ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DE COMPUTADORES

# IMPLEMENTACIÓN E INTEGRACIÓN DE UN SERVICIO DE AGENDA EN UN ROBOT DE ASISTENCIA DE PERSONAS MAYORES

# DESIGN AND INTEGRATION OF CALENDAR SERVICES FOR ASSISTANT ROBOTS

Realizado por
**Francisco Aragón Royón**

Tutorizado por
**Cipriano Galindo Andrades**

Departamento
**Ingeniería de Sistemas y Automática**

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO 2015

Fecha defensa:
El Secretario del Tribunal

## Resumen:

En los últimos años, el progresivo aumento del porcentaje de personas mayores en la población y la previsión de que siga aumentando en un futuro cercano está dando un papel importante a la tecnología, que se está introduciendo en las labores de asistencia doméstica de personas mayores para resolver esta problemática. Un ejemplo de la introducción de la tecnología es la *telepresencia*, que permite a una persona estar presente de forma virtual e interactuar con un lugar remoto mediante el control de un robot.

El objetivo de este trabajo de fin de grado es el de incorporar a un robot de telepresencia una función de calendario y la capacidad de interacción con el usuario mediante comandos de voz, permitiendo que el robot disponga de determinado nivel de autonomía para realizar estas tareas.

Con estas funcionalidades, el robot podrá recordar determinados eventos o citas al usuario mediante la reproducción de voz, lo cual podría requerir el desplazamiento semi-autonónomo del robot hasta el lugar donde se encuentre el usuario. De igual forma, éste podrá guardar eventos en el calendario o preguntar al robot por los eventos venideros gracias al reconocimiento de voz.

En este documento se describen las herramientas usadas, los desarrollos que se han implementado, así como un ejemplo ilustrativo de aplicación utilizando el robot de telepresencia Giraff.

## Palabras claves:

Asistencia a personas mayores, interacción, robot, telepresencia, calendario, citas, eventos, reproducir voz, reconocer voz.

## Abstract:

In the recent years, the progressive increase of elderly people in the population and the prediction of an increasing in the near future are giving an important role to the technology. Technology is being introduced to help in the assistance of the elderly people in a number of domestic tasks.

An example of the introduction of technology is the *robotic telepresence*, which allows a person to be virtually present in a remote location. This person can also interact with this remote location through the control of the robot.

The objective of this final degree project is to incorporate a calendar functionality to a telepresence robot, as well as the needed skills for interacting with users by means of voice commands, allowing a certain level of autonomy to the robot.

With these functionalities, the robot will be able to remind certain events or appointments to the user by means of voice speech, and the user will be able to add new events in the calendar or ask to the robot about coming events thanks to voice recognition system. It is important to remark that this tasks may involve the semi-autonomous navigation of the robot to the user's location.

In this document, the used tools, and the developments carried out are described, as well as an illustrative example of applicability to a particular telepresence robot, called Giraff.

## Keywords:

Elderly people assistance, interaction, robot, telepresence, calendar, appointments, events, voice reproduction, voice recognition.

# Índice

Contenido

# Capítulo 1

## Introducción

Desde hace unas décadas la mejora de las condiciones generales de vida está dando lugar a un aumento del porcentaje de personas mayores, provocando un notable envejecimiento de la población, siendo éste un problema que se extiende a toda Europa. En el caso particular de España, para mediados de siglo se estima que las personas mayores de 65 años superen más del 30% de la población total. Esto no hace más que reafirmar el envejecimiento de nuestra sociedad, hecho éste que se va a acusar cada vez más, por lo que esta problemática se extenderá también a las próximas generaciones.

Se espera que debido a este aumento del porcentaje de población anciana, se produzca también un aumento sustancial de los gastos relacionados con la atención sanitaria y cuidados sociales, pudiendo ser requerida una mano de obra tan numerosa para los servicios necesarios demandados por este colectivo que no resulte asequible.

La tecnología podría ofrecer soluciones útiles ante esta situación y por esto es necesario su incorporación a labores de asistencia doméstica, ya sea para poder facilitar tareas cotidianas, controlar el estado de salud, incluso también interactuar y poder mitigar la soledad de las personas mayores ofreciendo una asistencia personalizada según las necesidades de cada individuo.

La robótica es un campo de **la innovación tecnológica** que se encuentra en constante crecimiento, una expansión que veremos materializada en nuestra vida cotidiana en los próximos años pues las empresas de robótica ya tienen la vista puesta en el cuidado de los ancianos como un enorme mercado potencial. Más concretamente, la telepresencia robótica está recibiendo estos últimos años una gran atención, especialmente cuando se aplica a la interacción con las personas mayores. La robótica telepresencial hace referencia a un conjunto de tecnologías que permiten que una persona esté presente de forma virtual e interactúe en un lugar remoto determinado por medio de un robot al cual controla.

Respondiendo a esta demanda, numerosos grupos de investigación se están volcando en el estudio y desarrollo de sistemas robóticos para mejorar la calidad de vida de las personas mayores. Como ejemplo, cabe mencionar el proyecto GIRAFF+, financiado por la UE cuya misión es implantar y evaluar el uso de robots de telepresencia equipados y apoyados con una serie de sensores y dispositivos en domicilios particulares. El robot propuesto, denominado Jirafa, se desplaza por la vivienda y es capaz de interactuar mediante videoconferencia con familiares o

personal sanitario, permitiendo a personas de edad avanzada llevar una vida más segura e independiente dentro de sus limitaciones.

Pero éste es aún un campo nuevo que necesita continuar con su desarrollo, habiendo cabida para una gran cantidad de aplicaciones aún por desarrollar que permitan integrar todo tipo de funcionalidades a estos robots telepresenciales que cuidan personas mayores.

## 1.1. Objetivos

El objetivo de este trabajo de fin de grado es el de incorporar al robot Jirafa una funcionalidad de calendario y de dotarlo de una capacidad de interacción de forma sencilla, mediante comandos de voz. De esta forma el robot se convierte en un compañero capaz de recordarle a la persona mayor sus citas y eventos más importantes de forma autónoma sin que sea necesaria la telepresencia. El mecanismo de interacción será directo y mecánico mediante el uso de distintas frases preestablecidas que el robot reconocerá y a partir de las cuales actuará en consecuencia. El calendario utilizará servicios ampliamente extendidos permitiendo ser compartido por otros usuarios, por ejemplo familiares y personal sanitario quienes pueden acceder y añadir eventos relevantes, como el recordatorio de la toma de medicamentos.

El robot Jirafa (ver figura 1.1) es un robot de telepresencia, que consta de una plataforma motorizada con ruedas y con un sistema de videoconferencia, pantalla, micrófono, altavoz y cámara, además de un ordenador a bordo, sensores, etc. Este sistema permite a una persona controlar al robot remotamente, percibir el entorno (audio y video) y establecer una interacción cercana con el usuario.

Este trabajo de fin de grado contribuirá al robot Jirafa con las capacidades para realizar, sin la posible necesidad de telepresencia, una serie de tareas de servicio con respecto a una agenda como pueden ser la introducción mediante voz de eventos en el calendario o el recordatorio por parte del robot de determinados eventos del calendario al usuario.

La realización de este conjunto de tareas estará complementada por algoritmos ya desarrollados previamente y presentes en el robot que le permiten por ejemplo moverse autónomamente evitando los posibles obstáculos. Todas las capacidades del robot Jirafa se verán potenciadas con la función de calendario mediante voz ya que permitirá una interacción directa y natural con el usuario, pudiendo llegar a suplir la necesidad de telepresencia en determinados casos como por ejemplo el recordatorio de eventos. Cabe mencionar que los nuevos servicios implementados e integrados en este trabajo fin de grado servirán de base para futuros servicios que

podrán beneficiarse de una interacción (reproducción/reconocimiento) basada en voz con el robot.

Para la consecución de este objetivo nos apoyaremos en una serie de APIs [1] para el reconocimiento y reproducción de voz, así como de gestión de un calendario, que usaremos como capas de abstracción y que modificaremos adaptándolas según las necesidades de nuestro trabajo, concretamente se utilizarán los servicios que ofrece Google tanto de calendario como de reproducción de voz y los servicios de reconocimiento de voz que vienen integrados en HTML. También hay que mencionar la necesidad del uso de un protocolo de comunicación mediante paso de mensajes y de una serie de lenguajes de programación y herramientas:

- *Lenguajes de programación*: JavaScript, HTML, JSON.
- *APIs*: Chrome TTS, Google Calendar, Web Speech.
- *Protocolo de comunicación*: MQTT.
- *Herramientas*: Navegador web, servidor web, servidor MQTT, robot Jirafa.

Figura 1.1. Robot Jirafa.

## 1.2. Estructura de la memoria

La memoria se divide en siete capítulos principales, cada uno los cuales aporta descripción detallada de los puntos más importantes del desarrollo de este trabajo de fin de grado.

- *Capítulo 1:* **Introducción**

Este capítulo contiene la estructura de la memoria, presenta una descripción de la problemática por la cual se requiere la necesidad del desarrollo de este trabajo y los objetivos que se pretenden alcanzar.

[1] La interfaz de programación de aplicaciones, abreviada como API (del inglés: *Application Programming Interface*), es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

- *Capítulo 2:* **Tecnologías usadas**

Introduce los conceptos tecnológicos usados para la realización del trabajo. Se describen lenguajes de programación, protocolos de comunicación, APIs, así como las diversas herramientas tecnológicas empleadas.

- *Capítulo 3:* **Integración de los servicios de voz**

Se detallan las fases de desarrollo de los servicios de voz, la reproducción y el reconocimiento, así como también la integración de un protocolo de comunicación que hace posible el intercambio de información entre los servicios.

- *Capítulo 4:* **Integración de los servicios de calendario**

En este capítulo se detallan las fases de desarrollo del servicio de calendario, y como este servicio es el nexo de unión en torno al que giran los servicios de voz.

- *Capítulo 5:* **Integración de las aplicaciones en la arquitectura del robot**

En este capítulo se describe la forma en que las aplicaciones desarrolladas son integradas en la arquitectura del robot.

- *Capítulo 6:* **Resultado final y uso**

Este capítulo presenta las principales funcionalidades de la aplicación resultante mediante los distintos casos de uso de la aplicación.

- *Capítulo 7:* **Conclusiones**

Se presentan las conclusiones obtenidas en base a los estudios realizados y los resultados obtenidos. Posteriormente, se presentan unas posibles líneas de continuación de este trabajo que se pueden seguir en un futuro.

# Capítulo 2

# Tecnologías usadas

En este capítulo se presentan las tecnologías que se han empleado en el desarrollo del trabajo. Se realiza una breve introducción a los distintos lenguajes de programación usados y se especifica el motivo por el que han sido elegidos. Se describe el protocolo de paso de mensajes entre los diferentes módulos software, así como las APIs elegidas para dar la funcionalidad requerida. Al final del capítulo se presentan las herramientas tecnológicas sobre las cuales se ejecutará la aplicación resultante.

## 2.1. Lenguajes de programación

En esta primera sección se describen los lenguajes de programación utilizados en el desarrollo de las aplicaciones, el motivo de su elección depende de las restricciones que presentan algunas de las APIs que se usan:

- El acceso a la API de Google Chrome TTS solo es posible mediante la creación de una extensión/aplicación para el navegador Chrome, que se define en código JSON y se programa en HTML y JavaScript.

- El acceso a Web Speech API solo es posible desde HTML ya que es una librería JavaScript incluida en éste.

Por el contrario, no existen restricciones estrictas referentes a lenguajes de programación para acceder a la API de Google Calendar ni para implementar el protocolo de comunicación MQTT, ya que se pueden hacer desde diversos lenguajes de programación.

Las restricciones anteriormente nombradas han hecho necesario el uso tanto de JavaScript, como HTML y JSON para el desarrollo de las aplicaciones de servicio de voz. Por consiguiente, se ha usado también JavaScript como lenguaje de implementación de MQTT. Por último y para no crear disparidad, se ha usado JavaScript y HTML para implementar los servicios de interacción con el calendario.

### 2.1.1.JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para el desarrollo y diseño de sitios web dinámicos. Nació en 1995 diseñado por Nestcape con el objetivo de permitir a los diseñadores de sitios web crear páginas que permitieran interactuar con los usuarios, requiriendo la creación de webs de mayor complejidad que las meramente estáticas que se diseñaban hasta entonces con HTML.

JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los códigos escritos en JavaScript se pueden ejecutar directamente en cualquier navegador sin necesidad de procesos intermedios ni de instalar ningún otro programa para visualizarlos ya que es el navegador el que interpreta el código. Existen diversas formas de incluir código JavaScript en las páginas web, ya que la integración de JavaScript en HTML es muy flexible. Ya sea por ejemplo incluyendo el código JavaScript en el mismo código HTML encerrándolo entre etiquetas, o definiendo el código JavaScript en un archivo externo de tipo JavaScript que es enlazado por HTML.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Ambos son diferentes y tienen sus características singulares.

### 2.1.2.HTML

HTML (HyperText Markup Language) es el lenguaje de programación que hace posible representar información en Internet. Fue creado como tal por Tim Berners-Lee a principios de los años 90, a partir del lenguaje de etiquetas SGML**,** como una forma de facilitar a científicos de diferentes universidades el intercambio de los documentos de investigación de cada uno de ellos. Fue tal el éxito que obtuvo el proyecto que se sentaron las bases de la web tal y como la conocemos hoy día.

Este lenguaje, estandarizado a cargo de W3C [2], sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web, como texto, imágenes, videos, entre otros. El lenguaje HTML basa su filosofía de desarrollo en la referenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, etc.), éste no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la

[2] World Wide Web Consortium, abreviado como W3C es una organización dedicada a la estandarización de muchas de las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación.

página web contiene sólo texto mientras que recae en el navegador web, que interpreta el código, la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma por cualquier navegador web actualizado.

### 2.1.3.JSON

JSON (JavaScript Object Notation) es un formato de texto para la serialización y el intercambio de datos estructurados. Nació como una alternativa a XML, pues como sistema de almacenaje de datos, es perfecto para transportarlos de una página a otra, incluso de un sitio web a otro. Está basado en un subconjunto del lenguaje de programación JavaScript.

Básicamente JSON describe los datos con una sintaxis dedicada que se usa para identificarlos y gestionarlos de forma organizada y de fácil acceso. JSON puede representar cuatro tipos primitivos: *cadenas*, *números*, *booleanos*, y *valores nulos* y dos tipos estructurados: *objetos* (colecciones de pares nombre/valor) y *arrays* (listas ordenadas de valores). Éstas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Una de las mayores ventajas que tiene el uso de JSON es precisamente esto, que puede ser interpretado por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías.

## 2.2. Protocolos de comunicación

En este punto se describe el protocolo de comunicación MQTT que se ha utilizado para comunicar las diferentes aplicaciones implementadas en este trabajo de fin de grado. Tras una breve introducción inicial, se presentan los principios en los que se basa y el modo de funcionamiento para finalmente reseñar algunos detalles importantes. La elección de este protocolo en concreto y no otro, atiende a la razón de que es el protocolo de comunicación utilizado en la arquitectura de control del robot dónde se van a ejecutar las aplicaciones, el robot Jirafa.

MQTT (Message Queue Telemetry Transport) es un protocolo de conectividad abierto y enfocado a M2M (machine to machine) y al IOT (Internet de las cosas). Fue desarrollado por Andy Stanford-Clark de IBM y Arlen Nipper de Arcom, en 1999. La

especificación del protocolo ha sido publicada con licencia libre y desde principios de 2013 se encuentra en proceso de normalización por OASIS [3].

Los principios de su diseño son minimizar el requerimiento de ancho de banda de la red y los requisitos de recursos del dispositivo, mientras que también intenta asegurar la fiabilidad y cierto grado de garantía de entrega. Por tanto, podemos decir que ha sido diseñado para ser un protocolo de mensajería extremadamente ligero y simple, que permite el intercambio de datos en forma de mensajes entre dispositivos con conexiones remotas a través de redes con restricciones de ancho de banda, alta latencia o poco confiables. Una característica muy importante es que al ser un protocolo tan ligero existen clientes y servidores MQTT en diversos lenguajes, desde Java, C, Arduino, Javascript, etc.

MQTT se basa en publicar/suscribir mensajes en un modelo cliente/servidor. Consta de los siguientes elementos:

- Mensaje: son los datos transportados a través de la red por MQTT, los mensajes deben de tener asociados un *topic* y una QoS.

- *Topic*: es la etiqueta adjunta a un mensaje, se compara con las suscripciones conocidas por el servidor.

- Cliente: es un programa o dispositivo que usa MQTT, que establece una conexión de red y puede suscribirse a un *topic* (para especificar el tipo de mensajes que está interesado recibir), publicar mensajes (los cuales pueden interesar a otro cliente), cancelar su suscripción o desconectarse del servidor.

- Servidor: es un programa o dispositivo que actúa como intermediario entre clientes y puede aceptar conexiones de red desde los clientes, acepta mensajes de publicación de los clientes, procesa solicitudes de suscripción y de cancelación de suscripción, así como reenviar a clientes mensajes publicados en un *topic* al cuál están suscritos.

El funcionamiento de MQTT consiste en la conexión de clientes a un servidor, también llamado *broker*, al cual se pueden suscribir con un cierto *topic.* Este *tópic* representa el tema en el que está interesado y por tanto desea recibir mensajes. Los clientes, suscritos a un *topic* o no, puede enviar mensajes al servidor siempre con un *topic* determinado. El servidor, que se encarga de administrar los mensajes, filtra mediante el *topic* los clientes interesados en recibir ese mensaje y envía una copia a

[3] OASIS, acrónimo de *Organization for the Advancement of Structured Information Standards*, es un consorcio internacional sin fines de lucro que orienta el desarrollo, la convergencia y la adopción de los estándares de comercio electrónico y servicios web.

todos los clientes cuyo *topic* de suscripción coincida con el *topic* del mensaje entrante. (Ver figura 2.1).

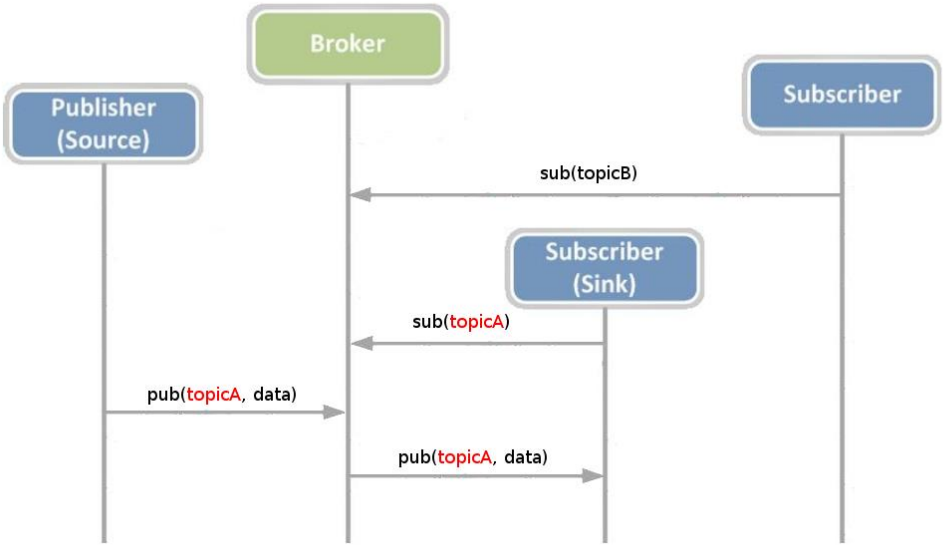


Figura 2.1. Ejemplo de una comunicación por MQTT. Dos procesos se suscriben a un *broker* con distintos *topic*, otro proceso publica en el *broker* un mensaje asociado a un *topic* en concreto, el proceso suscrito a ese *topic* recibe el mensaje.

El protocolo MQTT trabaja sobre TCP/IP, e incluye tres tipos de QoS (quality of service):

- "At most once" o QoS = 0, los mensajes son entregados según el mejor esfuerzo, por tanto pueden darse situaciones de pérdidas de mensajes.

- "At least once" o QoS = 1, donde se asegura que los mensajes llegaran al destino, aunque posiblemente duplicados.

- 'Exactly once' o QoS = 3, se asegura que los mensajes llegaran al destino exactamente una vez.

Hay puertos estándar para el uso de MQTT, el puerto TCP/IP 1883 está reservado por la IANA [4] para el uso de MQTT, y también el puerto 8883 para el uso de MQTT sobre SSL. Como vemos, también se pueden pasar nombres de usuarios y contraseñas por paquetes MQTT así como cifrar los datos que se reciben y se envían, aunque cabe destacar que SSL no es precisamente el más ligero de los protocolos y que puede añadir una sobrecarga significativa a la red, pudiendo dar lugar a que MQTT pierda su ligereza.

[4] Internet Assigned Numbers Authority (cuyo acrónimo es IANA) es la entidad que supervisa la asignación global de direcciones IP, sistemas autónomos, servidores raíz de nombres de dominio DNS y otros recursos relativos a los protocolos de Internet.

Por último mencionar que MQTT está muy extendido y se ha usado por ejemplo en comunicaciones entre sensores, conexiones GPRS, aplicaciones móviles, etc. Concretando aún más, se usa en conocidas aplicaciones como Facebook Messenger para iPhone y Android.

## 2.3. API's

Esta sección trata sobre las API's y librerías que se han utilizado en el desarrollo de las distintas aplicaciones de este proyecto. Se presenta una breve introducción y se describe la funcionalidad y las diversas opciones que ofrecen. El motivo de la elección de estas API's en concreto es que ofrecen las funcionalidades requeridas para llevar a cabo los distintos objetivos propuestos en la realización de este trabajo.

### 2.3.1. Text To Speech (Chrome TTS)

El navegador Chrome provee soporte nativo de reproducción de voz en Windows (usando SAPI 5), Mac OS X y Chrome OS, usando las capacidades de síntesis de voz que ofrece el propio sistema operativo. Google es el propietario de esta aplicación que ofrece su navegador Chrome, que puede ser usada por los usuarios tantas veces como deseen mediante la aplicación Chrome.tts, que ejecuta la sintetización de texto a voz (text-to-speech TTS). Es accesible desde la versión número 14 del navegador web y para poder acceder a ella, se debe crear una extensión o una aplicación para dicho navegador, que será definida mediante un archivo *Manifest* en el cuál se otorgan los permisos de uso de TTS. (Ver figura 2.2).

El funcionamiento es simple, basta con enviar una frase a la aplicación para que esta la reproduzca. Por defecto, Chrome usa la voz que considera más apropiada para la reproducción, pero se pueden establecer diferentes opciones de reproducción como por ejemplo la velocidad, el tono, el lenguaje, el género de la voz, etc. También permite la pausa o detención de la reproducción de voz en cualquier momento, así como el monitoreo de los distintos eventos que se pueden producir durante la reproducción. Ofrece también la posibilidad de desechar o colocar en la cola de reproducción una nueva frase a reproducir si en el momento de su entrada a la aplicación ya se está ejecutando una reproducción. Además, en cualquier plataforma, el usuario puede también instalar extensiones que registren un motor de voz alternativo.

```
{
  "manifest_version": 2,
  "name": "Robot Jirafa TTS",
  "version": "1.0",
  "description": "Extension chrome para hablar",
  "icons": {
     "16": "icon_16.png",
     "48": "icon_48.png",
     "128": "icon_128.png"
  },
  "app": {
    "background": {
      "scripts": ["ventana.js"]
    }
   },
  "permissions": ["tts"]
}
```

Figura 2.2. Ejemplo de un archivo *Manifest*. En concreto éste se corresponde con el archivo *Manifest* de la aplicación de reproducción de voz

### 2.3.2.Calendar

La API de Google Calendar permite desarrollar aplicaciones cliente que interactúen con un calendario de google. Actualmente la versión de esta API es la tercera. Esta API cliente es propiedad de Google, que limita su uso gratuito a un millón de llamadas por día. La API está disponible mediante librerías para diversos lenguajes de programación como por ejemplo Android, .NET, Java, JavaScript, PHP, Python, etc. Aunque no todas estas librerías están en fase estable, ya que por ejemplo en JavaScript, la librería está en fase beta.

Google Calendar API permite a un programa desarrollar la mayoría de operaciones disponibles en la interfaz web de un calendario de Google, como por ejemplo crear, editar o borrar eventos o calendarios. Con esta API se pueden acceder, buscar o modificar calendarios públicos y también calendarios privados, estos últimos mediante el previo establecimiento de una sesión de autenticación. Las llamadas a Google Calendar API se pueden hacer de dos formas, mediante:

- REST, la API define una URI [5] relativa a cada uno de los métodos y en la misma se especifican directamente los argumentos de cada llamada.

[5] URI son las siglas en inglés de Uniform Resource Identifier (en español identificador uniforme de recursos), que sirve para identificar recursos en Internet. Dicho identificador tiene un formato estándar y su propósito es permitir interacción con recursos disponibles en Internet -o en alguna red de cómputo-, como por ejemplo páginas, servicios, imágenes, vídeos, etc.

- RPC, permite ejecutar los distintos métodos ofrecidos por la API estableciendo en el código la llamada y los parámetros determinados con los que queremos que se ejecute.

Un detalle importante es que el uso de esta API cliente, al menos para la librería en JavaScript, no puede ser local (protocolo file://), ya que las llamadas locales no son soportadas, por lo que la aplicación que realice las llamadas debe estar alojada en un servidor.

### 2.3.3.Web Speech

Web Speech API es una librería de JavaScript, introducida a finales de 2012, que permite reconocimiento de voz y conversión de voz a texto en los navegadores web. Soporta un gran número de lenguas y el reconocimiento suele ser muy efectivo, características que típicamente no están disponibles cuándo se usa el reconocimiento de voz estándar. No está estandarizada para todos los navegadores luego solo se puede usar en algunas versiones de determinados navegadores web. Como podemos ver más abajo (ver figura 2.3), las versiones de los navegadores en verde son las soportadas, mientras las versiones en rojo las no soportadas.

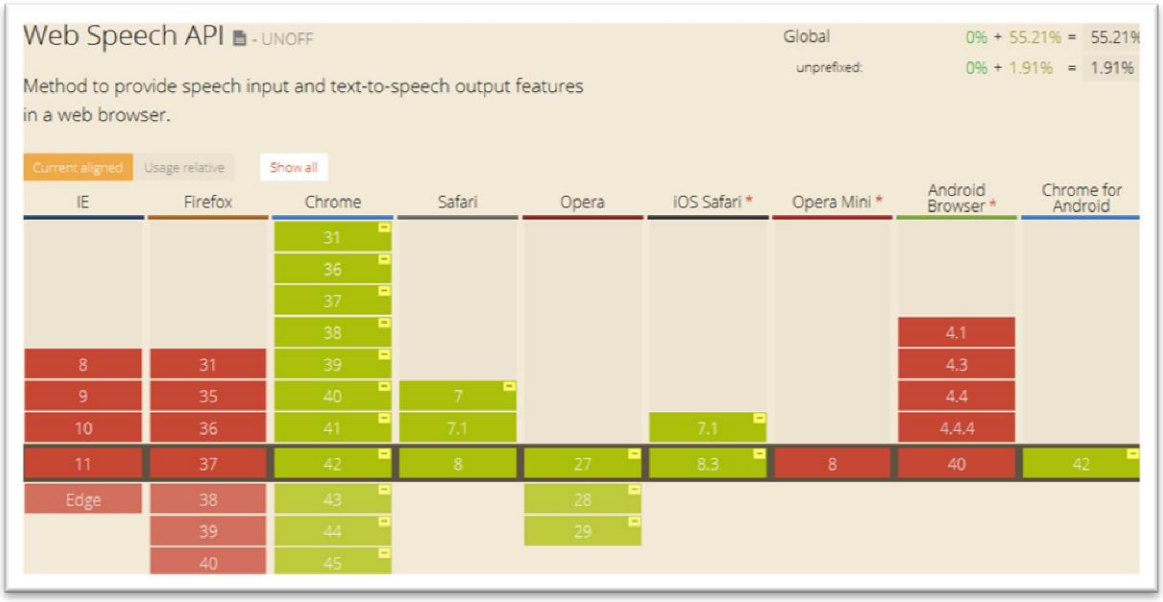


Figura 2.3. Versiones de los navegadores que soportan Web Speech API, en verde los soportados, en rojo los no soportados.

Esta API tiene conciencia de la privacidad de los usuarios, ya que antes de permitir al navegador el acceso a la voz mediante el micrófono, el usuario debe dar permiso explícito de uso del micrófono. Por otro lado, si la web que implementa Web

Speech API usa el protocolo HTTPS [6], el navegador solo pide permiso de uso del micrófono una sola vez, en otro caso, lo hace cada vez que se comience un nuevo proceso de reconocimiento.

La API está diseñada para permitir dos tipos de reconocimiento: entrada breve de voz o entrada de voz continua. En el primer caso, el reconocimiento termina tan pronto como el usuario deje de hablar, mientras que en el segundo caso el reconocimiento de voz es constante. También permite la detención del reconocimiento en cualquier momento. Con la ayuda de JavaScript, podemos tratar el resultado obtenido. Este resultado se presenta a la web, al código, como una lista de posibles reconocimientos o posibles hipótesis, junto con información relevante sobre cada una. En caso de haberse producido un error durante el reconocimiento, también devolverá un informe de la causa del error. Otra característica interesante de Web Speech API, es que permite especificar objetos gramaticales, como conjunto de reglas para definir un idioma. Pero esto es algo que se escapa al objetivo de este trabajo de fin de grado.

Por último entre otras ventajas de esta API podemos destacar la posibilidad de mostrar los resultados del reconocimiento interinos, es decir, en vivo sin necesidad de que haya terminado de hablar. Y entre las desventajas la limitación de tiempo de cada sesión de reconocimiento, debido a que el reconocimiento de la API no está desarrollado localmente en el navegador sino remotamente.

## 2.4. Herramientas

En esta sección se describen las herramientas tecnológicas sobre las que se ejecutarán las aplicaciones finales. Se presenta de manera breve una introducción a los navegadores web, Google Calendar, servidores MQTT y al robot Jirafa, así como también se describe el motivo de su elección. Todas estas herramientas combinadas con los demás pilares fundamentales, anteriormente descritos, han hecho posible el desarrollo de las aplicaciones finales de este trabajo.

### 2.4.1.Navegador web

Un Explorador Web o Navegador Web es un software, por lo general gratuito, que como funcionalidad básica nos permite visualizar páginas web a través de Internet además de acceder a otros recursos de información alojados también en servidores web, como pueden ser videos, imágenes, audios, etc.

[6] Hypertext Transfer Protocol Secure (en español: *Protocolo seguro de transferencia de hipertexto*), más conocido por sus siglas HTTPS, es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de Hipertexto, es decir, es la versión segura de HTTP.

El Navegador se comunica con los servidores web a través del protocolo HTTP y le pide el archivo solicitado en código HTML, después lo interpreta y muestra en pantalla para el usuario.

Los más populares son Mozilla Firefox, Internet Explorer, Google Chrome, Safari y Opera. (Ver figura 2.4).

Figura 2.4. Logotipos de los navegadores anteriormente nombrados.

El uso de un navegador web es una restricción imperativa para el desarrollo del trabajo ya que varios recursos como Web Speech API y TTS solo son accesibles desde un navegador. Más en concreto la aplicación TTS solo es accesible con la ejecución de una extensión/aplicación del navegador Google Chrome.

### 2.4.2.Google Calendar

Google Calendar es una agenda y calendario electrónico desarrollado por Google (ver figura 2.5). Está disponible desde el 13 de abril de 2006. Para su uso, los usuarios no están obligados a tener una cuenta de correo de Google, Gmail, pero si deben disponer de un Google Account.

Las dos principales funcionalidades que ofrece Google Calendar son la de poder crear calendarios y eventos, y compartirlos independientemente. Tanto los calendarios como los eventos, pueden ser públicos o privados. Si se eligen públicos, cualquier persona podrá verlos. Si se escoge que el calendario o el evento sea privado, solamente podrán acceder a él las personas con las que se comparta el calendario o el evento.

El motivo del uso de Google Calendar es que es uno de los calendarios electrónicos más extendidos, usados y completos hoy día, ya que cuenta con una gran variedad de funcionalidades. Otra razón indispensable es que provee una API que permite desarrollar aplicaciones cliente que interactúen con un calendario. La cuenta y contraseña de Google Calendar usada para este trabajo de fin de grado, se presenta en un documento adjunto a la memoria.

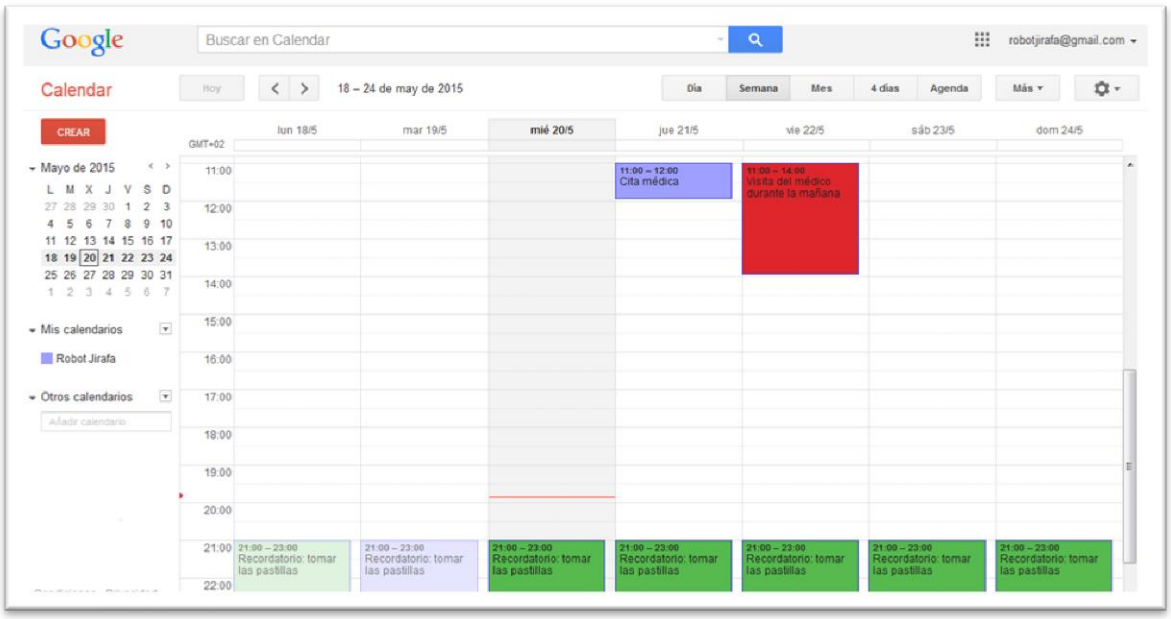


Figura 2.5. Interfaz web de Google Calendar con una serie de eventos almacenados.

### 2.4.3.Servidor MQTT

Un servidor MQTT, o *broker*, es una parte imprescindible de una conexión MQTT. Su funcionamiento ha sido descrito con anterioridad en la sección 2.2.

Entre los servidores MQTT que se pueden emplear, cabe destacar destacar los implementados por HiveMQ [17] o Mosquitto [18]. Para las pruebas locales se ha utilizado el servidor MQTT implementado por HiveMQ, ya que además de ser uno de los mejores servidores que existen, al ofrecer encriptación, portabilidad, etc., asegura compatibilidad con códigos clientes escritos en diversos lenguajes, como por ejemplo JavaScript. Mientras que para el caso real en el robot Jirafa se ha usado un servidor MQTT alojado en el laboratorio del cual depende del robot. Este laboratorio está ubicado en la Universidad de Málaga, y forma parte del Departamento de Ingeniería de Sistemas y Automática.

### 2.4.4.Servidor web

Un servidor web o servidor HTTP es un programa informático que sirve contenido estático a un navegador, cargando un archivo y sirviéndolo a través de la red al navegador de un usuario.

Esto es posible ya que procesa una aplicación del lado del servidor, a la cual se realizan conexiones cliente, una vez establecida la conexión se genera y cede una respuesta en cualquier lenguaje al cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos

datos suele utilizarse algún protocolo, siendo generalmente usado el protocolo HTTP para estas comunicaciones. El término también se emplea para referirse al ordenador que ejecuta el programa.

El motivo del uso de un servidor web es que será necesario para hacer las llamadas a la API de Google Calendar, las cuales no son soportadas de forma local. Por lo que se necesita que la aplicación que las haga esté alojada en un servidor. Para las pruebas locales se ha utilizado el servidor web Apache [19] contenido en XAMPP [7], mientras que para el caso real en el robot Jirafa se ha usado un servidor alojado en el laboratorio anteriormente nombrado, del cual depende del robot.

### 2.4.5.Robot Jirafa

El robot Jirafa (ver figura 2.6) es un robot de telepresencia desarrollado por Giraff AB Company. Consta de una plataforma motorizada con ruedas, un sistema de videoconferencia, pantalla, micrófono, altavoz y cámara, además de un conjunto de sensores, baterías, las cuales duran aproximadamente dos horas, y un ordenador interno del cual depende.

Está controlado remotamente por una persona, que puede percibir el entorno (audio y video) y establecer una interacción con el usuario. Esta interacción se lleva a cabo de una forma muy cercana gracias a la cámara y a la pantalla del robot que dan al visitante remoto la sensación de estar en el mismo lugar.

El robot contiene una aplicación software, *Giraff Pilot*, que ayuda a manejarlo de forma remota fácilmente. Este software es un interfaz gráfico para conducir el robot y controlar las diversas opciones que ofrece. También dispone de una arquitectura de control, denominada OpenMora [1], que permite la ejecución de tareas de alto nivel con navegación, evitación de obstáculos, interacción con el usuario, etc.

Todas las opciones que pueden necesitar las personas mayores para manejar el robot pueden ser fácilmente alcanzadas gracias a un controlador remoto. Por lo que una de sus grandes ventajas es que su manejo es algo fácil e intuitivo, no necesitando ningún conocimiento tecnológico para ello.

La razón del uso del robot Jirafa es que es la herramienta a la cual queremos dotar de funcionalidades de voz y calendario, siendo éste el motivo de realización de este trabajo de fin de grado.

[7] XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl.

Figura 2.6. Imagen del robot Jirafa con la interfaz gráfica.

# Capítulo 3

# Implementación de los servicios de voz

En este capítulo se presenta una descripción a nivel de **implementación** de las aplicaciones de voz, la aplicación de reproducción y la aplicación de reconocimiento. Así como se describe también la implantación en estas aplicaciones de un protocolo de paso de mensajes. Toda esta descripción está apoyada en el estudio previo de la documentación y manuales que ofrecen cada una de las herramientas usadas. Se desglosan los objetivos de cada aplicación y los medios disponibles para alcanzarlos, se describen las fases de desarrollo de la aplicación, los problemas surgidos durante ésta y el resultado final a nivel de código implementado de la aplicación. El código completo de estas aplicaciones no se describe en la memoria, pero se adjunta junto a esta.

## 3.1. Desarrollo de la aplicación de reproducción de voz

En este apartado se describe el proceso de realización de la aplicación de reproducción de voz.

**Objetivos y medios para alcanzarlos**

Esta aplicación tiene como objetivo ser una aplicación ejecutable en el navegador Chrome y ser también una aplicación cliente dentro de un modelo cliente/servidor MQTT, para poder recibir mensajes que sean enviados por otras aplicaciones cliente. Una vez se reciba un mensaje, el otro objetivo principal será administrar y adecuar ese mensaje para reproducir por voz su contenido. La integración de un protocolo de paso de mensajes, MQTT, que es igual para todas las aplicaciones de este trabajo de fin de grado, soluciona la recepción de mensajes en esta aplicación. Dicha integración es descrita en el punto 3.3 de esta memoria. Por otro lado, para la reproducción de voz se ha usado la API de Google Chrome, TTS (ver sección 2.3.1), que provee el siguiente método para dicho fin, y que es el que hemos usado:

```
chrome.tts.speak(utterance, options, function(){…});
```

Cuyos parámetros se definen como:

- *Utterance:* Es el texto que se quiere reproducir.

- *Options:* Establece las diversas opciones de reproducción. Entre ellas se puede establecer el idioma de reproducción, el género de la voz, la velocidad, los eventos que se quieren muestrear durante la ejecución o la acción a realizar una vez estos eventos sean capturados.

- *Function:* Es una función que se ejecuta antes de que comience el motor de voz. Su propósito es alertar de errores de sintaxis en la llamada a la función *speak*, retornando dicho error en caso de existir.

**Desarrollo**

El proceso de desarrollo de esta aplicación consta de una serie de fases que se detallan a continuación:

- **Periodo de documentación**: Es la fase previa al desarrollo, en la cual se adquieren los conocimientos y recursos necesarios para llevar a cabo el desarrollo correcto de la aplicación. Se adquieren conocimientos sobre la creación de aplicaciones para Chrome ([2] [3] [4] [5] [6]) y el uso de la aplicación Chrome TTS [7].

- **Desarrollo de los requisitos para ser una aplicación de Chrome:** En esta fase se desarrolla el requisito para ser una aplicación ejecutable en el navegador Chrome, la creación de un archivo *Manifest*.

- **Desarrollo del servicio de voz:** Es la fase que otorga la característica principal requerida de la aplicación. Se implanta y adapta la aplicación TTS de Chrome en el código para la reproducción de voz.

- **Implantación del protocolo MQTT**: Una vez se tenga la funcionalidad de voz, en esta fase se implanta el protocolo de paso de mensajes MQTT, para que la aplicación pueda recibir estos mensajes y posteriormente reproducirlos. Este será el punto de entrada inicial de la aplicación, y su correcto funcionamiento dependerá del estado de este protocolo.

- **Periodo de refinamiento**: Es la última fase, en ella se mejoran y optimizan todo lo posible las fases anteriores, como por ejemplo dotando a la aplicación de reconexión en caso de pérdida del servidor, etc. y se dota a la aplicación de un interfaz gráfico que hace más amigable la presentación al usuario.

**Problemas surgidos durante el desarrollo**

En este apartado se muestran los principales problemas acaecidos durante las distintas fases del desarrollo de esta aplicación y cómo se han podido solucionar.

- **Necesidad de una aplicación Chrome**: Inicialmente no estaba previsto que la aplicación de reproducción de voz fuera también una aplicación ejecutable en Chrome. Sin embargo, solo es posible el uso de la aplicación Chrome TTS desde una aplicación del navegador Chrome, definiendo permisos específicos para el uso de dicha aplicación. Luego la aplicación desarrollada se ha tenido que adaptar a dicha necesidad mediante la definición de un archivo *Manifest*.

- **MQTT no acepta el conjunto de símbolos latinos**: MQTT usa el formato de codificación UTF-8 [8], que es distinto del formato de codificación latino, luego determinados signos de puntuación latinos como exclamaciones o interrogaciones de apertura, signos de acentuación, o letras como "ñ", no son soportados como parte de un mensaje en una comunicación MQTT. El usuario de esta aplicación de reconocimiento va a usarlos, por lo que se ha desarrollado un codificador/decodificador que convierte las cadenas de caracteres en cadenas numéricas y viceversa para permitir así su correcta transmisión por MQTT. Más concretamente, la codificación consiste en convertir cada carácter de texto del mensaje a la codificación numérica del valor binario del mismo, y encerrarlo entre símbolos "|" formando una cadena. La decodificación es el proceso contrario.

- **Chrome TTS no reproduce textos con determinadas longitudes**: Si no se usa el lenguaje nativo de Chrome TTS, la aplicación contiene errores que la inutilizan para cadenas superiores a aproximadamente 200 caracteres, para solucionar esta falla, se ha desarrollado una función que recursivamente "corta" en signos de puntuación las cadenas de caracteres a reproducir que sean mayores de la longitud límite en secuencias de cadenas más pequeñas sí reproducibles, hasta que todas sean reproducibles.

**Resultado final**

Para cumplir el objetivo inicial y solucionar todos los problemas surgidos durante el desarrollo, se presentan las siguientes funciones contenidas en la aplicación desarrollada, que apoyándose en los medios brindados como ayuda, llevan a cabo las tareas requeridas:

- **Función *init():*** Es la función de entrada, llamada desde el código HTML, se encarga de conectar la aplicación al servidor MQTT.

- **Función *deCodifica(valor):*** Esta función se encarga de decodificar los mensajes que llegan a través de MQTT a la aplicación.

[8] UTF-8 (8-bit Unicode Transformation Format) es una norma de transmisión de longitud variable para caracteres codificados utilizando Unicode, usa grupos de bytes para representar el estándar de Unicode para los alfabetos de muchos de los lenguajes del mundo.

- **Función *dividirMensaje(mensaje):*** Esta función se encarga de dividir de forma recursiva el mensaje pasado como parámetro en submensajes más pequeños reproducibles por la aplicación, hasta que el mensaje completo haya sido reproducido.

- **Función *speak(str):*** Es la función a la cual se llama para reproducir un mensaje pasado como parámetro, contiene la llamada a Chrome TTS.

## 3.2. Desarrollo de la aplicación de reconocimiento de voz

Este apartado trata la descripción del proceso de realización de la aplicación de reconocimiento de voz.

**Objetivos y medios para alcanzarlos**

Esta aplicación tiene como objetivo ser una aplicación ejecutable en el navegador Chrome y ser una aplicación cliente dentro de un modelo cliente/servidor MQTT, para poder enviar mensajes a otras aplicaciones cliente. Una vez se consiga esto, el otro objetivo principal es permitir al robot mantener con un usuario una interacción para obtener mediante reconocimiento de voz una serie de datos para tratarlos, transformarlos, prepararlos y enviarlos como mensaje a los servicios de calendario. La integración de un protocolo de paso de mensajes, MQTT, que es igual para todas las aplicaciones de este trabajo de fin de grado, soluciona el envío y recepción de mensajes en esta aplicación. Dicha integración es descrita en el punto 3.3 de esta memoria. Por otro lado, para el reconocimiento de voz se ha usado Web Speech API (ver sección 2.3.3), que provee los siguientes métodos y funcionalidades para dicho fin, y que son los que hemos usado:

```
start();

stop();
```

Una vez definida una variable de clase reconocimiento, la llamada a estas funciones, *start* y *stop*, desde dicha variable marcan el principio y el final respectivamente del reconocimiento de voz. Para un mejor control de sus funciones, existen métodos en la API que tratan los eventos ocurridos durante el reconocimiento. La función que cada uno define es llamada cuándo su determinado evento se produce.

- La función de *onstart* es llamada una vez que empieza el reconocimiento.

```
onstart = function() { … };
```

- La función de *onend* es llamada cuando el reconocimiento termina.

```
onend = function() { … };
```

- La función definida en *onerror* es llamada cuándo se produce un error en durante el reconocimiento, presenta un parámetro en el cual se devuelve el error producido.

```
onerror = function(event) { … };
```

- La función de *onresult* es llamada cuando se ha obtenido un resultado final, tiene un parámetro que devuelve el resultado final del reconocimiento.

```
onresult = function (event) { … };
```

Además, Web Speech API también permite establecer a la variable de clase reconocimiento una serie de opciones iniciales que cambiarán el comportamiento del reconocimiento, como el lenguaje del reconocimiento o el número de resultados alternativos finales del reconocimiento.

**Desarrollo**

El proceso de desarrollo de esta aplicación consta de un conjunto de fases que se muestran a continuación:

- **Periodo de documentación**: Es la fase anterior al desarrollo, en la que se adquieren conocimientos y recursos necesarios para llevar a cabo el desarrollo correcto de la aplicación. Se adquieren conocimientos sobre la creación de aplicaciones para Chrome ([2] [3] [4] [5] [6]) y conocimientos sobre el uso de Web Speech API ([8] [9] [10]).

- **Desarrollo de los requisitos para ser una aplicación de Chrome:** En esta fase se desarrolla el requisito para ser una aplicación ejecutable en el navegador Chrome, la creación de un archivo *Manifest*.

- **Desarrollo del servicio de reconocimiento:** Es la fase que otorga la característica principal requerida de la aplicación, el reconocimiento de voz. Se implanta la aplicación Web Speech en el código y se adapta a los requerimientos necesitados.

- **Implantación del protocolo MQTT**: Una vez se tenga la funcionalidad de reconocimiento, en esta fase se implanta el protocolo de paso de mensajes

MQTT, para que la aplicación pueda enviar mensajes a las demás aplicaciones. Este será el punto de entrada inicial de la aplicación, y una vez más su correcto funcionamiento dependerá del estado de este protocolo.

- **Periodo de refinamiento**: En esta última fase, se mejoran en la medida de lo posible las fases anteriores, teniendo como objetivo el modelo de reconocimiento.

**Problemas surgidos durante el desarrollo**

En este apartado se muestran los principales problemas acaecidos durante las distintas fases del desarrollo de esta aplicación y cómo se han podido solucionar.

- **Web Speech API no soporta llamadas locales**: Inicialmente, no se contaba con la necesidad de usar un servidor web para alojar el código HTML y su extensión en JavaScript, pero Web Speech API solo es accesible desde código alojado en un servidor, no es accesible localmente por lo que para solucionarlo se ha necesitado durante el desarrollo de esta aplicación una herramienta como XAMPP que proporciona un servidor. Durante el desarrollo, esta restricción dejó de serla, ya que al convertir la aplicación desarrollada en una aplicación de Chrome se eliminó tal necesidad.

- **Al comienzo del reconocimiento se debe aceptar el uso del micrófono por la aplicación**: Dos de las restricciones que Web Speech API presenta, son que cada comienzo de reconocimiento se debe aceptar el uso del micrófono y que cada reconocimiento termina tras un periodo de tiempo. Nuestra aplicación realiza un reconocimiento continuo volviendo a empezarlo cada vez que se termina, luego no es aceptable aceptar el uso del micrófono constantemente. Inicialmente se planteó adaptar el código HTML de esta aplicación al protocolo HTTPS para que el navegador recordara el permiso de uso del micrófono, pero esto creaba problemas en MQTT difíciles de solucionar. Por consiguiente, se ha optado por que la aplicación desarrollada fuera también una aplicación del navegador Chrome y en su declaración como tal mediante el archivo *Manifest*, obtener los permisos de uso del micrófono para todo el periodo de ejecución de la aplicación.

- **MQTT no acepta el conjunto de símbolos latinos:** Este problema se ha tratado y resuelto tal y como se presenta en la sección 3.1.

**Resultado final**

Para cumplir el objetivo inicial, solucionar todos los problemas surgidos durante el desarrollo, se presentan las siguientes funciones, que apoyándose en los medios brindados como ayuda, llevan a cabo las tareas requeridas:

- **Función *init():*** Es la función de entrada, llamada desde el código HTML, se encarga de conectar la aplicación al servidor MQTT.

- **Función *initSTT():*** Es la función principal, se llama cuando llega un mensaje por MQTT que así lo solicita, inicia el reconocimiento de voz y establece todos los procedimientos a seguir una vez que se tiene un resultado del reconocimiento, así como enviarlo al calendario.

- **Funciones *obtenerHora(), obtenerFecha(), obtenerEvento, chequeoFinal():*** Son una serie de funciones llamadas desde *initSTT* en el momento adecuado, que se encargan de reconocer la hora, fecha, evento y aceptación general del reconocimiento respectivamente. También chequean que el reconocimiento haya sido coherente e igual a como el usuario ha querido transmitir.

- **Función *codifica(valor):*** Esta función se encarga de codificar los mensajes que la aplicación envía a través de MQTT.

## 3.3. Integración de un protocolo de comunicación de paso de mensajes

La integración de un protocolo de comunicación de paso de mensajes en las aplicaciones tiene como motivo dotar a estas de un medio para poder conectarse a un servidor, convirtiéndolas en aplicaciones cliente dentro de un modelo cliente/servidor en el cual pueden enviar y recibir mensajes de otras aplicaciones cliente [21]. Esta integración está presente en todas las aplicaciones y será común en ellas, usándose como protocolo de comunicación de paso de mensajes, el protocolo MQTT. Todo esto ha sido posible gracias a la implementación en código abierto de la parte cliente realizada por Paho Eclipse [11], que provee los siguientes métodos y funcionalidades para alcanzar dicha funcionalidad, y que son los que hemos usado:

```
connect(options);

subscribe(topic, qos);

send(mqttmessage);
```

Una vez definida una variable de clase cliente MQTT con la IP y puerto del servidor específico, desde dicha variable se pueden llamar a las funciones *connect*, *subscribe* y *send*.

- *Connect:* realiza una conexión al servidor, estableciendo las opciones pasadas como parámetro.

- *Subscribe:* suscribe al cliente en el servidor con un *topic* determinado y una calidad de servicio determinada.

- *Send:* envía un mensaje MQTT, en el cual previamente se definen el *topic* al cual se envía y el mensaje que se envía.

Estos otros dos métodos son de vital importancia ya tratan los eventos de perdida de conexión con el servidor y llegada de un mensaje respectivamente. La función que cada uno define es llamada cuándo su determinado evento se produce y los parámetros devuelven el error producido y el mensaje recibido respectivamente.

```
onConnectionLost = function (responseObject) { … };

onMessageArrived = function (message) { … };
```

# Capítulo 4

# Implementación de los servicios de calendario

En este capítulo se describe el proceso a seguir para obtener desde Google el permiso de uso de su API Calendar y se presenta una descripción a nivel de **implementación** de la aplicación de calendario. Toda esta descripción está apoyada en el estudio previo de la documentación y manuales que ofrecen cada una de las herramientas usadas. Se desglosan los objetivos de la aplicación y los medios disponibles para alcanzarlos, se describe la fase de desarrollo de la aplicación, los problemas surgidos durante esta y el resultado final a nivel de código implementado de la aplicación. El código completo de esta aplicación no se describe en la memoria, pero se adjunta junto a esta.

## 4.1. Permiso de uso de la aplicación de calendario

Para poder usar la API de Google Calendar, Google obliga a tener una cuenta su consola de aplicaciones, *Google API console,* en la que se gestionan y otorgan las APIs que se usan y los permisos de uso que se conceden. En esta consola se crea un proyecto al cual en el momento de su creación se le otorga un identificador de proyecto único, *API key* (ver figura 4.1). Entre los parámetros que se establecen en esta consola, se establecen las APIs que el proyecto usa (ver figura 4.2) y se establecen los credenciales que se otorgan al proyecto, como por ejemplo el permiso de acceso a datos privados de un calendario, clave *OAuth* (ver figura 4.1). El motivo de esta exigencia por parte de Google, aparte de medidas de seguridad y privacidad, es que algunas de las APIs que Google ofrece son de pago, y otras son de uso gratuito hasta un cierto número de solicitudes por día, como Google Calendar, que tiene un millón de llamadas gratuitas por día, por lo tanto Google mantiene un muestreo de la actividad realizada sobre sus APIs obligando a que el código que desee usar sus APIs necesite autenticarse con una clave identificadora de proyecto única y además establezca una sesión cada vez que se ejecute el código. Tanto la *API key* como la *OAuth* deben ser cargadas en el código para enlazar con el proyecto.

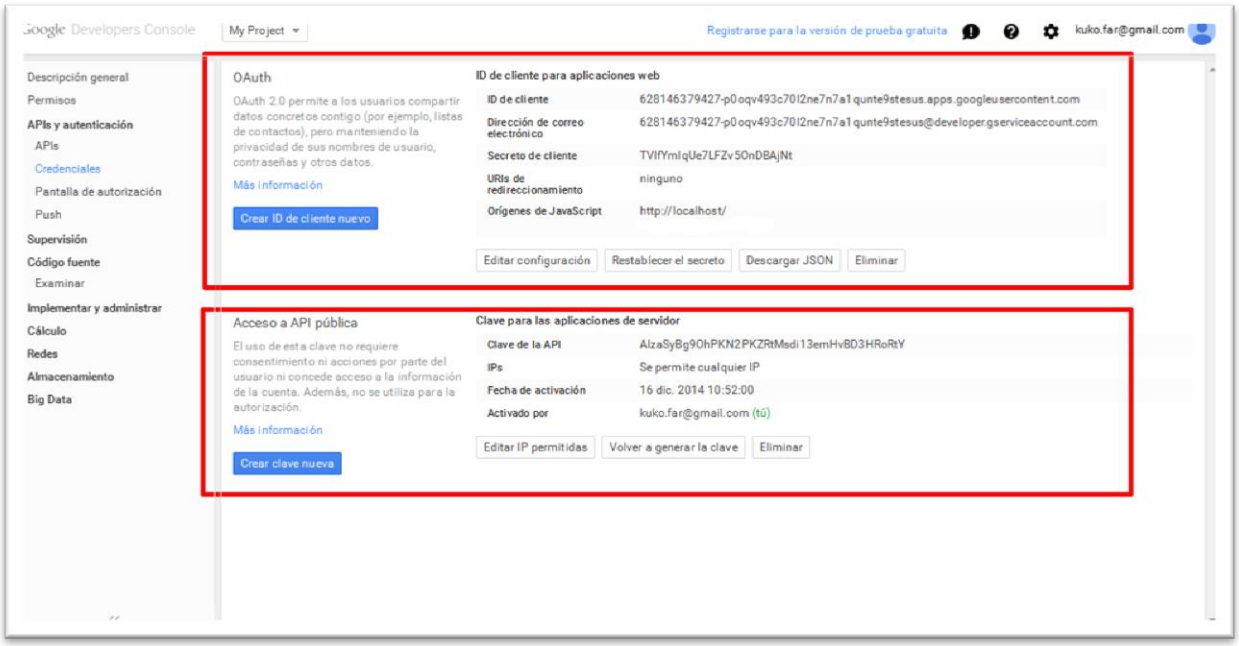


Figura 4.1. API key y OAuth para este proyecto en la consola de APIs de Google.

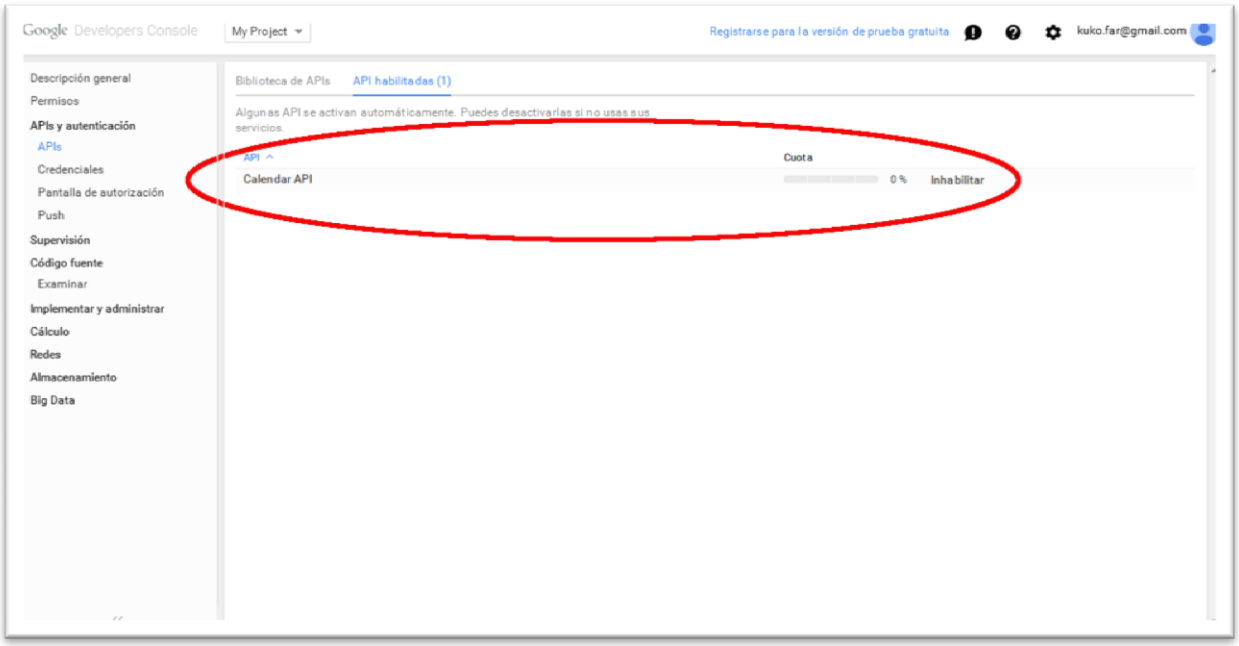


Figura 4.2. API habilitada para este proyecto en la consola de APIs de Google.

En el código de nuestro programa, las funciones que realizan este cometido de autenticación son las siguientes:

```
gapi.client.setApiKey(apiKey);
```

Establece una clave de aplicación que asocia nuestro código con nuestro proyecto por medio del identificador de proyecto único.

```
gapi.auth.authorize({ params }, callback );
```

Otorga una autorización explícita para el uso del calendario por la aplicación. Los parámetros definen:

- El conjunto *params*, incluye el ID del calendario a usar y el ámbito de la autorización.

- El parámetro *callback* define una función que se ejecutará una vez el proceso de chequeo haya finalizado.

Por último se obtiene un *token* con caducidad temporal para la sesión establecida.

```
gapi.auth.getToken();
```

## 4.2. Desarrollo de la aplicación de interacción con el calendario

A continuación se describe el proceso de realización de la aplicación de calendario.

**Objetivos y medios para alcanzarlos**

Esta aplicación tiene como objetivo ser una aplicación cliente dentro de un modelo cliente/servidor MQTT, para poder recibir y enviar mensajes desde y hacia otras aplicaciones cliente. Una vez recibido un mensaje, su otro cometido es realizar las acciones necesarias en el calendario atendiendo a lo solicitado con el mensaje, ya sea extraer una serie de eventos y mandarlos por mensaje para su reproducción mediante voz, o administrar, tratar y transformar los datos recibidos para insertarlos como evento en el calendario. La integración de un protocolo de paso de mensajes, MQTT, que es igual para todas las aplicaciones de este trabajo de fin de grado, soluciona la recepción y envío de mensajes en esta aplicación. Dicha integración es descrita en el punto 3.3 de esta memoria. Por otro lado, para la inserción o extracción de eventos en el calendario, se ha usado la API de Google Calendar (ver sección 2.3.2), que provee los siguientes métodos para dicho fin, y que son los que hemos usado:

```
gapi.client.load(name, version, callback);
```

Esta función carga la librería cliente a la API, sus parámetros definen:

- *Name*, el nombre de la API a cargar.

- *Version*, la versión de la API a cargar.

- *Callback*, define una función que será llamada una vez la librería sea cargada.

```
gapi.client.calendar.events.list({calendarId, orderBy,
timeMin, timeMax}, callback);
```

Esta función extrae del calendario una serie de eventos, los parámetros especifican las condiciones de extracción:

- *CalendarID*: ID del calendario del cual se extraerán los eventos.

- *OrderBy*: Especifica la forma de ordenar los eventos a extraer.

- *TimeMin*: Delimita la fecha mínima que un evento debe tener para ser extraído.

- *TimeMax*: Delimita la fecha máxima que un evento debe tener para ser extraído.

- Callback: Define una función que será llamada una vez se haya completado la extracción de eventos.

```
gapi.client.calendar.events.insert({calendarId, resource},
callback);
```

Esta última función de interacción con el calendario, se encargará de insertar los eventos en el mismo, sus parámetros definen:

- *CalendarID*: ID del calendario en el cual se insertarán los eventos.

- *Resource*: Se establece el evento a insertar.

- *Callback*: Define una función que será llamada una vez se haya completado la inserción del evento.

**Desarrollo**

El proceso de desarrollo de esta aplicación consta de una serie de fases que se detallan a continuación:

- **Periodo de documentación**: En esta fase se adquieren los conocimientos y recursos necesarios para llevar a cabo el desarrollo correcto. Se adquieren

conocimientos sobre Google Calendar API ([12] [20]), el uso de su cliente en JavaScript [13] y de las autenticaciones necesarias para su uso ([14] [15]).

- **Establecimiento de sesión:** Es la fase previa al desarrollo de las funcionalidades de calendario, y en ella se establece una identificación tanto del proyecto como del calendario ante Google para que este pueda monitorizar el uso de su API, y se establece una sesión temporal de uso. Todo ello mediante las funciones que brinda el cliente de Google Calendar API.

- **Desarrollo del servicio de extracción del calendario:** En esta fase se dota a la aplicación de la capacidad de leer un evento del calendario, para ello usamos la función que el cliente en JavaScript de Google Calendar API ofrece para tal tarea, estableciendo las características que sean necesarias.

- **Desarrollo del servicio de inserción en el calendario:** En esta fase se preparan y transforman los datos recibidos por mensaje y se dota a la aplicación de la capacidad de insertar estos datos en forma de eventos en el calendario gracias nuevamente a la función que el cliente en JavaScript de Calendar API ofrece.

- **Implantación del protocolo MQTT**: Una vez se tenga las funcionalidades de extracción e inserción, en esta fase se implanta el protocolo de paso de mensajes MQTT, para que la aplicación pueda recibir y enviar mensajes a las demás aplicaciones. Este será el punto de entrada inicial de la aplicación, y su correcto funcionamiento dependerá del estado de este protocolo.

- **Periodo de refinamiento**: En esta última fase, se mejoran y optimizan todo lo posible las fases anteriores.

**Problemas surgidos durante el desarrollo**

En este apartado se muestran los principales problemas acaecidos durante las distintas fases del desarrollo de esta aplicación y cómo se han podido solucionar.

- **Google Calendar API no soporta llamadas locales**: Inicialmente, no se contemplaba la necesidad de usar un servidor para alojar el código HTML y su extensión en JavaScript, pero la API de Google Calendar solo es accesible desde código alojado en un servidor, no es accesible localmente por lo que para solucionarlo se ha necesitado durante el desarrollo de esta aplicación una herramienta como XAMPP que proporciona un servidor. Una vez implantada en el robot Jirafa, se usa el servidor web del que éste dispone.

- **MQTT no acepta el conjunto de símbolos latinos:** Este problema se ha tratado y resuelto tal y como se presenta en la sección 3.1.

**Resultado final**

Para cumplir el objetivo inicial, solucionar todos los problemas surgidos durante el desarrollo, se presentan las siguientes funciones, que apoyándose en los medios brindados como ayuda, llevan a cabo las tareas requeridas:

- **Función *init():*** Es la función de entrada, llamada desde el código HTML, se encarga de conectar la aplicación al servidor MQTT. También se encarga de establecer la sesión de autenticación.

- **Función *leerDatos(msg):*** En esta función se carga la API de Calendar, se obtiene el periodo de tiempo de lectura de eventos según el parámetro pasado y se llama a la función de extracción de los eventos.

- **Función *llamadaAPI(tiempoActual,tiempoFinal):*** Esta es la función que hace la llamada a la API de Calendar para la extracción de los eventos existentes en el calendario durante el periodo de tiempo establecido por los parámetros.

- **Función *insertarDatos(msg):*** En esta función se preparan los datos a insertar en el calendario y se carga la API de Calendar, finalmente se llama a la función de la API de Calendar que inserta los datos en el calendario.

- **Funciones *obtenerTiempoActual(), obtenerTiempoAnyo(), obtenerTiempoFinalDia(), obtenerTiempoFinalManana(), obtenerTiempoFinalManana(), obtenerTiempoPrincipioManana(), obtenerTiempoFinalSemana():*** Son una serie de funciones que son llamadas desde *leerDatos* cuando es necesario, que se encargan de obtener la fecha y hora del momento que su nombre indica en el estándar RFC3339 [9] [16]. Este tiempo que obtienen es necesario para ser pasado como parámetro en las extracciones e inserciones del calendario, ya que el formato de tiempo en Google Calendar API está definido según dicho estándar.

- **Funciones *codifica(valor)* y *decodifica(valor):*** Ambas funciones ya han sido anteriormente descritas en las secciones 3.2 y 3.1 respectivamente.

[9] RFC3339, estándar de fecha y hora en internet.

# Capítulo 5

## Integración de las aplicaciones en la arquitectura del robot

Las aplicaciones implementadas en este trabajo están listas para ser integradas en la arquitectura del robot, denominada OpenMora [1]. OpenMora (Open Mobile Robot Arquitecture) es una arquitectura distribuida diseñada para robots móviles. Esta arquitectura está diseñada y desarrollada por el departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga. Dicha arquitectura implementa el paradigma de arquitectura robótica híbrida denominado ACHRIN [22]. OpenMora se basa básicamente en dos tecnologías, MOOS y MRPT. The Mobile Robot Programming Toolkit (MRPT) [23] es una biblioteca multiplataforma y abierta en C++ desarrollada para facilitar el diseño y la implementación de algoritmos relacionados con la localización, planificación de movimientos, etc. Múltiples centros de investigación la han utilizado para realizar sus proyectos. Por otro lado, The Mission Oriented Operating Suite (MOOS) [24] es un framework multiplataforma implementado en C++ para robótica, orientada a las comunicaciones entre módulos usando el protocolo TCP/IP. Los módulos de MOOS se conectan a una base de datos denominada MOOSDB configurando una topología de conexión de tipo estrella. Cada módulo se conecta a la base de datos para escribir o recoger información de sus tablas, quedando de esta forma excluidas las comunicaciones directas entre módulos.

La integración con OpenMora de las aplicaciones desarrolladas se ha llevado a cabo a través del protocolo de comunicación MQTT, teniendo como punto de entrada un módulo MQTT Listener, integrado en OpenMora, que traduce mensajes MQTT hacia la pizarra compartida de Openmora y viceversa. (Ver figura 5.1)

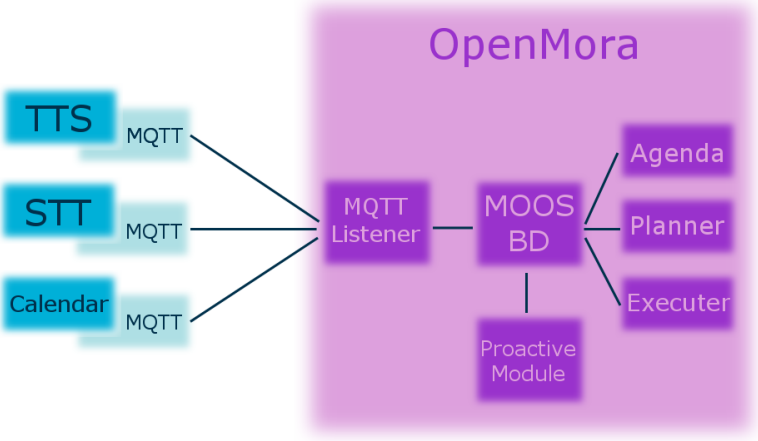


Figura 5.1. Integración de las aplicaciones en OpenMora.

Una vez implantadas las aplicaciones en el robot, también van a interactuar con un módulo monitor implementado por el equipo desarrollador del robot Jirafa, que implementa una de las posibles mejoras que se pueden añadir a este trabajo, la reproducción automática de un evento (ver sección 7.2.1). Más en concreto, el módulo de la arquitectura del robot *Proactive Manager* se comunica con la aplicación de Calendario para consultar si hay eventos próximos. En caso afirmativo, genera una tarea de alto nivel consistente en una navegación reactiva, y en el acceso a la aplicación de reproducción de voz para anunciar dichos eventos. Esta tarea es insertada en el módulo agenda y posteriormente planificada y ejecutada por el robot. (Ver figura 5.2).

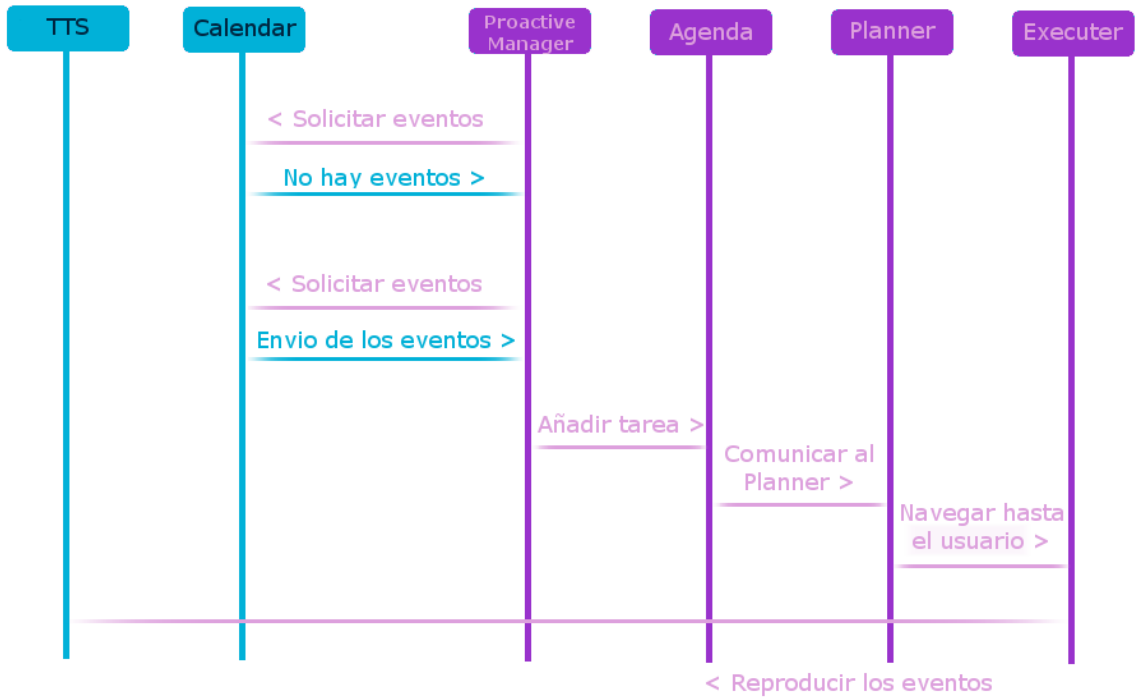


Figura 5.2. Ejemplo de una comunicación del módulo que monitoriza la existencia de eventos, Proactive Manager, y las aplicaciones desarrolladas. (En la figura, la comunicación con los módulos MQTT Listener y MOOS BD ha sido suprimida con el objetivo de que la figura fuera más intuitiva. En la figura, la comunicación ha sido realizada directamente con los módulos sin pasar por los dos anteriormente mencionados).

# Capítulo 6

# Resultado final y uso

En este capítulo se presenta el resultado final de este trabajo fin de grado como una **abstracción** de la implementación y de las tecnologías usadas durante el desarrollo. Para ello se realiza una descripción de las tres aplicaciones implementadas para alcanzar los objetivos inicialmente fijados y proporcionar las funcionalidades requeridas al robot Jirafa. En este capítulo también se destacan los casos de uso que el robot puede llevar a cabo con estas aplicaciones implantadas y funcionando. Como documento adjunto se presenta un video en el que se ilustra un ejemplo real del funcionamiento de las aplicaciones en el robot.

## 6.1. Aplicaciones implementadas

Una vez finalizado el trabajo de fin de grado, han sido tres las aplicaciones desarrolladas, que implantadas y funcionando, dotan al robot Jirafa de funcionalidades de servicios de voz, reproducción y reconocimiento, y de funcionalidad de calendario. (Ver figura 6.1).


Figura 6.1. Imagen que caricaturiza al robot Jirafa con sus nuevas aplicaciones.

A continuación, se describen las tareas que realizan cada una de estas aplicaciones, las tecnologías en las que están basadas y las herramientas en las que se apoyan para poder cumplir su función.

- **Aplicación TTS:** Es la aplicación encargada de la reproducción de voz, su principal tarea es la de administrar todos los mensajes que recibe de las demás aplicaciones para que sea posible su reproducción de forma correcta mediante voz por el robot Jirafa. Esto es posible gracias a la aplicación TTS del navegador Chrome, en la cual está basada y que permite la reproducción de voz, y en el apoyo en MQTT, gracias al cual puede recibir mensajes de otros componentes de la arquitectura del robot.

- **Aplicación STT:** Es la aplicación encargada del reconocimiento de voz, cuya principal tarea es la de realizar un reconocimiento constante desde la entrada del micrófono. Una vez reconocido un patrón concreto y predefinido, la aplicación realiza la labor requerida, por ejemplo insertar un evento definido por el usuario en el calendario, o solicitar al calendario una serie de eventos. Esta aplicación puede llevar a cabo su cometido gracias a la aplicación Web Speech sobre la cual se basa y que le permite el reconocimiento, y gracias a estar apoyada en MQTT, por el cual puede recibir o enviar mensajes a otras aplicaciones.

- **Aplicación Calendar:** Es la aplicación encargada del calendario, su cometido principal es el de administrar todas las acciones de inserción y extracción de eventos del calendario que la aplicación STT le solicite mediante mensaje, transformando los datos de forma adecuada para que esto sea posible. Estas tareas son posibles ya que esta aplicación está basada en la API de Google Calendar que permite desarrollar aplicaciones clientes. También se encarga mediante mensajes de enviar los datos a la aplicación TTS cuándo se tengan que reproducir. Finalmente reseñar que está apoyada en MQTT mediante el cual puede enviar o recibir mensajes a las otras aplicaciones y componentes de la arquitectura del robot.

Por último, se ha dotado al robot Jirafa de una interfaz que hace más agradable la interacción con el usuario, para ello se han desarrollado una serie de imágenes GIF (ver figura 6.2) (ver sección interfaz gráfica en anexo), las cuáles se alternan dependiendo del estado de ejecución de las aplicaciones, siendo éstas las imágenes visibles en el robot Jirafa durante las distintas fases de sus aplicaciones.

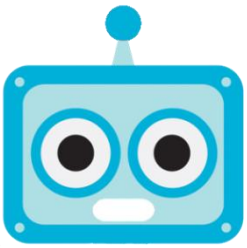

Figura 6.2. Imagen principal de la interfaz de las aplicaciones en el robot Jirafa.

## 6.2. Casos de uso

En este capítulo se describen los distintos casos de uso que el robot Jirafa puede realizar gracias a las aplicaciones desarrolladas. Los principales casos de uso son la inserción de un evento en el calendario y la reproducción por parte del robot de eventos contenidos en el calendario, pudiendo esta última hacerse de forma manual o automática. Para cada uno de estos casos de uso, se presenta una descripción y se detallan las fases de su desarrollo; condiciones previas, proceso y consecuencias.

A continuación se presentan las principales acciones que podrá llevar a cabo el robot gracias a las nuevas funcionalidades que ha adquirido. Es importante reseñar, que en el anexo (ver descripciones gráficas en anexo) se presentan las descripciones gráficas de estas tres acciones, así como las descripciones graficas también de las diferentes variantes que cada uno presenta.

### 6.2.1.Inserción de un evento en el calendario

La inserción de un evento es un proceso mediante el cual un usuario puede insertar un evento en el calendario gracias a la interacción directa con el robot. El usuario solicita al robot insertar un evento, y éste, le pregunta por la hora, fecha y descripción del mismo, las preguntas las hace de forma ordenada, una a una y espera a que el usuario responda antes de realizar la siguiente. Si el usuario no da una respuesta coherente con la pregunta realizada por el robot, éste vuelve a repetir la pregunta. Por último, el robot repite las respuestas que el usuario ha contestado y se cerciora mediante otra pregunta si esas han sido las respuestas que el usuario ha querido dar. Una vez la persona responda coherentemente las preguntas que el robot le va haciendo, éste procesa los datos obtenidos de manera adecuada y los inserta en el calendario en forma de evento. (Ver figura 6.3).

- **Previo / Precondición**: El robot Jirafa debe tener activado el reconocimiento de voz y debe estar lo suficientemente cerca de la persona para que ambos

puedan escucharse y entenderse lo suficientemente bien para mantener una conversación.

- **Proceso de desarrollo**: El usuario debe solicitar al robot la inserción de un evento mediante alguno de los patrones preestablecidos y reconocidos por el robot Jirafa (ver lista de patrones en el anexo) para la inserción de un evento. Una vez reconocido el patrón, el robot realiza al usuario las siguientes peticiones:
  - Introduce la hora del evento.
  - Introduce la fecha del evento.
  - Introduce la descripción del evento.

  Como se mencionó anteriormente, después de cada pregunta se espera la respuesta del usuario, en caso de no haber respuesta o de no ser ésta coherente, no se avanza en la conversación y el robot vuelve a solicitar la inserción de la siguiente manera:
  - Hora introducida incorrectamente, vuelva a introducir la hora.

    ó
  - Fecha introducida incorrectamente, vuelva a introducir la fecha.

  En el caso de la descripción del evento, si la descripción no es reproducible por el robot, el robot responderá:
  - Su descripción es demasiado larga, vuelva a darla usando puntos para separar las frases.

  Si por el contrario la descripción dada sí es reproducible por el robot, éste pregunta por su validez al usuario:
  - ¿Es *"la descripción dada"* su descripción?

  Una vez el usuario responde, si lo hace de forma negativa según el patrón o de una forma no entendible por el robot, éste lo expresa de las siguientes formas respectivamente:
  - Entonces vuelva a introducir su descripción.
  - No he podido entenderle, vuelva a repetirlo.

  En caso de que la respuesta sea positiva según el patrón, el robot reproduce la fecha, la hora y la descripción y pregunta por su validez:
  - ¿Es *"la descripción dada".* A la hora *"hora dada".* Del día *"día dado"* su evento?

  Nuevamente, en caso de una respuesta negativa según el patrón, se termina desechando el evento, o no entendida por el robot, las respuestas serán respectivamente:
  - En tal caso, el evento ha sido desechado.
  - No he podido entenderle, vuelva a repetirlo.

Si por el contrario la respuesta es positiva según el patrón, la respuesta del robot es:

- Evento introducido con éxito.

- **Consecuencias**: Una vez reconocido completamente el evento por el robot Jirafa, el objetivo final es introducirlo en el calendario, para ello lo procesa e inserta. Si se ha podido introducir en el calendario sin problemas, el robot responde:
  - Evento insertado en el calendario con éxito.

  En el caso contrario de que no haya sido posible introducirlo, el robot Jirafa lo comunica de la siguiente manera:
  - El evento no se ha podido insertar.

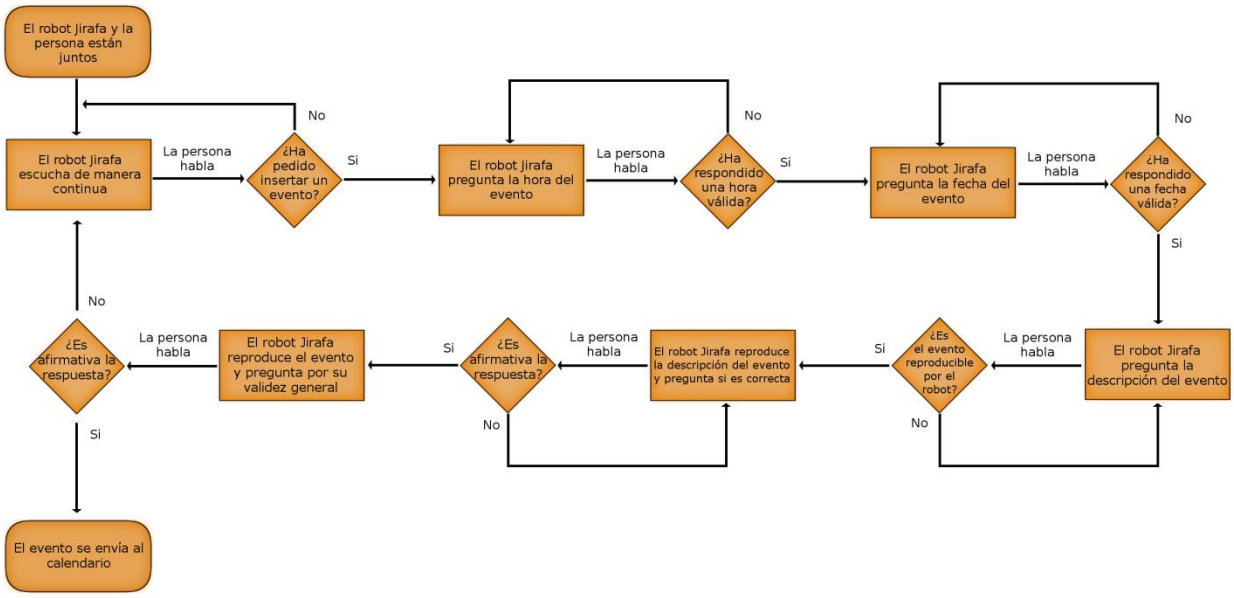


Figura 6.3. Diagrama de la inserción de un evento.

La inserción de un evento en el calendario no dispone de variantes como tal, ya que en este trabajo de fin de grado solo contempla la interacción con el usuario anteriormente descrita. Pero para este caso de uso en concreto, indirectamente se abre la posibilidad de que un tercero, ya sea la persona que controla remotamente al robot, alguna persona perteneciente al personal sanitario o cualquier otra persona, siempre que disponga de permiso y consentimiento por parte del usuario del robot, pueda insertar o borrar eventos directamente en el calendario por medio de la interfaz web de Google Calendar, lo que permitiría que la inserción de eventos al calendario no fuera exclusivo del usuario del robot, sino que cualquier otra persona como por ejemplo un médico o un familiar pudiera añadir citas o recordatorios que el robot podría reproducir al usuario. Esta funcionalidad indirecta dotará de un mayor potencial a los servicios desarrollados en este trabajo de fin de grado.

### 6.2.2. Reproducción de un evento en el calendario

La reproducción de un evento es un proceso mediante el cual el robot Jirafa reproduce de forma sonora una serie de eventos previamente solicitados por el usuario. El usuario solicita al robot que le transmita los eventos existentes en un determinado periodo de tiempo, si la solicitud por parte del usuario no es clara o no es coherente, el robot la ignora, pero si esta solicitud es correcta y es entendida por el robot, éste busca en el calendario los eventos solicitados y los reproduce mediante voz de forma ordenada por fecha y hora, en el caso de que no existan eventos en ese periodo de tiempo lo comunicará al usuario. (Ver figura 6.4).

- **Previo / Precondición**: El robot Jirafa debe tener activado el reconocimiento de voz y debe estar lo suficientemente cerca de la persona para que ambos puedan escucharse y entenderse lo suficientemente bien para mantener una conversación.

- **Proceso de desarrollo**: El usuario debe solicitar al robot la reproducción de un conjunto de eventos mediante alguno de los patrones preestablecidos y reconocidos por el robot Jirafa (ver lista de patrones en el anexo) para la reproducción de un evento. Una vez aceptado el patrón, el robot Jirafa responde de la siguiente manera dependiendo del periodo de tiempo solicitado:
  - Vamos a mostrar todos los eventos venideros.
  - Vamos a mostrar los eventos de la semana.
  - Vamos a mostrar los eventos de hoy y mañana.
  - Vamos a mostrar los eventos de hoy.
  - Vamos a mostrar los eventos de mañana.

  Por último extrae del calendario los eventos solicitados.

- **Consecuencias**: Finalmente, si existen los eventos solicitados el robot Jirafa los reproducirá uno a uno mediante voz al usuario de la siguiente forma:
  - El número de eventos leídos es de *"nº eventos extraídos".*
  - El contenido del evento *"nº evento"* es *"descripción del evento".* Su fecha es el *"fecha del evento".* Y la hora *"hora del evento".*

  En caso de que no existan eventos en el periodo de tiempo solicitado por el usuario, la respuesta del robot es:
  - No hay eventos para mostrar.

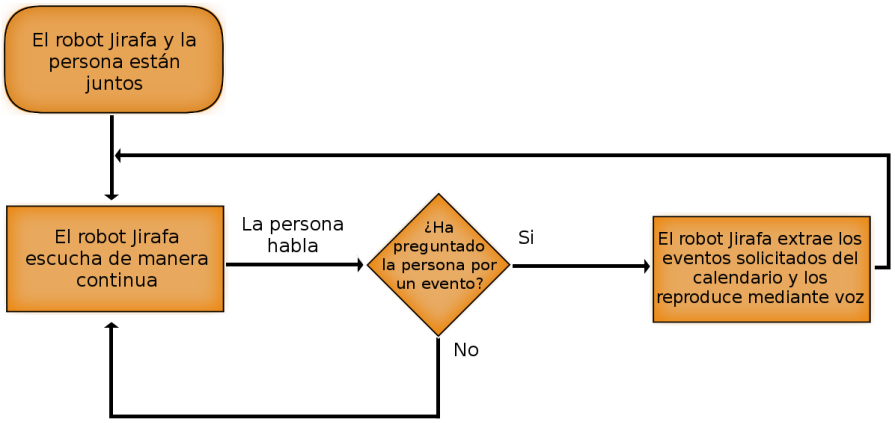


Figura 6.4. Diagrama de la reproducción de eventos.

La reproducción de un evento del calendario dispone de diversas variantes, estas variantes van a diferir en el periodo de tiempo en el que se encuentran los eventos solicitados para reproducir, así las modalidades son:

- Reproducción de los eventos de hoy.
- Reproducción de los eventos de mañana.
- Reproducción de los eventos de hoy y mañana.
- Reproducción de los eventos de la semana.
- Reproducción de todos los eventos.

El caso de la reproducción de eventos del calendario, abrió indirectamente la posibilidad de que un tercero, ya sea una persona que controle remotamente al robot o incluso una aplicación software pudiera activar la reproducción de alguna de estas modalidades mediante un mensaje MQTT. Esto ha dado lugar al desarrollo de un módulo complementario, para la reproducción automática de eventos, lo que ha permitido que la solicitud de reproducción de eventos del calendario no sea exclusiva del usuario.

### 6.2.3.Reproducción automática de un evento en el calendario

La reproducción de un evento de forma automática es el proceso mediante el cual el robot Jirafa reproduce mediante voz una serie de eventos, en este caso no solicitados por el usuario, sino por un módulo monitor. Este módulo software, desarrollado por el equipo que desarrolla el robot Jirafa, monitoriza regularmente y de forma automática la existencia de eventos próximos en calendario, en caso de hallar alguno, el robot Jirafa planifica y ejecuta la tarea de navegación autónoma hasta la posición del usuario, y una vez en presencia del usuario reproduce los eventos correspondientes mediante voz de forma ordenada por fecha y hora. En caso contrario, que no haya eventos próximos en el calendario, no hay actuación. (Ver figura 6.5).

- **Previo / Precondición**: El robot Jirafa debe tener el módulo monitor de eventos funcionando. En este caso de uso, al contrario que en todos los demás, no importa en el lugar que se encuentre con respecto al usuario.

- **Proceso de desarrollo**: El módulo continuamente monitoriza los eventos cercanos que se encuentran almacenados en el calendario.

- **Consecuencias**: Finalmente, si existen eventos en el calendario, el robot Jirafa navegará desde donde esté ubicado hasta donde se encuentre el usuario y los reproducirá uno a uno mediante voz de la siguiente forma:
  - Estos son los eventos próximos.
  - El contenido del evento *"nº evento"* es *"descripción del evento"*. Su fecha es el *"fecha del evento"*. Y la hora *"hora del evento"*.

  En caso de que no existan eventos próximos en el calendario, el robot Jirafa no realiza actuación alguna.

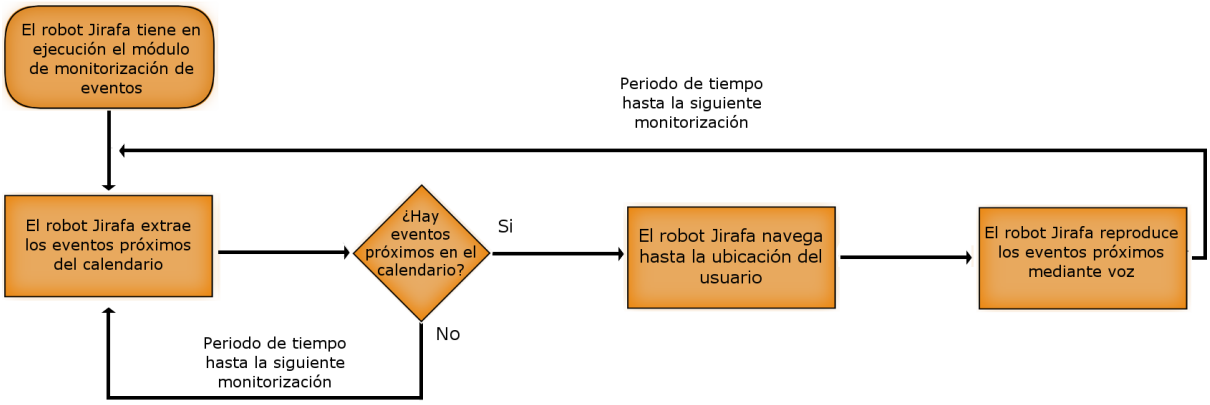


Figura 6.5. Diagrama de la reproducción automática de eventos.

Es de reseñar que la planificación y ejecución autónoma del plan para desplazar al robot hasta la ubicación del usuario está fuera del ámbito de este trabajo fin de grado. Para más información sobre la arquitectura del robot jirafa, consultar [1].

# Capítulo 7

# Conclusiones

El desarrollo de este trabajo fin de grado ha significado un pequeño aporte a un campo que aún es novedoso y que está por desarrollar en gran medida, como es la robótica telepresencial. Por consiguiente, ha aportado su pequeño granito de arena cumpliendo los objetivos inicialmente marcados que fueron dotar al robot Jirafa de una funcionalidad de calendario y de una capacidad de interacción con el usuario de forma simple mediante voz. Ahora, con este trabajo, el robot de asistencia dispondrá de mayores facilidades para desarrollar su cometido.

La realización de este trabajo ha supuesto un importante reto al combinar entre sí tecnologías tan diferentes como las que se han empleado durante su desarrollo, como por ejemplo un protocolo de comunicación de paso de mensajes o una API que permite desarrollar aplicaciones cliente, necesitándose de una exhaustiva fase de documentación acerca de las tecnologías y de su forma de uso. Como en todo trabajo, durante el desarrollo han surgido ciertos problemas relativos a las distintas tecnologías usadas, que han sido superados satisfactoriamente, de forma que todo el desarrollo del trabajo se ha ido cumpliendo tal cual se requería, sin la necesidad que modificar ningún requisito inicial establecido.

Uno de los aspectos que se ha tenido en cuenta a la hora del desarrollo, ha sido realizar el trabajo de forma escalable, estructurado y dividido en módulos organizados, de forma que la implementación de futuros cambios se pudieran realizar lo más fácilmente posible. Por tanto, el trabajo completado no es un trabajo rígido, sino que es un trabajo flexible que puede ser modificado y adaptado en un futuro para crear nuevas aplicaciones de mayor complejidad, o que puede ser reusado para servir de ayuda a otros trabajos futuros. Sin ir más lejos, la aplicación de reproducción de voz puede ser fácilmente incorporada en multitud de trabajos futuros que necesiten de tal funcionalidad.

Por tanto, todo el proceso de desarrollo de este proyecto ha sido enriquecedor y ha servido como una gran experiencia de cómo llevar a cabo un proyecto en su totalidad en el que se han podido aplicar los conocimientos adquiridos. Finalmente, no se puede saber el futuro que tendrá el campo en el cual se desarrolla este proyecto, la robótica telepresencial, ni si este tipo de robot evolucionará en la previsión que marca este documento, el cuidado futuro de la población anciana, ni tampoco si este trabajo formará parte de trabajos posteriores. Lo que sí parece claro es que la población europea está envejeciendo, y que nosotros como parte de la población envejecemos con ella, y que quizás este tipo de robot sea quién nos proporcione asistencia en un futuro más cercano de lo que puede parecer.

A continuación se describen un conjunto de mejoras que se podrían desarrollar para conseguir incrementar las funcionalidades que este trabajo ofrece. Incluso la primera de ellas, la reproducción de eventos de forma automática, ya se ha llevado a cabo sobre el resultado de este trabajo de fin de grado.

- **Reproducción de eventos de manera automática:** Esta es una mejora posible que ya ha sido llevada a cabo por el equipo desarrollador del robot. Ya que en este proyecto solo se podían reproducir eventos mediante solicitud del usuario por voz, era interesante que el robot Jirafa pudiera automáticamente chequear las horas de los eventos, y autónomamente buscar al usuario para reproducirle los eventos llegada la hora concreta en que estos empiezan. Con esta mejora ya desarrollada, se reduce la necesidad de la telepresencia y se reduce también la necesidad de interacción previa para el recordatorio de eventos o citas haciendo más independiente, autónomo y real el comportamiento del robot Jirafa.

- **Interacción robot-usuario para eliminar eventos:** En este trabajo, solo se pueden añadir eventos al calendario o leerlos del mismo, pero mediante una interacción de la persona y el robot no se pueden borrar los eventos existentes. Se podría modificar la aplicación de reconocimiento desarrollada, con el fin de obtener mediante una interacción robot-usuario los datos necesarios para enviarlos a una aplicación de calendario ampliada y eliminar un evento existente en el calendario. Esta mejora extendería los servicios de calendario y de voz, en su variante de reconocimiento y otorgaría al robot una nueva funcionalidad relativa al uso del calendario.

- **Interacción más inteligente robot-usuario:** Este trabajo de fin de grado permite al usuario mantener una conversación con el robot Jirafa, la forma de esta interacción es preestablecida y mecánica, solo posible siguiendo una serie de patrones que el robot reconoce. Aunque la complejidad sería elevada, se podría dotar de cierta inteligencia al robot para que no necesitara de patrones preestablecidos para reconocer lo que el usuario le dice en cada momento, de forma que a partir de una frase reconocida pudiera extraer el contexto y actuar en consecuencia. Esto dotaría al robot de una mayor facilidad y naturalidad a la hora tanto de pedir eventos como de reconocer eventos mediante una interacción con el usuario.

- **Reproducción de los eventos de cualquier día:** En este trabajo, solo se pueden reproducir eventos siguiendo una serie de patrones establecidos, mediante los cuales se pueden reproducir los eventos de la semana, de hoy, de mañana, de hoy y mañana o todos los eventos. Aunque estas solicitudes cubren todos los eventos existentes, la flexibilidad para su solicitud podría ampliarse dotando al robot de la capacidad de reconocer mediante voz

cualquier fecha solicitada por el usuario, permitiendo que el usuario pudiera requerir los eventos de cualquier día concreto.

Debido a las características de este proyecto, y al ser la robótica telepresencial un campo relativamente nuevo, son bastante numerosas las posibles líneas futuras a seguir para ampliar las capacidades actuales del robot Jirafa y potenciar la utilidad del robot en las tareas para las cuales ha sido desarrollado, el cuidado de las personas mayores. A continuación se van a mostrar unas ideas sobre posibles líneas que se podrían seguir en un futuro cercano.

- **Aplicación lúdica-cognitiva:** Los objetivos del desarrollo del robot Jirafa no son únicamente colaborar en las tareas cotidianas de las personas mayores, sino también poder hacer compañía y mitigar la soledad de este conjunto de personas. Por este motivo sería interesante dotar al robot Jirafa de una aplicación de entretenimiento amena, que cada cierto tiempo plantee un acertijo o un pequeño juego al usuario, para que éste pueda ejercitar su memoria. De esta forma, el robot Jirafa no solo ofrecería funcionalidades sobre tareas domésticas del día a día, sino que mantendría al usuario distraído y en cierto modo acompañado.

- **Integración de servicios estándar de comunicación:** Como una forma de potenciar la funcionalidad de asistencia del robot, se le podría dotar de una aplicación que permitiera al usuario de forma sencilla mediante voz acceder a una serie de programas de comunicación comúnmente usados hoy en día como lo es por ejemplo Skype. De esta forma el usuario con una simple palabra podría gracias al robot Jirafa establecer una sesión de video, lo cual permitiría que el robot sirviera también al usuario como un medio de comunicación.

# Referencias Bibliográficas

[1] J. Blanco, C. Galindo, J. G. Monroy, and J. Gonzalez-Jimenez. Open Mobile Robot Architecture (OpenMORA) [en línea]. [Fecha de consulta: marzo 2014] Disponible en: http://www.mapir.isa.uma.es/openmora

[2] Google Inc. Manifest File Format [en línea]. [Fecha de consulta: octubre 2014]. Disponible en: https://developer.chrome.com/extensions/manifest

[3] Scott Clark. Creating a Manifest for a Chrome Store Web App [en línea]. [Fecha de consulta: octubre 2014]. Disponible en: http://www.htmlgoodies.com/beyond/webmaster/toolbox/article.php/3900531

[4] Google Inc. Chrome.permissions [en línea]. [Fecha de consulta: octubre 2014]. Disponible en: https://developer.chrome.com/apps/permissions

[5] Google Inc. Declare Permissions [en línea]. [Fecha de consulta: octubre 2014]. Disponible en: https://developer.chrome.com/apps/declare_permissions

[6] Iván Lasso. Cómo hacer aplicaciones web para Chrome [en línea]. Actualizada 7 agosto 2010. [Fecha de consulta: octubre 2014]. Disponible en: http://www.genbeta.com/paso-a-paso/como-hacer-aplicaciones-web-para-chrome

[7] Google Inc. Chrome.tts [en línea]. [Fecha de consulta: octubre 2014]. Disponible en: https://developer.chrome.com/apps/tts

[8] Glen Shires y Hans Wennborg. Web Speech API Specification [en linea]. Actualizada: 19 octubre 2012. [Fecha de consulta: enero 2015]. Disponible en: https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html

[9] How to use the Web Speech API [en línea]. Actualizada: 2 mayo 2013. [Fecha de consulta: enero 2015]. Disponible en: http://stiltsoft.com/blog/2013/05/google-chrome-how-to-use-the-web-speech-api/

[10] Aurelio De Rosa. Introducing the Web Speech API [en línea]. Actualizado: 14 febrero 2014. [Fecha de consulta: enero 2015]. Disponible en: http://www.sitepoint.com/introducing-web-speech-api/

[11] Paho – Open source messaging for M2M [en línea]. [Fecha de consulta: noviembre 2014]. Disponible en: http://www.eclipse.org/paho/

[12] Google Inc. Google Calendar API [en línea]. Actualizada 18 febrero 2015. [Fecha de consulta: diciembre 2014]. Disponible en: https://developers.google.com/google-apps/calendar/?hl=es

[13] Google Inc. Google APIs Client Library for JavaScript (Beta) [en línea]. Actualizada 20 febrero 2015. [Fecha de consulta: diciembre 2014]. Disponible en: https://developers.google.com/api-client-library/javascript/start/start-js?hl=es#registeryourapp

[14] Google Inc. Google Developers Console Help [en línea]. Actualizada: 1 abril 2015. [Fecha de consulta: diciembre 2014]. Disponible en: https://developers.google.com/console/help/new/#usingkeys

[15] Google Inc. Google Identity Platform [en línea]. Actualizada: 7 abril 2014. [Fecha de consulta: enero 2015]. Disponible en: https://developers.google.com/identity/protocols/OAuth2

[16] IETF. Date and time on the Internet: timestamps [en línea]. Actualizada: julio 2002. [Fecha de consulta: enero 2015]. Disponible es: https://www.ietf.org/rfc/rfc3339.txt

[17] HiveMQ Enterprise MQTT [en línea]. [Fecha de consulta: noviembre 2014]. Disponible en: http://www.hivemq.com/

[18] Mosquitto – An open sourcer MQTT Broker [en línea]. Actualizada: 18 febrero 2015. [Fecha de consulta: noviembre 2014]. Disponible en: http://mosquitto.org/

[19] Como instalar y configurar el servidor web Apache en Windows [en línea]. [Fecha de consulta: enero 2015]. Disponible en: http://norfipc.com/internet/instalar-servidor-apache.html

[20] Google Inc. Google APIs Explorer [en línea]. [Fecha de consulta: diciembre 2014]. Disponible en: https://developers.google.com/apis-explorer/#p/

[21] Welcome to the MQTT community wiki [en línea]. Actualizada 15 septiembre 2014. [Fecha de consulta: noviembre 2014]. Disponible en: https://github.com/mqtt/mqtt.github.io/wiki

[22] C. Galindo, J. Gonzalez, y J.A. Fernandez-Madrigal. 2005. An architecture for cognitive human–robot integration. Application to rehabilitation robotic. *IEEE Int, volumen 1*, páginas 329-334.

[23] J. L. Blanco. The mobile robot programming toolkit (MRPT) [en línea], [Fecha de consulta: abril 2015]. Disponible en: http://www.mrpt.org

[24] P. M. Newman. Moos - a mission oriented operating suite. 2003 MIT Dept. of Ocean Engineering, Tech. Rep. OE2003-07.

# Anexo

A continuación se presenta el material adicional que se ha generado durante el desarrollo del trabajo y también el generado específicamente para complementar esta memoria.

## Anexo A. Material gráfico

El Anexo A incluye el material gráfico adicional. Comprende el material creado para la interfaz gráfica de las aplicaciones, en este caso imágenes y GIFs, que dotan el robot de una interfaz más amigable y cercana. También comprende una serie de imágenes que grafican los diferentes casos de uso. Todo este material gráfico ha sido desarrollado mediante la herramienta de edición de imágenes GIMP.

### Interfaz Gráfica

En este apartado se muestran las diferentes imágenes que van a formar los GIF que aparecen en la interfaz gráfica del robot, así como otras imágenes que también aparecen en la interfaz gráfica sin formar ningún GIF.

- **Estado normal del robot**

  Estas son las dos imágenes principales que componen el GIF en la interfaz del robot, pues reproducen los ojos abiertos y el parpadeo (ver figura A.1).

Figura A.1. Izquierda) Figura de la interfaz del robot con los ojos abiertos. Derecha) Figura de la interfaz del robot con los ojos cerrados durante un parpadeo.

- **Robot aburrido**

  Por otro lado, estas otras tres imágenes forman parte secundaria del GIF de la interfaz del robot, aparecen por unos instantes cuando ninguna acción se ha realizado en el robot en un periodo de tiempo y quieren aparentar aburrimiento en el robot. (Ver figura A.2).

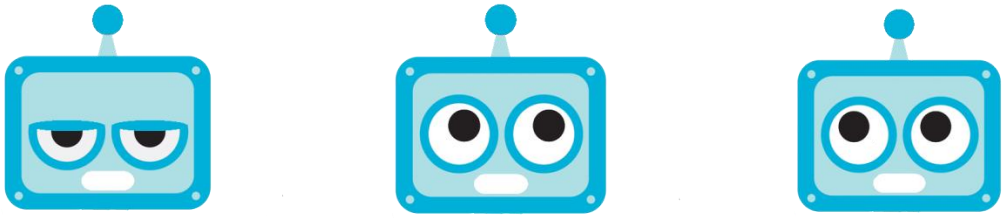

Figura A.2. Izquierda) Figura de la interfaz del robot con los ojos entreabiertos. Centro) Figura de la interfaz del robot con los ojos hacia arriba a la derecha. Derecha) Figura de la interfaz del robot con los ojos hacia arriba a la izquierda.

- **Error de conexión en el robot**

Si se produce un error en la conexión MQTT entre la aplicación y el servidor MQTT, el error se muestra en la interfaz gráfica del robot mediante una imagen con una “x” como ojos. (Ver figura A.3).

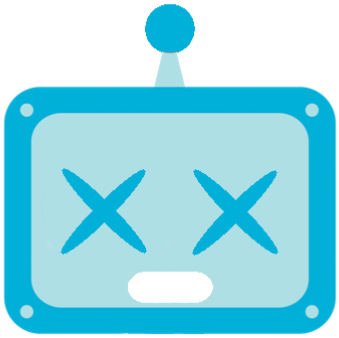

Figura A.3. Figura de la interfaz del robot que representa una pérdida de conexión de las aplicaciones.

- **Reproducción de un mensaje de voz por el robot**

Si se está usando la aplicación de reproducción de voz, TTS, y por lo tanto se está reproduciendo un mensaje, durante esa reproducción un GIF formado por imágenes como la siguiente (ver figura A.4) y similares a ella aparecen en la interfaz del robot.

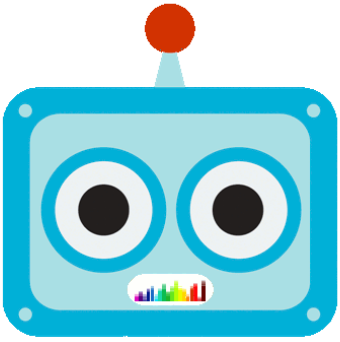

Figura A.4. Figura de la interfaz de robot que forma parte de un GIF que se inicia cuando la aplicación TTS reproduce una cadena mediante voz.

## Casos de uso

A continuación se muestran las imágenes que representan cada uno de los casos de usos y sus variantes que el robot puede realizar debido a la implantación de las aplicaciones. (Ver casos de uso).

### Insertar evento

- Insertar un evento por el usuario (ver figura A.5).

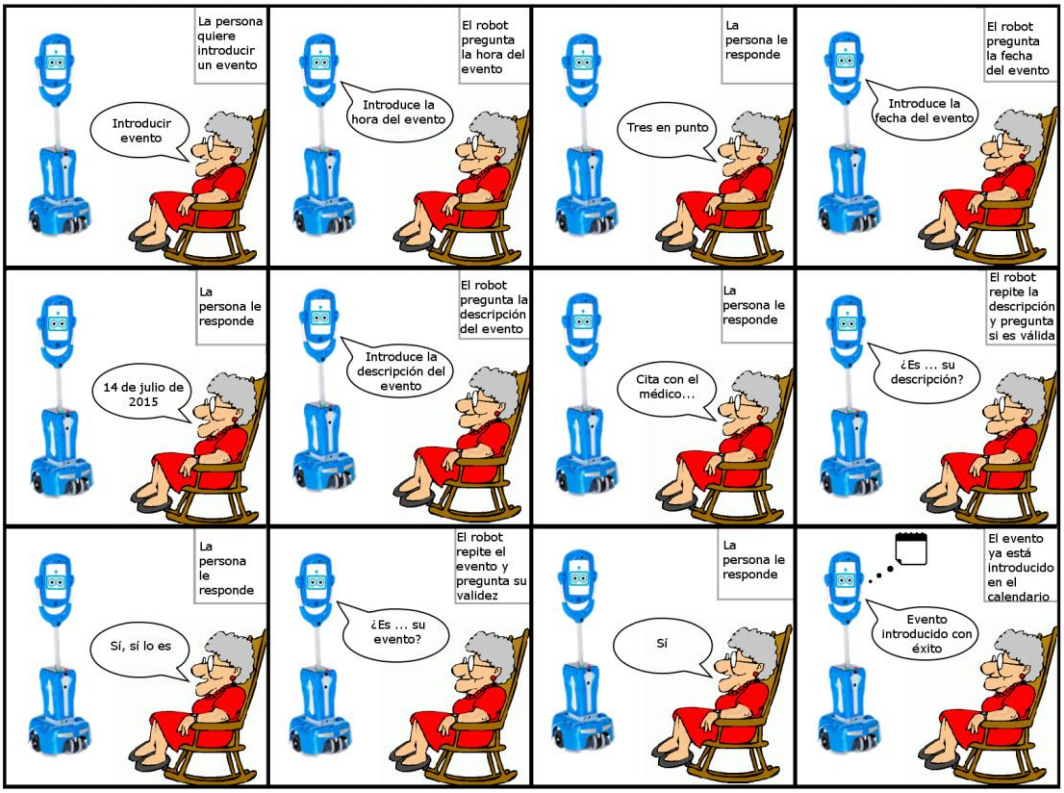


Figura A.5. Figura que representa la interacción entre el robot y el usuario y los pasos a seguir para la inserción de un evento en el calendario.

- Insertar un evento por un tercero (ver figura A.6).


Figura A.6. Figura que representa los pasos a seguir por un tercero para la inserción de un evento.

**Reproducir evento**

- Reproducir los eventos solicitados por el usuario (ver figura A.7).

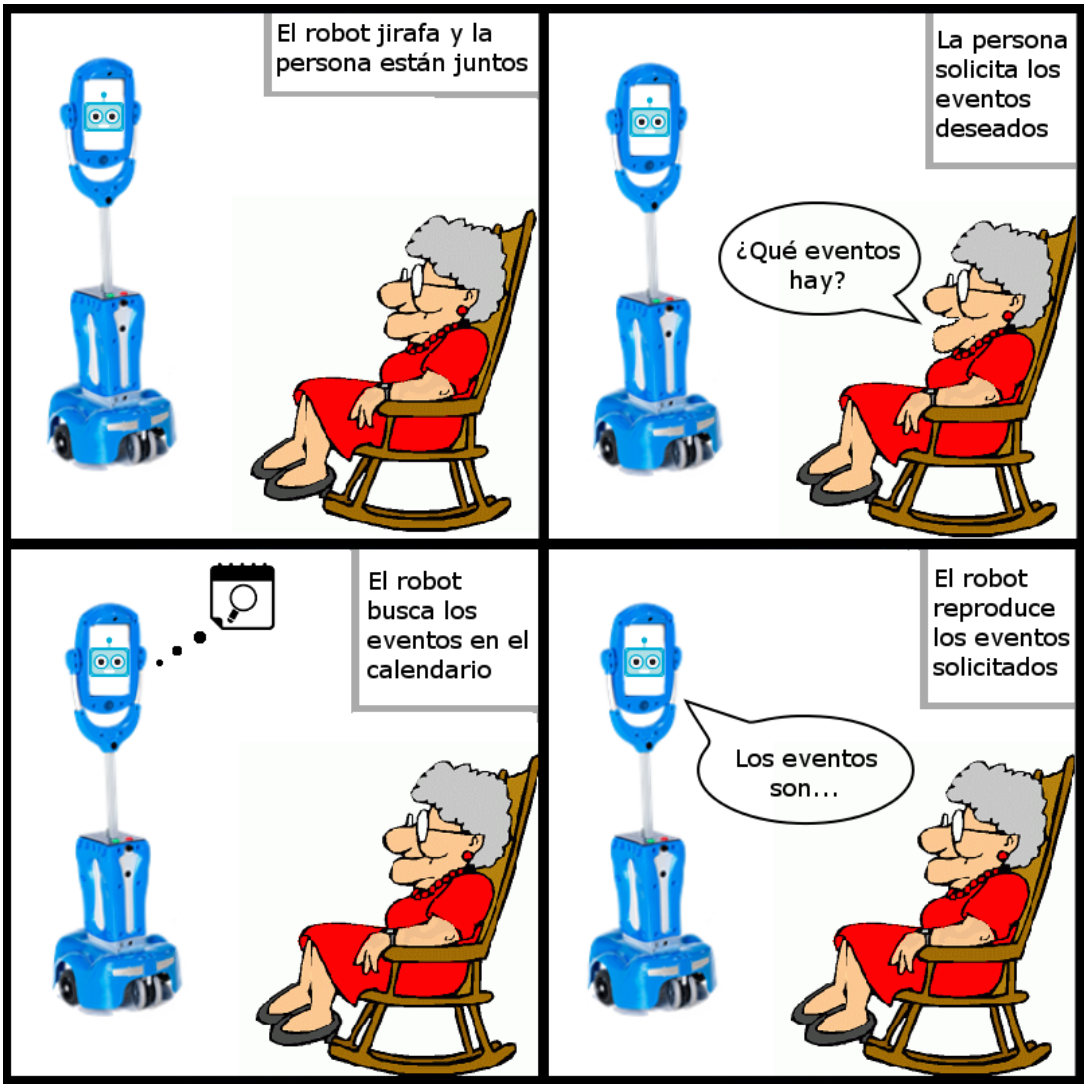


Figura A.7. Figura que representa la interacción entre el robot y el usuario junto con los pasos a seguir para la reproducción de un evento del calendario.

Existen diversas modalidades de solicitud de reproducción de un evento por parte del usuario (reproducción de todos los eventos, de los eventos de hoy, de los eventos de mañana, de los eventos de hoy y mañana o de los eventos de la semana). Los pasos para solicitar cada modalidad y la reproducción de los eventos por parte del robot son los mismos alterando únicamente el patrón de solicitud, por tanto se muestra solo una imagen genérica para la reproducción de eventos por el robot.

- Reproducir evento automáticamente (ver figura A.8).

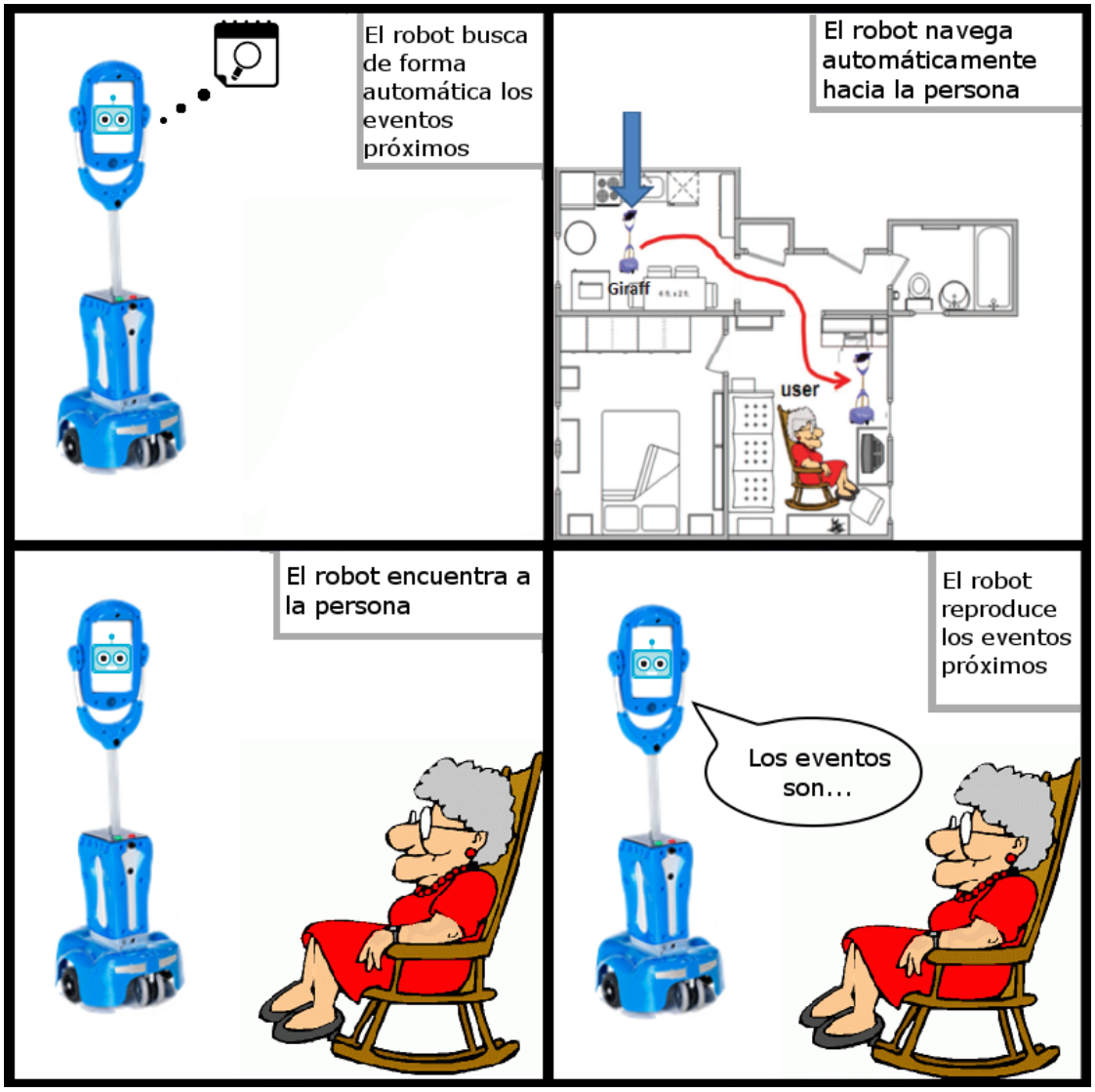


Figura A.8. Figura que representa la interacción entre el robot y el usuario para la reproducción de un evento del calendario de forma automática.

# Anexo B. Video

El Anexo B incluye un guion detallando el desarrollo de un video que tiene como objetivo mostrar de forma real la ejecución y uso de las aplicaciones desarrolladas en funcionamiento en el robot Jirafa interactuando con un usuario.

- En la primera escena se muestra en funcionamiento una de las mejoras llevadas a cabo por el equipo desarrollador del robot, la reproducción de un evento de forma automática. El robot Jirafa que se encuentra estacionado en su estación de carga, monitoriza periódicamente la existencia de eventos

venideros, y al encontrar un evento próximo activa la navegación hasta el usuario. (Ver figura A.9).

Figura A.9. Arriba) El robot Jirafa ha encontrado un evento en el calendario y ha comenzado la navegación hacia el usuario. Abajo) El robot Jirafa continúa su navegación hacia el usuario.

- En la segunda escena, se muestra cómo el robot Jirafa localiza al usuario y reproduce el evento que anteriormente había leído como próximo. Una vez terminado esto, el usuario le pregunta por los eventos existentes en el día de mañana y el robot Jirafa los reproduce. En esta escena se muestran en acción todas las aplicaciones realizadas, el reconocimiento, la reproducción y la función de calendario, así como también la mejora nombrada en la escena anterior. (Ver figura A.10).


Figura A.10. El robot Jirafa está reproduciendo automáticamente al usuario los eventos cercanos.

- En la tercera escena se puede observar cómo el usuario solicita la introducción de un evento en el calendario y el robot Jirafa pregunta por los detalles del evento a introducir, produciéndose así una interacción entre ambos. (Ver figura A.11).


Figura A.11. El usuario interactúa con el robot Jirafa para introducir un evento.

- Finalmente en la última escena se puede observar cómo el evento introducido por el usuario aparece en la interfaz web de Google Calendar. (Ver figura A.12).

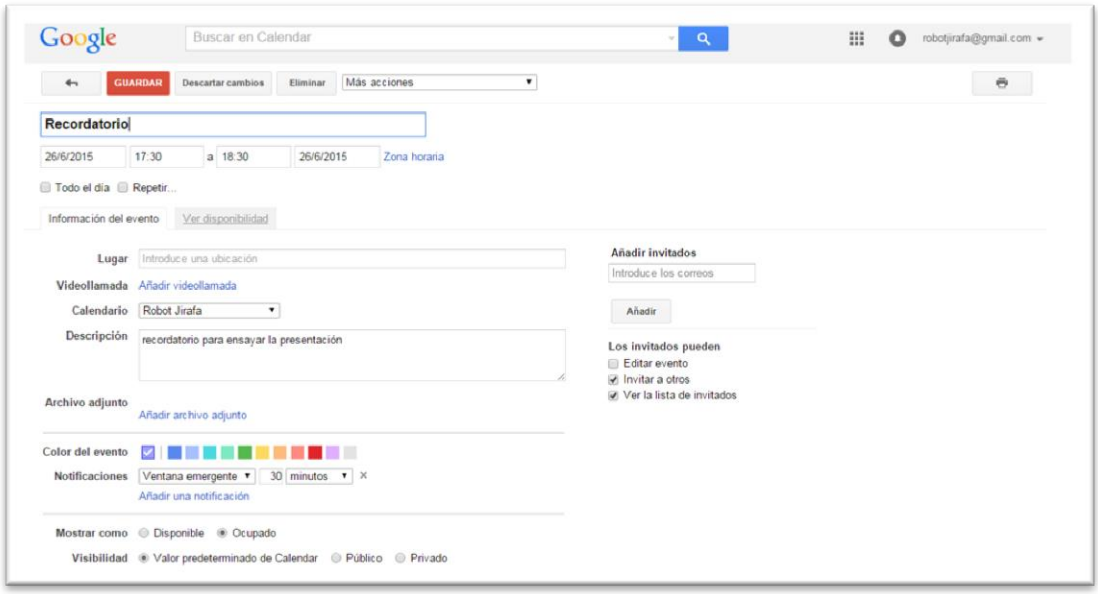


Figura A.12. Figura que muestra el nuevo evento con la descripción insertada por el usuario.

Anteriormente se han mostrado algunas imágenes del video para complementar el guion, el video completo se encuentra en la siguiente ubicación:

https://www.dropbox.com/s/byiuztcwjnl4dwx/Video%20_TFG.mp4?dl=0

# Anexo C. Patrones de reconocimiento

Se presentan los patrones que la aplicación de reconocimiento de voz tiene incluidos, estos patrones pueden ser reconocidos por el robot Jirafa una vez el usuario los repita mediante voz. Se enumeran tanto los patrones de voz para la inserción de un evento en el calendario, como los patrones de voz para la reproducción de un evento en sus diferentes variantes. Es reseñable que estos patrones se pueden modificar, eliminar o aumentar en número, para ello se tiene que modificar el documento que los contiene, adjunto con la memoria.

**Patrones para la inserción de un evento**

- “Introducir evento"
- "Insertar evento"

**Patrones para la reproducción de todos los eventos**

- "Dime todos los eventos"
- "Quiero saber todos los eventos"
- "Quiero todos los eventos"
- "Qué eventos hay"

- "Cuáles son los eventos"

**Patrones para la reproducción de los eventos de la semana**

- "Dime los eventos de la semana"
- "Quiero saber los eventos de la semana"
- "Quiero los eventos de la semana"
- "Qué eventos hay esta semana"
- "Cuáles son los eventos de la semana"

**Patrones para la reproducción de los eventos de hoy y mañana**

- "Dime los eventos de hoy y mañana"
- "Quiero saber los eventos de hoy y mañana"
- "Quiero los eventos de hoy y mañana"
- "Qué eventos hay hoy y mañana"
- "Cuáles son los eventos de hoy y mañana"

**Patrones para la reproducción de los eventos de hoy**

- "Dime los eventos de hoy"
- "Quiero saber los eventos de hoy"
- "Quiero los eventos de hoy"
- "Qué eventos hay hoy"
- "Cuáles son los eventos de hoy"

**Patrones para la reproducción de los eventos de mañana**

- "Dime los eventos de mañana"
- "Quiero saber los eventos de mañana"
- "Quiero los eventos de mañana"
- "Qué eventos hay mañana"
- "Cuáles son los eventos de mañana"

**Patrones de afirmación o negación**

- 'Si'
- 'Sí sí lo es'
- 'Sí lo es'

- 'No'
- 'No no lo es'
- 'No lo es'

**Patrones de reconocimiento de algunas horas en punto**

- "12 de la noche"
- "Una de la noche"
- "2 de la noche"
- "Tres de la noche"
- "4 de la noche"
- "5 de la noche"
- "6 de la mañana"
- "7 de la mañana"
- "8 de la mañana"
- "9 de la mañana"
- "10 de la mañana"
- "11 de la mañana"
- "12 de la mañana"
- "Una de la tarde"
- "Dos de la tarde"
- "3 de la tarde"
- "4 de la tarde"
- "Cinco de la tarde"
- "6 de la tarde"
- "7 de la tarde"
- "8 de la tarde"
- "9 de la noche"
- "10 de la noche"
- "Once de la noche"