ESCUELA DE INGENIERÍA INFORMÁTICA DE VALLADOLID

SINCRONIZACIÓN DE INFORMACIÓN PARA UN ASISTENTE VIRTUAL OFFLINE

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Mención Tecnologías de la Información

Alumno:

Santiago Rabadán Fernández

Tutores:

Jesús María Vegas Hernández

César Llamas Bello

2021

Índice general

Resumen

Los asistentes basados en voz son herramientas muy útiles para ayudar a personas con necesidades especiales de movilidad o que tengan una situación de dependencia, por ejemplo, algunas personas ancianas que viven solas. Este tipo de dispositivos les podría permitir acceder a información útil que de otra forma sería difícil de obtener para ellas.

La presente memoria describe el Trabajo de Fin de Grado que trata el problema de la creación y puesta en funcionamiento de un sistema de sincronización de información para estos asistentes virtuales offline. Consiste principalmente en el diseño de un software que sea capaz de generar cachés atomizadas dependiendo del núcleo de población en el que se encuentre situado el asistente virtual, el paso de las cachés del servidor a los clientes, y en el diseño de una *skill* que sea capaz de narrar esa información. Esto permitirá que las personas con alguna de estas condiciones que vivan en entornos rurales donde la conexión a internet no sea fluida, puedan seguir utilizando el dispositivo, y este les pueda dar el mínimo servicio exigible.

Este proyecto debería facilitar a las personas que trabajen en el en el futuro, toda la parte relacionada con la interacción entre el altavoz y el usuario, y deberá ser integrada en el sistema final, cuando se junten todas las partes, como servicio esencial teniendo en cuenta la naturaleza de uso del dispositivo.

Abstract

Voice assistants are nowadays very useful tools to help people with special needs, in movility for example, o who lives in a dependency situation, for example, old persons who live alone. This type of devices could help them to have access to some information that in other way they can´t obtain.

This document describes the Final Degree Work that treat to solve the problem of make function a system of information synchronization between the server an those intelligent assistants. It consists mainly in the design of a software that creates atomized caches for a concrete location, the passage of this file between server al clients, and also a skill that permits the speaker to read the downloaded information. This will able to people who lives in tiny villages or rural zones, where internet connection probably is not too fluid, to use the system, and it could give them the minimum exigible service.

This project will make easier to the future developers of the project the task of create the interaction between the user and the speaker, and this funcion should be integrated in the final prototype of the project.

Capítulo 1

Introducción

Hoy en día los dispositivos inteligentes pueden adoptar muchas formas y tener funcionalidades muy especificas que nos ayuden en muchos aspectos cotidianos de la vida. Pensamos primeramente en los smartphones, el dispositivo central de todo ecosistema que se precie hoy en día, a los que podemos añadir todos los complementos que este puede tener, desde relojes inteligentes, pulseras contabilizadoras, gafas y numerosos dispositivos de control domótico, como termostatos, bombillas, enchufes, e incluso la lavadora o el lavavajillas. Todos estos dispositivos están diseñados para ayudarnos y facilitarnos algunas de las tareas que hacemos en nuestro día a día.

Hay un tipo de dispositivo que no se ha mencionado y sobre el que va a radicar este proyecto, y son los altavoces inteligentes, también llamados asistentes inteligentes de voz que se pueden colocar en tu domicilio y a los que les puedes encargar de viva voz muchas de las tareas que realizan los dispositivos antes mencionados, desde apagar la luz a encender la lavadora. Asimismo les puedes preguntar información que te haga falta de manera inmediata, como el tiempo o que te recuerde que tenías anotado hoy en el calendario.

Un problema de estos dispositivos es que para que funcionen a pleno rendimiento se necesita una conexión a internet estable y fluida. Hay dispositivos cuyo principal nicho de mercado es el de las personas jóvenes, un colectivo que lleva conviviendo con la tecnología desde que entró en el colegio, que tuvieron sus primeros teléfonos inteligentes en cuanto salieron al mercado, y debido a todo ello, tienen una barrera de entrada muy pequeña a la hora de utilizar nuevos dispositivos inteligentes. Pero si no nos paramos a pensarlo, no nos damos cuenta de lo útil que puede resultar esta tecnología a personas mayores, que con un leve aprendizaje podrían contar con asistentes que les ayudasen a realizar esas tareas cotidianas de las que hablamos. Además, en nuestro país estas personas suelen vivir en entornos rurales que habitualmente están poco poblados.

El problema de vivir en estos lugares es que el acceso a internet es limitado, y esto supone un problema de cara a intentar que estos dispositivos se puedan implantar. Debido a este problema aparece la necesidad de crear algun tipo de sistema que permita que con una conexión mínima el servicio que puedan dar estos dispositivo sea razonable. Estos mecanismos no orientados a conexión no pueden aplicarse a todos los dispositivos, pero si a uno del tipo que nosotros tenemos entre manos, los altavoces inteligentes.

Aquí empieza el trabajo. La principal tarea que hay por delante es la creación de un sistema que, primero, tome datos de ciertas web de referencia de las que el altavoz extraería la información cuando el usuario le hiciese las preguntas. Segundo, que crease un archivo a modo de caché para cada una de esas webs de referencia, y que sea capaz de concatenar varios de ellos para que solo haya que enviar un archivo a los altavoces, ahorrando transacciones. Tercero, la creación del sistema que permita el paso de estos archivos entre el servidor que los genera y los altavoces que los usan. Y por último, el diseño de una aplicación para estos altavoces, denominadas *skills*, que permita que el usuario pueda hacer preguntas al altavoz referidas a alguno de los datos almacenados, y este sea capaz de leerlos a viva voz.

Para el desarrollo de este trabajo, aunque lo detallaremos más adelante, se ha seguido un plan de trabajo que consistió en, al principio, intentar adaptar dos trabajos realizados previamente sobre este sistema de altavoces, y sobre el que luego habría que añadir esta nueva funcionalidad. A continuación, se diseñaría todo el sistema de creación y paso de cachés y finalmente la *skill* para su lectura.

1.1. Estructura de la memoria

Esta memoria está estructurada de tal manera que pueda ser comprendida fácilmente y que se entienda paso a paso como ha ido avanzando el proyecto.

1. En este primer capítulo introductorio se ha explicado brevemente en que consiste el proyecto y que lo motiva, para ir ahondando sobre ello en el resto del trabajo.

2. En el segundo capítulo, después de esta introducción, se explicará en que tecnologías ya existentes se basa el sistema que estamos creando que avances ha habido en la materia en los últimos años, y con datos en la mano se justificará la necesidad de crear una solución de este tipo para estos sistemas. Finalmente explicaremos la solución planteada.

3. En el capitulo tres, se hablará sobre la planificación temporal del proyecto, se hará un estudio de riesgos que puede sufrir el proyecto durante su realización, y se detallará como se va a desarrollar y en qué fases.

4. En el cuarto capítulo se hará un análisis del proyecto, con explicaciones a los trabajos que preceden a este y que es lo que se va hacer con ellos. Además se identifican los requisitos que debe tener el sistema, así como los casos de uso, y se hará un estudio de costes que permita conocer que costes podría tener un proyecto de este tipo en la vida real.

5. En el quinto capítulo se pasaría a dar detalles del diseño y la implementación del proyecto, explicando primeramente el software utilizado para el desarrollo del mismo, y ya después los diagramas de tipo de tipo de proceso del sistema cliente-servidor y la lógica conversacional de la *skill* desarrollada.

6. En el sexto capítulo se detallan las pruebas realizadas para comprobar que el sistema funciona correctamente.

7. A continuación en el séptimo capítulo, se encuentra la guía de despliegue, donde se explicará como hacer funcionar el sistema para que las personas que tengan que probarlo no tengan problemas a la hora de ponerlo en funcionamiento.

8. Y finalmente en el capítulo ocho serán comentadas las conclusiones extraídas de este trabajo y que es lo que quedaría para hacer en el futuro, y que se podría hacer con el trabajo aquí realizado, en consonancia con los planes de futuro de los dos trabajos previamente realizados.

Tras esta introducción comenzamos con las explicaciones que dan motivación a la realización de este proyecto y y a explicar en que tecnologías ya existentes nos basamos para llevarlo a cabo.

Capítulo 2

Estado del Arte

2.1. ¿Qué es el internet de las cosas?

Durante los últimos años con el auge de las tecnologías que se engloban en lo que coloquial y tecnológicamente ha empezado a denominarse de forma mayoritaria como el ”Internet de las cosas”, o IoT, del Inglés ”Internet of Things”, han aparecido numerosos dispositivos. Desde medidores de actividad, pulseras o relojes, hasta altavoces, pantallas interactivas o incluso robots, pasando por instrumentos de cocina, termostatos o bombillas, y en realidad cualquier aparato susceptible de incluir un chip de comunicación, ya sea para proveerle directamente de una conexión a internet, como una tarjeta SIM, o una tecnología que haga de puente entre el dispositivo en cuestión, y otro que si que tenga conexión a internet, como por ejemplo Bluetooth.

Estos dispositivos han sido creados para ser complementos tecnológicos que lejos de ser esenciales, si que pueden facilitarnos algunas tareas y ayudarnos en nuestro día a día. Muchos de ellos suelen estar muy orientados a la domotización de nuestros hogares.

Aunque el internet de las cosas cada vez está más presente en nuestras vidas a través de todos los dispositivos que se engloban dentro de esta clasificación, es un concepto que aun no tiene una definición estandarizada [16]. Pero para entenderlo todo mejor, podría decirse que el IoT está compuesto por todos los dispositivos de uso cotidiano que se encuentran interconectados mediante internet, y que cuentan con algún tipo de inteligencia.

2.2. Los altavoces inteligentes

En los últimos años se ha popularizado especialmente el uso de este tipo de dispositivos englobados en lo que hemos explicado como İnternet de las cosas”. Numerosas compañías tecnológicas, principalmente Google, Apple y Amazon, han apostado por ellos como complementos para el hogar que nos permiten automatizar tareas siempre y cuando tengamos montado un ecosistema completo. Son aparatos, a priori, simples de utilizar y que pueden ser muy útiles en diferentes aspectos.

Si bien, es cierto, que este tipo de tecnología parece no estar diseñada para la población más adulta, y que para la gente más joven, tiene una barrera de entrada muy pequeña, este proyecto tiene como finalidad el hacer que personas de ese rango de edad para las que no suelen estar dirigidos estos aparatos, lo usen en su día a día y les puedan servir de gran ayuda, e incluso de compañía. Y sobretodo, darnos cuenta de que complementos de este tipo pueden ser muy útiles para paliar en parte problemas generados por vivir en zonas despobladas, tener situaciones de dependencia o vivir en soledad, ya que puede ponerse al alcance de su voz mecanismos para pedir ayuda inmediata, dar avisos importantes, o incluso mantener conversaciones.

Este tipo de aparatos tienen un problema añadido que habrá que tener en cuenta en el desarrollo del proyecto, y es la privacidad. Estamos poniendo en nuestras casas unos dispositivos que se encuentras en un estado de escucha constante a la espera de que les despiertes con una palabra de activación que le haga hacer algo. Pero durante todo ese tiempo el altavoz sigue escuchando.

El funcionamiento interno de estos altavoces intenta imitar como sería una conversación humana. En una conversación humana cualquiera de los interlocutores podría iniciar la conversación, en este tipo

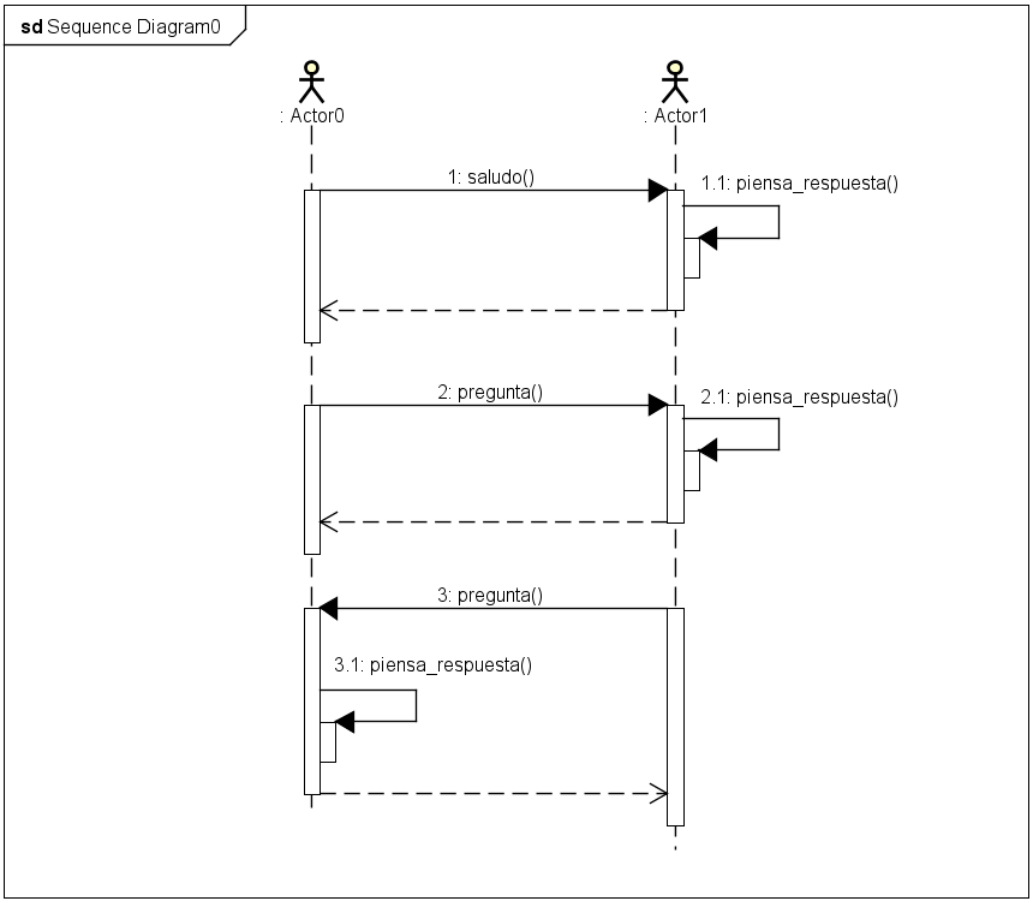

Figura 2.1: Diagrama de secuencia de una conversación humana

de dispositivos no es lo habitual. Sin embargo una función interesante para el sistema a diseñar, ya que está pensado para personas que vivan solas y que puedan tener algún problema, sería que los altavoces narrasen en ciertos momentos del día algunas alertas y ya de paso preguntasen a la persona si necesita algo.

Más adelante cuando expliquemos los antecedentes de este trabajo, hablaremos de uno cuyo objetivo era la creación de una funcionalidad de este tipo.

Un diagrama de secuencia que adapte esta lógica al problema conversacional entre una persona y un altavoz sería así. El usuario diría la palabra de activación que despierte al altavoz. Este habitualmente responde con alguna señal sonora que indica que puedes comenzar a hablar y hacerle una petición. Esa petición sería enviada en crudo por el altavoz y ya en el servidor sería procesada. Se saldría a internet para buscar la respuesta más apropiada y lo que devolviese el servidor sería de nuevo la respuesta convertida a texto para que la narre el altavoz de vuelta al usuario. Más adelante veremos que esta secuencia difiere en Mycroft y explicaremos las modificaciones que se hacen al respecto.

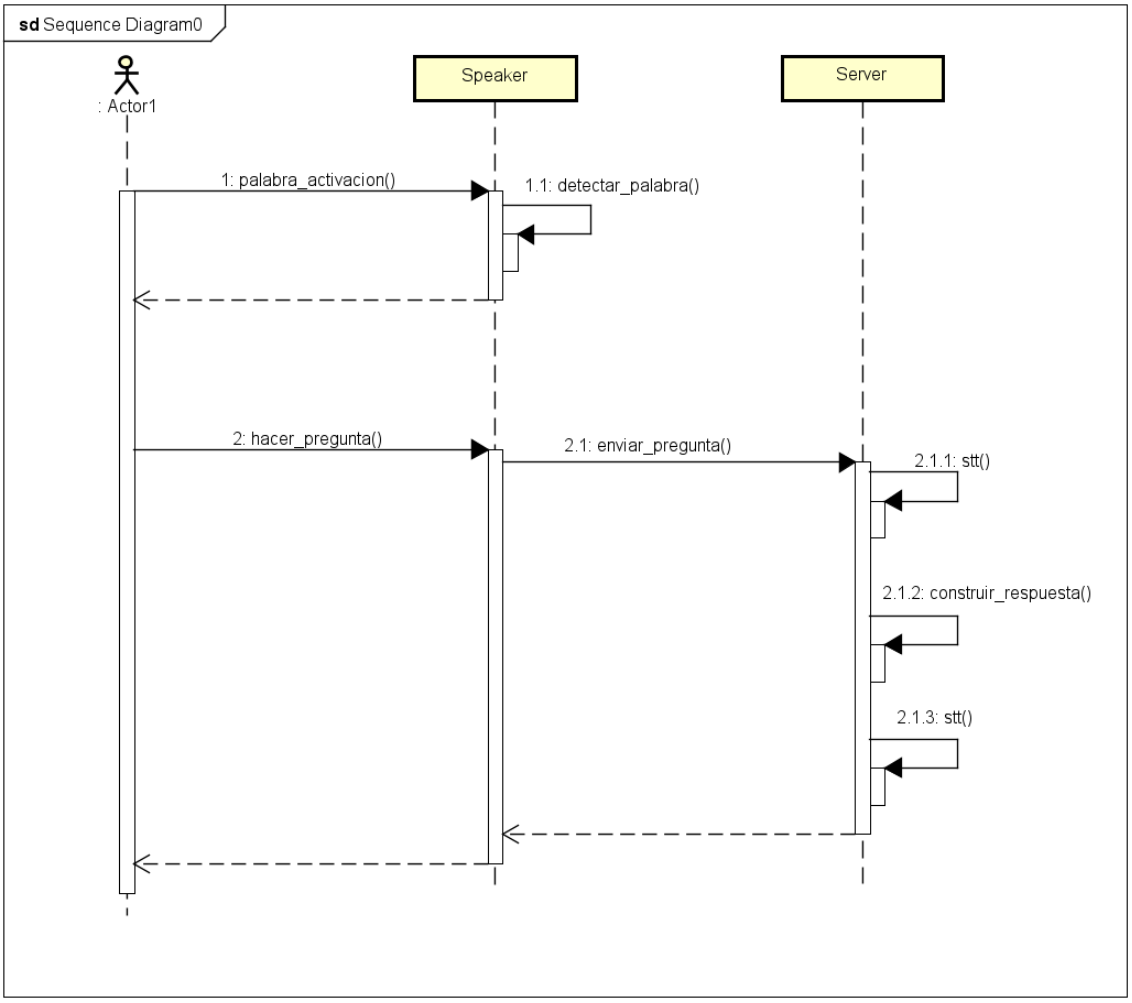

Figura 2.2: Diagrama de secuencia usuario-altavoz

2.3. Motivación: El problema de la necesidad de conexión a Internet

Como hemos comentado anteriormente la mayoría de estos dispositivos necesitan una conexión a internet fluida y constante. La naturaleza de un dispositivo de estas características, pensado para atender a personas dependientes, y residentes en zonas con muy bajas densidades de población, confronta totalmente con el poder tener acceso a una conexión a internet fluida.

Según un estudio realizado en el año 2019 por el Ministerio de Asuntos Económicos y Transformación Digital [6], un 28,87 % del territorio Español aún no dispone de una conexión a internet de velocidad superior a los 10 Mbps, concentrándose las zonas sin esta conexión principalmente en Castilla y León, Aragón, y el norte de Castilla la Mancha. La falta de conexión a internet en estas zonas se localiza principalmente en aquellos lugares donde se concentra poca población. Se puede extraer de este mismo estudio, que, de media, el acceso a conexiones de internet de velocidad superior a 10 Mbps llega al 72,43 % de los hogares en poblaciones con más de 2000 habitantes, pero que sin embargo, en poblaciones de menor tamaño, este porcentaje disminuye hasta el 39,83 %.

Las zonas con baja cobertura de internet coinciden con las más despobladas de nuestro país. En los últimos años se están haciendo esfuerzos por dar a conocer las dificultades añadidas que suponen vivir en alguno de estos lugares que se encuentran en lo que hoy comúnmente se conoce como ”la España Vaciada”. Una de las principales consignas de los colectivos que defienden la necesidad de mejorar la calidad de vida en estos lugares es, además de aumentar los servicios prestados en estas zonas, la mejora de la cobertura de internet. En un mundo globalizado y conectado como el que vivimos desde hace casi ya dos décadas, internet se ha convertido en un bien de primera necesidad, y la falta de acceso a la red agrava un problema que ya venia dándose desde hace años con la migración de los habitantes de los pueblos a las ciudades por la falta de oportunidades en el medio rural. Sin internet ”no se puede vivir”.

Según un estudio realizado por la OCU en Julio de 2020 [2], un 28 % de la población encuestada (personas de 25 a 74 años) se ha planteado cambiar su lugar de residencia, y al preguntar a esas mismas personas por su preferencia sobre donde les gustaría vivir, de las que viven en un entorno urbano o semiurbano, es decir, una ciudad, o en los alrededores de una ciudad, un 30 % querrían irse a vivir a un entorno rural, es decir, al campo o la montaña, o bien a un pueblo (semirrural). Además, del total de encuestados que viven en un entorno rural o semirrural, que es el 24 %, un 80,5 % prefiere seguir viviendo en el mismo hábitat o uno similar. Una de las principales requerencias de las personas que viven en ciudades y que se plantearían volverse al campo pasa por poder teletrabajar, para lo cual hace falta una conexión a internet que sigue sin ser suficiente en muchos lugares para poder trabajar a distancia. Y ya no solo teletrabajar. Cada vez más emprendedores encuentran en el campo una oportunidad para poder montar nuevos negocios.

Por esto se están haciendo muchos avances en nuevas maneras de llevar internet a los pueblos. Algunas de las empresas más grandes del sector tecnológico como Google o SpaceX están avanzando en este tipo de tecnologías, que permitirían, sin la necesidad de tener que desplegar miles de kilómetros de cable desde los grandes núcleos de población más cercanos, dotar a estas zonas de una conexión a internet de alta velocidad. Los más importantes son los siguientes:

- Google esta poniendo en pruebas un proyecto llamado LOON. Consiste en el despliegue de globos aerostáticos similares a los que se usan en meteorología, que van equipados con paneles solares que alimentan unas antenas de radio, que permiten recibir la señal de internet desde unos emisores que el proveedor tenga en superficie, para posteriormente redistribuirla desde el aire el internet, tanto a los usuarios finales como a otros globos cercanos, y así extender la red, sin necesidad de una infraestructura física al uso [14].
- SpaceX por su parte esta probando un sistema más costoso, pero con más perspectiva a largo plazo, Starlink, un sistema compuesto en la actualidad por más de 1000 satélites, pero que cuando la constelación completa esté en orbita rondarán los 30000. A diferencia del sistema de Google funciona mediante antenas satélite, por lo que los futuros usuarios necesitarán de un receptor en el lugar donde quieran darle uso. Actualmente se encuentra en fase de pruebas en Estados Unidos, Canadá y Reino Unido. Samsung tiene un proyecto similar en desarrollo [3].

Debido a toda esta casuística, que no tiene vistas a ser solucionada en el corto plazo, el sistema que estamos diseñado trata de incorporar ciertas funcionalidad que lo hagan mínimamente dependiente de internet.

2.4. Caching

A modo introductorio podría decirse que las cachés son espacios reservados en memoria para guardar algún dato que esperamos tener que volver a usar pronto, bien porque lo hemos usado recientemente o porque es de uso recurrente. Durante años las caches se han usado a bajo nivel, siendo su principal uso el ahorro de operaciones de acceso a disco, disminuyendo en muchos casos el tiempo de respuesta de los ordenadores [7]. A pesar de su extendido uso como herramienta en la gestión de sistemas operativos, con la aparición de internet se le dio un nuevo uso. [10]

2.5. Web caching

Con la generalización del uso de internet el ancho de banda de las primeras instalaciones existentes estaba empezando a coparse. Hacía falta un parche que solventase en parte este problema. Entonces, se adaptó el sistema de caches y se aplico a la navegación web. Básicamente consiste en crear una copia en local de una web la que hayamos accedido muy recientemente, y en caso de querer volver a acceder a ella en un corto plazo de tiempo, en vez de ir al servidor y descargarla de nuevo, si no ha pasado más de ciertos minutos desde la creación de la caché, se utilizará la versión local. Cabe la posibilidad de que la pagina web que está descargada cambie dentro del tiempo que se supone que debería ser válida. En ese caso la web no mostrará la información actualizada [1].

2.6. Web scrapping y los datos abiertos

El Web Scrapping es una técnica que consiste en el análisis de paginas web, para la extracción de datos contenidos en las mismas. Luego esos datos pueden almacenarse para su posterior uso. Técnicas de este tipo son utilizadas por ejemplo por los comparadores web entre otros muchos servicios.

Los datos que almacenan las paginas web puede que a veces estén sometidos a cierta normativa de propiedad, por ello en EEUU ya ha habido sanciones en contra de empresas y personas por hacer scrapping de sitios web [17] y entender que estaban violando una propiedad personal. Por ello en los últimos tiempos, sobretodo en el plano político, se ha hecho un avance en transparencia de datos, y gobiernos municipales y nacionales, así como empresas privadas, están poniendo en marcha proyectos de Datos Abiertos.

Las webs de Datos Abiertos, son sitios donde se publican datos de muy diversos aspectos para el uso general evitando generar conflictos de propiedad de la información, ya que están diseñadas precisamente para darle uso a esos datos. Además en muchas ocasiones estas webs ofrecen los datos ya recopilados en archivos de tipo XML, JSON o similares, facilitando la tarea del programador que necesite utilizarnos y haciendo casi innecesario tener que hacer un scrapper.

2.7. Solución Planteada

Para solucionar el problema que se planteaba para el sistema de altavoces inteligentes que se quiere desplegar, se ha pensado en una solución que combina el web caching y el web scrapping. Luego explicaremos de manera más detallada como se ha hecho cada parte, pero aquí vamos a dar una explicación general de como funciona el sistema.

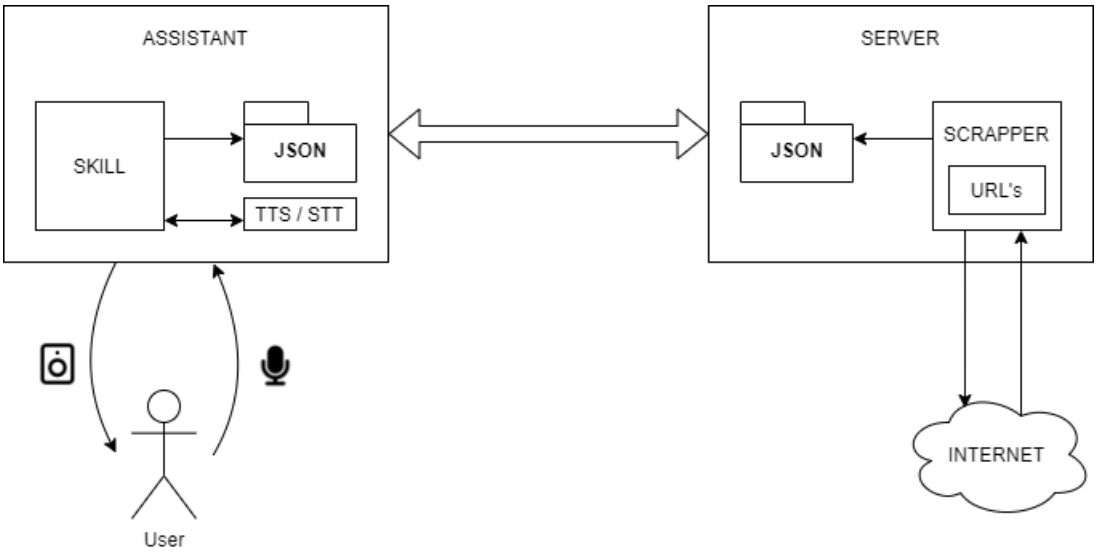

Figura 2.3: Diagrama del sistema

Se creará un scrapper que hay que programar manualmente para cada una de las webs de las que se quiera extraer información. El scrapper lo que hace es, mediante el uso de algunas librerías de Python, que luego detallaremos, analizar las webs en busca de ciertos campos, que son los que queremos extraer para posteriormente almacenarlos en un archivo JSON. Este archivo JSON tendrá catalogados mediante etiquetas los datos que quisiéramos extraer de esa web concreta. Para cada nueva web de la que queramos obtener los datos habría que diseñar un nuevo scrapper. Posteriormente todos los archivos JSON se juntan en uno solo que será la cache atomizada para el municipio en el que se encuentre el altavoz. Esa cache general será la que se envíe a los altavoces para su lectura en caso de no disponer de conexión a internet.

El paso de la cache atomizada al servidor se hace mediante un protocolo que se ha diseñado para este fin que se basa en encapsular en peticiones HTTP el contenido a enviar, y que es desencapsulado en los clientes, ya como archivo JSON [13], preparado para su lectura.

Finalmente se ha diseñado una *skill* para Mycroft [12] cuya funcionalidad consiste en leer la caché recibida desde el servidor para darle al usuario la información que requiera. Se ha diseñado intentando abarcar el máximo número posible de construcciones verbales de las que se puedan formular las preguntas que deberían responderse.

Capítulo 3

Plan de Desarrollo

3.1. Introducción

A continuación detallaremos como se ha desarrollado el proyecto, a que metodología se adapta, cuales han sido las fases de desarrollo del mismo, así como las herramientas utilizadas, incluyendo software de desarrollo, IDE, librerías, o los sistemas operativos.

3.2. Calendario de realización del Trabajo de Fin de Grado

La duración del TFG en este grado es de 300 horas. Debido a que las practicas las he desarrollado junto con este Trabajo de Fin de Grado, y en el mismo laboratorio, y al no tener asignaturas por cursas, ni ningún compromiso laboral externo, la dedicación del desarrollador ha podido ser completa.

Desde el inicio de este segundo cuatrimestre hasta la fecha limite de entregas ha habido 154 días, a los que hay que restar como mínimo los 44 días de fin de semana que ha habido de por medio. Teniendo en cuenta que entre practicas y TFG suman 600 horas, si repartimos estas horas entre los 110 días disponibles, nos sale un promedio de 5,45 horas al día. Si bien desde Febrero hasta Junio la jornada habitual ha sido de 9 a 2 de la tarde, es decir, 5 horas/día, y para compensar el resto de horas, durante el mes de Junio y Julio, además de las 5 horas de por la mañana se ha seguido con el trabajo por la tarde para cumplir los plazos exigidos, teniendo en cuenta otras festividades como Semana Santa o el Día de Castilla y León. Se ha utilizado todo el tiempo disponible.

3.3. Plan de desarrollo del software

La metodología que hemos utilizado para el desarrollo de este proyecto ha sido una metodología Incremental. En esta metodología de desarrollo de software se va construyendo el proyecto final de manera progresiva. En cada etapa incremental se añade una nueva funcionalidad, lo que permite ver resultados de una forma más rápida en comparación con el modelo en cascada.

Además permite que el software se puede empezar a utilizar incluso antes de que se complete totalmente y, en general, es más flexible que otras metodologías.

La ventaja de haber usado una metodología incremental además es que por necesidades de desarrollo todo el proyecto ha sido desarrollado de una forma muy modular, de tal manera que añadir nuevas funcionalidades a lo que ya está diseñado no supondrá un problema, ya que todos los posibles elementos a añadir tienen un claro lugar de encaje.

El uso de esta metodología se ha dado sobretodo en la creación del sistema de paso de las cachés entre cliente y servidor, donde hasta hacerlo funcionar completamente, ha habido 4 iteraciones:

- En la primera iteración se creó un cliente y un servidor básicos que fuesen capaces de comunicarse entre ellos pasándose un mensaje el uno al otro.
- Para la segunda iteración se crearon los diagramas de estados que veremos más adelante tanto del lado del cliente como del lado del servidor y se fueron implementando sobre la estructura previamente creada las primitivas que se diseñaron.
- Hizo falta hacer una tercera iteración para cambiar como estaba montando el servidor ya que debido a un fallo de diseño hubo que incluir una primitiva más y para que fuese más modular se cambio de uno a varios endpoints, de tal manera que cada nueva primitiva ejecutaba una porción de código del servidor y así no había casi que cambiar lo anterior.
- La cuarta iteración simplemente añadió otra primitiva más de la que nos dimos cuenta mientras implementábamos la tercera.

Luego en el análisis del proyecto explicaremos que esto forma parte de un proyecto más amplio, para el que ya se habían desarrollado dos trabajos previos que necesitaban de cierta adaptación. Por ello la primera parte del desarrollo de este proyecto no se adapta a la metodología indicada, pero si el desarrollo de la mejora propuesta para el sistema general de altavoces desarrollada en este trabajo, que es el diseño e implementación de un software que crea caches atomizadas por población para su futuro uso offline.

3.4. Riesgos

En esta sección analizaremos los riesgos por los que el proyecto puede verse afectado. Estos riesgos pueden suponer desde un retraso en la fecha de entrega del proyecto hasta un posible aumento de los costes. Para ello nos hemos guiado del libro estandarte sobre la gestión de proyectos [5].

Los riesgos deben analizarse, según el libro de referencia, teniendo en cuenta la probabilidad de que acabe pasando y el posible impacto que pueda tener en el proyecto en general.

Los riesgos detectados para este proyecto son los siguientes:

Código	Descripción
R-01	Baja del desarrollador
R-02	Pérdida del trabajo realizado
R-03	Cierre de algun servicio externo
R-04	Cierre del centro de trabajo
R-05	Mala planificación del proyecto

A continuación detallaremos para cada uno de los riesgos el impacto que puede tener el proyecto, la probabilidad de que ocurran y como podría solucionarse o mitigarse.

- R-01: La baja del desarrollador, por enfermedad o cualquier causa sobrevenida sobre la persona que se está encargando de la realización del proyecto es un riesgo que tiene distintas posibles soluciones en función del tiempo que dure el problema:

  Si la circunstancia sobrevenida no se alarga demasiado en el tiempo (identificamos esta variante como R-01.1):

Impacto	BAJO
Probabilidad	SIGNIFICANTE
Mitigación	Dar descanso al desarrollador
Solución	Alargar brevemente el proyecto, o aumentar la intensidad de trabajo a la vuelta de la baja

  Si la circunstancia sobrevenida se alarga en el tiempo (identificamos esta variante como R-01.2):

Impacto	SIGNIFICANTE
Probabilidad	MODERADA
Mitigación	Evitar sobrecargar al desarrollador
Solución	Alargar bastante la duración del proyecto, o teletrabajar desde casa

- R-02: Si debido a algun problema físico con los dispositivos con los que se está desarrollando el código del proyecto, se perdiese el trabajo realizado, supondría un importante retraso en la entrega del proyecto. Estos problemas van desde la rotura de los propios equipos, hasta un posible incendio que arrase el edificio en el que se encuentran estos equipos.

Impacto	ALTO
Probabilidad	BAJA
Mitigación	Utilizar software de almacenamiento en linea para los archivos que conforman el proyecto.
Solución	Rehacer el proyecto o continuarlo desde una versión más antigua que puede que no se haya perdido.

- R-03: Si algun servicio externo como por ejemplo la API de AEMET, o el propio Mycroft, cerrasen o se hiciesen privativos habría que buscar alternativas de uso para ellos.

Impacto	ALTO
Probabilidad	MODERADA
Mitigación	Hacer un proyecto modular que permita el cambio de estos servicios sin tener que alterar la estructura del código.
Solución	Sustituir los servicios.

- R-04: El cierre del centro de trabajo, es un problema que a priori tendría una baja probabilidad de ocurrir, y es un riesgo que habría sido incluso difícil de identificar, pero tras la pandemia generada por el COVID-19 hemos visto que este riesgo puede aparecer. Si bien es cierto que en un mundo informatizado como en el que vivimos este riesgo no debería suponer un gran problema para el desarrollo del proyecto.

Impacto	BAJO
Probabilidad	BAJA
Mitigación	No es posible mitigarlo.
Solución	Teletrabajar desde casa.

- R-05: Una mala planificación del proyecto o unos objetivos excesivamente grandes para el tiempo de desarrollo estipulado del proyecto son riesgos a tener en cuenta.

Impacto	ALTO
Probabilidad	MODERADA
Mitigación	Espaciar menos las reuniones de seguimiento.
Solución	Aumentar la duración del proyecto.

A continuación vamos a colocar en una matriz los riesgos identificados. Esta matriz nos indica si hay algun riesgo inasumible. Los riesgos inasumibles son los que se encuentran por encima de la linea de tolerancia de la matriz. No se ha detectado ningún riesgo inasumible en el proyecto.

3.5. Plan de trabajo

3.5.1. Estudio y puesta en funcionamiento de los sistemas previos

Las primeras semanas de mi trabajo consistieron en hacer una lectura exhaustiva de los dos trabajos anteriores y posterior intento de comprensión del código para tratar de aunar los dos proyectos en uno solo, y ya a partir de ahí construir la ampliación de funcionalidad que iba a realizar. Finalmente debido a la complejidad de la integración tanto del segundo proyecto como del que yo iba a desarrollar, en el sistema de back-end y front-end diseñado en el primero de los proyectos, mi trabajo finalmente se ha limitado a ser una adaptación del primer proyecto para funcionar en Picroft, y al posterior diseño y realización del sistema encargado de crear, pasar y narrar información almacenada en caché para cuando los dispositivos no tengan internet.

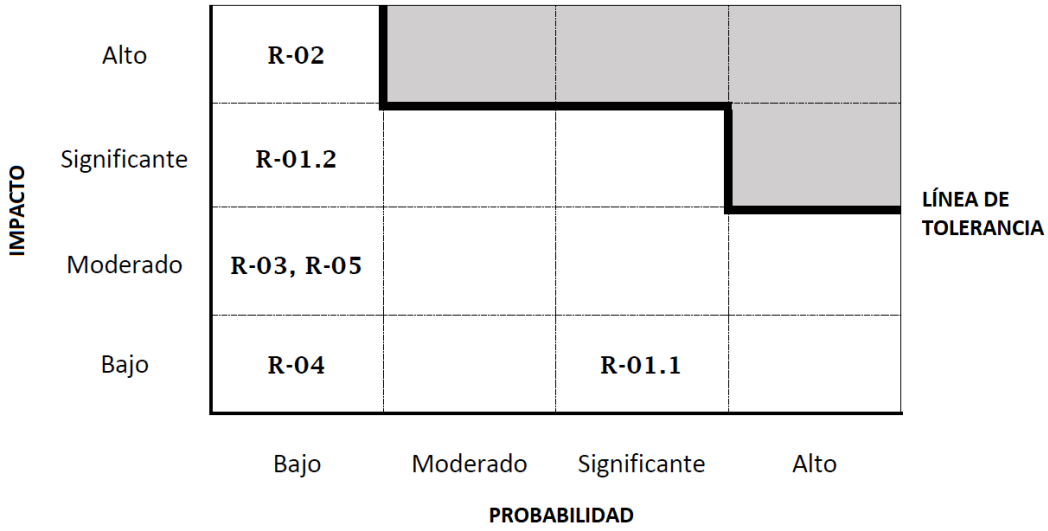

Figura 3.1: Matriz de tolerancia de riesgos

3.5.2. Migración a Mycroft

A continuación hice funcionar en el sistema operativo actual, tras el cambio de Sonos a Mycroft el software de gestión remota de los altavoces. Las las tareas que podían realizar solo se han adaptado las de reinicio y apagado remoto ya que para adaptar el resto era necesario trabajar en la API creada por el anterior estudiante y como hemos explicado previamente a pesar de su modularidad y buen diseño, la carga del trabajo habría aumentado radicalmente. Hay otras funciones como el envío de la información de la última tarea realizada por el altavoz al servidor que tampoco funcionan. La identificación de dispositivos así como la localización geográfica y el servicio que detectaba si estaba conectado y en caso de no estarlo cuanto tiempo llevaba en ese estado, también funcionan.

Respecto al trabajo realizado durante el segundo proyecto, no merecía la pena integrarlo, ya que al igual que con este, se espera que en un futuro se junten en un único cliente que pueda ser controlado por la API diseñada en el Proyecto 1. Además no suponía ningún trabajo de adaptación ya que simplemente habiendo copiado la *skill* al directorio de *skills* de Mycroft, y haber habilitado en el servidor algun endpoint a mayores que realizase las funciones diseñadas para el servidor de este proyecto, habría funcionado sin mayor problema.

3.5.3. Documentación

Antes de dejar de intentar adaptar el sistema completo del Proyecto 1 a Mycroft se estuvo buscando información sobre Kotlin, lenguaje en el que está programado para tratar de finalmente poder integrarlo todo en la misma plataforma. Además de esto hubo un importante trabajo de documentación sobre Mycroft y su funcionamiento, estructura de las *skills*, como no hacerlo depender de internet, como sustituir los servicios de Google por otros que dotaran al sistema se algo más de privacidad y sobre como extraer información de páginas web, y encapsular datos en peticiones HTTP.

3.5.4. Creación de un web scrapper

Una vez finalizado este proceso documental se comenzó con el desarrollo del trabajo que aquí se presenta. Lo primero en desarrollarse fueron los scrapper de la AEMET y de los datos abiertos de la web de la Diputación de Valladolid. Primero se busco un paquete de Python que facilitase esta tarea. Con la consulta a los tutores se utilizo el previamente mencionado BeautifulSoup4 [18].

De modo general, lo primero que se hace es una petición HTTP con el módulo requests [19] que posteriormente utilizaremos para la gestión de la comunicación entre el servidor y la aplicación. La respuesta obtenida es analizada y parseada en función de de qué tipo sea lo que devuelve la petición. En el primer código parsea la respuesta de tipo XML con lxml y el segundo parsea HTML con html.parser:

```
response = requests.get(url, verify=False)
text_response = response.text
soup = BeautifulSoup(text_response, 'lxml')
```

```
response = requests.get(url, verify=False)
text_response = response.text
soup = BeautifulSoup(text_response, 'html.parser')
```

A continuación con la respuesta ya parseada y guardada en este caso en la variable `soup`, con la función `find` se van buscando las palabras clave que identifican los campos que queremos extraer para posteriormente almacenarlos en un fichero, en este caso, de tipo JSON. Para ello abrimos el archivo en modo escritura y vamos guardando los datos extraídos.

Esto se hace para cada web de la que queramos extraer información. Para cada una de ellas se crea un nuevo scrapper. Posteriormente hemos creado un programa que junta todas las caches creadas en una sola para así enviar un único archivo al cliente.

Cuando hay que elaborar las caches tras la petición del cliente el servidor ejecuta un programa que las crea de forma secuencial. Para añadir nuevos scrappers solo habría que incluir una llamada a la clase que la crea, y añadir, copiando y pegando, unas lineas al final del código para que lo concatene con lo anterior.

3.5.5. Creación de una estructura cliente-servidor HTTP básica

Una vez teníamos dos scrappers funcionando, comenzamos con el desarrollo del cliente-servidor que sería el encargado de una vez estuviese creada la caché, enviarla a los dispositivos para su uso. Comenzamos creando una estructura básica y comprobando que el cliente se podía conectar al servidor y este devolver algun tipo de respuesta. Se usa un servidor embebido de la librería Flask. Una vez funcionaba lo básico simplemente habría que adaptar el protocolo que diseñásemos al prototipo básico [8] [15].

Uno de los trabajos más importantes de este trabajo radicaba en la creación del protocolo de paso de mensajes entre cliente y servidor. Tras su diseño, que tuvo 3 iteraciones en las que se fueron cambiando las llamadas que se hacían desde el cliente al servidor para que se adaptase mejor al funcionamiento que queríamos conseguir, se fueron poco a poco creando las peticiones desde el cliente, así como los 3 endpoints que tiene el servidor y las respuestas que le envían de vuelta.

Un valor añadido del servidor es que hace comprobaciones sobre la información que ya tiene descargada de las web en las que se hace scrapping y si la información no ha sido actualizada no vuelve a descargarla. Para ello lo que hace es comprobar la fecha de la última actualización de la información de AEMET y lo compara con la fecha y hora de descarga de la última información obtenida. Si la fecha de actualización de la web es más reciente que la fecha almacenada no vuelve a descargar los datos. Esto sumado a la verificación de que la caché solo se envíe a los clientes cada cierto tiempo hace que el uso de internet se reduzca casi al mínimo.

3.5.6. Creación de una *skill* para Mycroft

Una vez estaba diseñado y funcionando el sistema de creación y paso de cachés, se comenzó con el diseño de la skill que se encargaría de leer los datos almacenados en la cache. Debido a la estructura que hemos explicado previamente, crear una *skill* en Mycroft es relativamente sencillo y por eso se dejó este paso para el final, ya que se pensó que no supondría una carga de trabajo importante.

3.5.7. Redacción del Proyecto

Cuando ya estaba a punto de finalizarse el software con el que funcionaba el proyecto se comenzó de forma paralela con la redacción de la presente memoria. Se hace el presente documento utilizando el editor de LaTeXen línea Overleaf, cumpliendo con los requisitos de estilo que figuran en la web de la escuela.

Capítulo 4

Análisis del Proyecto

4.1. Introducción

A continuación procederemos a explicar brevemente los dos trabajos previos sobre los que se sustenta este proyecto e identificar los requisitos que debería cumplir la versión final, teniendo en cuenta además de las adaptaciones de estos trabajos, los requisitos propios de las nuevas funcionalidades creadas para el sistema. Para la realización de los análisis de coste nos hemos guiado del libro Software Project Management [5]

4.2. Antecedentes

Como hemos comentado anteriormente, este proyecto forma parte de un proyecto más general que consiste en la creación de un sistema de altavoces inteligentes pensados para ayudar a personas en situación de dependencia que residen en lugares apartados o poco poblados en los que es habitual no disponer de una conexión fluida a internet.

Previo a este se han desarrollado otros 2 Trabajos de Fin de Grado que han avanzado hacia la consecución del proyecto general. Es importante explicar los antecedentes a este proyecto ya que las primeras semanas de este trabajo se basaron en la adaptación del código para permitir el funcionamiento del primer proyecto en Picroft y actualizar algunas características de la configuración del segundo, ya que Mycroft había sido actualizado recientemente, para ya luego proceder al desarrollo del trabajo en sí.

4.2.1. Proyecto 1

El primero de los trabajos, y el más importante hasta la fecha, consistió en el diseño de un sistema cliente-servidor distribuido que permitía configurar de forma remota los altavoces inteligentes. Se creo un back-end, compuesto por una API-REST, y una interfaz web.

La API se diseñó en Kotlin y es la encargada de gestionar todas las operaciones entre el cliente, en este caso cada uno de los altavoces, y el servidor donde estaba almacenada la información sobre el usuario que tenia asignado cada dispositivo, así como su estado o su localización geográfica. Para el diseño de la web de gestión, desde la que se ejecutan a través de la API las operaciones para controlar los altavoces, se utilizó Vue, un lenguaje que combina HTML, JavaScript y CSS en un único archivo, facilitando mucho la labor del programador, que no tenia 3 archivos distintos para cada vista de la web.

Las operaciones que pueden llevarse a cabo con este sistema van desde el cambio de ciertos parámetros de los altavoces, así como reinicios, apagados, o actualizaciones del sistema. Este proyecto es de una gran envergadura, y a pesar de que funciona a la perfección, y ser además bastante modular en forma, resulta bastante complicado incorporar las funcionalidades que se desarrollaron en el segundo trabajo, así como las que hemos desarrollado en este.

La parte de front-end, es decir los clientes, se diseñó para funcionar sobre un sistema de código abierto para la gestión precisamente de altavoces inteligentes llamado Snips, que en este caso correría sobre Raspbian. Al final del desarrollo del mismo se anunció que Snips se iba a convertir en un sistema privativo tras su compra por la compañía Sonos. Esto obligó a tener que buscar una alternativa open-source, que fue Mycroft, y su sistema operativo Picroft basado también en Raspbian, pero menos desarrollado que Snips. Esto provocó que el desarrollo de la parte que se refiere al habla no pudiese realizarse finalmente.

4.2.2. Proyecto 2

El segundo proyecto, se realizó de forma casi paralela al primero debido a que la pandemia retrasó los plazos de entrega, y aunque empezó lo suficientemente tarde como para que ya pudiese ser realizado desde cero en Picroft y no en Snips, la idea original era continuar con el desarrollo del trabajo realizado por el otro estudiante.

En este segundo proyecto se ahondó en la parte conversacional de los altavoces y se creo un sistema de alertas con prioridad que buscaba que tras cada conversación de un usuario con el altavoz, o que en momentos concretos del día, este narrase algún aviso importante que el usuario debiese conocer (como por ejemplo sobre la potabilidad del agua). También se comenzó a desarrollar un sistema que extraía información de sitios web para su lectura posterior. Para todo ello de nuevo se realizó un sistema cliente-servidor de cero, con una web de gestión sencilla, un cliente, y el desarrollo de una base de datos dinámica que ordenaba las alertas que debían narrarse en función de su prioridad.

4.3. Identificación de requisitos

A continuación expondremos los requisitos que deberá satisfacer el sistema:

4.3.1. Requisitos funcionales

ID	Descripción
RF-1	El dispositivo debe ser capaz de identificarse ante el sistema.
RF-2	El dispositivo debe poder recibir la caché del servidor.
RF-3	El sistema debe recopilar información de la API de AEMET.
RF-4	El sistema debe recopilar información de la web de la Diputación de Valladolid.
RF-5	El dispositivo debe poder actualizar de forma periódica.
RF-6	El sistema debe comprobar si el dispositivo está registrado.
RF-7	El sistema debe comprobar si la caché descargada en el dispositivo es reciente.
RF-8	El sistema debe comprobar si la información recopilada de la API de AEMET ha sido actualizada.
RF-9	El sistema deberá permitir conversaciones simples de pregunta-respuesta entre el usuario y el altavoz.
RF-10	El sistema deberá permitir al usuario conocer la última información disponible sobre el tiempo.
RF-11	El sistema deberá permitir al usuario conocer la información disponible del ayuntamiento.
RF-12	El sistema deberá permitir al usuario saber de cuando es la última información que está disponible.
RF-13	El sistema deberá permitir al usuario saber en qué localidad se encuentra situado el altavoz.

4.3.2. Requisitos de información

ID	Descripción
RI-1	El sistema deberá almacenar información del municipio en el que esté colocado.
RI-2	El sistema deberá almacenar información sobre la fecha de descarga de la última caché.
RI-3	El sistema deberá almacenar información sobre los dispositivos con acceso al servicio, concretamente, su dirección MAC como identificador, y un token generado en función de la MAC.
RI-4	El sistema deberá almacenar información del ayuntamiento, concretamente, dirección postal, numero de teléfono, numero de fax y el correo electrónico.
RI-5	El sistema deberá almacenar información de la previsión del tiempo, concretamente, la fecha de la previsión, la probabilidad d precipitación, la cota de nieve, el estado del cielo, la velocidad y la dirección del viento, las temperaturas máxima y mínima.

4.3.3. Requisitos No Funcionales

ID	Descripción
RNF-1	Los clientes deberán poder ser desplegados en máquinas Raspberry Pi 3 Modelo B+.
RNF-2	El servidor deberá poder ser desplegado en cualquier sistema capaz de ejecutar código Python.
RNF-3	El sistema dispondrá de una capa de seguridad básica basada en autenticación de los dispositivos mediante usuario y contraseña.
RNF-4	El asistente deberá poder comunicarse en Castellano.
RNF-5	El asistente deberá minimizar el numero de veces que acude al servidor para descargar la caché.
RNF-6	El sistema deberá minimizar el número de veces que descarga nueva información de la API de AEMET.
RNF-7	Los asistentes deberán procesar la mínima información necesaria para ser funcionales.

4.4. Casos de uso

Debido a que este proyecto aunque presenta una parte de interacción con el usuario, principalmente está enfocado al diseño del sistema de creación de caches y paso de estas desde los servidores a los clientes, este diagrama de casos de uso tiene tres:

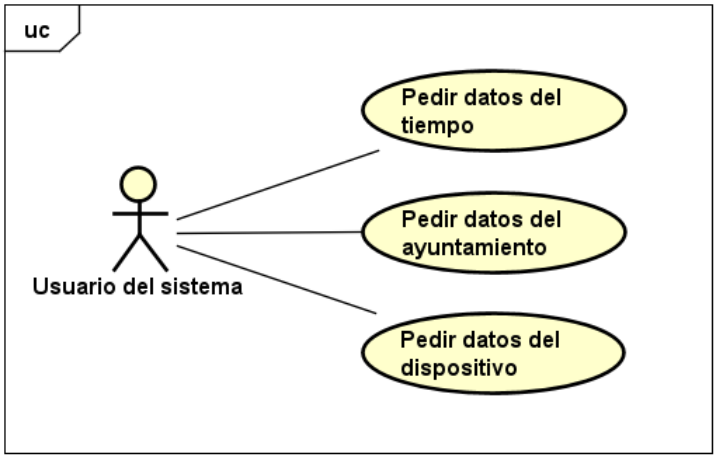

Figura 4.1: Diagrama de casos de uso

Estos casos de uso representan los dos tipos de operaciones que el usuario puede hacer contra el cliente, que son pedir información del ayuntamiento, pedir información del tiempo atmosférico, y pedir información sobre el propio dispositivo, como la fecha de la caché descargada, o el municipio para el que está configurado. El resto de acciones del sistema, al no haber desarrollado un servidor con interfaz gráfica si no que es un servidor que está en constante espera de peticiones de los clientes y cuyo funcionamiento una vez llegan estas es totalmente automático.

4.5. Costes

A continuación haremos una estimación de los costes que podría haber tenido el proyecto si este hubiese sido realizado por una empresa.

- La duración del proyecto son 300 horas. El salario medio de un Programador Junior, es decir un estudiante de Ingeniería Informática que aún no está en posesión del titulo debido a que no ha finalizado sus estudios, en España es de 19477€ [11].

  Si hacemos los cálculos teniendo en cuenta que en España en 2021 habrá 254 días laborables, tenemos que:

$$\frac{9477 \text{ euros al año}}{254 \text{ días laborables}} = 76{,}68 \text{ euros / día laborable}$$

$$\frac{76{,}68 \text{ euros por día laborable}}{8 \text{ horas por jornada}} = 9{,}59 euros/hora$$

$$9{,}59 \text{ euros por hora trabajada} * 300 \text{ horas} = 2877€$$

- Los costes del hardware utilizado son aproximadamente los siguientes [20] [9]:

Componente	Coste
Raspberry Pi 3 Modelo B+	40€
ReSpeaker 2-Mics Pi HAT	10€
Carcasa de plástico impresa en 3D	15€

- Por lo que sumando los costes anteriores, el precio final que tendría el proyecto sería de 2942€.

Capítulo 5

Diseño e implementación

5.1. Recursos y herramientas

A continuación detallaremos todas las herramientas y software utilizado durante el desarrollo del proyecto, así como las librerías más importantes que permiten el correcto funcionamiento del mismo.

5.1.1. Mycroft

Es un sistema de gestión de asistentes de voz de código abierto [12], que permite la creación de *skills* que permitan la interacción humano-altavoz de una manera muy sencilla. Dispone de multitud de herramientas ejecutables mediante linea de código que autogeneran *skills* y permiten editar configuraciones sobre las ya creadas. Para la conversión de texto a lenguaje y viceversa utiliza lo siguiente:

- Google STT (Speech-to-text): Por el momento Mycroft para realizar la conversión de voz a texto utiliza un servicio externo de Google. Es cierto que este proyecto busca poder independizar el sistema de altavoces de internet hemos de explicar que eso de momento no es posible, y es debido a que al correr estos en una Raspberry, resulta imposible que el análisis sintáctico de las ordenes del usuario y la síntesis de voz del altavoz para generar la respuesta se hagan de manera local, ya que este tipo de dispositivos tiene una potencia de computo limitada.
- Adapt: Es el analizador sintáctico que viene configurado por defecto para realizar el análisis de intención, que es lo que se conoce comúnmente como ïntent parser". Es el encargado de detectar si en las palabras que hemos pronunciado se encuentra la palabra de activación.
- Google TTS (Text-to-speech): El sintetizador de voz que utiliza Mycroft por defecto para convertir el texto en voz para que lo narre de vuelta al usuario tras realizar este alguna petición es Mimic, sin embargo, este sistema de momento no comprende entre sus idiomas el castellano, por lo que de nuevo nos vemos obligados a utilizar los servicios de Google.

Gracias a la modularidad de Mycroft estos componentes pueden cambiarse por otros. Actualmente Mozilla se encuentra desarrollando junto a Mycroft un software de speech-to-text de código abierto para dejar de depender de los servicios de Google, aunque por el momento no puede usarse en producción.

5.1.2. Picroft

Picroft es un sistema operativo basado en Raspbian Lite, de código abierto y creado por Mycroft [12] que trae incluido todo lo necesario para poder utilizar Mycroft de forma nativa.

5.1.3. Astah Professional

Programa utilizado para la realización de los diagramas de secuencia y de casos de uso que se exponen en este trabajo, y que hemos utilizado en varias asignaturas a lo largo de la carrera.

5.1.4. draw.io

Para otros diagramas más sencillo he hecho uso de la aplicación online draw.io que permite crearlos de una forma sencilla e intuitiva, y además tiene una gran cantidad de símbolos que permiten hacer representaciones muy variadas, incluso UML.

5.1.5. Microsoft Visual Studio

Es el IDE que se ha elegido para desarrollar el código del proyecto. Es muy versátil y tiene muchas extensiones que permiten visualizar el código muy variados lenguajes, así como ejecutarlos, incluso en paralelo, función que ha sido muy útil para el desarrollo del cliente-servidor.

5.1.6. Overleaf

Editor online de archivos LaTeXpara hacer la presente memoria.

5.1.7. Python

Es el lenguaje utilizado para el desarrollo del proyecto completo. Cliente Servidor y *skill* están diseñados en Python, un lenguaje muy conocido por muchos motivos, entre otros por la simplicidad de su código, su flexibilidad, y una comunidad muy grande que la respalda, creando un gran numero de librerías que simplifican nuestros trabajos. Para las caches así como en los mensajes que se pasan entre cliente y servidor hemos utilizado archivos JSON, por su facilidad de manejo. Las librerías más importantes que hemos utilizado son las siguientes:

- requests: Esta librería [19] nos permite hacer peticiones HTTP de una forma mucho más simple. Además con urllib3 automatiza el uso de conexiones HTTP persistentes, que consiste en utilizar una única conexión TCP y enviar sobre ella múltiples peticiones HTTP. La manera de manejar las peticiones sería la siguiente:

```
r = requests.post(date_url, json=payload)
```

```
cache_status = r.text
```

Se haría la petición y en el payload de la misma se introduciría la información a enviar al servidor. Después la respuesta sería analizada utilizando alguna de las múltiples funciones que incluye la librería como `text` para ver el contenido de la respuesta, `status` para el status que nos envía de vuelta el servidor o `status_code` para ver el código de respuesta.

- json: Nos permite el manejo de este tipo de archivo. Podemos codificar y descodificar cadenas de texto, parsear Strings a JSON y viceversa... Estas funciones son muy útiles para extraer la información devuelta por el servidor tras una petición HTTP y en función de la cadena que contenga ejecutar unas tareas u otras.
- beautifulsoup4 (bs4): Es una librería [18] que nos permite hacer scrapping de sitios web. Es básicamente un analizador de HTML y XML.
- Flask: librería que utilizaremos para ejecutar el servidor.

5.2. Diagrama de tipos de procesos del sistema

En esta sección explicaremos el protocolo que hemos diseñado para la creación y paso de cachés entre los altavoces y el servidor, los posibles procesos del sistema en cada punto y los mensajes. Se ha empleado notación SDL en vez de UML por la facilidad a la hora de representar los mensajes enviados por el cliente y el servidor y poder verlos comparados secuencialmente.

5.2.1. Diagrama del cliente

En el lado del cliente es donde se da el inicio de la comunicación.

- El cliente con el envío de un mensaje de alive inicia el proceso de creación o actualización de la caché remota, y posterior recepción. Con este primer mensaje se envían el usuario y la contraseña que deberán estar previamente registradas en el servidor. El user o id está conformado por la MAC del cliente, y la contraseña se genera mediante un mecanismo diseñado para el servidor del proyecto del back-end, y ha sido reutilizado para ser consistente y hacer más fácil una futura integración. En caso de no coincidir con ninguna de las credenciales guardadas en el servidor este devolverá un mensaje de error y se interrumpirá la comunicación.
- Si el id es correcto, el cliente comprobará si ya tiene una caché descargada.

  En caso de tenerla, extrae la fecha de creación de esa caché y se la envía al servidor para que este compruebe cuanto tiempo tiene. El servidor responde con la diferencia de tiempo, que, en caso de ser inferior a 12 horas, lo entiende como reciente, no descarga una nueva y se interrumpe la conexión. Si han pasado más de 12 horas continua el flujo normal del diagrama.

  Si por el contrario no hay ninguna caché descargada en el dispositivo, se salta esta comprobación y el flujo continua normalmente.

- Una vez se ha comprobado si existe o no cache, y si en caso de existir la caché descargada está actualizada, lo siguiente que se hace es enviar al servidor la localidad en la que está situado el altavoz. El servidor devuelve si puede generar una cache completa para esa población. En caso de no poder se interrumpe la conexión. Si por el contrario se puede el cliente descarga directamente la caché del servidor y continua el flujo normal.
- Si la caché ha sido recibida en el paso anterior, finalmente se guarda en un archivo tipo JSON en la ruta `/home/pi/caches`, para que la pueda leer la *skill* diseñada a estos efectos.

A continuación vemos la figura que representa este flujo.

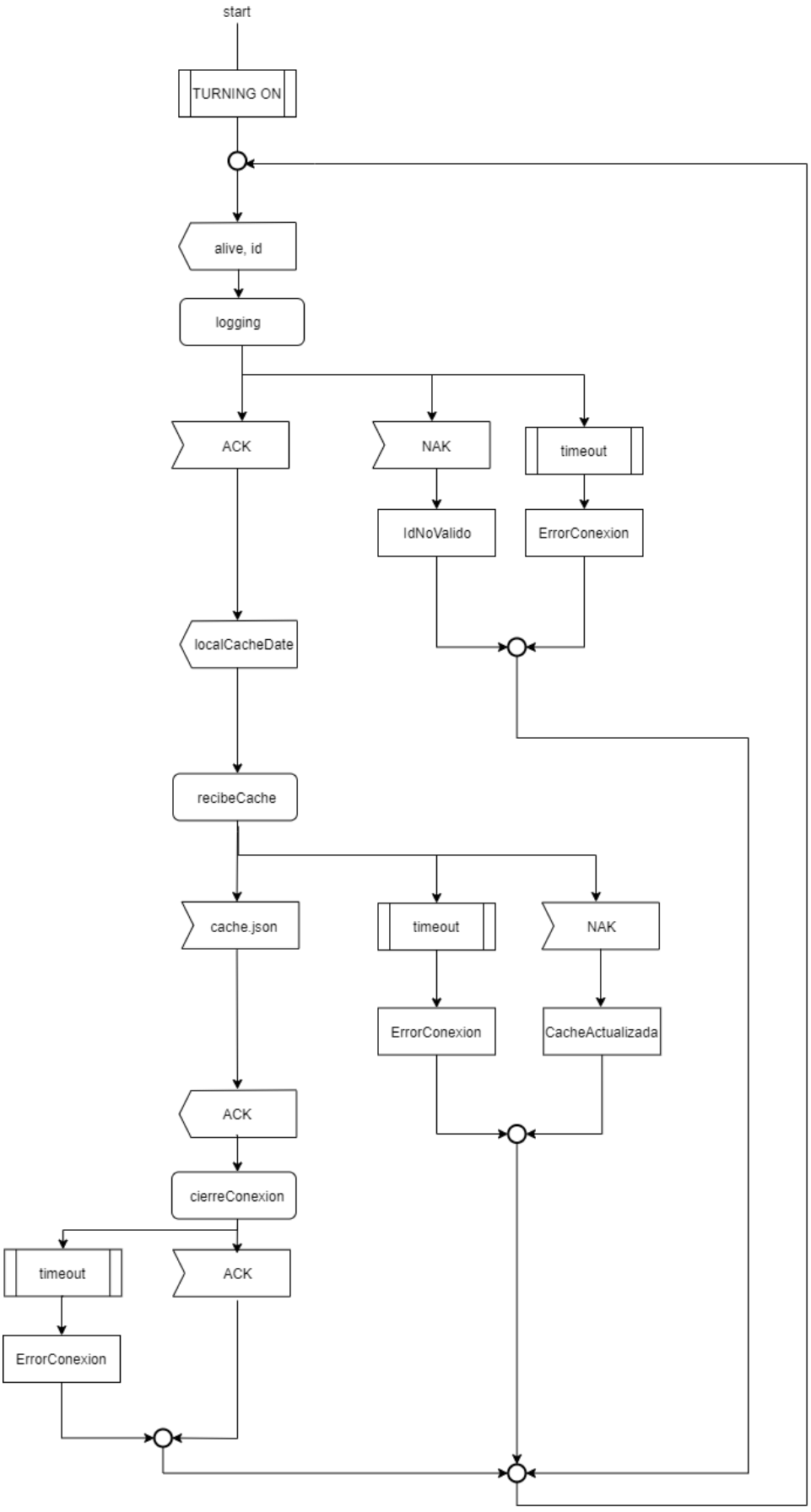

Figura 5.1: Diagrama de tipo de proceso del lado del cliente

5.2.2. Diagrama del servidor

El servidor espera a recibir el alive del cliente. Es el encargado de hacer todas las comprobaciones que solicita el cliente y con sus respuestas permitir o no el avance del flujo del sistema.

El servidor tiene 3 endpoints, uno para el login, otro para la fecha, y finalmente el último que es el que comprueba el municipio y envía la caché.

- El primer mensaje que recibe el servidor es el alive del cliente. Lo recibe en el endopint `/login`. En el se incluyen las credenciales descritas anteriormente. Al llegar al endpoint diseñado para esta función, comprueba si las credenciales recibidas coinciden con alguna de las credenciales almacenadas en un archivo JSON de credenciales. Si se encuentra alguna coincidencia exacta, el servidor devuelve al cliente el status `"Logged in"`, y si no, comprueba si la contraseña es incorrecta. En ese caso devuelve el status `"Wrong password"` y el cliente interrumpe la conexión. En cualquier otro caso, es decir, que el usuario sea incorrecto, o ambos, contraseña y usuario sean incorrectos, devuelve el status `"Wrong username"` y el cliente también interrumpe la conexión.
- En caso de que el cliente ya tenga descargada una caché, se accede al endpoint que comprueba en el endpoint `/date` la diferencia de tiempo entre la fecha de la caché y la fecha actual. Si la diferencia es menor de 12 horas el servidor devuelve el status `"cache updated"` y el cliente interrumpe la conexión. Por el contrario, si es mayor devuelve el status `"actualiza cache"`. Este tiempo puede ser ajustado cambiando un parámetro si considerásemos que 12 horas no no es un valor adecuado para actualizar la caché.
- El último endpoint y el más importante es `/cache` el encargado de comprobar si existe la población que envía el cliente para generar las cachés, si existe si está disponible, y generar los propios archivos.

  Primeramente se comprueba si la población es un municipio elegible de la provincia de Valladolid, que es para donde de momento se ha generalizado la aplicación. Si el ayuntamiento no está contenida en la web de datos abiertos de la diputación o el municipio no figura en la lista de municipios de AEMET, le devuelve al cliente el status `"población no existe o no disponible"` y el cliente interrumpe la conexión.

  Si la comprobación es satisfactoria llama al scrapper y genera la cache, y posteriormente es encapsulada en la respuesta que se envía al cliente para que este disponga de ella.

Incluimos a continuación de nuevo el diagrama de estados para esta secuencia.

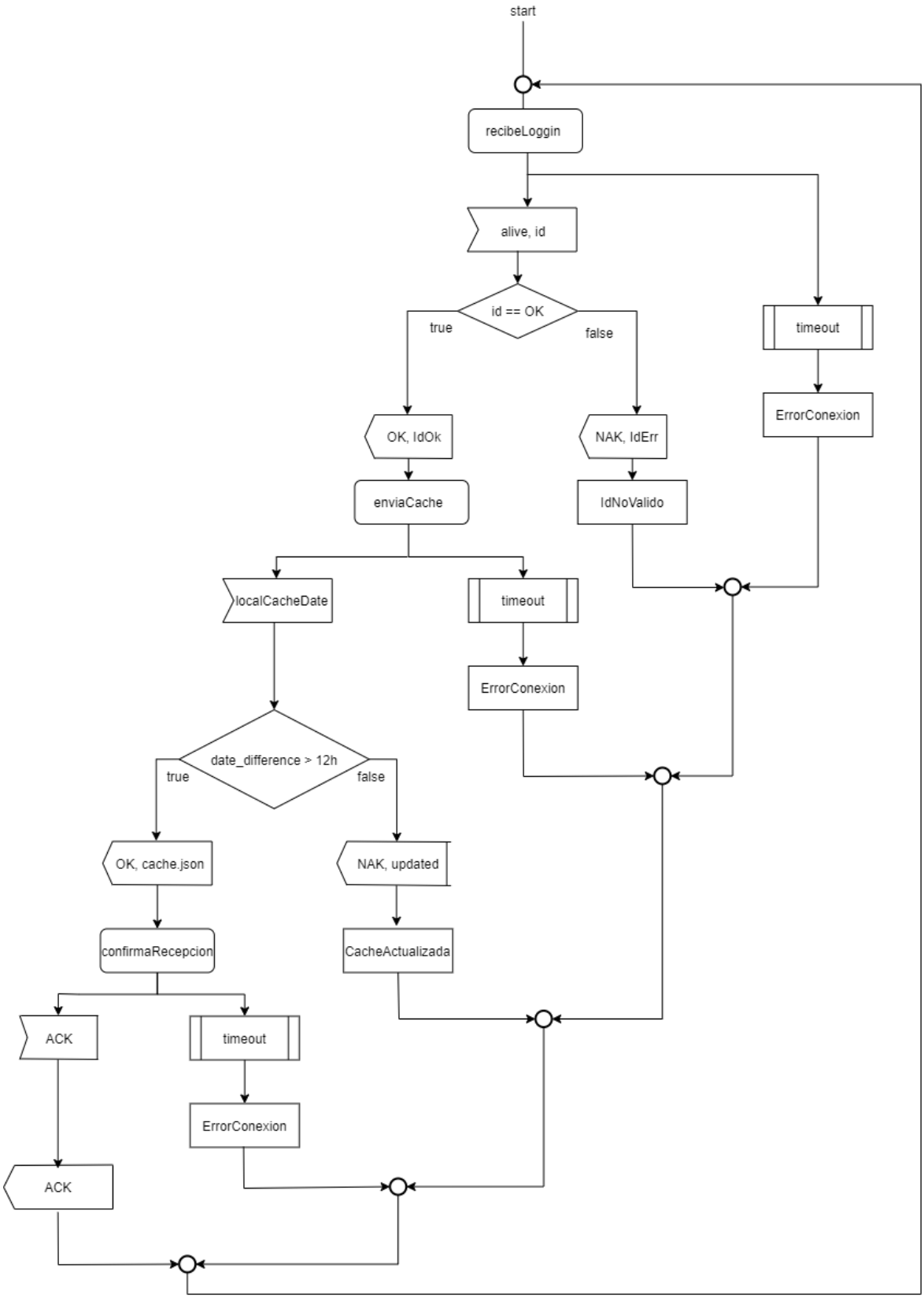

Figura 5.2: Diagrama de tipo de proceso del lado del servidor

5.3. Lógica conversacional y diálogos

La otra parte de este trabajo radica en el diseño de una *skill* para Mycroft [12] que lea las caches recibidas por los altavoces.

Inicialmente el sistema estará basado en conversaciones simples del tipo pregunta-respuesta. Cada vez que requiramos cierta información del altavoz, primero habrá que invocarle mediante el uso de una palabra de activación, en nuestro caso ”Hey Mycroft”. Una vez detecte la palabra, este devolverá una señal sonora que indicará que podemos hacer la pregunta que queramos.

A diferencia de como funcionan los altavoces explicados en el primer capitulo de este trabajo, aquí entran en juego dos sistemas más, un servidor, y el de los servicios externos de Google en este caso. Cuando el usuario activa al altavoz de abre un canal de escucha que graba nuestra petición. Esta se envía en crudo al servidor de Mycroft y este a su vez se lo reenvía a los servidores de Google para que estos hagan la transcripción de audio a texto. Una vez se recibe, el servidor de Mycroft manda la petición al servidor que nosotros hemos creado y este la procesa. La respuesta que se vaya a dar se envía de vuelta al servidor de Mycroft que lo vuelve a enviar a Google, pero esta vez para generar el audio de la respuesta, se envía de nuevo a Mycroft y Mycroft lo envía al asistente para que sea narrado.

Como podemos comprobar, en este punto, se presenta uno de los principales problemas que tenemos para que estos altavoces que se están diseñando funcionen de forma completamente autónoma, y que además deberían poder funcionar de manera offline, no puedan hacerlo. Por el momento es obligatorio que este dispositivo esté conectado al servidor de Mycroft [12] [4], ya que es la puerta de entrada a los servicios de Google que son los encargados de hacer el STT y el TTS.

La traducción de texto a voz y de voz a texto es un proceso computacional de alto coste y por el momento es imposible llevarlo a cabo de manera offline en una Raspberry. Además si no se provee a Picroft de una API externa, es necesario que pase por el servidor por defecto en el que el dispositivo además debe estar previamente registrado. Esto implica que el sistema que estamos creando para el uso offline de estos altavoces no pueda desplegarse por el momento sin conexión a internet, pero el trabajo aquí realizado si serviría en el futuro para ello.

A continuación mostramos el diagrama de secuencia que representa una petición de información a Mycroft tal y como ocurre actualmente con la necesaria conexión a internet.

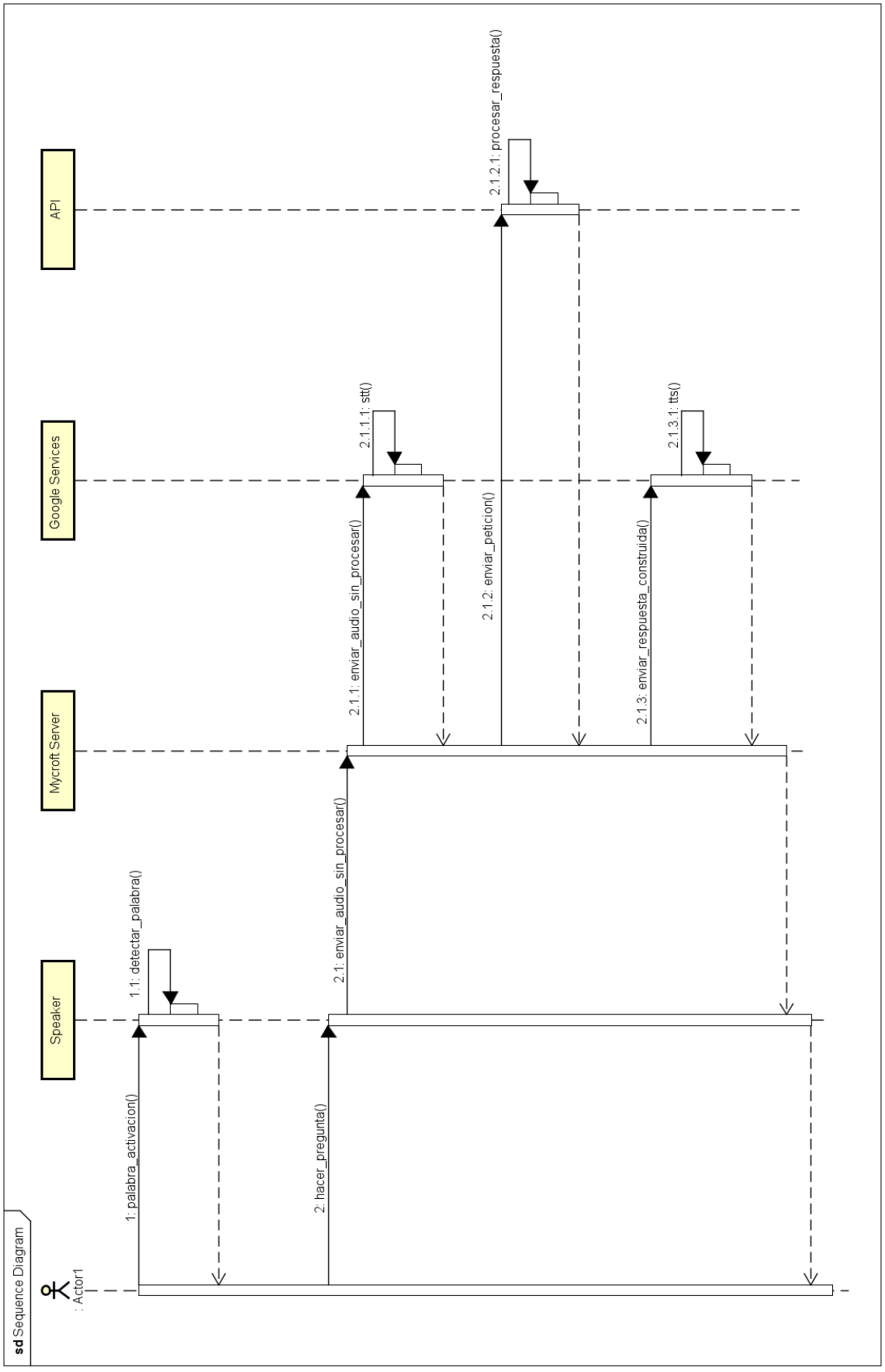

Figura 5.3: Diagrama de secuencia de Mycroft

Los diálogos que se han creado para invocar a las distintas funciones de la *skill* que lee los datos almacenados en la caché contemplan diversidad de formas de hacer las preguntas para tratar de naturalizar lo máximo posible la conversación entre el humano y el altavoz.

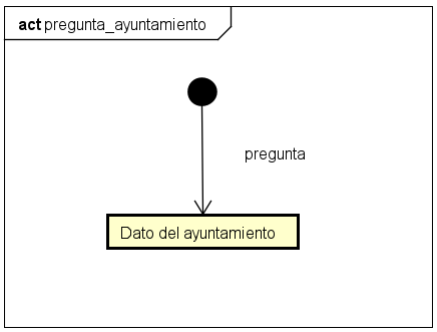

Figura 5.4: Diagrama de interacción de Mycroft para una pregunta sobre el ayuntamiento

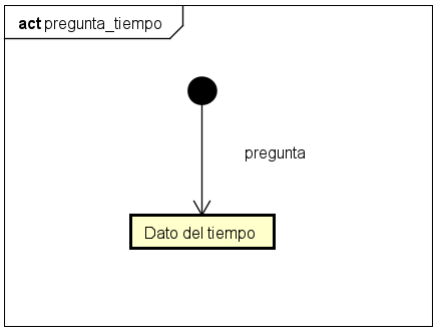

Figura 5.5: Diagrama de interacción de Mycroft para una pregunta sobre el tiempo

Las *skills* de Mycroft tienen la siguiente estructura:

- En la raíz del directorio que almacena la *skill* se encuentra en fichero __init__.py. Este fichero tiene definido en su interior una función que devuelve un objeto del tipo MycroftSkill, y los intent handlers, que son los encargados de la lógica para leer los datos que le pidamos. En este caso solo se ha diseñado un intent handler que lee todos los datos de la caché, pero podría hacerse uno por cada modulo de la caché y cuando se incluyan nuevas funciones, solo habría que crear un nuevo intent handler sin hacer falta modificar los demás.

  Además en este directorio raíz hay una carpeta denominada locale que explicaremos a continuación, archivos de configuración, y por defecto Mycroft crea otros archivos para la sincronización con directorios remotos de GitHub.

- En la carpeta locale hay un subdirectorio de nombre es-es que almacena los tres siguientes archivos:

  - nombreSkill.intent tiene las frases con las que el altavoz va a identificar qué le estas preguntando, y cuál es la *skill* que se debe encargar de responder a esa pregunta. Es en este archivo donde tenemos que dotar a la *skill* del máximo número posible de intents, que es como se denominan a estas frases, para que la conversación sea natural.

- nombreSkill.dialog tiene las frases con las que responde el altavoz a las preguntas que le haga el usuario. Esto además de con este archivo puede hacerse directamente en el fichero `__init.py__` escribiendo en la línea de output el texto que queremos que el altavoz lea para cada posible petición.
- nombreSkill.entity almacena palabras que posteriormente utilizaremos para filtrar en `__init.py__` qué información está pidiendo el usuario y ejecutar una parte del intent handler u otra.

A continuación, en la sección de pruebas, se detallaran los distintos diálogos que se pueden mantener con el asistente.

Capítulo 6

Pruebas

A continuación se detallan las pruebas realizadas con el asistente, donde además se incluyen los diálogos que se pueden mantener con el dispositivo.

6.1. Pruebas del sistema de creación y paso de cachés

Para probar el sistema de creación de cachés y paso de las mismas entre el servidor y el cliente se fueron probando las funciones según se implementaban, y como ya hemos explicado previamente, debido al uso de una metodología incremental todo se iba probando tras su desarrollo. Una vez ha habido un cliente-servidor completamente funcional se han hecho las siguientes pruebas:

P-01	Solicitar caché cuando no hay una descargada
Resultado esperado	Caché descargada
Resultado obtenido	Caché descargada
Test satisfactorio	Sí

P-02	Solicitar caché cuando hay una descargada y tiene más de 12 horas
Resultado esperado	Caché descargada.
Resultado obtenido	Caché descargada.
Test satisfactorio	Sí

P-03	Solicitar caché cuando hay una descargada y tiene menos de 12 horas
Resultado esperado	Caché actualizada. No se descarga una nueva.
Resultado obtenido	Caché actualizada. No se descarga una nueva.
Test satisfactorio	Sí

P-04	Intentar loguearse en el sistema estando el dispositivo bien registrado
Resultado esperado	Logged in.
Resultado obtenido	Logged in.
Test satisfactorio	Sí

P-05	Intentar loguearse en el sistema estando el dispositivo no registrado
Resultado esperado	Wrong username.
Resultado obtenido	Wrong username.
Test satisfactorio	Sí

P-06	Intentar loguearse en el sistema estando el dispositivo mal registrado (nombre de usuario incorrecto)
Resultado esperado	Wrong username.
Resultado obtenido	Wrong username.
Test satisfactorio	Sí

P-07	Intentar loguearse en el sistema estando el dispositivo mal registrado (contraseña incorrecta)
Resultado esperado	Wrong password.
Resultado obtenido	Wrong password.
Test satisfactorio	Sí

P-08	Solicitar una caché para un municipio que no está en la lista de municipios de la AEMET.
Resultado esperado	El municipio no existe o no está disponible para hacer la caché.
Resultado obtenido	El municipio no existe o no está disponible para hacer la caché.
Test satisfactorio	Sí

P-09	Solicitar una caché para un municipio cuya web no está integrada en la web de datos abiertos de la Diputación de Valladolid.
Resultado esperado	El municipio no existe o no está disponible para hacer la caché.
Resultado obtenido	El municipio no existe o no está disponible para hacer la caché.
Test satisfactorio	Sí

6.2. Pruebas de la skill que lee la caché descargada

Primeramente veremos las pruebas que se han hecho respecto de los diálogos relacionados con el tiempo atmosférico. Expondremos todas las maneras de las que se le puede hacer las preguntas al dispositivo, sobre qué se se le puede preguntar, y finalmente el resultado de las pruebas:

- Preguntas sobre el tiempo atmosférico

  1. Dime/Dame la última información disponible sobre el/acerca del/sobre la/acerca de la/sobre las/acerca de las _____
  2. Cuál es la última información disponible sobre el/acerca del/sobre la/acerca de la/sobre las/acerca de las _____
  3. Qué información tienes disponible sobre el/acerca del/sobre la/acerca de la/sobre las/acerca de las _____

4. Dime lo último que sepas sobre el/acerca del/sobre la/acerca de la/sobre las/acerca de las _____
5. Cuál es la ultima información que tienes sobre el/acerca del/sobre la/acerca de la/sobre las/acerca de las _____
6. Qué sabes del/de la _____
7. Cuál es el/la _____
8. Cuáles son las _____
9. A cuánto está la cota de nieve
10. A que altura habrá hoy nieve

- Opciones

1. Tiempo
2. Cielo
3. Temperaturas
4. Temperatura máxima
5. Temperatura mínima
6. Viento
7. Lluvia
8. Nieve

- Resultado de las pruebas: Al dotar a la *skill* de una gran diversidad de preguntas, que pueden ser formuladas en singular o plural y masculino o femenino, y a las que se podría acoplar cualquiera de las posibles opciones de información disponibles, las pruebas son satisfactorias y siempre responden lo que se esperaba.

A continuación expondremos las preguntas que se le pueden hacer al asistente sobre los datos del ayuntamiento:

- Preguntas sobre el ayuntamiento:

1. Dime/Dame la última información disponible sobre el/acerca del _____ del ayuntamiento
2. Dime/Dame el _____ del ayuntamiento
3. Cuál es la última información disponible sobre el/acerca del _____ del ayuntamiento
4. Qué información tienes disponible sobre el/acerca del _____ del ayuntamiento
5. Dime lo último que sepas sobre el/acerca del _____ del ayuntamiento
6. Cuál es la ultima información que tienes sobre el/acerca del _____ del ayuntamiento
7. Qué sabes del _____ del ayuntamiento
8. Cuál es el/la _____ del ayuntamiento

9. Dime/Dame la última información disponible sobre el ayuntamiento
10. Cuál es la última información disponible sobre el ayuntamiento
11. Qué información tienes disponible sobre el ayuntamiento
12. Dime lo último que sepas sobre el ayuntamiento
13. Cuál es la ultima información que tienes sobre el ayuntamiento
14. Qué sabes del ayuntamiento

- Opciones

1. Email
2. Correo electronico
3. Correo
4. Teléfono
5. Fax

- Resultado de las pruebas: Al dotar a la *skill* de una gran diversidad de preguntas a las que se podría acoplar cualquiera de las posibles opciones de información disponibles, las pruebas son satisfactorias y siempre responden lo que se esperaba.

Por último exponemos las preguntas que se pueden hacer sobre el propio dispositivo, para ayudar a los usuarios a determinar si la información que están recibiendo está vigente:

- Preguntas sobre el dispositivo:

1. Dime para qué pueblo estoy configurado
2. Dime en qué pueblo estoy
3. Cuál es mi pueblo
4. En qué pueblo estoy
5. De cuándo es la última información disponible
6. De cuándo es esta información
7. De qué día es esta información
8. De qué fecha es esta información

- Las preguntas son cerradas y no tienen opciones
- Resultado de las pruebas: Permitir a los usuarios que conozcan de cuando es la información almacenada les ayudará a determinar si es válida o no. Ante estas preguntas se comprueban la fecha de descarga de la caché y el municipio para el que se ha creado. Todas las preguntas reciben la respuesta que se esperaban para ellas.

Capítulo 7

Despliegue

El despliegue del sistema se llevará a cabo en 3 fases. La primera consistirá en la instalación de Picroft en una Raspberry y en hacer todas las configuraciones necesarias para que funcione con el hardware que hemos utilizado, e integrar la funcionalidad del Proyecto 1 en el sistema operativo.

La segunda fase consistirá en el despliegue y ejecución del servidor que hemos creado. Esta tarea debería ser la más sencilla de todas si se configura de forma correcta un entorno virtual de ejecución en Windows o si no ejecutarlo en una máquina Linux.

La tercera consistirá en el despliegue y ejecución del cliente en Picroft, así como la instalación de la *skill* en el sistema montado en el paso 1 de esta guía de despliegue.

7.1. Preparación de Picroft

A continuación exponemos los pasos que deben realizarse para la correcta puesta en funcionamiento de Picroft.

1. Flasheamos Mycroft en la microSD, y elegimos configuración manual, y ejecutamos `sudo raspi-confi` para cambiar la contraseña a ”piz la distribución de teclado a español.

2. Ejecutamos las siguientes órdenes para instalar los controladores de la tarjeta de sonido que nos permitirá interactuar vocalmente con la Raspberry:

   - `pip install requests`
   - `pip install pyopenssl`
   - `sudo apt-get update`
   - `sudo apt-get upgrade`
   - `git clone https://github.com/respeaker/seeed-voicecard.git`
   - `cd seeed-voicecard`
   - `sudo ./install.sh`

3. Reiniciamos la Raspberry, y al encenderse de nuevo ejecutamos `mycroft-cli-client` para que se abra el cliente de Mycroft y poder registrarlo en la web https://account.mycroft.ai/devices. Reiniciamos de nuevo la Raspberry y comprobamos que las *skills* básicas funcionan en inglés.

4. Editamos el fichero de configuración `/etc/mycroft/mycroft.conf` y sustituimos las lineas expuestas a continuación para configurar las lineas de entrada y salida de audio que vienen configuradas por defecto en la Raspberry, para que detecte correctamente los micrófonos de la tarjeta de sonido, y que la salida de audio se produzca por el altavoz conectado a la misma . Las líneas a sustituir son las siguientes:

```
''play_wav_cmdline'': ''aplay -Dplughw:ArrayVAC10,0 %1''
```

   por

```
''play_wav_cmdline'': ''aplay -Ddmix:CARD=seeed2micvoicec,DEV=0
   %1''
```

   y

```
''play_mp3_cmdline'': ''mpg123 -a plughw:ArrayUAC10,0 %1''
```

   por

```
''play_mp3_cmdline'': ''mpg123 -a
   plughw:CARD=seeed2micvoicec,DEV=0 %1''
```

5. Ejecutamos a continuación los dos siguientes comandos para cambiar el idioma en el que Mycroft comprende nuestras palabras y también el idioma con el que responderá:

   - `mycroft-config set lang "es-es"`
   - `mycroft-config set stt.mycroft.lang "es-es"`

6. Ejecutamos el comando `mycroft-config edit user` y sustituimos el contenido por lo siguinte para acabar de configurar el idioma de Mycroft:

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

7. Reiniciamos de nuevo la Raspberry, y al encenderla probamos que Mycroft ejecute *skills* básicas ya en Castellano.

8. Copiamos el contenido de pregonero-controller del Proyecto 2 en `/home/pi/pregonero/` y ejecuto `setup.sh`, y el contenido de pregonero-skill también del Proyecto 2 en `/opt/mycroft/skills/`.

9. Copiamos el contenido de assistant.task del Proyecto 1 en `/home/pi/`.

10. Ejecutamos `sudo apt install python3-pip` y `sudo apt install python-pip`.

11. Accedemos al directorio de assistant.task y ejecutamos `sh requeriments.sh`. Por lo general nos obliga a ejecutarlo dos veces.

12. Reiniciamos la Raspberry y al encenderse Mycroft es completamente funcional con sus *skills* básicas y las operaciones de gestión remota de la Raspberry pueden hacerse desde el backend del Proyecto 1, ya que se conecta directamente tras ejecutar el paso 11.

7.2. Instalación del servidor

Copiamos la carpeta server en la máquina donde queramos ejecutarlo. Esta contiene los scrappers, asi como el programa que crea las caches de cada scrapper y el que las junta en un solo archivo, además del propio servidor. Habría que ejecutar en el entorno en el que quisiéramos hacer correr el servidor el comando `pip install -r requirements.txt` que instala los paquetes necesarios para la correcta ejecución del entorno virtual en el que el servidor se ha comprobado funciona.

Después, para ejecutar el servidor simplemente hay que hacer `python server.py`

Para que pueda funcionar correctamente debemos introducir en el archivo `credentials.json` la MAC de la máquina en la que se ejecute el cliente y su clave asociada.

7.3. Instalación de la *skill*

Para la instalación de la *skill* que lee los datos de la caché simplemente tenemos que copiar la carpeta speaker-reader-skill a la ruta `/opt/mycroft/skills` y reiniciar la Raspberry para que cuando Mycroft se inicie cargue la *skill*.

7.4. Instalación del cliente

Para la instalación del cliente copiamos la carpeta client en el directorio `/home/pi` de Picroft. Es necesario previamente ejecutar el programa `setup.py`, que crea los directorios necesarios para el correcto funcionamiento del cliente, uno en el que se almacenan las caches que se descargan del servidor, y un archivo JSON con una localidad por defecto para que el cliente no falle a la hora de pedir una caché al servidor. La localidad por defecto será Wamba.

A continuación, para ejecutar el cliente simplemente hay que hacer `python client.py`

Capítulo 8

Conclusiones y trabajo futuro

A continuación y para finalizar esta memoria expondremos unas breves conclusiones y que trabajo queda para el futuro.

Un trabajo de este tipo te ayuda a comprender en mejor medida la dificultad que puede llegar a tener sacar adelante ciertos proyectos, y haberme dado cuenta con este que es sencillo, me hace pensar en todas las dificultades que tienen que surgir en proyectos de mayor calado, de cuanto tiempo de reflexión y análisis hay que emplear para que las cosas funcionen como deben.

Otra conclusión que he sacado es que el trabajo en equipo es mejor, porque tener a alguien al lado que te ayude o al que poder ayudar cuando no se sabes por donde seguir. Entiendo ahora que durante toda la carrera se nos haya forzado a hacer trabajos de este tipo ya que considero que es algo completamente necesario para la vida laboral.

También es muy importante tener una buena planificación del proyecto, marcarse plazos y horarios, y metas asequibles, o hitos, que te permitan comprobar que avanzas en la buena dirección.

Como trabajo futuro remarcaría las siguientes tareas:

- Se podría crear un script que sea capaz de crear nuevos archivos JSON de forma semiautomática, y no haya que programarlo manualmente para cada nuevo sitio web del que queramos hacer scrapping. Buscando en internet aparecen grupos de trabajo que han avanzado en analizadores de webs de este tipo que luego son utilizados en otras webs del tipo comparador, como www.trivago.es, www.kayak.es u otras webs del estilo.
- Integrar el sistema aqui creado en el back-end diseñado en el desarrollo del Proyecto 1.
- Ajustar de forma dinámica el tiempo que tardan las cachés en actualizarse, en vez de hacerlo mediante un parámetro fijo.
- Cambiar la palabra de activación de Mycroft.
- Crear diálogos complejos con menús que permitan al usuario explorar todas las funcionalidades que le da la *skill* e ir ampliándola con las nuevas funciones que se añadan si se incluye alguna

nueva web para ser scrappeada.

Bibliografía

[1] Barish, G., y K. Obraczke. «World Wide Web caching: trends and techniques». IEEE Communications Magazine, vol. 38, n.º 5, mayo de 2000, pp. 178-84. IEEE Xplore, doi:10.1109/35.841844.

[2] «Ciudad, campo... ¿dónde te gustaría vivir?» https://www.ocu.org/consumo-familia/derechos-consumidor/noticias/encuesta-preferencias-donde-vivir.

[3] Foust, Jeff. «SpaceX's space-Internet woes: Despite technical glitches, the company plans to launch the first of nearly 12,000 satellites in 2019». IEEE Spectrum, vol. 56, n.º 1, enero de 2019, pp. 50-51. IEEE Xplore, doi:10.1109/MSPEC.2019.8594798.

[4] How to Make Mycroft run Offline to Have Real Privacy. https://wltd.org/posts/how-to-make-mycroft-run-offline-to-have-real-privacy.

[5] Hughes, Bob, y Mike Cotterell. Software Project Management. 5. ed, McGraw-Hill, 2009.

[6] Informe de Cobertura de Banda Ancha en España en el Año 2020. Vicepresidencia Segunda del Gobierno - Ministerio de Asuntos Económicos y Transformación Digital - Secretaria de Estado de Telecomunicaciones e Infraestructuras Digitales, mayo de 2021, https://avancedigital.mineco.gob.es/banda-ancha/cobertura/Documents/Cobertura-BA-2020.pdf?csf=1&e=lVCXmu.

[7] Mertz, Jhonny, y Ingrid Nunes. «Understanding Application-Level Caching in Web Applications: A Comprehensive Introduction and Survey of State-of-the-Art Approaches». ACM Computing Surveys, vol. 50, n.º 6, noviembre de 2017, p. 98:1-98:34. January 2018, doi:10.1145/3145813.

[8] Olivia Smith. Performing an HTTP Request in Python. https://www.datacamp.com/community/tutorials/making-http-requests-in-python.

[9] Overview - Seeed Wiki. https://wiki.seeedstudio.com/ReSpeaker_2_Mics_Pi_HAT/.

[10] Pablo Rodriguez, et al. Web Caching Architectures: Hierarchical and Distributed Caching. https://www.researchgate.net/profile/Pablo-Rodriguez-49/publication/2596067_Web_Caching_Architectures_Hierarchical_and_Distributed_Caching/links/02e7e52f42835e74e9000000/Web-Caching-Architectures-Hierarchical-and-Distributed-Caching.pdf.

[11] Salario de un Programador/a junior en España. https://es.indeed.com/career/programador-junior/salaries.

[12] Schweppe, Derick. «Mycroft – The Open Source Privacy-Focused Voice Assistant». Mycroft, https://mycroft.ai/.

[13] «Sending JSON request with Python». Stack Overflow, https://stackoverflow.com/questions/8634473/sending-json-request-with-python.

[14] Soujanya Katikala. «Google Project Loon». RIVIER ACADEMIC JOURNAL, VOLUME 10, NUMBER 2, FALL 2014, https://www2.rivier.edu/journal/ROAJ-Fall-2014/J855-Katikala_Project-Loon.pdf.

[15] Triajianto, Junian. Simple HTTP Web Server and Client in Python. 20 de diciembre de 2016, https://www.godo.dev/tutorials/python-http-server-client/.

[16] Wortmann, Felix, y Kristina Flüchter. «Internet of Things». Business & Information Systems Engineering, vol. 57, n.º 3, junio de 2015, pp. 221-24. Springer Link, doi:10.1007/s12599-015-0383-3.

[17] Martí, Marq. «¿Qué es el Web scraping? Introducción y herramientas». Sitelab España, de abril de de 2016, https://sitelabs.es/web-scraping-introduccion-y-herramientas/

[18] Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation. https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[19] Requests: HTTP for HumansTM — Requests 2.26.0 documentation. https://docs.python-requests.org/en/master/

[20] Raspberry Pi. «Teach, Learn, and Make with Raspberry Pi». Raspberry Pi, https://www.raspberrypi.org/