seguridad integradas para que sea menos vulnerable a los ataques de usuarios externos.

5.2.3. Configuración general de las APIs

La mayor parte de la configuración de cada API es la misma. Esta sección define los pasos comunes a todas las API. Los pasos específicos de cada API se definirán en sus respectivas secciones.

1. Usamos el sistema de administración de paquetes para instalar Python 3.7, Python 3 pip y Python 3.7 Developer Tools:

```
sudo apt install python3.7 python3-pip python3.7-dev
sudo python3.7 -m pip install pipenv
```

2. Configuramos el directorio de aplicaciones backend:

```
sudo mkdir -p /opt/selene
sudo chown -R dial:users /opt/selene
```

3. Configuramos el directorio de registro:

```
sudo mkdir -p /var/log/mycroft
sudo chown -R mycroft:users /var/log/Mycroft
```

4. Clonamos el repositorio backend de Selene:

```
cd /opt/selene
git clone https://github.com/MycroftAI/selene-backend.git
```

Una vez realizado lo anterior, vamos a crear los entornos virtuales para cada API.

- API de inicio de sesión único (Single Sign On, SSO)

Esta API da soporte al inicio de sesión de los usuarios, validándoles con su correspondiente token. De nuevo, la configuración de servidor recomendada es Ubuntu 18.04 LTS, 1 CPU, 1 GB de RAM y disco de 5 GB. Vamos a crear el entorno virtual:

```
cd /opt/selene/selene-backend/api/sso
pipenv install
```

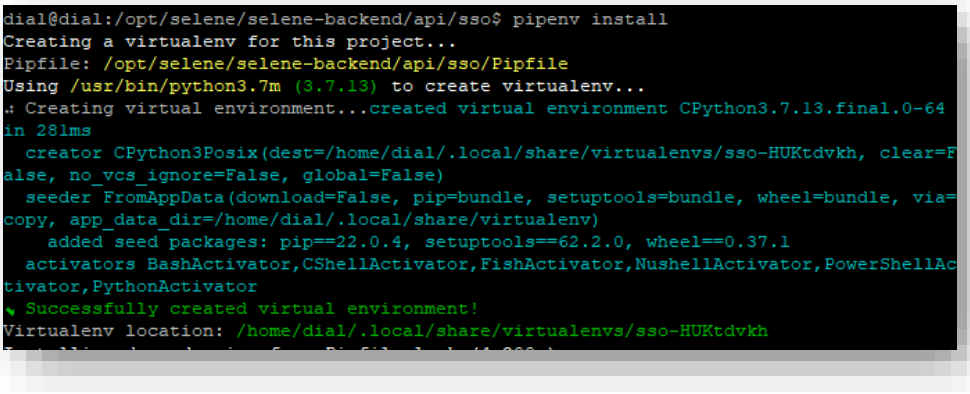

Fig. 42. Entorno virtual para la API de sesión único

- API de cuenta (Account)

Esta API da soporte a la parte de configuración de las cuentas de usuarios. De nuevo, la configuración de servidor recomendada es Ubuntu 18.04 LTS, 1 CPU, 1 GB de RAM y disco de 5 GB. Vamos a crear el entorno virtual:

```
cd /opt/selene/selene-backend/api/account
pipenv install
```

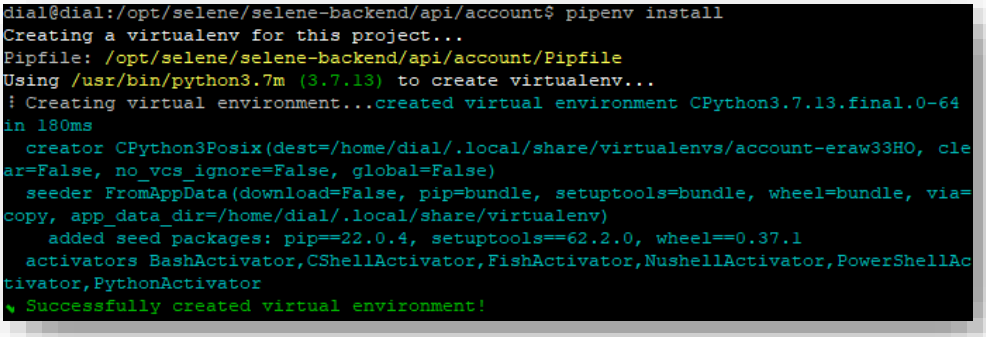

Fig. 43. Entorno virtual para la API de cuenta

- API de “market” (Market)

Esta API da soporte a los menús de instalación de nuevas skills. De nuevo, la configuración de servidor recomendada es Ubuntu 18.04 LTS, 1 CPU, 1 GB de RAM y disco de 10 GB. Vamos a crear el entorno virtual:

```
cd /opt/selene/selene-backend/api/market
pipenv install
```

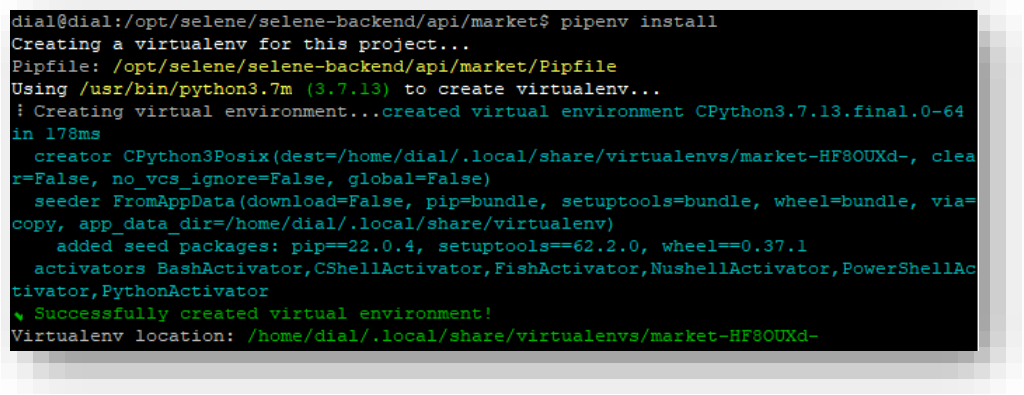

Fig. 44. Entorno virtual para la API de “market”

- API de dispositivo (Public)

Esta API es necesaria para la administración de los dispositivos, es decir, es la API a la que se conectan los dispositivos para su configuración y administración. De nuevo, la configuración de servidor recomendada es Ubuntu 18.04 LTS, 2 CPU, 2 GB de RAM y disco de 50 GB. Vamos a crear el entorno virtual:

```
cd /opt/selene/selene-backend/api/public
pipenv install
```

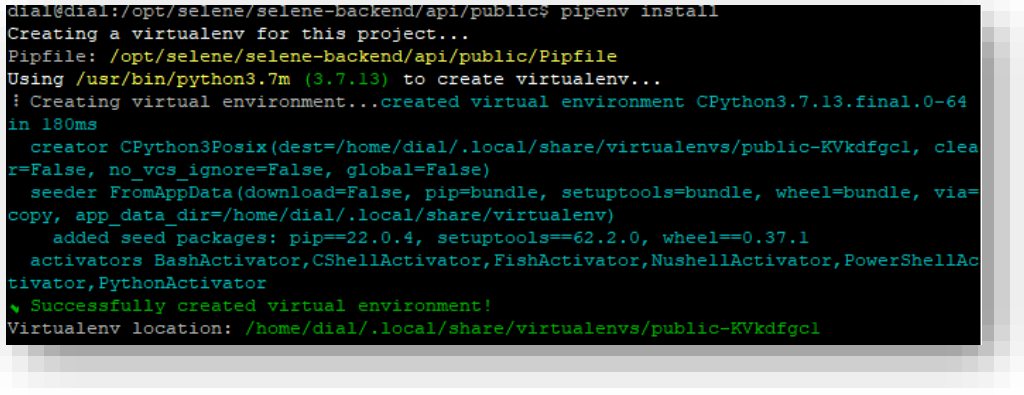

Fig. 45. Entorno virtual para la API de dispositivo

- API “precise”

Esta API es necesaria para la escucha de la “wake word” y de los sonidos ambiente. Es en realidad un servidor de Speech-To-Text. De nuevo, la configuración de servidor recomendada es Ubuntu 18.04 LTS, 1 CPU, 1 GB de RAM y disco de 5 GB.

En el caso del despliegue oficial de Mycroft, esta API se usa para recopilar palabras de activación grabadas por los dispositivos Mycroft oficiales, que se usan para entrenar y mejorar modelos de reconocimiento de “wake word”. En nuestro caso, se iba a utilizar en la definición inicial del proyecto, pero finalmente no se utilizará esta API. Igualmente, las instrucciones para crear el entorno virtual son muy parecidas a las anteriores.

```
cd /opt/selene/selene-backend/api/precise
pipenv install
```

```
dial@dial:/opt/selene/selene-backend/api/precise$ pipenv install
Creating a virtualenv for this project...
Pipfile: /opt/selene/selene-backend/api/precise/Pipfile
Using /usr/bin/python3.7m (3.7.13) to create virtualenv...
⠸ Creating virtual environment...created virtual environment CPython3.7.13.final.0-64
in 189ms
  creator CPython3Posix(dest=/home/dial/.local/share/virtualenvs/precise-Fh25EY_c, cle
ar=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=
copy, app_data_dir=/home/dial/.local/share/virtualenv)
    added seed packages: pip==22.0.4, setuptools==62.2.0, wheel==0.37.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellAc
tivator,PythonActivator
✔ Successfully created virtual environment!
Virtualenv location: /home/dial/.local/share/virtualenvs/precise-Fh25EY_c
```

Fig. 46. Entorno virtual para la API “precise”

5.2.4. Configuración de puertos de las APIs

Por defecto, cada API está configurada para ejecutarse en el puerto 5000. Esto no es un problema si cada una se ejecuta en su propia máquina virtual, pero será un problema si todas las API se ejecutan en el mismo servidor o si el puerto 5000 ya está en uso.

En nuestro caso, actualmente el puerto 5000 está en uso y no se puede utilizar (este puerto se utiliza por el servidor TTS de OpenTTS para sintetizar voz). Para abordar este escenario, cambiamos la numeración de puertos en el archivo ‘uwsgi.ini’ para cada API. Los puertos van del 5001 al 5005.

```
[uwsgi]
master = true
module = market_api.api:market
processes = 4
socket = :5001
die-on-term = true
lazy = true
lazy-apps = true
```

```
[uwsgi]
master = true
module = precise_api.api:precise
processes = 4
socket = :5002
die-on-term = true
lazy = true
lazy-apps = true
```

```
[uwsgi]
master = true
module = public_api.api:public
processes = 10
socket = :5003
die-on-term = true
lazy = true
lazy-apps = true
```

```
[uwsgi]
master = true
module = sso_api.api:sso
processes = 4
socket = :5004
die-on-term = true
lazy = true
lazy-apps = true
```

```
[uwsgi]
master = true
module = account_api.api:acct
processes = 4
socket = :5005
die-on-term = true
lazy = true
lazy-apps = true
```

Fig. 47. Puertos en los que se ejecuta cada API

5.2.5. API de inicio de sesión único (SSO)

En este apartado se detallan todos los pasos necesarios para poner en funcionamiento la API de inicio de sesión único (SSO).

5.2.5.1. JSON Web Tokens

Antes de nada, la aplicación SSO utiliza tres JJON Web Tokens (JWT) para la autenticación. Un JSON Web Token es un token de acceso estandarizado en el RFC 7519 que permite el intercambio seguro de datos entre dos partes. Es decir, lo utilizamos para enviar datos entre aplicaciones o servicios y garantizar que sean válidos y seguros. El primero que se precisa es para la clave de acceso, que se requiere para autenticar a un usuario para las llamadas a la API. El segundo es una clave de actualización que actualiza automáticamente la clave de acceso cuando caduca. Y el tercero es una clave de restablecimiento, que se utiliza en un escenario de restablecimiento de contraseña. Para ello, comenzaremos generando una clave publica y una clave privada con el algoritmo RSA, que se guardaran en dos archivos llamados ‘pkcs8.key’ para la clave privada y ‘publickey.crt’ para la clave pública. El formato correcto es PKCS#8. El siguiente comando genera un par de claves RSA de 2048 bits, las cifra con una contraseña que proporciona y las escribe en un archivo.

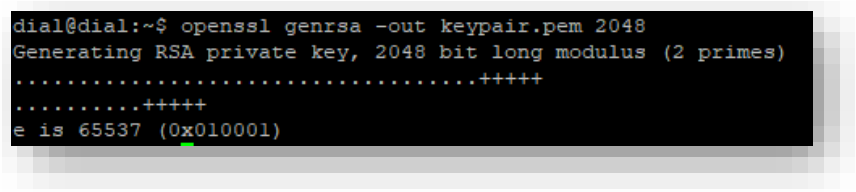

Fig. 48. Claves RSA

A continuación, debemos extraer la contraseña de clave pública y guardarla en ‘publickey.crt’ y la contraseña de clave privada y guardarla en ‘pkcs8.key’.

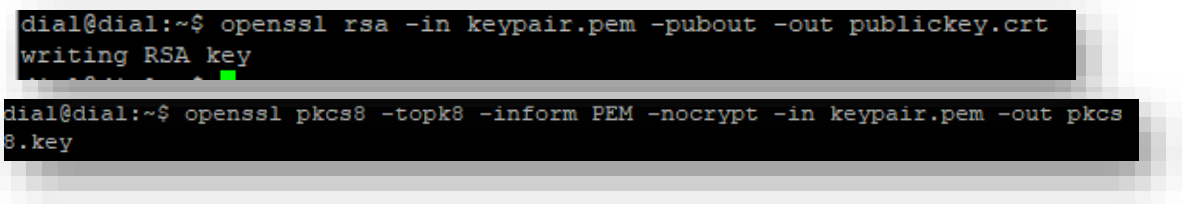

Fig. 49. Archivos finales que contienen las claves

Una vez hecho esto, vamos a generar los tres JWT. Vamos a la web https://jwt.io/, donde seleccionamos el algoritmo RS256. Y rellenamos la cabecera con el tipo de algoritmo y token, la parte de datos con la clave de acceso, y por ultimo las claves pública y privada ya generadas en formato PKCS#8, obteniendo el JSON Web Token codificado:

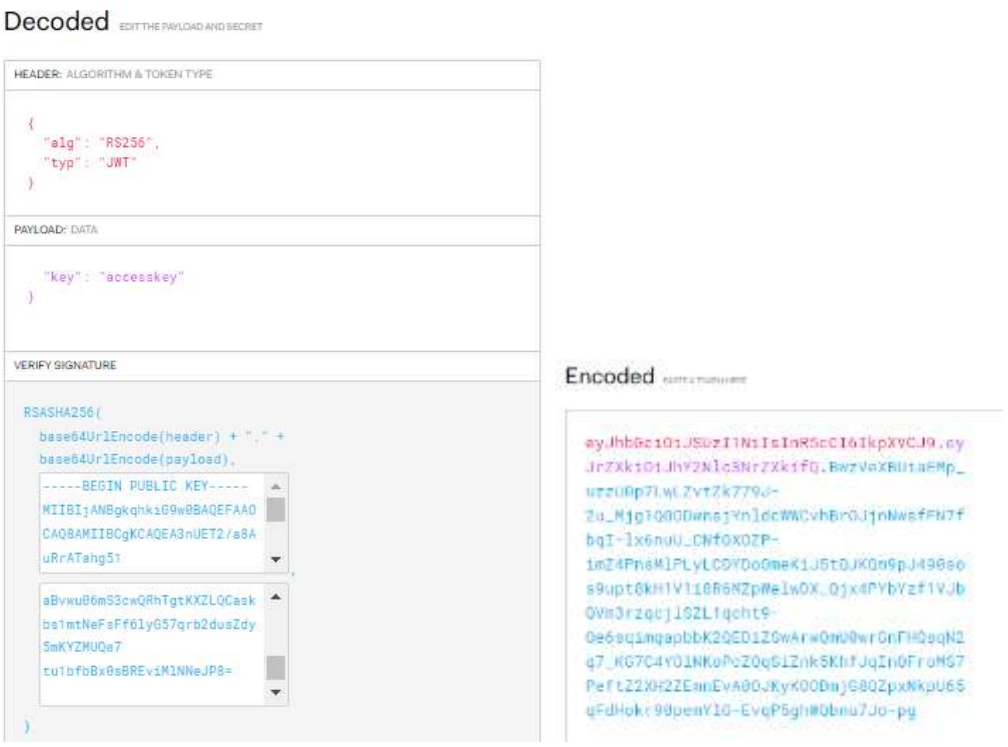

Fig. 50. JWT de acceso

Realizamos los mismos pasos cambiando la clave para obtener tres JWT codificados.

5.2.5.2. Salt

Por otra parte, cualquier dato que pueda identificar a un usuario está encriptado. Por lo que generamos un ‘salt’ que se utilizará con el algoritmo de cifrado. Una de las constantes en seguridad que siempre se exponen, es que las contraseñas nunca deben almacenarse como texto plano porque el riesgo es total, siempre deben estar en forma de hash. El hash es el proceso mediante el cual una contraseña se convierte en una cadena diferente, usando algunos algoritmos complejos, además del uso de datos aleatorios (salt), logrando que el resultado no sea siempre el mismo.

El método usado para generar un hash de contraseña es con el uso del comando ‘mkpasswd’, que nos permite generar contraseñas complejas y seguras a través de diversos algoritmos criptográficos. Por lo tanto, lo primero es agregar el paquete whois del que forma parte esta funcionalidad. Una vez instalado podremos usarlo para codificar nuestra contraseña.

A continuación, tenemos que especificar qué algoritmo hash queremos usar. El método recomendado es ‘sha512crypt’. Tan pronto como ejecutamos el comando, se nos solicita que ingresemos la contraseña que queremos usar. El programa funciona de forma interactiva por razones de seguridad: si tuviéramos que ingresar la contraseña de texto plano directamente como argumento de alguna opción, sería visible en la salida del comando ‘ps’ (estado de procesos) y en el historial de la shell.

Desafortunadamente, Selene no acepta el formato sha512crypt tal cual, sino que necesita una Salt alfanumérica de 16 caracteres. Como incluyo aquí, se generan Salt seguras, pero no funcionará debido al formato. En la subsección sobre el desarrollo Docker se explica esto con más detalle. Para generar la salt de 16 caracteres se pueden usar herramientas online o hacerlo a mano siempre que sea suficientemente aleatorio.

La contraseña hash se devuelve como resultado del comando.

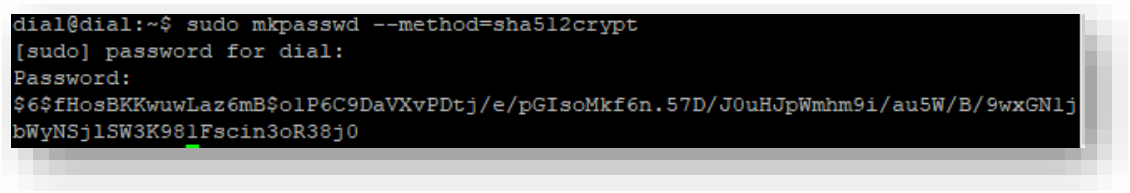

Fig. 51. Clave “salt”

5.2.5.3. Acceso SSO mediante Github

Una vez realizado todo lo necesario para tener contraseñas seguras, se requiere acceso a la API de Github para permitir el inicio de sesión con una cuenta de Github. En principio el desarrollo de este apartado se deja como segunda opción ya que usaremos nuestras propias cuentas creadas en el backend y no autenticaremos mediante Github. En cualquier caso, es útil tenerlo como opción para futuro. Crearemos una “mini-app” para la autenticación mediante Oauth de Github, pero finalmente será la Web de Mycroft SSO quien se encargue de ello.

En este punto, vamos a enfocarnos en lo básico de la autenticación, para poder autenticarnos en la aplicación. Específicamente, vamos a crear un servidor en Ruby (utilizando Sinatra) que implemente el flujo web de la aplicación en varias formas diferentes. Sinatra es un lenguaje específico de dominio (DSL) para crear rápidamente aplicaciones web en Ruby.

```
#Actualizamos la lista de paquetes

sudo apt update
sudo apt upgrade
#Instalamos las dependencias necesarias para instalar Ruby
sudo apt install -y build-essential libssl-dev libreadline-dev zlib1g-dev
# Instalamos rbenv
wget -q https://github.com/rbenv/rbenv-installer/raw/master/bin/rbenv-installer -O- | bash
#A continuación, agregamos ~/.rbenv/bin a su $PATH para poder usar la utilidad de línea de comandos rbenv. Alteramos el archivo ~/.bashrc para que afecte a las futuras sesiones de inicio de sesión:
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
rbenv init
```

```
echo 'if which rbenv > /dev/null; then eval "$(rbenv init -)"; fi' >> ~/.bashrc
#Aplicamos los cambios realizados al archivo ~/.bashrc en la sesión de shell actual:
source ~/.bashrc
#Instalaremos Ruby 2.4.1 con rbenv:
rbenv install 2.4.1
sudo apt-get install ruby-dev
#Una vez completada la instalación, la configuramos como nuestra versión predeterminada de Ruby con el subcomando 'global':
rbenv global 2.4.1
```

Las gemas son el medio de distribución de las bibliotecas de Ruby. Se utiliza el comando 'gem' para administrar estas gemas. Por otra parte, el uso de rbenv proporciona un entorno sólido para desarrollar las aplicaciones de Ruby, ya que permite cambiar de forma sencilla las versiones de Ruby y mantener la misma versión. Vamos a utilizar Sinatra que no es más que un marco web (framework) simple escrito en Ruby útil para aplicaciones web pequeñas, como es nuestro caso.

```
#Instalamos la "gema" Sinatra
sudo gem install Sinatra
#Instalamos la "gema" haml
sudo gem install haml
#Instalamos la "gema" rest-client
gem install rest-client
#Bundler es una herramienta que administra dependencias de gemas para proyectos.
sudo gem install bundler
#Creamos un "Gemfile"
sudo bundler init
sudo bundler install
ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
echo 'eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"' >> /home/dial/.profile
eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"
```

Aquí usamos la fuente de rubygems para especificar dónde obtener las dependencias de Sinatra, además, hemos añadido cualquier otra dependencia que vayamos a necesitar:

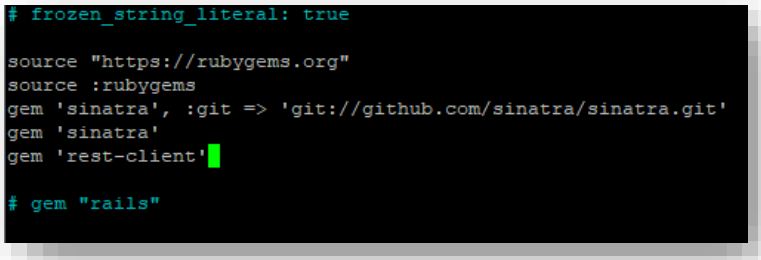

Fig. 52. Archivo Gemfile en root

Completado el proceso de instalación de Sinatra, necesitamos registrar la aplicación. Uno de los aspectos críticos de la seguridad informática es poder proporcionar una experiencia de acceso sin interrupciones y de inicio de sesión único (SSO) entre varios dispositivos. Y en este campo, OAuth es uno de los más utilizados. Cuando hablamos de OAuth (Open Authorization), nos referimos a una solución de administración de identidad y acceso (IAM). Su finalidad es otorgar autorizaciones a los usuarios.

Mediante la plataforma de desarrollo Github, podemos crear y registrar una App de OAuth bajo nuestra cuenta personal o bajo cualquier organización en la que tengas acceso. El flujo de la aplicación web para autorizar a los usuarios para su propia aplicación es la siguiente:

1. Los usuarios son redirigidos para solicitar su identidad de GitHub
2. Los usuarios son redirigidos a su sitio por GitHub
3. Su aplicación accede a la API con el token de acceso del usuario

A cada aplicación de OAuth que se registra se le asigna una ID de Cliente única y un número Secreto de Cliente, el cual no puede compartirse. Hemos registrado nuestra aplicación bajo el nombre “Mycroft Single Sign On Api”.

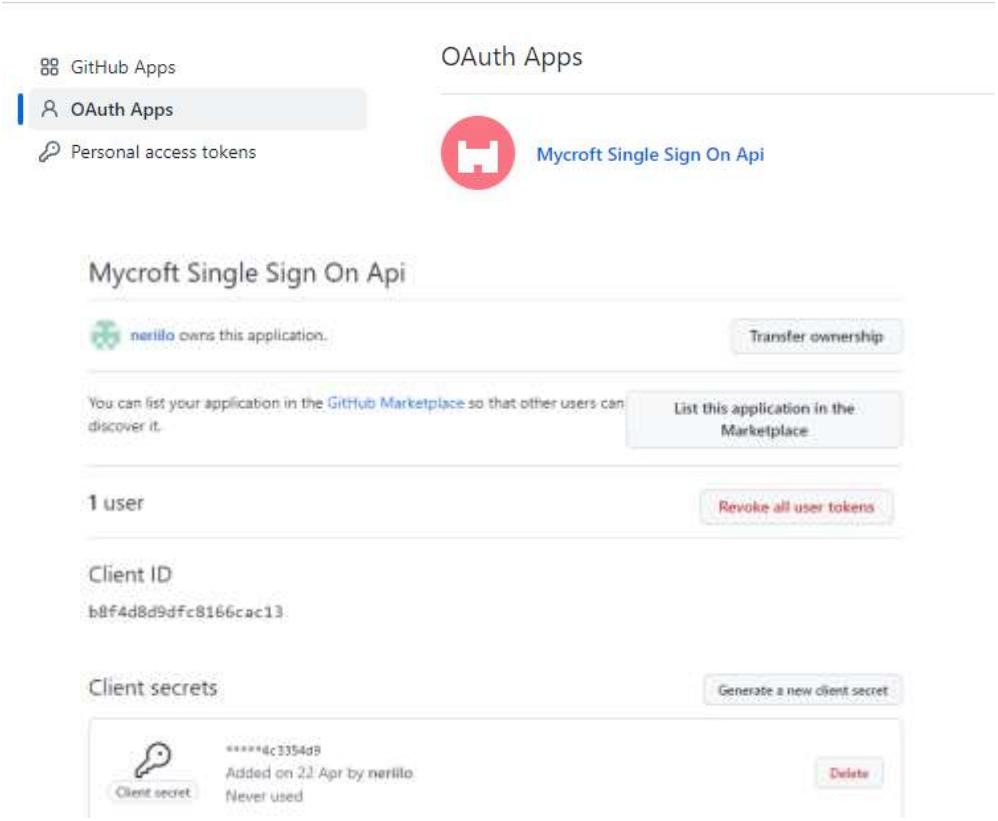

Fig. 53. ID de cliente para la app

Podemos llenar toda la información como decidamos, con excepción de la URL de rellamado para la autorización. Esta la parte más importante para configurar la aplicación. Es la URL de rellamado a la cual GitHub devuelve al usuario después de una autenticación exitosa. La aplicación encargada de recibir el token de autorizacion de

Github es el módulo SSO de Selene-UI. Por lo tanto la URL de callback sería la url que tiene registrada para ello http://158.42.167.31:8081 (usando el puerto 8081 asumiendo que es el que le toca a SSO en Apache). El módulo Selene UI SSO luego redirigirá el token al módulo Selene Backend SSO para autenticarlo.

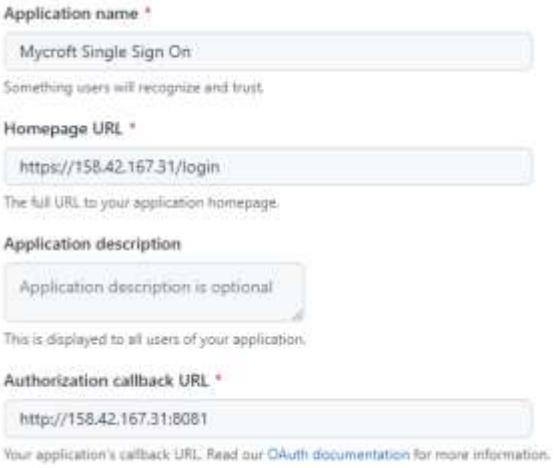

Fig. 54. Configuración de la app SSO

Ahora vamos a pasar a la autorización del usuario. Vamos a comenzar a llenar nuestro servidor. Creamos un archivo que se llame 'server.rb' con la siguiente información:

```
Cat > server.rb (en /)
require 'sinatra'
require 'rest-client'
require 'json'
CLIENT_ID = ENV['GH_BASIC_CLIENT_ID']
CLIENT_SECRET = ENV['GH_BASIC_SECRET_ID']
get '/' do
  erb :index, :locals => {:client_id => CLIENT_ID}
end
```

La ID de cliente y las claves secretas de cliente vienen en la página de configuración de la aplicación. Nunca deberíamos almacenar estos valores en GitHub ni en otro lugar público. Vamos a almacenarlas como variables de entorno:

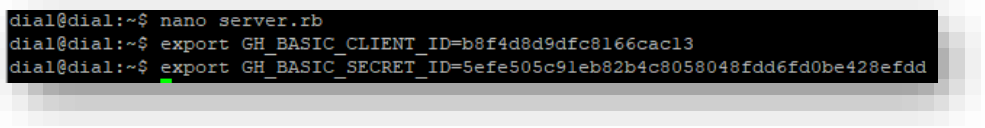

Fig. 55. Variables de entorno que contienen las claves

Posteriormente, vamos a realizar una prueba para comprobar que la API funciona correctamente. Introducimos el siguiente código en views/index.erb para poder probarlo y visualizarlo en localhost:

```
<html>
  <head>  </head>
  <body>
    <p>
      Well, hello there!
```

```
  </p>
  <p>
    We're going to now talk to the GitHub API. Ready?
    <a href="https://github.com/login/oauth/authorize?scope=user:email&client_id=<%=
client_id %>">¡Haz clic aquí</a> para comenzar!
  </p>
  <p>
    Si ese enlace no funciona, recuerda proporcionar tu propia <a href="/apps/building-oauth-
apps/authorizing-oauth-apps/">ID de cliente</a>!
  </p>
 </body>
</html>
```

Instalamos Lynx para poder acceder al localhost mediante otro terminal:

```
sudo apt install Lynx
Lynx 127.0.0.1:4567
```

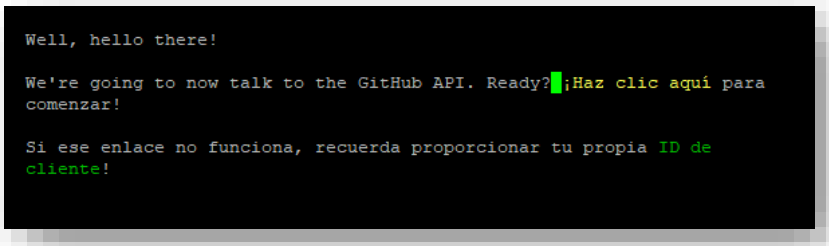

Fig. 56. Test de funcionamiento de la API

La API pide al usuario que se autorice con su cuenta de Github, que sería el primer paso, como se observa en la Fig. 57.

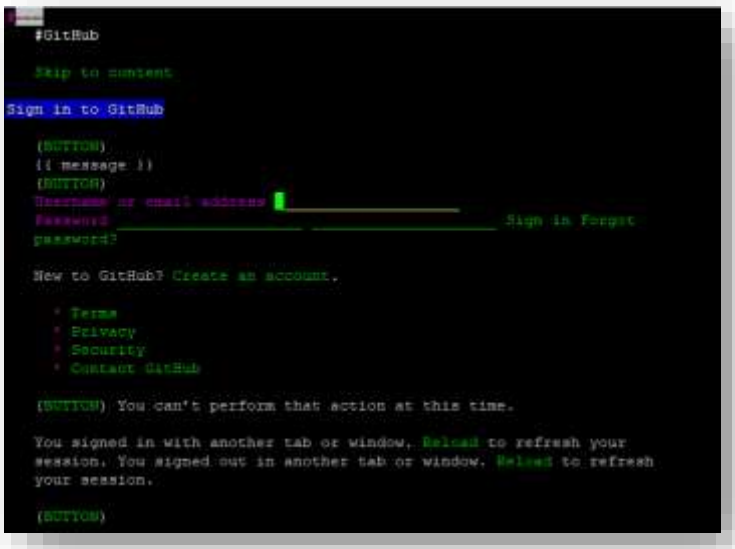

Fig. 57. Primer paso. Log in en cuenta Github

A continuación, te indica que verifiques tu cuenta de email mediante un código enviado.

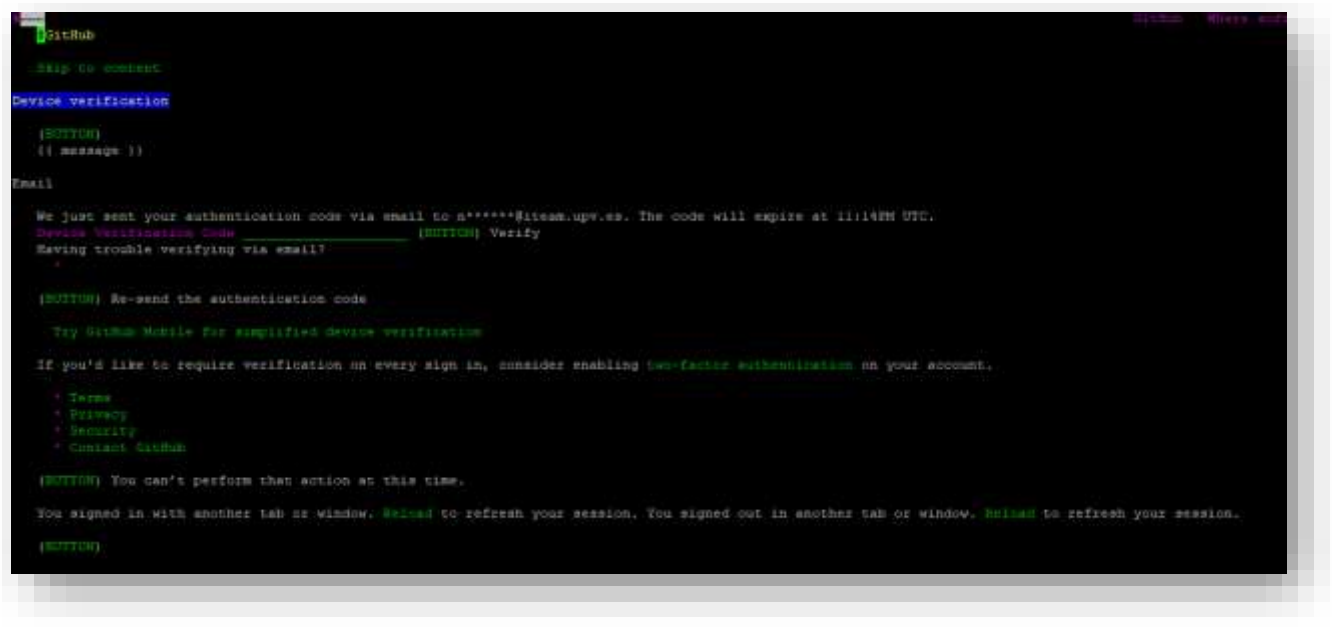

Fig. 58. Verificación de email

Después de autorizar al usuario, Github desconoce donde dejar al usuario, por lo que tenemos que proporcionar una ruta para la URL de rellanado. Además, podemos mejorar este proceso implementando la autenticación “persistente”, es decir, para que los usuarios no tengan que iniciar sesión en la app cada vez que necesiten acceder a la página web.

```
dial@dial:~$ ruby advanced_server.rb
[2022-06-18 22:58:05] INFO  WEBrick 1.3.1
[2022-06-18 22:58:05] INFO  ruby 2.4.1 (2017-03-22) [x86_64-linux]
== Sinatra (v2.2.0) has taken the stage on 4567 for development with backup from WEBrick
[2022-06-18 22:58:05] INFO  WEBrick::HTTPServer#start: pid=150795 port=4567
127.0.0.1 - - [18/Jun/2022:22:58:14 +0000] "GET / HTTP/1.1" 200 450 0.0098
127.0.0.1 - - [18/Jun/2022:22:58:14 UTC] "GET / HTTP/1.0" 200 450
- -> /
```

Fig. 59. Autenticación persistente

```
require 'sinatra'
require 'rest_client'
require 'json'
CLIENT_ID = ENV['GH_BASIC_CLIENT_ID']
CLIENT_SECRET = ENV['GH_BASIC_SECRET_ID']
use Rack::Session::Pool, :cookie_only => false
#Para verificar si el usuario ya se autenticó:
def authenticated?
  session[:access_token]
end

#Si no se ha autenticado, realiza el flujo de OAuth y actualiza la sesión con el token otorgado:
def authenticate!
  erb :index, :locals => {:client_id => CLIENT_ID}
end
```

```
get '/' do
  if !authenticated?
    authenticate!
  else
    access_token = session[:access_token]
    scopes = []

    begin
      auth_result = RestClient.get('https://api.github.com/user',
                                   {:params => {:access_token => access_token},
                                    :accept => :json})
    rescue => e
      session[:access_token] = nil
      return authenticate!
    end
    if auth_result.headers.include? :x_oauth_scopes
      scopes = auth_result.headers[:x_oauth_scopes].split(', ')
    end
#Con el token de acceso, podemos hacer solicitudes autenticadas como el inicio de sesión:
    auth_result = JSON.parse(auth_result)
    if scopes.include? 'user:email'
      auth_result['private_emails'] =
        JSON.parse(RestClient.get('https://api.github.com/user/emails',
                    {:params => {:access_token => access_token},
                     :accept => :json}))
    end
    erb :advanced, :locals => auth_result
  end
end

#Para proporcionar el rellanado:
get '/callback' do
  session_code = request.env['rack.request.query_hash']['code']

  result = RestClient.post('https://github.com/login/oauth/access_token',
                           {:client_id => CLIENT_ID,
                            :client_secret => CLIENT_SECRET,
                            :code => session_code},
                            :accept => :json)
  session[:access_token] = JSON.parse(result)['access_token']
  redirect '/'
end
```

5.2.5.4. Sengrid

Por otra parte, la funcionalidad de restablecimiento de contraseña envía un correo electrónico al usuario con un enlace para restablecer su contraseña. Selene usa SendGrid para enviar estos correos electrónicos, por lo que se requiere una cuenta de SendGrid y una clave API.

Fig. 60. API Sengrid

5.2.6. Servicios para las APIs

5.2.6.1. Servicio API SSO

Continuamos definiendo un servicio “systemd” para ejecutar las APIs. El servicio define las variables de entorno que utilizan las claves de la API, incluidas la secreta, generadas en los pasos anteriores. Vamos rellenando los datos en función a nuestra aplicación:

```
sudo nano /etc/systemd/system/sso_api.service
[Unit]
Description=Mycroft Single Sign On Api
After=network.target
[Service]
User=dial
Group=www-data
Restart=always
Type=simple
WorkingDirectory=/opt/selene/selene-backend/api/sso
ExecStart=/usr/local/bin/pipenv run uwsgi --ini uwsgi.ini
Environment=DB_HOST=127.0.0.1
Environment=DB_NAME=mycroft
Environment=DB_PASSWORD=XXX
Environment=DB_PORT=5432
Environment=DB_USER=selene
Environment=GITHUB_CLIENT_ID= b8f4d8d9dfc8166cac13
Environment=GITHUB_CLIENT_SECRET= XXX
Environment=JWT_ACCESS_SECRET= XXX
Environment=JWT_REFRESH_SECRET= XXX
Environment=JWT_RESET_SECRET= XXX
Environment=SALT=XXX
Environment=SELENE_ENVIRONMENT=test
```

```
Environment=SENDGRID_API_KEY=XXX
Environment=SSO_BASE_URL= http://158.42.167.31:5004
[Install]
WantedBy=multi-user.target
```

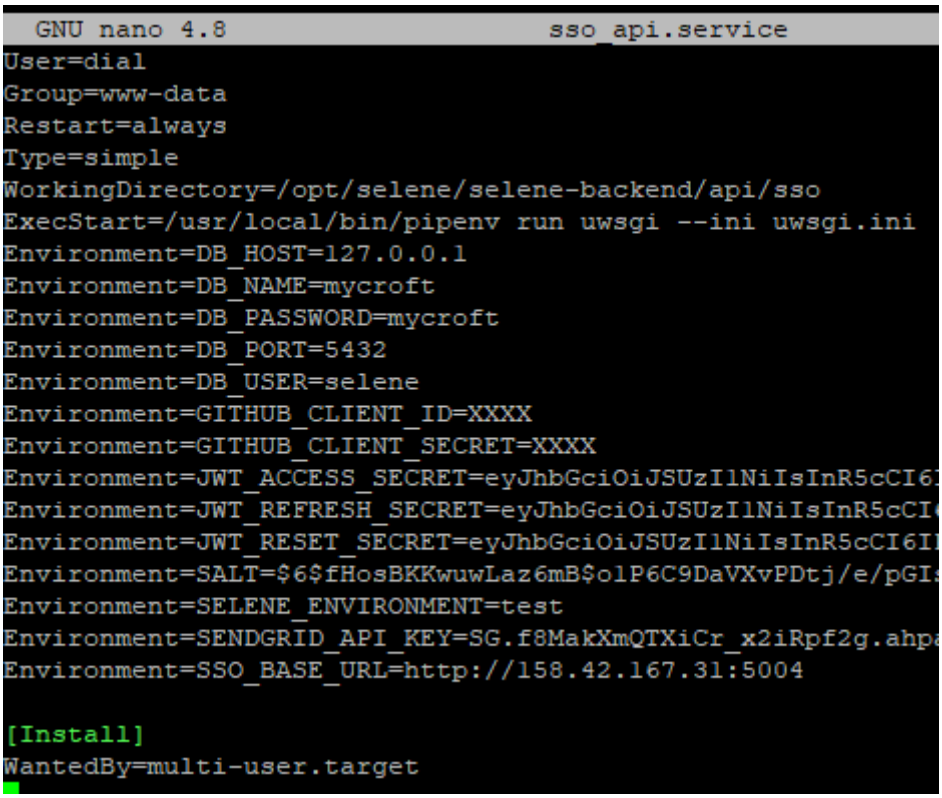

Fig. 61. Servicio “systmd” para la API SSO

Una vez completado, lo iniciamos y lo configúrelo para que se inicie en el arranque:

```
sudo systemctl start sso_api.service
sudo systemctl enable sso_api.service
```

Ya tenemos todo correctamente configurado para la API SSO.

5.2.6.2. Servicio API de inicio de cuenta (Account)

La API de la cuenta utiliza el mismo mecanismo de autenticación que la API de inicio de sesión único. Las variables de entorno JWT_ACCESS_SECRET, JWT_REFRESH_SECRET y SALT deben tener los mismos valores que los de la API de inicio de sesión único. Esta aplicación utiliza la base de datos de Redis, por lo que el servicio necesita saber dónde reside. Vamos a definir el servicio “systemd” para ejecutar la API:

```
[Unit]
Description=Mycroft Account API
After=network.target
```

```
[Service]
User=dial
Group=www-data
Restart=always
Type=simple
WorkingDirectory=/opt/selene/selene-backend/api/account
ExecStart=/usr/local/bin/pipenv run uwsgi --ini uwsgi.ini
Environment=DB_HOST=127.0.0.1
Environment=DB_NAME=mycroft
Environment=DB_PASSWORD=<selene user database password>
Environment=DB_PORT=5432
Environment=DB_USER=selene
Environment=JWT_ACCESS_SECRET=<same as value for single sign on>
Environment=JWT_REFRESH_SECRET=<same as value for single sign on>
Environment=OAUTH_BASE_URL=http://127.0.0.1:4567/callback
Environment=REDIS_HOST=127.0.0.1
Environment=REDIS_PORT=6379
Environment=SELENE_ENVIRONMENT=test
Environment=SALT=<same as value for single sign on>
[Install]
WantedBy=multi-user.target
```

Iniciamos el servicio lo configuramos para que se inicie en el arranque:

```
sudo systemctl start account_api.service
sudo systemctl enable account_api.service
```

5.2.6.3. Servicio API de “market”

La API del mercado utiliza el mismo mecanismo de autenticación que la API de inicio de sesión único, y la de cuenta. Las variables de entorno JWT_ACCESS_SECRET, JWT_REFRESH_SECRET y SALT deben tener los mismos valores. Esta aplicación también utiliza la base de datos de Redis, por lo que el servicio necesita saber dónde reside. Definimos el servicio “systemd” para ejecutar la API.

```
sudo nano /etc/systemd/system/market_api.service

[Unit]
Description=Mycroft Marketplace API
After=network.target
[Service]
User=dial
Group=www-data
Restart=always
Type=simple
WorkingDirectory=/opt/selene/selene-backend/api/market
```

```
ExecStart=/usr/local/bin/pipenv run uwsgi --ini uwsgi.ini
Environment=DB_HOST=<db host IP address or name>
Environment=DB_NAME=mycroft
Environment=DB_PASSWORD=<selene user database password>
Environment=DB_PORT=5432
Environment=DB_USER=selene
Environment=JWT_ACCESS_SECRET=<same as value for single sign on>
Environment=JWT_REFRESH_SECRET=<same as value for single sign on>
Environment=OAUTH_BASE_URL=<url for oauth service>
Environment=REDIS_HOST=<IP address or name of redis host>
Environment=REDIS_PORT=6379
Environment=SELENE_ENVIRONMENT=<test/prod>
Environment=SALT=<same as value for single sign on>
[Install]
WantedBy=multi-user.target
```

De nuevo, iniciamos el servicio y lo configuramos para que se inicie en el arranque:

```
sudo systemctl start market_api.service
sudo systemctl enable market_api.service
```

La API de “market” asume que las skills que proporciona a la aplicación web están en la base de datos de Postgres. Para llevarlos allí, debemos ejecutar un script para descargarlas de una cuenta de Github que puede ser cualquier cuenta, simplemente para descargar las skills del repositorio de Mycroft.

El script requiere las variables de entorno GITHUB_USER, GITHUB_PASSWORD, DB_HOST, DB_NAME, DB_USER y DB_PASSWORD para ejecutarse:

```
Export GITHUB_USER=neriilo
Export GITHUB_PASSWORD=XXXX
Export DB_HOST=127.0.0.1
Export DB_NAME=mycroft
Export DB_USER=selene
Export DB_PASSWORD=XXXX

cd /opt/selene/selene-backend/batch
pipenv install
pipenv run python load_skill_display_data.py --core-version <specify core version>
```

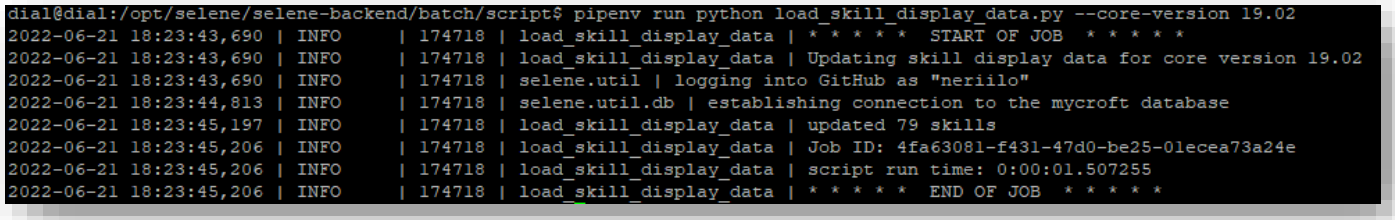

Fig. 62. Proceso de implementación de skills

5.2.6.4. Servicio API de dispositivo

La API del dispositivo utiliza los mismos pasos que para las APIs anteriores, pero en este caso, se requiere proporcionar las claves para las APIs concretas que utilizan las skills. Por ejemplo, podemos configurar skills básicas. La skill para la meteorología, utiliza la API Open Weather Map y requiere su clave privada. El motor de voz a texto, si utilizamos la API STT de Google, también requiere su propia clave. O si utilizamos Wolfram Alpha, se requiere la clave para la API de Wolfram Alpha. Si queremos que el asistente mande y reciba correos por ejemplo para concertar citas con un terapeuta, incluimos una configuración de email.

Definimos el servicio “systemd” para ejecutar la API

```
[Unit]
Description=Mycroft Public API
After=network.target
[Service]
User=dial
Group=www-data
Restart=always
Type=simple
WorkingDirectory=/opt/selene/selene-backend/api/public
ExecStart=/usr/local/bin/pipenv run uwsgi --ini uwsgi.ini
Environment=DB_HOST=127.0.0.1
Environment=DB_NAME=mycroft
Environment=DB_PASSWORD=mycroft
Environment=DB_PORT=5432
Environment=DB_USER=selene
Environment=EMAIL_SERVICE_HOST=<email host>
Environment=EMAIL_SERVICE_PORT=<email port>
Environment=EMAIL_SERVICE_USER=<email user>
Environment=EMAIL_SERVICE_PASSWORD=<email password>
Environment=GOOGLE_STT_KEY=<Google STT API key>
Environment=JWT_ACCESS_SECRET=<same as value for single sign on>
Environment=JWT_REFRESH_SECRET=<same as value for single sign on>
```

```
Environment=OAUTH_BASE_URL=<url for oauth service>
Environment=OWM_KEY=<Open Weather Map API Key>
Environment=OWM_URL=https://api.openweathermap.org/data/2.5
Environment=REDIS_HOST=<IP address or name of redis host>
Environment=REDIS_PORT=6379
Environment=SELENE_ENVIRONMENT=<test/prod>
Environment=SALT=<same as value for single sign on>
Environment=WOLFRAM_ALPHA_KEY=<Wolfram Alpha API Key
Environment=WOLFRAM_ALPHA_URL=https://api.wolframalpha.com
Environment=DB_SSLMODE=prefer
[Install]
WantedBy=multi-user.target
```

5.3. Interfaz gráfica de usuario (GUI)

Una vez que se completa la configuración de las bases de datos y de las APIs (Backend), el siguiente paso es configurar la interfaz gráfica de usuario (Frontend). El repositorio de Selene UI para la interfaz de usuario contiene todo lo necesario para configurar las aplicaciones web. Selene proporciona los servicios utilizados por Mycroft Core para administrar dispositivos, skills y configuraciones. Como ya mencionamos en capítulos anteriores, consta de dos repositorios. El que vamos a utilizar a lo largo de esta sección es el que contiene las aplicaciones web GUI creadas con el framework Angular, la capa de acceso a datos, APIs y scripts.

Hay tres aplicaciones web definidas en este repositorio, inicio de sesión único (SSO), de administración de cuentas (Account) y "skills market" (Market). Cada aplicación está diseñada para ejecutarse independientemente de las demás. Este repositorio también incluye dos bibliotecas que contienen código común a cada una de las aplicaciones.

1. Instalamos Angular 7 (npm 6.14.12) y node.js (v10.24.1) con sus paquetes necesarios y descargamos este repositorio a través de git:

```
Mkdir gui-selene
cd gui-selene
git clone https://github.com/MycroftAI/selene-ui.git
cd selene-ui
curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -
sudo apt update
sudo apt -y install nodejs
sudo npm install -g @angular/cli@7
cd ..
ng –version
cd selene-ui
npm install
```

Fig. 63. Versión Angular instalada

2. Compilamos el código de Angular TypeScript. De forma predeterminada, el código se compila en el directorio "selene-ui/dist":

```
ng build --project shared
ng build --project globalnav
ng build --project account --configuration test
ng build --project market --configuration test
ng build --project sso --configuration test
```

3. Una vez compilado, copiamos el código en el directorio /var/www del servidor para cada aplicación. Los módulos shared y globalnav son bibliotecas que deben copiarse en todos los servidores. Los módulos account, market y sso solo necesitan copiarse en sus respectivos servidores. De esta forma tenemos que renombrar los "index.html" generados con nombres distintos para utilizarlo con Apache como se explica en el siguiente capítulo.

5.4. Servidor Apache

Apache es un servidor web HTTP de código abierto el que nos va a permitir servir contenido (aplicación web UI) desde los clientes web. Comenzamos escribiendo el comando que iniciará la instalación de Apache:

```
sudo apt install apache2
```

Antes de nada, comprobamos los perfiles por defecto para Apache en UFW:

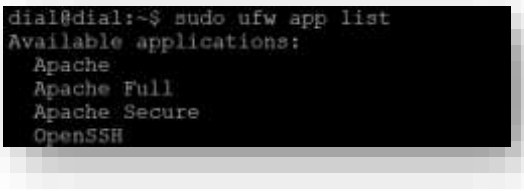

Fig. 64. Perfiles por defecto Apache en UWF

- Apache: Apertura solo del puerto 80, que permite tráfico sin cifrar (No seguro).
- Apache Full: Apertura tanto del puerto 80 como del puerto 443, permitiendo el tráfico cifrado (Seguro).
- Apache Secure: Apertura solo del puerto 443, permitiendo conexiones seguras (HTTPS).

-Ahora permitimos el perfil “Apache” y verificamos que el servicio Apache esté ejecutándose correctamente, con el siguiente comando:

```
sudo ufw allow 'Apache'
sudo ufw status
sudo systemctl status apache2
```

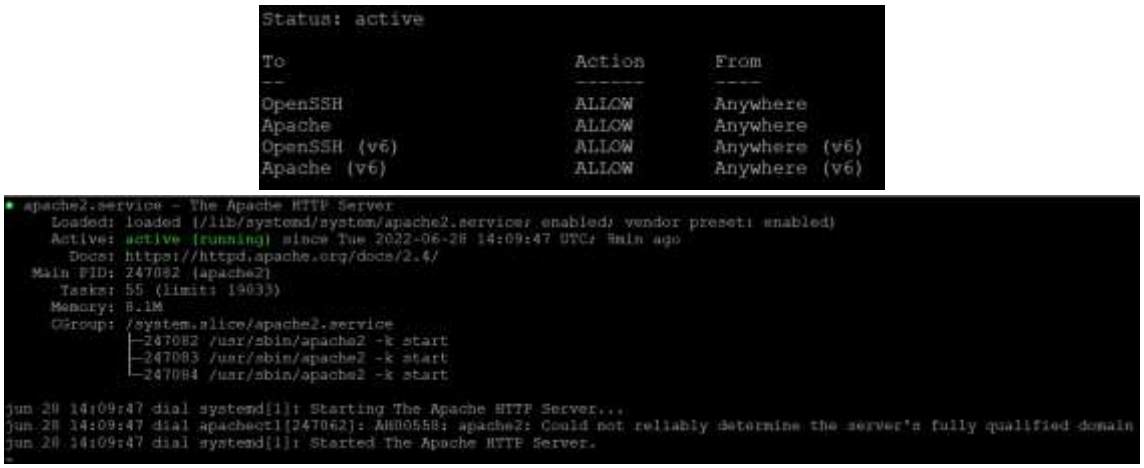

Fig. 65. Perfil “Apache”

-Comprobamos que funciona correctamente, accediendo desde el navegador ingresando la IP publica del servidor (127.0.0.1) o con la IP del servidor 158.42.167.31:

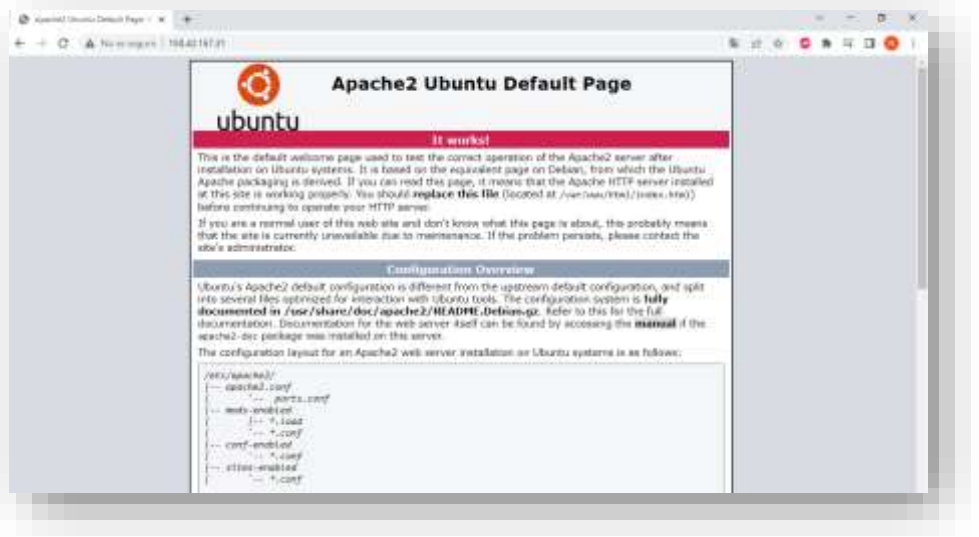

Fig. 66. Comprobación del servicio Apache en el navegador

-Antes de nada, realizamos las configuraciones necesarias en los dominios que vienen por defecto en los archivos de configuración. Vamos a cambiar todas las referencias de los dominios Mycroft (account.mycroft.ai, sso.mycroft.ai, etc) a la IP nuestra: http://158.42.167.31 con los puertos correspondientes a cada aplicación web. Definimos tres sitios: account, sso y market, con sus respectivos puertos 8081, 8082 y 8083.
Esto se cambia en los archivos /selene-backend/shared/selene/api/base_config.py y en todos los /selene-backend/projects/*/src/environments/environment*.ts:

```
find account / -name "*.js" -print | xargs sed -i
"s/account.mycroft.ai/http://158.42.167.31:8081
find account / -name "*.js" -print | xargs sed -i
"s/sso.mycroft.ai/http://158.42.167.31:8082
find account / -name "*.js" -print | xargs sed -i
"s/market.mycroft.ai/http://158.42.167.31:8083
```

-Además, cambiamos en el archivo de configuración "/home/dial/gui-selene/selene-ui/src/environments /environment.test.ts la url con nuestra ip:

```
export const environment = {
    production: false,
    mycroftUrls: {
        account: 'http://158.42.167.31/account',
        chat: 'http://chat.mycroft.ai',
        forum: 'http://community.mycroft.ai',
        marketplace: 'http://158.42.167.31/market',
        mimic: 'http://mimic.mycroft.ai',
        singleSignOn: 'http://158.42.167.31/sso',
        translate: 'http://translate-test.mycroft.ai',
        wordPress: 'http://test.mycroft.ai'
    }
};
```

Fig. 67. Archivo "environment.test.ts"

-Una vez compilado vamos a guardar los correspondientes "index,html" en los tres sitios creados dentro de "/www" de la siguiente forma:

```
Cp -a account/. /var/www/html/account
Cp -a sso/. /var/www/html/sso
Cp -a market/. /var/www/html/market
```

-Creamos los archivos de configuración y los rellenamos con la siguiente información:

```
cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/sso.conf
cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-
available/account.conf
cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/market.conf
```

Fig. 68. Archivo de configuración para cada uno de los “sites”

-Deshabilitamos el archivo predeterminado de configuración, habilitamos los creados y reiniciamos Apache:

```
a2dissite 000-default.conf

systemctl restart apache2
```

-En este punto ya tenemos las aplicaciones web sirviendose en las direcciones: 158.42.167.31:8081, 158.42.167.31:8082 y 158.42.167.31:8083. A continuación, podemos ver las vistas de la aplicación web:

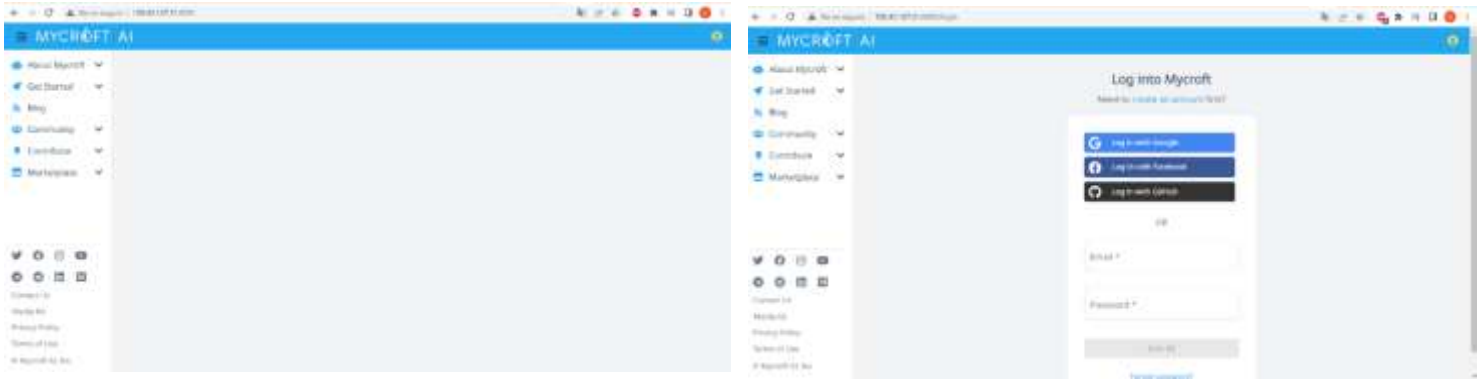

Fig. 69. Vistas “account” y “login” con Apache

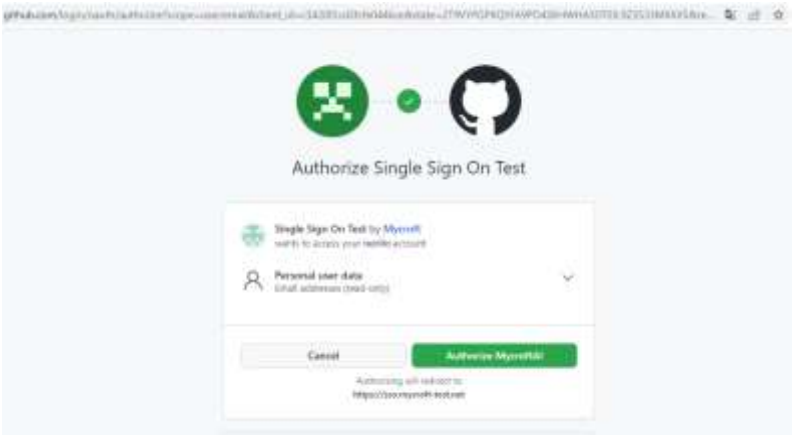

Fig. 70. Autenticación por GitHub

Llegado este punto, faltaría añadir un proxy para servir todas las urls de Mycroft y continuar con tareas de depuración, pero debido a la dificultad de comprender las instrucciones originales de Myrcoft y a la cantidad de archivos configurados ha sido complicado continuar y realizar todo el enrutamiento correctamente. Es por ello que se decidió empezar de nuevo realizándolo con Docker

5.5 Docker

En este apartado, se realiza la “dockerización” de todo lo desplegado previamente. Una vez llegado el punto de desplegar Apache, aparecieron muchos problemas de enrutamiento y de depuración de difícil solución, por lo que el desarrollo se decidió enfocar a contenedores Docker.

Finalmente, se va a utilizar todo lo anterior como base a seguir y orientarlo a un despliegue con Docker donde la configuración está más organizada y donde se definen claramente los parámetros y servicios para instalar Selene. Adicionalmente, se añadirá un proxy con Nginx para que sirva correctamente las urls.

5.5.1. Arquitectura Docker

El esquema Docker de interconexión de contenedores que contiene tanto los contenedores desarrollados para el frontend como los correspondientes para el backend, donde se incluyen los puertos y los correspondientes para las bases de datos, sería el siguiente:

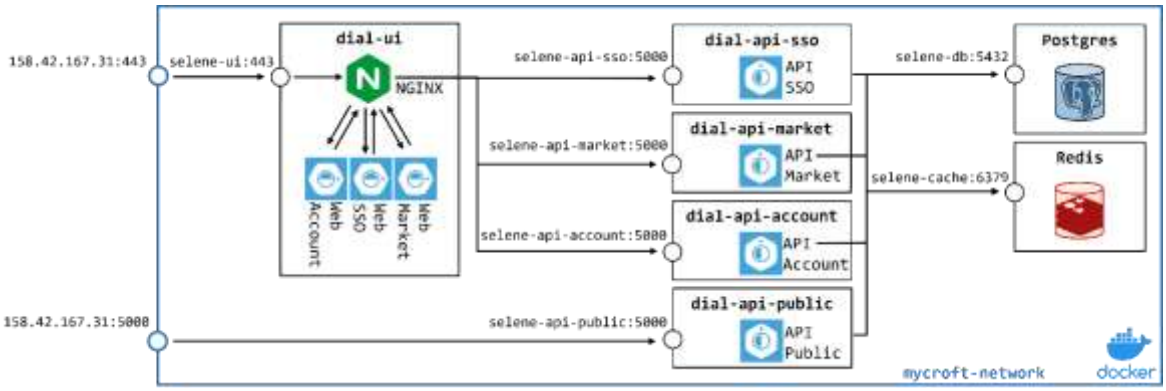

Fig. 71. Arquitectura Backend y Frontend con despliegue Docker

Por una parte, tenemos el Frontend con las tres aplicaciones web que serán desplegadas bajo el mismo servidor Nginx a las que se accede por el puerto 443. Por otro lado, tenemos el backend con los contenedores que servirán a las aplicaciones web. A la API "public" se tendrá acceso de forma externa mediante el puerto 5000. Por último tenemos los contenedores para la base de datos PostgreSQL y Redis.

5.5.2. Archivos Docker de configuración

Estos archivos se han creado "desde cero" para convertir las instrucciones de instalación recopiladas a Docker.

/selene-backend/Dockerfile-api-base

Este Dockerfile genera una imagen Docker base con todas las variables de configuración necesarias para el resto de las imágenes. Se han recogido todas las variables de entorno que deben personalizarse para ejecutar los distintos módulos Selene y se han unificado aquí, en un único lugar.

/selene-backend/Dockerfile-init-db

Este Dockerfile realiza la imagen que crea las tablas y esquemas de la BD e inicializa su contenido. Es necesario que estén presentes los archivos privacy_policy.md y terms_of_use.md junto a este Dockerfile. Estos son respectivamente el texto de la política de privacidad y los términos de uso que aparecen durante la creación de una nueva cuenta, pudiendo ser modificados si fuera necesario.

/selene-backend/Dockerfile-init-defaults

Crea la imagen que se usa para inicializar las tablas de palabra de activación (wake word) con los valores por defecto. Es necesario que esté presente el archivo neo4j-postgres.py junto a este Dockerfile. Aquí están las instrucciones para insertar en la BD las wake word por defecto (como "Hey mycroft").

/selene-backend/Dockerfile-init-market

Este Dockerfile crea la imagen que inicializa el contenido de Skills en la BD. Añade las skills oficiales por defecto al Marketplace.

/selene-backend/Dockerfile-api-*

Se han creado cuatro archivos Dockerfile, uno por cada módulo (account, market, sso, public) que crean una imagen cada uno para ejecutarlos. Son equivalentes a los servicios de Linux creados en subsecciones anteriores.

.

/selene-ui/Dockerfile-ui

Este Dockerfile genera una imagen con los tres frontend de Selene: Account, Market y SSO, combinados en un único servidor Nginx. Tener cada uno de los frontend como un servidor por separado complicaba el enrutamiento y el proxy. En vez de eso se despliega todo el código compilado en una única carpeta combinada dentro de la imagen, renombrando los respectivos index.html a account.html, market.html y sso.html. Nginx se encargará de enrutarlos adecuadamente, para lo cual usa el archivo default.conf.

/selene-ui/default.conf

Archivo de configuración del servidor Nginx para servir los tres frontend de Selene como un único servidor y hacer de proxy hacia las APIs. Es decir, se usa para desplegar, conectar y enrutar todas las peticiones entre todos los módulos de Selene, conforme al esquema del anexo. Las URL web de account, market y sso se enrutan a sus respectivos index.html que han sido renombrados como hacíamos con Apache. Además, las peticiones base (como las de archivos javascript) se enrutan a la carpeta raíz del frontend UI. Por su parte, las URL de las API de account, market y sso se transfieren como proxy a sus respectivos módulos backend. Además, se ha activado SSL usando los certificados/clave nginx-certificate.crt y nginx.key.

/selene-backend/*/uwsgi.ini

En estos archivos (uno por cada módulo) se cambia el protocolo de socket a http o de lo contrario las API no serían accesibles. Podrían dejarse como socket y resolver el problema en la configuración de proxy de Nginx, pero resulta más sencillo de este modo.

5.5.3. Instalación de Selene

En este punto recogemos los pasos desarrollados es subsecciones anteriores y los adaptamos a Docker para instalar Selene:

1. Crear una carpeta raíz selene-docker: mkdir selene-docker

2. Descargar dentro de la carpeta los repositorios GitHub de selene-backend y selene-ui:

```
cd selene-docker
git clone https://github.com/MycroftAI/selene-backend.git
git clone https://github.com/MycroftAI/selene-ui.git
```

3. Copiar los archivos Docker creados al servidor en sus respectivas carpetas con el comando scp (desde fuera de la sesión ssh):

```
scp archivo1 archivo2 … dial@158.42.167.31:/carpeta/destino/
```

4. Editar los siguientes archivos cambiando “socket” por “http”:
   /selene-backend/api/account/uwsgi.ini
   /selene-backend/api/market/uwsgi.ini
   /selene-backend/api/sso/uwsgi.ini
   /selene-backend/api/public/uwsgi.ini

5. Editar el archivo /selene-backend/shared/selene/api/base_config.py y cambiar los dominios de Mycroft por la IP del servidor, ya que no se configuran por variables de entorno, se han introducido manualmente.: mycroft.test, .mycroft-test.net y .mycroft.ai por la IP del servidor 158.42.167.31.

6. Editar todos los archivos en selene-ui que siguen la nomenclatura environment*.ts (***/selene-backend/projects/*/src/environments/environment*.ts***). que son 16 en total, 4 por cada módulo, y cambiar las urls de Mycroft por la IP del servidor (158.42.167.31):
   account.mycroft.ai, account.mycroft.test, account.mycroft-test.net, market.mycroft.ai, market.mycroft.test, market.mycroft-test.net, sso.mycroft.ai, sso.mycroft.test, sso.mycroft-test.net, home.mycroft.ai, home-test.mycroft.ai

   Estos archivos configuran la referencia a las URLs de las APIs de backend que usan los frontend. Se puede realizar mediante el siguiente comando:

```
find . -type f -name "*environment*ts" -exec sed -i 's/account.mycroft.ai/158.42.167.31/g' {} +
```

7. Creamos las base de datos siguiendo las instrucciones de Selene como hemos hecho anteriormente:

```
export DB_PASSWORD=mycroft
export POSTGRES_PASSWORD=mycroft
sudo -u postgres psql -c "ALTER USER postgres PASSWORD '$POSTGRES_PASSWORD'"
sudo -u postgres psql -c "CREATE ROLE selene WITH LOGIN ENCRYPTED PASSWORD '$DB_PASSWORD'"
```

5.3.4. Puesta en marcha

Una vez creados y modificados todos los archivos de configuración, se puede iniciar la instalación con el script “***install.bat***”. Este script ejecuta todos los pasos necesarios para compilar las imágenes Docker y lanzarlas con Docker Compose.

Usa docker build para compilar todos los Dockerfile anteriores, crea la red Docker mycroft-network, lanza los contenedores con Docker Compose, y ejecuta los contenedores de inicialización de la BD una única vez.

```
cd selene-backend
docker build -t dial-base -f Dockerfile-api-base .
docker build -t dial-api-public -f Dockerfile-api-public .
docker build -t dial-api-sso -f Dockerfile-api-sso .
docker build -t dial-api-market -f Dockerfile-api-market .
docker build -t dial-api-account -f Dockerfile-api-account .
cd ..
cd selene-ui
docker build -t dial-ui -f Dockerfile-ui .
cd ..
docker network create mycroft-network
docker compose up -d
cd selene-backend
docker build -t dial-init-db -f Dockerfile-init-db .
docker build -t dial-init-defaults -f Dockerfile-init-defaults .
docker build -t dial-init-market -f Dockerfile-init-market .
docker run -it --rm --net mycroft-network --name dial-init-db dial-init-db
docker run -it --rm --net mycroft-network --name dial-init-defaults dial-init-defaults
docker run -it --rm --net mycroft-network --name dial-init-market dial-init-market
```

Lo ejecutamos mediante el comando: sudo sh install.sh

Una vez haya terminado se puede comprobar que está en marcha con: docker ps -a

En este punto, si está en marcha, se puede acceder a la web de Selene a través de https://158.42.167.31.

Y después se puede simplemente reiniciar con los comandos docker compose stop y up.

Por otro lado, tenemos el archivo "***docker-compose.yml***". Este archivo de Docker Compose orquesta todas las imágenes generadas por los archivos Dockerfile anteriores conforme al esquema del anexo y permite iniciarlos, eliminarlos, pararlos, etc. de una sola vez. Es decir, configura y pone en marcha todos los contenedores necesarios para arrancar Selene.

Capítulo 6

Resultados y conclusiones

En esta sección, se exponen los resultados obtenidos en la validación del prototipo en un entorno de laboratorio y pruebas iniciales con usuarios reales.

En primer lugar, se incluyen los resultados de la evaluación de los usuarios con distintos asistentes comerciales para comprobar la interacción, aceptación y compresión de la tecnología.

Por otra parte, también se han llevado a cabo las pruebas que incluyen el correcto funcionamiento de acceso remoto al prototipo, el despliegue backend y frontend con contenedores Docker, comprobando el correspondiente flujo de la aplicación web desplegada en el servidor, así como la persistencia y registro de los datos de los dispositivos en la base de datos PostgreSQL.

Finalmente, se ha validado en un entorno de laboratorio el diseño del prototipo hardware.

6.1. Sesión de pruebas iniciales con usuarios reales

Para esta validación inicial, se utilizó una muestra representativa de la población adulta mayor.

Para definir el número de usuarios necesarios para alcanzar los objetivos, se siguieron la recomendación de Nielsen (1994), de utilizar una muestra de aproximadamente 8-10 usuarios, pues según este autor, con este número de usuarios se recoge el 97% de la información relacionada con una interfaz tecnológica.

Metodología: Con el objetivo de obtener información sobre la experiencia de los usuarios con los asistentes de voz, se realizó un estudio inicial con las dos soluciones tecnológicas Alexa y Google Assistant.

Cada uno de los participantes seleccionados acudió por separado a la Universidad y testeó las dos soluciones, proporcionando su opinión. Además, también se probó la síntesis del habla (Text-to-Speech) tanto de Google Assistant como de Microsoft, y el reconocimiento del habla (Speech-to-Text) de Microsoft.

Al estar el proyecto enmarcado dentro de la Comunidad Valenciana, se incluyó la evaluación de los idiomas español y catalán.

Muestra: generamos un perfil de usuario para nuestra solución, en consonancia con las características clave correspondientes al proyecto. Se escogieron voluntarios que cumplieran las siguientes características:

- Hombres y mujeres de una edad aproximada de 65 años en adelante.
- Con capacidades cognitivas, visuales, auditivas y psicomotrices normales para su rango de edad y suficientes como para mantener una conversación o leer y seguir unas instrucciones.
- Que hablaran y entendieran el idioma valenciano.

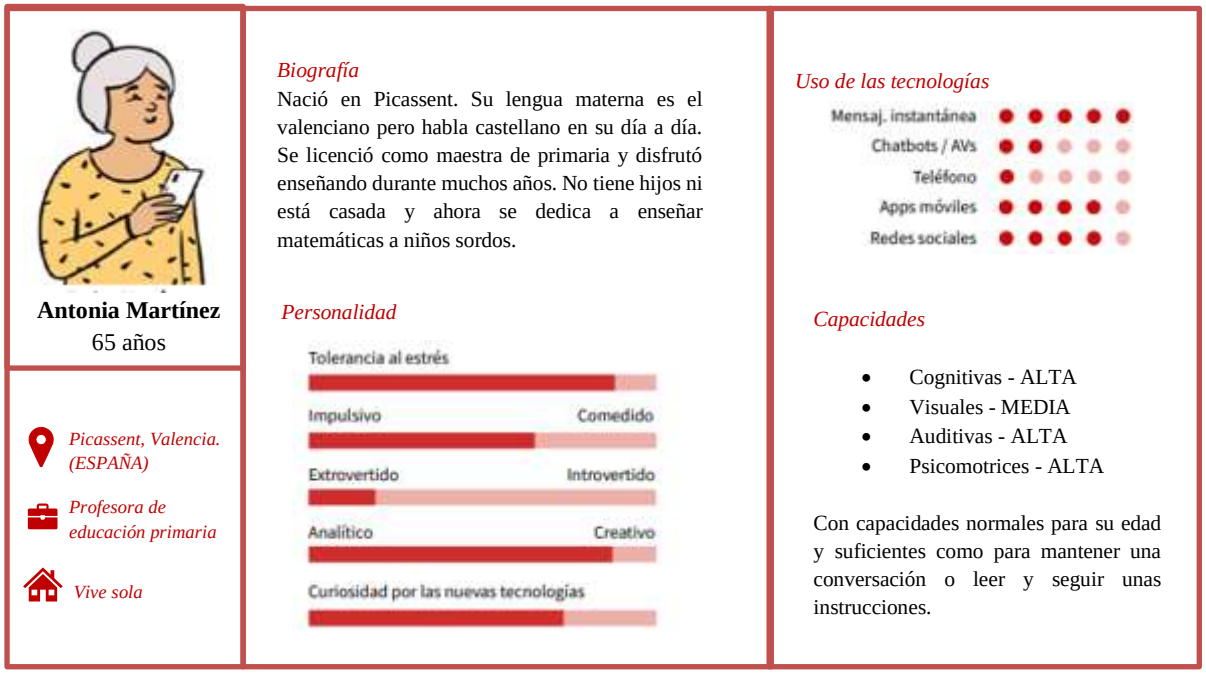

Fig. 72. Perfil de usuario

La muestra es representativa tanto de hombres como mujeres y se tuvieron en cuenta las diferentes edades y los diferentes niveles educativos.

La muestra estuvo formada por 8 personas mayores (50% mujeres; Medad = 74,63, DT = 9,36), de edades comprendidas entre 66 y 88 años. La mitad de ellos habían accedido a la Universidad (4 personas), 3 de ellos cursaron estudios básicos, y 1 de ellos no fue al colegio.

Todos residen en Valencia capital. La lengua materna de 6 de ellos es el castellano, 1 de ellos tiene como lengua materna el valenciano, y otro el inglés.

Todos hablan y entienden el catalán/valenciano a la perfección.

De los 8 participantes, 6 viven solos y 2 viven acompañados.

Fig. 73. Muestra representativa tanto de hombres como de mujeres

Procedimiento y datos recogidos:

En concreto, se evaluaron las siguientes funcionalidades y se obtuvieron los siguientes resultados:

Variables	Descripción	Prueba	Resultados
Conocimiento previo de los dispositivos	Se evaluó si conocían los dispositivos	Se les preguntó si los tenían, si los habían visto en la televisión, si conocían a alguien que los tuviese, si habían visto a alguien usarlos y si alguna vez los habían usado.	Más de la mitad de los participantes los conocía (5 personas). Los 5 que lo conocían afirmaban que era porque tenían un familiar que los tenía en casa y al que habían visto usarlos (3 de estos lo habían visto en televisión). Sin embargo, ninguno de ellos afirmó tenerlo en casa o haber usado los dispositivos en ningún momento de su vida.
Interacción con los dispositivos	Se les invitó a probar ambos dispositivos, con el objetivo de evaluar la interacción con cada uno de los aparatos. También se les invitó a que preguntaran y pidieran lo que ellos quisieran.	Se les pidió que les preguntaran a los dispositivos lo siguiente: "qué día es", "qué tiempo hace", "¿tienes amigos?", "¿te gusta ver la televisión?", "dime cómo preparar un cocido", "cuéntame un chiste" o "define un término concreto".	Todos los participantes disfrutaron de la experiencia, y se sorprendieron de que los dispositivos contestasen a las preguntas que les hacían. Cuando les preguntamos qué les había parecido hablar con los dispositivos, 6 de ellos contestaron que les había gustado y les parecía interesante (por ejemplo, "es interesante, divertido y útil"). Sin embargo, 2 de los participantes lo encontraron extraño y artificial ("es extraño hablar con una pelota").
Comprensión de los dispositivos (teniendo en cuenta la distancia física)	Se evaluó el nivel de comprensión de las dos tecnologías.	Se les preguntó si oían bien la voz de cada dispositivo, de un 1 (nada) a un 10 (perfectamente) y sí entendían bien la voz de ambos, también de 1 (nada) a 10 (perfectamente). Se les situó a una distancia mayor (5 metros) para que respondieran si seguían oyendo y entendiendo bien la	Siete participantes oyeron y entendieron perfectamente la voz de ambos dispositivos (Si 1 es nada, y 10 es perfectamente, los siete participantes situaron en 10 las preguntas de "oyes bien la voz de cada dispositivo" y "entiendes bien la voz de cada dispositivo"). Una participante tenía problemas auditivos, por lo que tuvimos que subir el volumen de voz de ambos dispositivos para que los escuchara bien. Aun así, cuando la participante hacía alguna pregunta, le costaba entender alguna respuesta proporcionada por los dispositivos, y fue una experiencia menos agradable para ella.

		voz de ambos dispositivos.	
Utilidad de los dispositivos	Se les preguntó la utilidad de ambos dispositivos, tanto para ellos como para otras personas	A través de la siguiente pregunta: "¿Crees que (Alexa o Google Assistant) puede ser útil para ti o para alguien que conozcas? ¿Por qué?".	La mayoría de ellos lo consideró útil para ellos mismos (6/8 participantes). Además, el dispositivo se consideró útil para que pusiera música, para la seguridad de la gente mayor (si caen al suelo, que llamen a algún familiar), para que llamen a alguien que les haga la compra, que encienda y apague las luces, o para que la gente no se sienta sola. Dos de ellos señalaron que sería útil que el aparato les preguntara "cómo estás" de vez en cuando, o que les ayudara a saber el tiempo de ese mismo día o a poner la radio.
Preferencias en los dispositivos	Se evaluó qué es lo que más y menos les había gustado y qué cosas cambiarían de cada dispositivo.	Mediante las preguntas: ¿Qué es lo que más te ha gustado? ¿Qué es lo que menos te ha gustado? ¿Qué cosas cambiarías de Alexa?	Cuatro participantes señalaron que lo que más les gustó fue que el aparato les contestara a las preguntas, o dicho de otro modo, que les hiciera caso. Uno de ellos señaló que lo que más le gusta es que el aparato le haga compañía, otro señalo lo que se había reído interaccionando con ellos, y el último participante señaló que lo que más le gusta es que el aparato pudiera ayudar a gente a tener conversación. En general, a 6 de 8 participantes les gustó interactuar con ambos aparatos y se divirtieron con él.
Dificultades en los dispositivos	Se evaluó cómo responden los participantes cuando hay problemas con los aparatos	Se recogió su reacción cuando no entienden lo que dice el aparato o el aparato no les entiende. Por ejemplo: se ríen, se enfadan, etc.	Cuando hubo problemas con los aparatos (no les entendía o no sabía responder), todos los participantes se reían. Solamente 3 de ellos se molestaron en algún momento cuando el aparato no respondía en diversas ocasiones lo que ellos esperaban.
Preguntas de comparación	Se realizaron preguntas de comparación de los dos dispositivos.	En concreto, se evaluó qué voz les gustaba más, a qué aparato entendían mejor, qué aparato les gustaba más y por qué.	Cinco participantes señalaron que les gustaba más la voz de Google, 2 de ellos señalaron que mejor la de Alexa, y a 1 de ellos le resultó indiferente.
Tipo de interacción	Se evaluó la forma en la que preferían que se les preguntara algo. Se evaluó la preferencia de si iniciar ellos el aparato (por ejemplo, para pedir una canción) o que el aparato les hablara un día sin avisar (¿por ejemplo, "hola, te apetece escuchar una canción?").	Se les preguntó "¿qué cantante te apetece escuchar?" y ellos responder cuál quieren, o que el aparato dijera varias opciones de respuesta (por ejemplo, varios cantantes) y ellos eligieran. Se les preguntó si, por ejemplo, a la hora de escuchar una canción, preferían que el aparato les dijera los cantantes y ellos escogieran, o si preferían decirle ellos mismos el	Seis de ellos señalaron la preferencia por ser ellos mismos los que dijeran qué cantante querían, y 2 de ellos señalaron que preferían que el aparato les diera varias opciones y ellos escogieran al cantante.

		cantante que querían escuchar.	
Text to Speech (síntesis del habla) y Speech To Text (reconocimient ol del habla) en castellano y catalán	Se evaluó el Text-to-speech tanto de Google como de Microsoft. Los participantes escucharon textos en castellano y textos en catalán de cada herramienta.	Se evaluó qué les parecía la voz y el grado en el que entendían cada voz de 1 (nada) a 10 (perfectamente) de cada herramienta. También jugaron con las distintas voces de cada herramienta, y con las barras de velocidad y entonación, para evaluar qué combinación les parecía mejor. El objetivo era observar qué tono y velocidad eran más adecuados para ellos, qué voz entendían más y cuál era la voz preferida para cada uno de los participantes (femenina o masculina, voz más aguda o más grave, y voz de Google o de Microsoft).	En cuanto al Text-to-Speech de Microsoft en español todos entendieron bien el idioma en todas las voces, y el tono y velocidad que había por defecto fue el más aceptado por todos. Siete de los 8 participantes señalaron que preferían la voz de chico, ya que era menos aguda y se entendía mejor. Un participante consideró que tanto la voz de hombre como la de mujer eran adecuadas. En catalán, todos señalaron que la voz tenía demasiado acento catalán y, a veces, costaba entenderla. Aunque la mitad de ellos entendieron todas las voces por igual, 5 de ellos prefirieron la voz de mujer, y 3 de ellos, la de hombre. En cuanto al Text-to-Speech de Google en español, también todos entendieron bien el idioma español en todas las voces, y el tono y velocidad que había por defecto fue el más aceptado por todos. En catalán, todos señalaron que la voz en este caso era más "valenciana" y se entendía más que la de Microsoft, por tanto, la herramienta preferida por unanimidad para la voz en catalán fue la de Google. Además, Google también fue la herramienta preferida en general para el idioma español, aunque en este caso la diferencia no fue tan significativa. A la pregunta de si preferían que el texto estuviese en valenciano en lugar de catalán, todos respondieron que mejor valenciano, dado que era mucho más sencillo de entender.
Se evaluó el Speech-to-text de Microsoft.	El objetivo era comprobar si el sistema los reconocía bien.	Realizaron la lectura de varios textos en castellano y catalán así como dijeron algunas frases que se les ocurrió para probar el reconocimiento del habla en lenguaje natural del día a día.	Se transcribieron bien todos los textos y frases en castellano y catalán que leyeron y produjeron los participantes. Los únicos fallos se derivaron de fallos en el habla (por ejemplo, si el participante aumenta el tiempo entre una palabra y otra que deberían ir seguidas, el sistema lo reconoce como un punto y seguido).

Tabla 4. Funcionalidades evaluadas y resultados con usuarios reales

6.2. Funcionamiento del prototipo

El dispositivo final desarrollado con Raspberry Pi permite interactuar con el usuario mediante el asistente virtual introducido con la imagen Picroft. Al dispositivo desarrollado funcional se añadirán más adelante las funcionalidades de machine learning correspondientes, que se enlazarán con el módulo Mycroft correspondiente como se ha descrito anteriormente en su arquitectura.

Gracias al acceso remoto a la Raspberry, podemos enviar comandos y comprobar su estado de forma remota para el acceso en los domicilios de los usuarios. Como vemos en las siguientes imágenes, es posible su conexión remota.

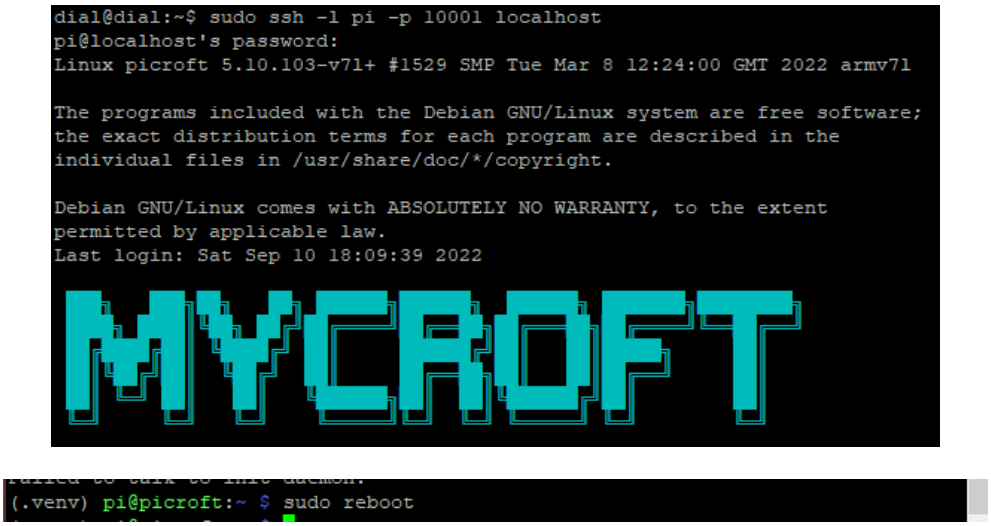

Fig. 74. Reinicio remoto de Raspberry

Por otro lado, se ha comprobado el flujo web servido como resultado del enrutamiento de URLs. El contenedor de Nginx en dial-ui está configurado con el archivo "default.com" para dirigir las peticiones según se ha recogido en las siguientes tablas:

Destino (Proxy)
http://selene-api-account:5000
Request URL path
/api/account
/api/cities
/api/countries
/api/defaults
/api/devices
/api/device-count
/api/geographies
/api/memberships
/api/pairing-code
/api/preferences
/api/regions
/api/skills
/api/software-update
/api/ssh-key
/api/timezones
/api/voices
/api/wake-words
/api/account[1]

Destino (Proxy)
http://selene-api-market:5000
Request URL path
/api/user
/api/skills
/api/skills/available
/api/skills/status
/api/skills/install

Destino (Proxy)
http://selene-api-sso:5000
Request URL path
/api/agreement/
/api/internal-login
/api/github-token
/api/logout
/api/password-change
/api/password-reset
/api/validate-email
/api/validate-federated
/api/validate-token
/api/account[1]

Destino (Estático)
{raíz archivos}/account.html
Request URL path
/ (por defecto)
/dashboard
/devices
/devices/*
/devices/add
/profile
/new

Destino (Estático)
{raíz archivos}/sso.html
Request URL path
/login
/logout
/change-password
/new-account
/profile
/new

Destino (Estático)
{raíz archivos}/market.html
Request URL path
/skills
/skills/*

Tabla 5. Enrutamiento de URLs

A continuación, se muestran la mayoría de vistas de la aplicación web. Por ejemplo, desde la página “/login” da paso a la página “/new account” junto con la política de privacidad y los términos de uso añadidos. Después sigue la vista “/new”.

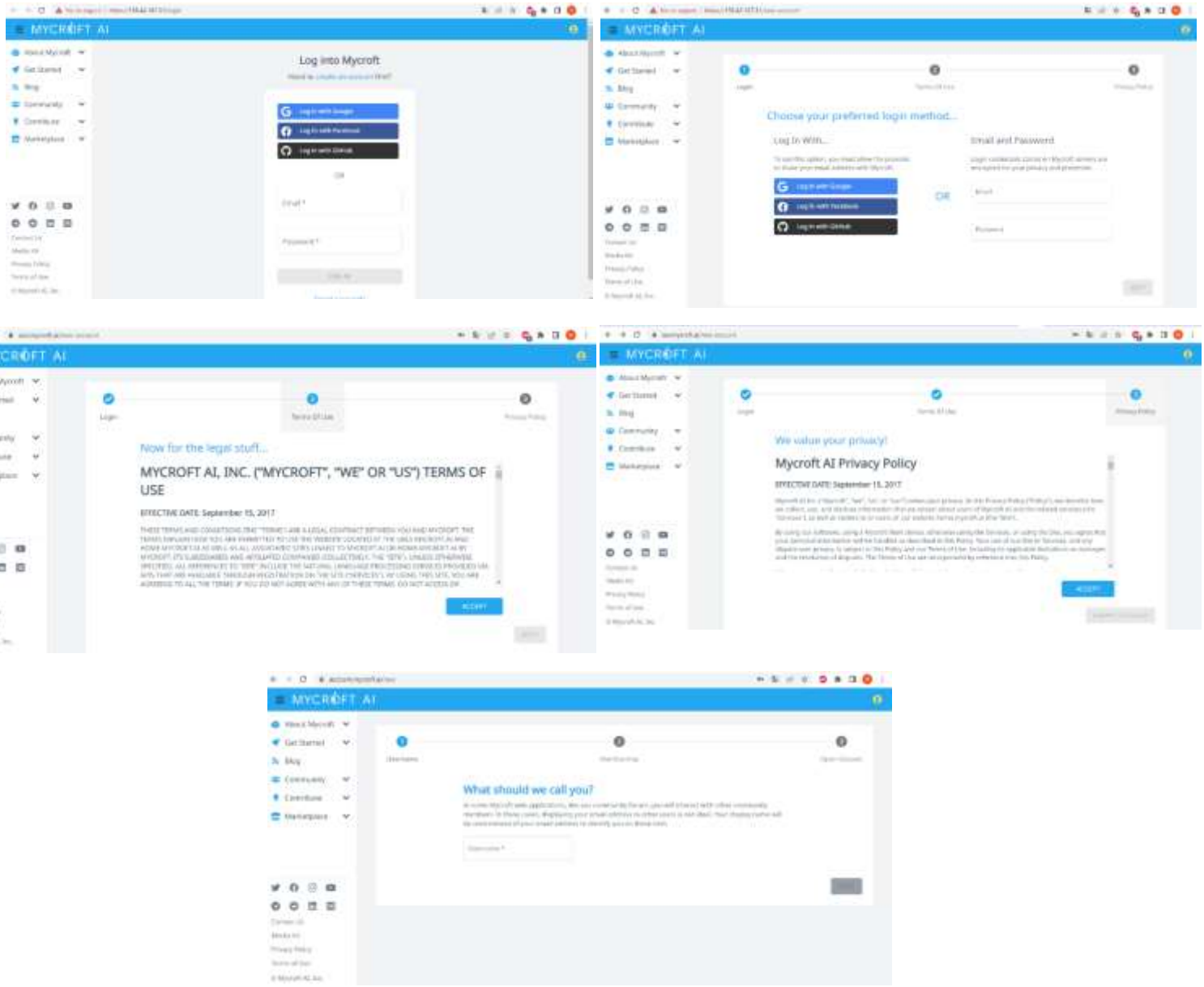

Fig. 75. Vistas /login, /new account, política de privacidad, términos de uso y /new.

A continuación, vemos las vistas “/skills” y “/skills/*.

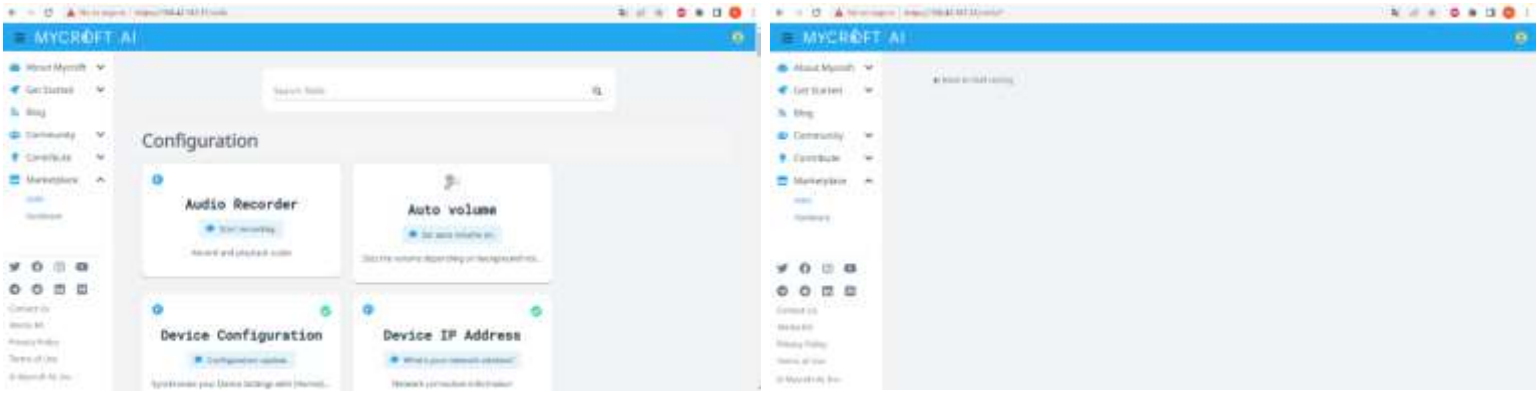

Fig. 76. Vistas “/skills” y “/skills/*.

También se muestra a continuación la vista “/dashboard” y “/profile”

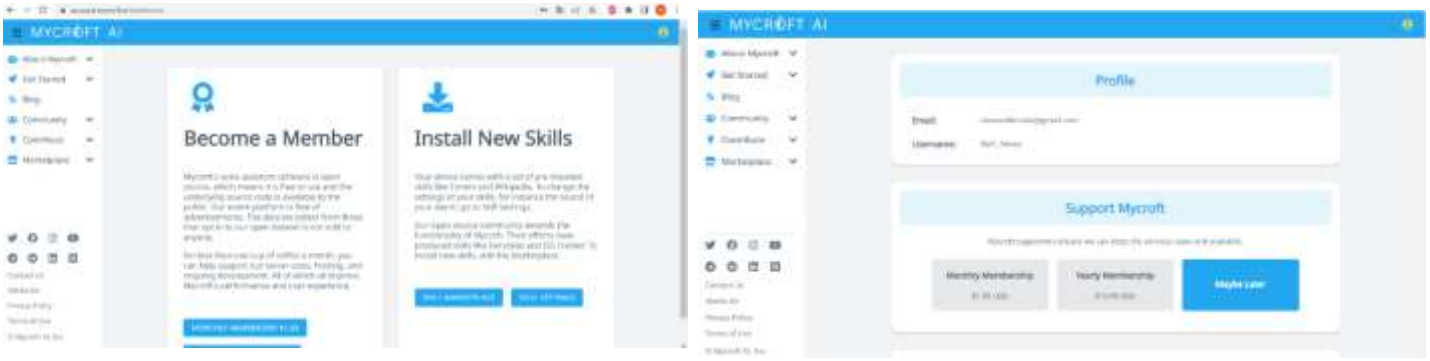

Fig. 77. Vistas “/dashboard” y “/profile”.

Con respecto a la vista “/devices” y “/devices/add”. Esta última utiliza las tablas de geonames.org utilizados para completar las tablas del esquema geográfico.

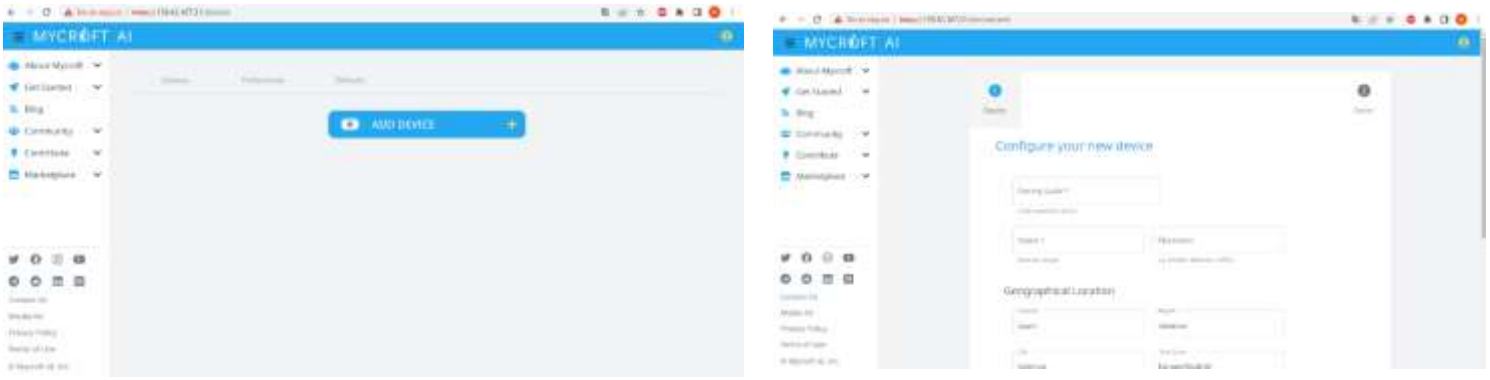

Fig. 78. Vistas “/devices” y “/devices/add”.

6.3. Conclusiones

Abordar la realidad de la soledad no deseada de personas mayores, requiere un enfoque desde muchos puntos de vista. Para trabajar por el bienestar de las personas mayores, muchos organismos deben trabajar en común para construir soluciones adecuadas para esta etapa de la vida.

Requiere también de la sensibilización e implicación de la sociedad en su conjunto tanto en la detección como en la intervención y prevención de estas situaciones. Promover la sensibilización social y la transmisión de valores de inclusión y solidaridad al conjunto de la sociedad, y la idea de apoyo mutuo e interdependencia. Las actividades comunitarias y culturales, creando espacios de encuentro que faciliten las relaciones, incluidas las intergeneracionales, contribuyen a esa sensibilización social y a la generación de redes.

Es importante contar con la opinión de las propias personas mayores, con necesidades y demandas, pero también con propuestas e iniciativa para paliar situaciones de soledad no deseada.

Por ejemplo, una gran mayoría (alrededor del 88%) de las personas mayores desean vivir en su casa el mayor tiempo posible, por lo que el incremento y la mejora de servicios como la atención a domicilio o la teleasistencia, es una demanda clara y creciente. Sin olvidar también los recursos personales, es necesario potenciarlos y trabajar para que las personas aprendamos a vivir en soledad.

Los resultados principales de esta validación señalan varios aspectos a tener en cuenta. En primer lugar, tras las pruebas iniciales con usuarios reales, la mayoría de los participantes disfrutaron de la experiencia de probar ambos dispositivos (Alexa y Google Assistant) y les pareció interesante. En segundo lugar, se observó una adecuada comprensión de los dispositivos por parte de las personas mayores, lo cual nos indica que la interacción entre estas y los dispositivos fue adecuada. Además, también hubo una adecuada interacción tras situar a los participantes a una distancia mayor. En tercer lugar, la mayoría de los participantes consideró que los dispositivos podían ser útiles para ellos, especialmente para mejorar su seguridad (por ejemplo, en caso de caídas poder llamar a alguien) y su compañía. En cuarto lugar, y tras evaluar la comparación entre los dos dispositivos, el 62,5% de los participantes señalaron que preferían Google Assistant, por la claridad de su voz y porque el dispositivo comprendía mejor a los participantes. El mismo porcentaje de participantes señalaron que preferían iniciar ellos la conversación, en lugar de que el aparato les preguntara alguna cosa durante el día. Finalmente, en quinto lugar, todos prefirieron la síntesis del habla (en español y catalán) de Google en lugar de Microsoft, y el reconocimiento del habla de Microsoft (el único que se pudo probar en esta validación) fue adecuado y todos los textos y frases se transcribieron de forma correcta.

El asistente de voz desarrollado es una herramienta útil para tratar las soledad no deseada, siempre que se consiga una conversación fluida que no se vea dificultada por la detección de voz.

Con respecto a los objetivos propuestos al inicio del trabajo, se han desarrollado con éxito la mayoría de ellos.

Tanto el objetivo 1 (*O1: Análisis de la soledad no deseada y el impacto en la sociedad*) como el 2 (*O2: Estudio de las tecnologías asistivas y los diferentes asistentes de voz de libre distribución que podemos encontrar*) se han llevado a cabo satisfactoriamente realizándose un estudio completo pero no exhaustivo de la soledad no deseada y de las tecnologías asistivas. Este estudio ha sido suficiente para la comprensión del problema que se trata en este proyecto.

Por otra parte, los objetivos relacionados con el desarrollo del prototipo (*O3: Diseño del prototipo, O4: Compilación y despliegue de la infraestructura de backend de Mycroft en servidores propios de la UPV, O6: Implementación y configuración de la interfaz gráfica de usuario y O7: Administración remota de los dispositivos*) se han cumplido resultando en un dispositivo funcional con su debida arquitectura corriendo en servidores propios de la UPV. Docker ha sido una mejor opción para el desarrollo del backend y frontend en este proyecto al poder definir ordenadamente cada contenedor con su aplicación. Además, al poder disponer de un sencillo script que ejecuta todos los pasos necesarios para compilar las imágenes Docker, podemos ejecutarlo todo de una sola vez cuando sea necesario. En relación con el objetivo 7, el acceso remoto a los dispositivos se ha cumplido exitosamente. Esta solución se puede desarrollar añadiendo funcionalidades como el reinicio programado a las 45 Raspberrys.

Con respecto al objetivo relacionado con la seguridad de los datos (*O5: Garantizar la seguridad de los datos)* se ha cumplido parcialmente ya que se han utilizado cortafuegos básicos por el lado del servidor así como contraseñas cortas debido a ser un desarrollo en un entorno de pruebas y no de producción.

Para terminar, el objetivo 8 (*O8: Análisis de los resultados de las sesiones de prueba con usuarios reales y pruebas de funcionamiento*) se ha llevado a cabo obteniendo resultados significativos del uso de estas tecnologías con usuarios y se ha demostrado el correcto funcionamiento de la solución.

En resumen, el prototipo cumple con los objetivos establecidos al inicio del trabajo correspondientes al diseño del prototipo y elección de componentes router y conectividad por tarjetas SIM M2M. También se ha desarrollado la compilación y despliegue de backend incluyendo la interfaz gráfica de usuario. Al desarrollo de este prototipo funcional, también se ha añadido la administración de los dispositivos de forma remota.

6.4. Líneas futuras

Dentro del marco del proyecto DIAL, se continúa con el desarrollo de un segundo prototipo “beta” al que le seguirá el tercer y último dispositivo desarrollado como final. A la vista de los resultados obtenidos en este trabajo, se pueden contemplar como mejoras directa de este dispositivo las siguientes:

- En las posteriores validaciones, se tratará de mejorar el número de participantes de la muestra y se observará si hay diferencias en las respuestas según el nivel educativo de los participantes.
- Con respecto a los elementos hardware y software, se pueden buscar otras alternativas que abaraten costes al igual que se pueden ir actualizando los componentes según nuevas versiones que vayan apareciendo en el mercado. Debido a la situación de desabastecimiento de chips provocada por las disrupciones en la fabricación y la cadena de suministro debido a la pandemia, el modelo de Raspberry Pi 4 B tiene stock bajo o inexistente en casi todos los proveedores por lo que se pueden buscar alternativas que funcionen con Picroft.
- La realización de una interfaz de usuario propia del proyecto que esté diseñada e incluya las funcionalidades que necesita el proyecto.
- La implementación de Grafana para recolectar los datos consumidos por el asistente y poder visualizar cuando tenemos mayor y menor consumo de datos para identificar con que asiduidad el usuario utiliza el asistente y a qué horas.
- De forma similar a usar Grafana, se pueden utilizar las interfaces proporcionadas por los operadores de las SIMs para comprobar el consumo de datos.
- Implementar la persistencia de los datos en el servidor ya que, si este cae, se reinicia o si necesitamos compilar de nuevo, los datos registrados se perderán. Crear volúmenes en Docker sería una buena implementación que permitiría con servar los datos.
- El desarrollo de asistentes de voz similares para tratar otros problemas de salud como pueden ser trastornos depresivos, fóbicos, del sueño o incluso para tratar ataques de pánico o ansiedad.

II. PRESUPUESTO

II. Presupuesto

La planificación y los objetivos se establecen teniendo en cuenta unos criterios técnicos y también económicos. Este Trabajo Final de Máster conlleva unos gastos de material y de personal determinados. Cabe mencionar que en el desglose de costes ya está incluido el 21% de IVA. A continuación, se detallan los costes divididos en estas categorías.

A. Costes de material

Como material hardware utilizado en este trabajo se recoge el implementado en el prototipo (altavoz, Raspberry Pi, etc) y los componentes necesarios para el desarrollo en pruebas (teclado, cables, etc). El detalle del cálculo de los costes de material por cada dispositivo se detalla en la Tabla 6. Adicionalmente, se ha aproximado el coste para la implementación de los 45 dispositivos necesarios para realizar el Piloto en DIAL.

Concepto	***Cantidad***	***Coste unitario (€/ud)***	***Coste Piloto***
Desarrollo Prototipo			
Altavoz Bluetooth eMeet Luna	45	99.99	4,495.50
Teclado	1	20.00	20.00
SanDisk Extreme Pro 32GB	45	12.70	571.50
Caja Oficial Raspberry Pi 4	45	9.00	405.00
Fuente alimentación Oficial Raspberry Pi 4	45	7.19	323.55
Raspberry Pi 4 B 4GB	45	60.00	2,700.00
Ratón RPI-MOUSE-BLACK/GREY	1	7.42	7.42
Cable microHDMI a HDMI	1	5.00	5.00
Monitor	1	70.00	70.00
Lineas telefónicas			
Bono de datos 10GB	45	16.10 (10GB)	724.50
Conectividad			
Router WiFi SIM	45	69.00	3,105.00
Coste Total		**376.40€**	**12,427.47€**

Tabla 6. Presupuesto de material para el proyecto y para el piloto

El coste final del prototipo variará en función de los componentes que se utilicen. Se pueden elegir otro tipo de altavoces, monitor o router que abaratará costes en caso de tener que reducir el presupuesto final.

B. Costes de personal

En el máster de Ingeniería Biomédica de la UPV, el Trabajo de Fin de Master cuenta con 20 créditos ECTS, lo que corresponde aproximadamente a 500 horas. Se han dividido estas horas en función al tiempo dedicado a los objetivos que se definieron al inicio del proyecto, mencionados de nuevo a continuación.

-Objetivo 1: Análisis de la soledad no deseada y el impacto en la sociedad.

-Objetivo 2: Estudio de las tecnologías asistivas y los diferentes asistentes de voz de libre distribución que podemos encontrar.

-Objetivo 3: Diseño del prototipo para el despliegue de los pilotos, realizando el correspondiente estudio de mercado, que incluye los router SIM a utilizar y las alternativas comerciales de conectividad IoT por tarjetas SIM M2M. El prototipo desarrollado deberá ser funcional con la debida arquitectura funcionando correctamente.

-Objetivo 4: Compilación y despliegue de la infraestructura de backend del asistente de voz utilizando Mycroft en servidores propios de la UPV, implementando la base de datos y las APIs definidas.

-Objetivo 5: Garantizar la seguridad de los datos, ya que en este tipo de aplicaciones la información privada de pacientes es crítica.

-Objetivo 6: Implementación y configuración de la interfaz gráfica de usuario

-Objetivo 7: Administración remota de los dispositivos.

-Objetivo 8: Análisis de los resultados de las sesiones de prueba con usuarios reales y pruebas de funcionamiento.

El desglose de los costes de personal se detalla en la Tabla 7, donde se han incluido los costes por alumno y por tutor. Se ha dedicado 45 horas aproximadamente a la redacción de la memoria de este trabajo.

Concepto	*Precio*	*Horas Alumno*	*Horas Tutor*	*Total Horas*	*Coste Total*
Desarrollo Prototipo					
Objetivo 1	13€/h	25	10	35	455.00
Objetivo 2	13€/h	25	10	35	455.00
Objetivo 3	13€/h	50	30	80	1,040.00
Objetivo 4	13€/h	170	70	240	3,120.00
Objetivo 5	13€/h	20	5	25	325.00
Objetivo 6	13€/h	110	70	180	2,340.00
Objetivo 7	13€/h	35	5	40	520.00
Objetivo 8	13€/h	25	15	40	520.00
Elaboración de la memoria					
Documento final	13€/h	40	5	45	585.00
Horas de trabajo totales	-	500	220	**720**	-
Costes indirectos					
Gastos de luz, agua, seguro	0.15€/h	75	33	108	216.00
Coste Total	-	-	-	-	9,576.00
Margen de beneficio*					
10% de margen	10%	-	-	-	957.60
Coste Total	-	-	-	-	**10,533.60€**

Tabla 7. Presupuesto de personal para el proyecto y para el piloto

**Margen de beneficio.*

En el caso de que este estudio estuviera enfocado a obtener un beneficio por su realización deberíamos aplicar un margen de beneficio estimado en un 10% sobre el total de costes.

ANEXO I - Código Docker

/selene-backend/Dockerfile-api-base

```
FROM python:3.7-slim as base-build
RUN apt-get update && apt-get -y install gcc git
RUN python3 -m pip install pipenv
RUN mkdir -p /root/allure /opt/selene/selene-backend /root/code-quality
/var/log/mycroft
WORKDIR /opt/selene/selene-backend
####### Variables de entorno con la configuración del servidor
#### BD Selene (Postrges). Para poder usar DB_HOST como nombre "selene-db" en
vez de IP, el container de Docker con la BD debe estar en la misma network y tener
como hostname: selene-db
ENV DB_HOST selene-db
ENV DB_PORT 5432
ENV DB_NAME mycroft
ENV DB_PASSWORD mycroft
ENV DB_USER selene
ENV POSTGRES_PASSWORD Mycroft
# Esta variable es necesario para conectar a la BD sin problemas causados por SSL:
ENV DB_SSLMODE prefer
#### BD Selene Cache (Redis). Para poder usar DB_HOST como nombre "selene-
cache" en vez de IP, el container de Docker con la BD Redis debe estar en la misma
network y tener como hostname: selene-cache
ENV REDIS_HOST selene-cache
ENV REDIS_PORT 6379
#### El servicio de autenticacion lo provee la imagen de SSO. Para poder usar las URL
como nombre "selene-api-sso" en vez de IP, el container de Docker con la SSO debe
estar en la misma network y tener como hostname: selene-api-sso
#Al instalarse como contenedores de Docker, se puede usar el hostname de cada
contenedor en vez de la IP y puerto. Estas URLs deben ser las del modulo SSO:
ENV SSO_BASE_URL http://selene-api-sso:5000
ENV OAUTH_BASE_URL http://selene-api-sso:5000
#### JSON Web Tokens (y salt) usados como base para la autenticación. Si se cambian
se invalidan todos los usuarios y hay que empezar de cero.
ENV JWT_ACCESS_SECRET XXX
ENV JWT_REFRESH_SECRET XXX
ENV JWT_RESET_SECRET XXX
#### Salt aleatorio para contraseñas. 16 caracteres alfanumericos
ENV SALT XXX
#### Keys de terceros.
ENV STRIPE_PRIVATE_KEY XXX
ENV GITHUB_CLIENT_ID XXX
ENV GITHUB_CLIENT_SECRET XXX
```

```
ENV EMAIL_SERVICE_HOST smtp-mail.outlook.com
ENV EMAIL_SERVICE_PORT 25
ENV EMAIL_SERVICE_USER dialemailservice@outlook.es
ENV EMAIL_SERVICE_PASSWORD mandacorreopordial
ENV GOOGLE_STT_KEY XXX
ENV SENDGRID_API_KEY XXX
ENV WOLFRAM_ALPHA_KEY XXX
ENV WOLFRAM_ALPHA_URL https://api.wolframalpha.com
ENV OWM_KEY XXX
ENV OWM_URL https://api.openweathermap.org/data/2.5
ENV GITHUB_USER neriilo
ENV GITHUB_PASSWORD XXX
ENV SELENE_ENVIRONMENT prod
####### Copiar el codigo base compilado
FROM base-build as config-build
COPY shared shared
```

/selene-backend/Dockerfile-init-db

```
FROM dial-base
WORKDIR /opt/selene/selene-backend
COPY db db
WORKDIR /opt/selene/selene-backend/db
RUN pipenv install
COPY privacy_policy.md /opt/mycroft/devops/agreements/privacy_policy.md
COPY terms_of_use.md /opt/mycroft/devops/agreements/terms_of_use.md
ENTRYPOINT ["pipenv", "run", "python", "scripts/bootstrap_mycroft_db.py"]
```

/selene-backend/Dockerfile-init-defaults

```
FROM dial-base
WORKDIR /opt/selene/selene-backend
COPY db db
WORKDIR /opt/selene/selene-backend/db
RUN pipenv install
COPY privacy_policy.md /opt/mycroft/devops/agreements/privacy_policy.md
COPY terms_of_use.md /opt/mycroft/devops/agreements/terms_of_use.md
ENTRYPOINT ["pipenv", "run", "python", "scripts/neo4j-postgres.py"]
```

/selene-backend/Dockerfile-init-market

```
FROM dial-base
WORKDIR /opt/selene/selene-backend
COPY batch batch
WORKDIR /opt/selene/selene-backend/batch
RUN pipenv install
```

```
ENTRYPOINT ["pipenv", "run", "python", "script/load_skill_display_data.py", "--core-version", "21.02"]
```

/selene-backend/Dockerfile-api-*

```
# Dockerfile para la imagen de la API Account
FROM dial-base
ENV PYTHONPATH=$PYTHONPATH:/opt/selene/selene-backend/api/account
COPY api/account api/account
WORKDIR /opt/selene/selene-backend/api/account
RUN pipenv install --system
RUN pipenv install --dev
ENTRYPOINT ["uwsgi", "--ini", "uwsgi.ini"]

# Dockerfile para la imagen de la API SSO
FROM dial-base
ENV PYTHONPATH=$PYTHONPATH:/opt/selene/selene-backend/api/sso
COPY api/sso api/sso
WORKDIR /opt/selene/selene-backend/api/sso
RUN pipenv install --system
RUN pipenv install --dev
ENTRYPOINT ["uwsgi", "--ini", "uwsgi.ini"]

# Dockerfile para la imagen de la API Public
FROM dial-base
RUN mkdir -p /opt/selene/data
ENV PYTHONPATH=$PYTHONPATH:/opt/selene/selene-backend/api/public
COPY api/public api/public
WORKDIR /opt/selene/selene-backend/api/public
RUN pipenv install --system
RUN pipenv install --dev
ENTRYPOINT ["uwsgi", "--ini", "uwsgi.ini"]

# Dockerfile para la imagen de la API Marketplace
FROM dial-base
RUN mkdir -p /opt/selene/data
ENV PYTHONPATH=$PYTHONPATH:/opt/selene/selene-backend/api/market
COPY api/market api/market
WORKDIR /opt/selene/selene-backend/api/market
RUN pipenv install --system
RUN pipenv install --dev
ENTRYPOINT ["uwsgi", "--ini", "uwsgi.ini"]
```

/selene-ui/Dockerfile-ui

```
FROM node:latest as build
ENV NODE_OPTIONS --openssl-legacy-provider
```

```
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build -- --project=shared
RUN npm run build -- --project=globalnav
RUN npm run build-prod -- --project=account
RUN npm run build-prod -- --project=market
RUN npm run build-prod -- --project=sso
####### Utilizar Nginx como servidor para publicar los frontend
FROM nginx
## default.conf contiene la configuracion de servidor y proxy de nginx
COPY default.conf /etc/nginx/conf.d/default.conf
#### Copiar los certificados para SSL
COPY nginx-certificate.crt /etc/nginx/certificate/nginx-certificate.crt
COPY nginx.key /etc/nginx/certificate/nginx.key
#### Copiar el codigo compilado de cada frontend y las dependencias en el mismo sitio, cambiando el nombre de cada index.html para que sea distinto
COPY --from=build /usr/src/app/dist/account /usr/share/nginx/html
COPY --from=build /usr/src/app/dist/market /usr/share/nginx/html
COPY --from=build /usr/src/app/dist/sso /usr/share/nginx/html
COPY --from=build /usr/src/app/dist/account/index.html /usr/share/nginx/html/account.html
COPY --from=build /usr/src/app/dist/market/index.html /usr/share/nginx/html/market.html
COPY --from=build /usr/src/app/dist/sso/index.html /usr/share/nginx/html/sso.html
COPY --from=build /usr/src/app/dist/shared /usr/share/nginx/html/shared
COPY --from=build /usr/src/app/dist/globalnav /usr/share/nginx/html/globalnav
```

/selene-ui/default.conf

```
server {
  ####### Configuracion basica del servidor
  #### Configuracion SSL
  listen 443 ssl;
  listen [::]:443 ssl;
  ssl_certificate /etc/nginx/certificate/nginx-certificate.crt;
  ssl_certificate_key /etc/nginx/certificate/nginx.key;
  #### El nombre del servidor debe ser el mismo que el hostname del container: selene-ui
  server_name selene-ui;
  #### Configurar DNS resolver como el interno de Docker, para que pueda resolver los hostname de cada container
  resolver 127.0.0.11 valid=10s;
  resolver_timeout 5s;
```

```
  #### Configurar headers para prevenir error de CORS, o si no el navegador no
permitira transferencias de tantas direcciones/puertos distintos
  add_header 'Access-Control-Allow-Origin' '*';
  add_header 'Access-Control-Allow-Credentials' 'true';
  add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, PATCH,
DELETE';
  add_header 'Access-Control-Allow-Headers' 'DNT,X-CustomHeader,Keep-
Alive,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type';
  ####Configuracion Proxy para redirigir las url de Account a la API Account
  location ~ ^/(api/cities|api/countries|api/defaults|api/devices|api/device-
count|api/geographies|api/memberships|api/pairing-
code|api/preferences|api/regions|api/software-update|api/ssh-
key|api/timezones|api/voices|api/wake-words) {
    proxy_pass http://selene-api-account:5000;
  }
  ####Configuracion Proxy para redirigir las url de Marketplace a la API Market
  location ~ ^/(api/user|api/skills|api/skills/available|api/skills/status|api/skills/install) {
    proxy_pass http://selene-api-market:5000;
  }
  ####### Configuracion Proxy para redirigir las url de SSO a la API SSO
  location ~ ^/(api/agreement|api/internal-login|api/github-
token|api/logout|api/password-change|api/password-reset|api/validate-email|api/validate-
federated|api/validate-token) {
    proxy_pass http://selene-api-sso:5000;
  }
  ####### Configuracion Proxy para el caso especial /api/account. Metodo POST debe
ir a SSO y los demas a Account.
  location /api/account {
    if ($request_method = POST ) {
      proxy_pass http://selene-api-sso:5000;  }
    if ($request_method = GET ) {
      proxy_pass http://selene-api-account:5000;   }
    if ($request_method = PATCH ) {
      proxy_pass http://selene-api-account:5000;    }
    if ($request_method = DELETE ) {
      proxy_pass http://selene-api-account:5000;    }
  }
 ######Configuracion sel servidor para redirigir cada url al frontend que toca
  #### La url base apunta a Account
  location / {
    # Alias hace que todas las peticiones empiecen desde esta raiz, ignorando subpaths
    alias /usr/share/nginx/html/;
    # Index por defecto si no se pide uno
    index account.html;
```

```
    #### Urls de Account
    location = /dashboard {
      # Redirigir cualquier peticion al html con los javasccripts (si no se encuentra, 404)
      try_files /account.html =404;    }
    location = /devices {
      try_files /account.html =404;    }
    location /devices/ {
      # Redirigir sub-urls varias como /devices/deviceid....
      try_files /account.html =404;    }
    location = /devices/add {
      try_files /account.html =404;    }
    location = /profile {
      try_files /account.html =404;    }
    location = /new {
      try_files /account.html =404;    }
    #### Urls de Market
    location = /skills {
      try_files /market.html =404;    }
    location /skills/ {
      # Redirigir sub-urls varias como /skills/skillid....
      try_files /market.html =404;    }
    #### Urls de SSO
    location = /login {
      try_files /sso.html =404;    }
    location = /logout {
      try_files /sso.html =404;    }
    location = /change-password {
      try_files /sso.html =404;    }
    location = /new-account {
      try_files /sso.html =404;
    }  }
}
```

/docker-compose.yml

```
services:
  ####Contenedores de las BD
  ##Contenedor POSTGRES para Selene DB. POSTGRES_HOST_AUTH_METHOD trust evita tener que configurar el hba pg_hba.conf.
  selene-db:
    image: postgres:10
    container_name: selene-db
    hostname: selene-db
    environment:
      - POSTGRES_PASSWORD=mycroft
```

```
    - POSTGRES_HOST_AUTH_METHOD=trust
  networks:
    - mycroft-network

 ##Contenedor REDIS para cache Selene.
 selene-cache:
  image: redis:6
  container_name: selene-cache
  hostname: selene-cache
  networks:
    - mycroft-network

 ####Contenedores de las API
 ## Contenedor API Public. No es necesario hostname ya que solo se accede desde el exterior. Asegurarse de que 5000 esta abierto en firewall.
 selene-api-public:
  image: dial-api-public
  container_name: selene-api-public
  ports:
    - "5000:5000"
  networks:
    - mycroft-network
  depends_on:
    - selene-db
    - selene-cache

 ##Contenedor API SSO. No es necesario ports ya que solo se accede desde el interior.
 selene-api-sso:
  image: dial-api-sso
  container_name: selene-api-sso
  hostname: selene-api-sso
  networks:
    - mycroft-network
  depends_on:
    - selene-db
    - selene-cache

 ##Contenedor API Market. No es necesario ports ya que solo se accede desde el interior.
 selene-api-market:
  image: dial-api-market
  container_name: selene-api-market
  hostname: selene-api-market
  networks:
    - mycroft-network
```

```
    depends_on:
      - selene-db
      - selene-cache

  ##Contenedor API Account. No es necesario ports ya que solo se accede desde el interior.
  selene-api-account:
    image: dial-api-account
    container_name: selene-api-account
    hostname: selene-api-account
    networks:
      - mycroft-network
    depends_on:
      - selene-db
      - selene-cache

  #### Contenedor de los frontend. Asegurarse de que 443 esta abierto en firewall.
  selene-ui:
    image: dial-ui
    container_name: selene-ui
    hostname: selene-ui
    ports:
      - "443:443"
    networks:
      - mycroft-network
    command: [nginx-debug, '-g', 'daemon off;']
    depends_on:
      - selene-api-account
      - selene-api-market
      - selene-api-sso

####Definicion de la red. External es por si se ha creado antes desde fuera.
networks:
  mycroft-network:
    # name: mycroft-network # Solo funciona en version 3.9
    external: true
```

Referencias

[1] INE:1900-2011: Censos de Población y Vivienda. 2019: Estadística del Padrón continuo a 1-1-2019. Consulta enero 2020; 2028-2068: Proyecciones de población. Consulta enero 2019.

[2] INE. Demografía de Europa. ESTADÍSTICAS VISUALIZADAS - EDICIÓN 2021. ISBN 978-92-76-37873-0. ISSN: 2600-3368. doi:10.2785/428873. Cat. N.º: KS-FW-21-001-EN-Q

[3] INE. Evolución de la esperanza de vida al nacimiento. Brecha de género. España. Serie 1991-2020.

[4] INE. Encuesta Continua de Hogares (ECH) - Año 2018

[5] Página sobre la prevención de la soledad no deseada del Ayuntamiento de Madrid https://soledadnodeseada.es/

[6] Asistente de voz Mycroft. https://mycroft.ai/

[7] Asistente Leon. https://getleon.ai/

[8] Asistente de voz Rhasspy. https://rhasspy.readthedocs.io/en/latest/

[9] Plataforma de voz LinTO. https://linto.ai/index

[10] Plataforma de AI Jasper. https://www.jasper.ai/

[11] Estimación uso de datos. https://tecnobestias.com/domotica-para-casas-inteligentes/cuantos-datos-usa-echo-dot/

[12] Cliente SSH: PuTTY. https://www.putty.org/

[13] Repositorio de Mycroft para backend. https://github.com/MycroftAI/selene-backend

[14] Repositorio de Mycroft para frontend. https://github.com/MycroftAI/selene-ui

[15] Script de comprobación del proceso SSH. https://www.tunnelsup.com/raspberry-pi-phoning-home-using-a-reverse-remote-ssh-tunnel/

[16] Comandos para generar las claves publico-privadas con PuTTY en Windows. https://www.it-react.com/index.php/2020/01/12/create-and-install-ssh-keys-on-linux-and-windows/

[17] P. Hernandis, Sacramento; D. Bellegarde, Monica; La soledad de las personas mayores Conceptualización, valoración e intervención. Universidad de Valencia (2018).