UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de una aplicación para la Web utilizando el Web Speech API

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Antonio Roig Boronat

Tutor: Manuel Agustí Melchor

Curso: 2018/2019

Agradecimientos

A Manolo Agustí Melchor tutor de este proyecto, por descubrirme las funcionalidades del reconocimiento de voz y ayudarme todo lo posible en este proyecto, a los profesores que he tenido a lo largo de mis estudios y a la Escola Tècnica Superior d'Enginyeria Informàtica.

Gracias.

Resumen

En este trabajo, orientado al estudio de los sistemas de entrada y salida mediante el uso de la voz, que, aunque son habituales las aplicaciones que utilizan la voz como medio de entrada y salida en la actualidad, la mayor parte de las soluciones aplicadas son propietarias y cerradas. Desde el entorno de las aplicaciones web está emergiendo una propuesta que puede hacer cambiar esta situación que estudiaremos con más detalle. En este caso, vamos a explorar la librería *Web Speech API* de código abierto, que pretende ser la base de las pautas a seguir para desarrollar este tipo de aplicaciones de reconocimiento, es decir la entrada de audio y la síntesis de la voz, salida del audio.

Este proyecto aborda la realización de una aplicación que a modo de un asistente personal nos permita la realización de diferentes funciones visuales mediante el uso de la voz. Con este objetivo se abordarán tanto los mecanismos que hacen posible el reconocimiento de voz, como la síntesis, utilizando los eventos y funciones de esta API.

Palabras clave: JavaScript, HTML, CSS, API, Reconocimiento de la voz, Síntesis de la voz.

Abstract

This project is oriented to the studies of voice input and output systems in different applications, which amount of these applications that use voice as a means of input and output are common in actuality, most of these applications are private and closed. From the environment of web applications is emerging a proposal that can change this situation and we will study in more detail. In this case we are going to explore about the library that intends to be the base of the guidelines to follow to develop this type of applications for voice recognition, possible the audio input, as the synthesis, audio output.

This project deals with the development of an application that as a personal assistant receives voice commands and returns some preconfigured actions as a test of this API methods, which will address both the mechanism that make voice recognition and the synthesis, using the events and functions of this API.

Keywords: JavaScript, HTML, CSS, API, Speech Recognition, Speech Synthesis.

Resum

En aquest treball, orientat a l'estudi dels sistemes d'entrada i eixida mitjançant la veu en diferents aplicacions, encara que hui en dia son habituals les aplicacions que utilitzen la veu com a mitjà d'entrada i eixida, la major part de les solucions aplicades son de propietari i tancades. En l'entorn de les aplicacions web està sorgint una proposta que pot fer que canvií aquesta situació i que l'estudiarem amb mes detall. En aquest cas anem a explorar sobre la llibreria que pretén d'alguna forma ser la base de les pautes a seguir per desenvolupar aquest tipus d'aplicacions de reconeixement de veu, es a dir l'entrada d'àudio i de síntesis de veu, eixida de veu.

Aquest projecte aborda el desenvolupament d'una aplicació que a mode d'assistent personal, ens permetrà la realització de diferents funcions visuals mitjançant l'ús de la veu. Amb aquest objectiu s'abordaran tan els mecanismes que fan possible el reconeixement de veu com la síntesis de la veu, mitjançant els esdeveniments i funcions d'aquesta API.

Paraules clau: JavaScript, HTML, CSS, API, Reconeixement de la veu, Síntesis de la veu.

Tabla de contenidos

Tabla de figuras

1. Introducción

Hace tiempo que las aplicaciones generan sonido música e imitan y reconocen la voz humana. Las soluciones actuales dependen de plataforma y son, mayormente, soluciones propietarias.

Este trabajo se centra en la búsqueda y experimentación sobre el uso de la voz en plataforma web a través del estándar propuesto por el W3C para la web. Cabe nombrar que se encuentra la Web Audio API [1], que es la interfaz que permite modificar y aplicarle efectos al sonido, pero que no es el objetivo de estudio en nuestro proyecto.

En este aspecto se abordarán los mecanismos que hacen posible tanto la síntesis (salida del audio), como el reconocimiento (entrada del audio) de voz en plataforma Web, basándonos en el uso de este estándar y su API (Interfaz de Programación e Aplicaciones).

1.1. Motivación

Una de las cosas que más me motivaron a elegir este TFG fue sin duda el gran crecimiento de la utilización del reconocimiento de voz en diferentes dispositivos del mercado. Esto se puede constatar en los resultados que se obtienen en estadísticas actuales [2] y que se puede ver en la figura 1, en que se muestran según el rango de edades el uso de estos asistentes y las perspectivas del incremento de la utilización desde el 2016 a 2019.

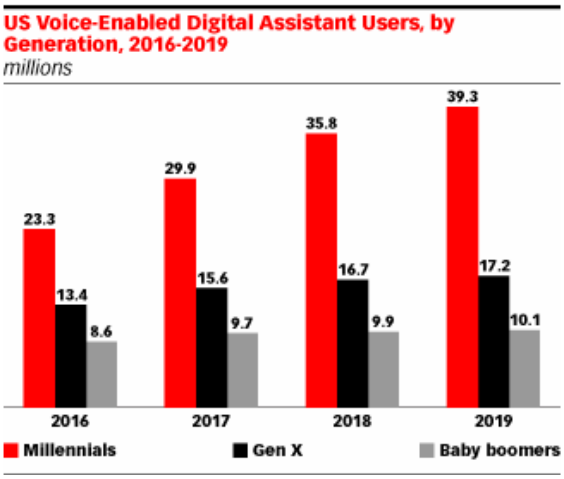

Figura 1: Comparativa uso de asistentes digitales según generaciones.

Estas estadísticas reflejan como según el grupo de edades separadas entre *Millennials* (1981-2000), *Gen X* (1965-1980) y *Baby boomers* (1945-1964). En los casos las estimaciones esperadas para esta gráfica de uso de los sistemas asistentes en Estados

Unidos, observamos que el grupo con mayor uso y con un incremento más grande es el del grupo más joven *Millennials* donde se espera un mayor incremento en la utilización de estos dispositivos.

La figura 2, resume el estudio [3] que lleva a cabo una comparativa de las diferentes opciones del mercado que ofrecen reconocimiento de la voz y de aquí se ha extraído las mejores relaciones, funcionalidades y costes en cada plataforma en la actualidad. Se realiza esta tabla a modo informativo con las características generales de cada dispositivo que destaca en su plataforma.

Plataforma	***Nombre del software***	***Características***	***Coste***
Mejor software para reconocimiento de voz y dictado en la industria.	Dragon NaturallySpeaking	Se puede usar como interfaz principal de su ordenador, transcribe la voz a texto con la precisión más alta del mercado, puede ejecutar programas, acceder a páginas web y buscar en ellas entre otras cosas. Tiene versiones disponibles para Windows y macOS. Disponible en 7 idiomas.	150 a 1000 dólares.
Mejor Software para reconocimiento de voz y dictado basado en Web	Google Docs Voice Typing	Esta herramienta de Google docs además de transcribir a escritura la voz te permite aplicar diferentes funciones como copiar, pegar, seleccionar, utilizando la voz, tiene más de 100 comandos disponibles. Además, está disponible en 7 idiomas diferentes.	Disponible en la web de manera gratuita junto Google docs.
Mejor software de reconocimiento de voz para Android	Google Now	Las funciones De Google Now y Assistant, están integradas al sistema operativo Android. Es capaz de abrir aplicaciones, buscar información entre otras funcionalidades. También puede funcionar en iOS siempre dentro de su funcionalidad de buscador. También soporta unos 43 idiomas.	Disponible de manera gratuita en Android y iOS.
Mejor software de reconocimiento de voz para iOS	Siri	Al configurar Siri en tu dispositivo Apple, tiene unas funciones similares al Google Now, búsquedas por voz, apertura de aplicaciones, sin embargo, el sistema de transcripción online está limitado a 30 segundos. Disponible en 21 idiomas diferentes.	Disponible de manera gratuita en iOS.
Mejor software de reconocimiento	Cortana	Las funciones generales que contiene este asistente personal permiten: enviar texto por voz, abrir	Disponible de manera gratuita en

de voz para Windows		aplicaciones... Soporta unos 7 idiomas.	Windows iOS y Android.

Figura 2: Mejor software según plataforma.

En el futuro en este extenso campo y con un crecimiento rápido, se esperan bastantes cambios en el futuro. Uno de estos incrementos vendrá dado por el aumento de las ventas en los altavoces inteligentes donde, según informes de *voicebot [4]* se espera que el incremento de compras de estos dispositivos augmente significativamente. Pero si hay un campo que empleará estos sistemas en el futuro este es sin duda alguna, el campo de ayudas a la conducción, donde según datos de *voicebot [5],* el número de usuarios que utilizan los asistentes de voz en los vehículos es superior a los altavoces inteligentes. Ya en la actualidad las funciones que se pueden realizar mediante la voz en los automóviles son muy extensas debido a las compatibilidades con Android, IOS entre otros sistemas, que hacen que este campo puede desempeñar un papel importante en el desarrollo de dispositivos de reconocimiento de voz.

Después de realizar un estudio de las diferentes alternativas que hay actualmente para el reconocimiento i síntesis de la voz, se procede a utilizar la API, de código abierto, *Web Speech de* W3C, para así poder incrustar sus funciones de reconocimiento de voz en una aplicación web realizada mediante *HTML, CSS i JavaScript*.

1.2. Objetivos marcados

El primer objetivo de este proyecto es investigar sobre los diferentes sistemas que hay en el mercado. Se pretende hablar de las características principales que nos permiten estos aplicativos y la manera con las que son implementados. En este caso se pretende caracterizar las posibilidades de la *Web Speech API*, frente a los sistemas de reconocimiento y síntesis actuales.

Una vez se investiga la *Web Speech API* se empezarán a hacer pequeñas aplicaciones con funciones básicas para comprobar el funcionamiento tanto de la interfaz de reconocimiento como de la interfaz de síntesis de esta API. Las aplicaciones nos servirán para hacer unas primeras pruebas sobre los métodos más generales de la API de una forma simple.

Como objetivo principal, desarrollaremos una aplicación que, a modo de asistente, interactúe con el usuario a través de la voz como medio principal. Para ello diseñaremos una aplicación mediante HTML, JavaScript y CSS que nos ayude a ejemplificar los métodos de esta API de una manera fácil y visual.

1.3. Metodología empleada

En primer lugar, se fija un objetivo en acuerdo con el tutor de este proyecto, para establecer unas pautas de trabajo a seguir, después de haber debatido el abanico de diferentes propuestas para tratar con el tema.

En segundo lugar, se empieza la búsqueda de información sobre el tema acordado y diferentes propuestas en Internet que podemos utilizar para la realización de este proyecto.

Posteriormente en cuanto se encuentra y se selecciona la información a utilizar se empieza a realizar la primera parte de la memoria donde consiste en explicar el estado del arte del campo elegido, así como la explicación de la API seleccionada.

A continuación, se desarrollan algunas pequeñas aplicaciones que nos permiten la utilización de diferentes métodos de esta API y así nos familiarizamos con la estructura de esta API

Finalmente se desarrolla una aplicación web que permitirá exponer alguna de las funcionalidades más importantes que tiene esta API, tanto la interfaz de reconocimiento como la de síntesis de voz.

1.4. Estructura de la memoria

En esta memoria capítulo 2, se encuentra la información del tema que vamos a desarrollar. Se realiza una explicación de los principales sistemas de las grandes compañías. Aportaremos información sobre los diferentes dispositivos y aplicativos que hay actualmente en el mercado y su desarrollo.

En el capítulo 3, se detalla el porqué de la elección de la API de W3C y sus métodos y características.

En el capítulo 4, en el punto 4.2, se desarrollan dos pequeñas aplicaciones, una para el reconocimiento de la voz y otra para la síntesis, para poder probar la funcionalidad de algunos métodos. Seguidamente en el punto 4.3, se desarrolla un pequeño prototipado de la aplicación final y las características que queremos que tenga.

En el capítulo 5, se pasa a desarrollar la aplicación, que nos servirá para ejemplificar posibles usos de esta API en las aplicaciones web.

En el capítulo 6, se ejecutarán una serie de pruebas para comprobar el correcto funcionamiento y sus limitaciones.

Finalmente, en el capítulo 7 se proporcionar conclusiones sobre el proyecto proporcionando un tema a tratar para trabajos futuros a partir de este proyecto.

2. Estado del arte

Históricamente la palabra hablada se ha utilizado con anterioridad a su contraparte escrita [6], es decir la naturalidad de la voz lleva ya muchos años como elemento de comunicación. A efectos prácticos la voz es más fluida para ejecutar la comunicación comparado con la escritura de palabras donde el volumen de palabras disminuye significativamente. Es a partir del 2008 cuando las grandes compañías (Google, Apple), culminan un proyecto de aprendizaje de varios años con el *Google Voice Search y el Apple Siri* [6].

Con la mejora de la tecnología, tanto en los dispositivos móviles (como la creación de nuevas herramientas, ya sea uso en los teléfonos móviles), han hecho que el uso del reconocimiento de la voz se utilice cada vez en más tareas diarias [4]. También el desarrollo de la domótica ha hecho que ciertos aparatos del hogar requieran de la voz para diferentes funciones, mediante el uso de dispositivos como el *Google Now o Alexa* entre otros [7]. En el campo de la domótica, en la actualidad ya existen dispositivos de diferentes compañías que permiten, mediante la voz, llevar a cabo algunas funciones prácticas en uso diario como consultar el tiempo, las noticias, crear listas de la compra, encender luces, entre otras funcionalidades.

Diferentes compañías han entrado fuertes en este apartado tecnológico creando sus propias herramientas y con esto creando un código propio hecho a medida para sus herramientas [8].

2.1. Herramientas y plataformas

En este punto vamos a nombrar y explicar de manera general algunas de las diferentes herramientas que hay actualmente, para el uso de la voz, es decir, reconocimiento de voz y síntesis de voz. Se van a ver soluciones para diferentes plataformas: web, escritorio, móvil y aplicativos domésticos.

2.1.1 Uso de la voz en los computadores

Desde la aparición del primer ordenador personal, siempre en mayor o menor medida se ha tenido en cuenta el aspecto del reconocimiento de voz. En sus principios esta tecnología se empezó a desarrollar mediante el propósito de poder mantener una conversación simple con la computadora, diferentes científicos experimentaron con la tecnología, como Audrey que fue capaz de crear una máquina que por primera vez podía interpretar ciertas cadenas no solo grabarlas como en anteriores experimentos [6].

Con la aparición de los dispositivos móviles cambió y en este campo, en concreto el escritorio web, se han realizado diferentes aplicaciones para diferentes plataformas que usan el reconocimiento de voz [8].

2.1.1.1 Cortana

Cortana es un sistema de reconocimiento propio de *Microsoft*. Este sistema en su versión de escritorio se encuentra como un buscador respaldado por el motor de búsqueda *Bing* y por el navegador *Microsoft Edge* [9]. Como características principales de esta aplicación encontramos que se puede buscar mediante reconocimiento de voz preguntas básicas en diferentes idiomas o abrir aplicaciones del sistema mediante la voz. En este caso nos vamos a centrar en las características del software que se utiliza para ofrecer el reconocimiento de la voz.

Para las versiones con sistemas operativo Windows, tanto en la versión escritorio como en la versión del móvil, Cortana utiliza el *Azure Node SDK* [10] de voz. El *SDK* proporciona a la aplicación acceso a la conversión de la voz en texto, así como la traducción y el análisis de la intención que se transmite. Para las versiones de escritorio se puede utilizar C# (.NET Standard), donde tiene una librería de consulta generalizada de métodos disponibles, *Microsoft.CognitiveServices.Speech* [11]. Para la versión de navegador se utiliza C#, Python o JavaScript, que utiliza esta librería propia de Azure, *Speech SDK de Cognitive Services*. Esta librería pretende simplificar el desarrollo de aplicaciones habilitadas para voz.

Como ventajas de esta aplicación para las versiones de escritorio, web y dispositivos móviles se pretende que la funcionalidad sea compatible con los tres campos. Esto siempre será posible cuando se ejecute mediante el motor Bing y Microsoft Edge que según Microsoft es un requerimiento obligatorio a excepción de versiones beta con otros navegadores creados por desarrolladores externos a la empresa.

Como desventajas podemos observar que, según el modo, es decir en modo interactivo y dictado, *Microsoft Speech Recognition API* admite un total de 31 idiomas disponibles. Por el contrario, en el modo conversación *Microsoft Speech Recognition API* solo admite 5 idiomas más generales. Al igual que lo dicho en el párrafo anterior solo deja usar Cortana con Bing y Microsoft Edge.

2.1.1.2 Festival

Este sistema desarrollado por el CSTR (centro de investigación tecnológico del habla) de la universidad de Edimburgo, es capaz de construir sistemas de síntesis de voz, ofreciendo la transformación de texto a voz. Utiliza varios compiladores basados en C++ [12]. Tiene una versión en escritorio con dos lenguajes en inglés (británico y americano) y español, el más avanzando hasta el momento es el inglés. También hay otros desarrolladores que están creando nuevos lenguajes para el sistema, para esto está disponible un programa llamado *FestVox*.

El sistema está implementado en C++ utilizando la librería *Edinburgh Speech Tools* para la arquitectura al más bajo nivel, luego tiene un intérprete de comandos basados en el esquema SIOD, de código libre.

Tiene una versión demo en plataforma web [13] donde puedes elegir diversas voces en inglés, y puedes escribir un máximo de 70 caracteres, se puede reproducir y descargar en formato .wav. Por lo que se observa en las pruebas en la versión 70.0.3538.77 de Google Chrome funciona correctamente, aunque no es dependiente de la versión, ya que se ejecuta en el servidor, al no utilizar una interfaz (API) dependiente de la compatibilidad de diferentes navegadores.

2.1.1.3 CMUSphinx

CMUSphinx, es el nombre que emplea el departamento de sistemas de reconocimiento de la Universidad de Carnegie Mellon para este sistema. Tiene soporte para diferentes idiomas entre ellos, inglés de EE.UU. como también el del Reino Unido, francés, mandarín, alemán, español, holandés y ruso entre otros. Además, posee la capacidad para desarrollar reconocimiento de voz en nuevos idiomas.

Las herramientas de CMUSphinix están diseñadas específicamente para plataformas de bajo coste, bajo licencia BSD, y está compuesta por una amplia gama de herramientas relacionadas con el reconocimiento de la voz:

- *Sphinix*, es un sistema de reconocimiento de la voz continua que utiliza modelos ocultos de Márkov. A lo largo de los años se ha ido mejorando la versión inicial de Kai Fu-Lee. La versión más reciente, que es la 4, utiliza una librería en lenguaje de programación Java [14], esto contribuye a que pueda ser utilizado en una amplia gama de sistemas operativos y hardware. Al utilizar la librería en Java el uso de *sphinx4* se puede integrar fácilmente mediante el uso de *.jar* en los proyectos creados.
  En la versión probada en Ubuntu 18.04.01 comprobamos que reconoce frases de forma continua hasta que cortamos la voz y devuelve el resultado por la consola.

- *Pocketsphinx*: Es una de las grandes librerías de código abierto que tiene *CMUSphinix*. Este, a diferencia de *Sphinix,* es una versión que está más enfocada al uso de sistemas embebidos. Tiene versiones Unix, MacOS versión para IPhone, Android y Windows [15].

- *Sphinixtrain*: Esta herramienta es utilizada para construir los modelos de reconocimiento para *Sphinix*, disponible en versión Unix y Windows.

- *Sphinixbase*: Este es el paquete que contiene las librerías básicas compartidas por *Sphinix* y *Pocketsphinix*, así como también permite manipular características acústicas y archivos de audio.

2.1.1.4 *Dialogflow*

Es una herramienta [16] propiedad de la compañía Google, que nos permite crear *chatbots* con reconocimiento del lenguaje. Esta herramienta permite crear interfaces de conversaciones de voz, además de texto, basados en la inteligencia artificial. Cabe destacar que permite conectarlo a diferentes plataformas como *"Google Home", "Amazon Alexa"* entre otras.

Permite establecer varias respuestas de forma natural para el reconocimiento ya sea de la voz o por escrito, permitiendo que los *chatbots* sean más naturales cuando se interactúa con el cliente [16].

El atractivo de esta herramienta es que puede utilizar tanto la voz para recibir la información a ser tratada como la saliente. Basando su pilar de programación en Node.js, "Dialogflow" en el caso de enviar una respuesta por voz utiliza la API de Google "Cloud Text-to-Speech" con soporte para diferentes idiomas, además tiene un abanico amplio de librerías en diferentes lenguajes de programación, como C#, Java, Node.js, Python, Ruby entre otros [17].

2.1.1.5 *Annyang*

Annyang es una librería en *JavaScript* [18], permitiendo así crear aplicaciones web que nos permiten controlarlas por la voz. Esta herramienta es muy práctica en el caso de las páginas web a la hora de rellenar formularios que pueden ser engorrosos.

Este SDK, tiene una programación bastante sencilla, se introducen los comandos que se necesite la página web y directamente la web se mantiene en escucha activa para reconocer los datos que se requieran. Es compatible con varios idiomas diferentes y, por lo que cuenta en el tema de la compatibilidad en navegadores, por ahora tiene un buen funcionamiento en Chrome y Opera.

Esta librería se puede usar junto a Speech KITT, que es una interfaz gráfica para el navegador que permite al usuario iniciar o detener el reconocimiento de voz, como también ver el estado actual. Además, puede proporcionar sugerencias como instrucciones y comandos de muestra [19].

A parte de esta opción que hemos utilizado de ejemplo de librerías, para configurar nuestra aplicación de manera sencilla para introducir comandos mediante la voz, tiene alternativas entre ellas *Voix, Voice-Commands y Julius*, todas estas librerías también están escritas en JavaScript [20].

2.1.2 Uso de la voz en los móviles y domótica.

Los dispositivos móviles de la actualidad han evolucionado a un ritmo vertiginoso, en la última década, el reconocimiento de voz en estos también ha dado un salto cualitativo

enorme. En la figura 3 [21] podemos observar el porcentaje en la actualidad, de funciones por voz utilizadas para diferentes aplicaciones, donde las más predominantes son las de marcación de una llamada, escribir un mensaje o encontrar una dirección.

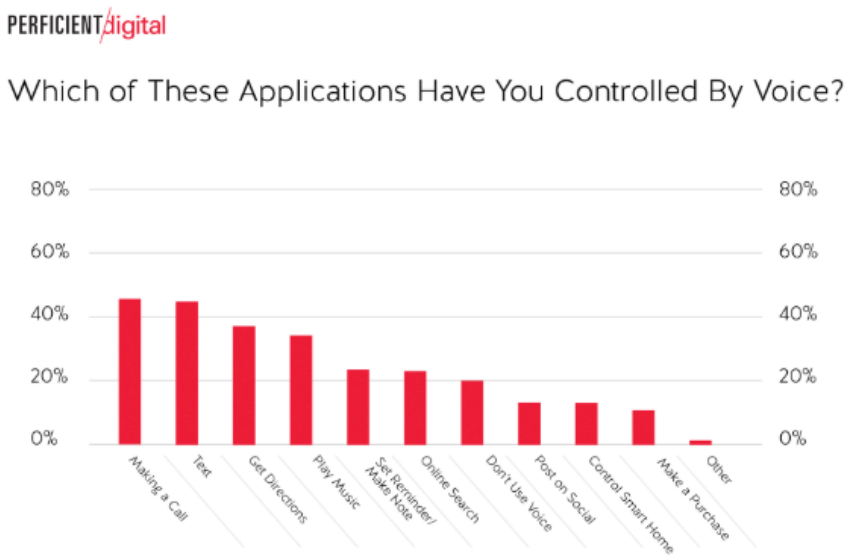

Figura 3: Estadísticas de porcentaje de acciones realizadas mediante la voz.

En este apartado vamos a hablar de las funciones en las que se emplea la voz y de cómo está implementado en los dispositivos de las grandes empresas del mercado, individualmente en móviles y últimamente en asistentes en el hogar.

2.1.2.1 Google Now /Google Assistant

Google lleva siendo desde hace varios años una de las principales empresas que más ha mejorado sus sistemas para implementar el reconocimiento de voz, y aquí vamos a explicar de forma general el funcionamiento y las características principales de algunos de ellos.

Google Now es el asistente personal inteligente desarrollado por la compañía Google. Este asistente esta incrustado dentro de la aplicación, también propia de la compañía, *Google Search*.

Este asistente está basado en una interfaz que interpreta el lenguaje natural para así proporcionar respuestas al usuario. Entre sus características más destacables estaría la búsqueda mediante el navegador *Google Chrome*, que también está disponible para la plataforma de escritorio, también mediante el histórico de búsquedas del cliente proporciona un predictor de búsqueda. Este asistente a parte de búsqueda también proporciona interacciones con los contactos, entre otras funciones.

En el 2016 Google lanzó *Google Assistant* [22], que es una mejora del *Google Now* ya que permite interactuar con el usuario. Este asistente permite la interacción mediante conversaciones bidireccionales. Esta aplicación fue la lanzada con la plataforma *Google Allo* y, ahora mismo es usada por el altavoz activado por voz *Google Home*.

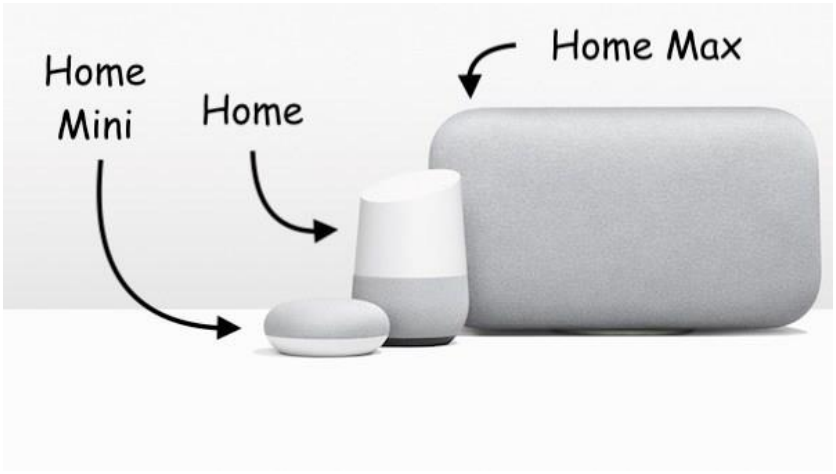

Figura 4: Altavoces inteligentes de Google.

Este asistente al igual que *Google Now* tiene funciones parecidas como buscar por internet, programar eventos, alarmas, gestionar configuraciones del dispositivo entre otras, lo que lo diferencia es que puede gestionar la interacción con el usuario de manera bidireccional. Google Assistent soporta varios lenguajes y es capaz de responder también en este grupo de lenguajes: alemán, inglés (Australia), inglés (Canadá), inglés (Inglaterra), inglés (Estados Unidos), francés (Canadá), francés (Francia), italiano, japonés, castellano y portugués.

Este asistente utiliza la librería "*Google Assistant Library*" [12] para el control básico de la funcionalidad de *Google Assistant*, así como el manejo del ciclo de vida de este, compatible con Python. Después de este proceso se utiliza “Google Assistant Service", que proporciona una gran flexibilidad y amplio soporte a la plataforma. Proporciona una API de bajo nivel que manipula el audio y genera enlaces para diferentes lenguajes de programación como JavaScript, Go, C++ y Java entre otras.

Este asistente ha permitido crear el Google Home, en figura 4, que es un altavoz inteligente con base en Google Assistant, que permite al usuario entre otras cosas inicializar la música, controlar videos la televisión (siempre que tengamos un dispositivo como el Google Chromecast) recibir noticias o controlar dispositivos varios. Este dispositivo trata de automatizar diferentes funciones en el hogar.

Por último, el Cloud Speech-to-Text, es una API de Google que permite convertir la voz en texto de una forma bastante básica. Cabe la posibilidad de reconocimiento de 120 idiomas, siendo gratuita los primeros 60 minutos diariamente. Permite gestionar la base de contactos, permitiendo habilitar el control por comandos de voz, así como poder transcribir las conversaciones de diferentes idiomas.

2.1.2.2 Siri

Otro de los gigantes de la comunicación como es Apple y después de la adquisición [23] de Siri, se ha convertido en uno de los buques insignia de las aplicaciones de inteligentes de reconocimiento de la voz. A continuación, veremos dos.

Siri fue creada por el grupo "SRI Venture Group" en el año 2007, lo que posteriormente fue comprada por la compañía Apple. Tras la adquisición por parte de la compañía empieza a incluirse en sus dispositivos en su sistema operativo iOS.

Entre las características y funciones más destacables, Siri se adapta a las características del usuario, se personalizan las búsquedas web o también se pueden programar mediante el reconocimiento de voz diferentes tareas, como consultas meteorológicas o la gestión de eventos, además de tener un motor para conversaciones bidireccionales que permiten al usuario interaccionar con el dispositivo.

Siri utiliza otra empresa como es *Nuance* para pasar la voz comprimida previamente y pasarla a texto, que luego será transmitido mediante librerías creadas en X-Code [24], para la utilización de la información introducida mediante la voz.

Este sistema de reconocimiento utiliza un sistema cerrado hecho que hace que el desarrollo de nuevos lenguajes diferentes a los ya desarrollados por Apple solo se pueda hacer desde dentro de la empresa.

Con la llegada de los altavoces inteligentes, Apple también se unió a la creación de su propio dispositivo. "*HomePod*" que funciona con Siri, respecto a otras compañías el precio es bastante más alto [25] que su competencia, la calidad de sonido del altavoz es superior. Este dispositivo es capaz de reproducir música, interactuar con el usuario etc.

Figura 5: Altavoz de Apple.

Aunque es compatible con muchas aplicaciones, en el ejemplo de reproductores de música externos a la compañía sí que aparecen limitaciones. Por ejemplo, en el uso de la plataforma de música *streaming Spotify*, se limita a subir y bajar el volumen parar la

música… Si utilizamos el servicio de *Apple Music* además podemos avanzar 20 segundos y otras cosas como pedir información del artista, todo esto mediante órdenes por voz

Por otra parte, hay que decir que, por ahora, *HomePod* no tiene funciones [27] como las de recordar eventos u otras tareas que sí se puede hacer en sus competidores como *Google Home o Amazon Echo*.

2.1.2.3 Alexa

Alexa es un asistente virtual desarrollado por Amazon, su primer uso fue dentro de la gama de altavoces inteligentes "Amazon Echo" en la figura 6 y derivados.

Figura 6: Altavoz de Amazon.

El servicio de voz de Alexa permite acceder a las capacidades ofrecidas por Alexa basadas en la nube con el soporte de las API de "Alexa Voice Service". Este servicio permite a los desarrolladores integrar Alexa directamente en sus productos, proporcionando que se puedan controlar estos productos mediante la voz.

"Alexa Voice Service Device SDK" [28] proporciona librerías en C++ que aprovechan la API de "Alexa Voice Service" para crear el software adecuado para productos habilitados de Alexa. Se puede modular, proporcionando diversas componentes para manejar las funciones como el reconocimiento de la voz. Soporta diferentes idiomas entre ellos inglés, alemán, japonés, francés, italiano y español. Para la conversión de la voz a texto Amazon adquirió Ivona [29], que es donde se encuentra la API Amazon Polly que es la que permite convertir su audio a texto o viceversa. Tiene un abanico de voces diferentes para ciertos idiomas, entre otras funciones tiene la de ajustar la velocidad de la voz a la de la imagen que se proyecta, es decir si el vídeo está en inglés y se quiere traducir al alemán esta aplicación corregirá la velocidad automáticamente para que dure igual.

Es compatible con varias plataformas y lenguajes de programación como Java, JavaScript, PHP, entre otros o SDK para móviles como iOS y Android. Amazon también

soporta una API de HTTP para así poder implementarse, con el uso de “Alexa Web Services” y derivados la propia capa de acceso.

Amazon Echo [30] es un altavoz inteligente de la compañía Amazon. Este altavoz funciona con *Amazon Web Services* que, como se ha comentado en el punto anterior, se trata de los servicios de computación en la nube que tiene Amazon. El dispositivo por defecto está a la escucha de la conversación hasta que escucha la palabra Alexa. Entre otras funciones Echo es capaz de reproducir música en servicios streaming como Spotify desde un teléfono o tableta. Como también puede controlar por voz la configuración y acciones sobre alarmas, listas de eventos, entre otros.

Recientemente [31] desde la compañía Alexa ha incorporado a su catálogo de aplicaciones y herramientas, nuevas apps de reconocimiento de voz para Android y IOS. En Android esta aplicación no permite realizar acciones similares al “Google Assitant”, como iniciar la escucha de música, enterarte de diversas noticias entre otros servicios. Como punto a favor a Alexa puede activar diversas “Skills” que nos permiten mejorar esta aplicación y ampliar sus funciones.

En el caso de Alexa en Windows 10 escritorio [32], ya se puede ejecutar via voz en cualquier PC, es decir, hay algunas funciones que sí que están disponibles en las versiones de "Amazon Echo", Por ahora esta versión para Windows 10 está disponible en inglés (Estados Unidos y Inglaterra) y alemán.

2.1.2.4 Cortana

De este sistema de reconocimiento ya hemos hablado en el punto 2.1.1.1 y en este caso habría que apuntar que, en dispositivos móviles, aparte de estar disponible en las versiones con Windows10 para móvil, también existe la versión propia para Android y iOS como asistentes personales.

Cortana como otros asistentes como *Google Assistant*, recopila información mediante su sistema de reconocimiento de voz, para así poder realizar una búsqueda más detallada para el usuario.

Como ya se ha hablado en el punto 2.1.1.1 Cortana utiliza el motor de búsqueda Bing. Entre las funciones destacadas muchas de ellas compartidas con los otros asistentes están las de añadir recordatorios, las típicas búsquedas por navegador y la configuración de funciones del sistema, entre otras.

Microsoft junto a Harman Kardon (una compañía que diseña altavoces y otros sistemas de sonido), han diseñado el primer altavoz inteligente con el uso de Cortana como sistemas para el reconocimiento de voz, el “Harman Kardon Invoke” [33].

Figura 7: Altavoz con Cortana Harman Kardon Invoke.

Este altavoz tiene semejanzas con los populares Google Home y Amazon Echo, entre sus funciones se encuentran, el usuario puede interactuar con el asistente, para saber información general sobre eventos, metrología entre otra información y se puede utilizar junto a la aplicación Skype para realizar llamadas. Como en el caso de altavoces ya citados de otras compañías posibilita utilizarlo como asistente personal en algunas funciones en el uso doméstico.

2.1.2.5 Comparativa entre altavoces inteligentes en el reconocimiento de voz

En este apartado vamos a adentrarnos en una pequeña comparativa entre los dos altavoces inteligentes más populares, el Google Home y el Amazon Echo. Basándonos en información de las propias webs, así como blogs donde se comparan sus funciones. En este caso no entrarían los dispositivos de Apple porque aún no tiene suficientes funciones para igualarse a los demás y debido a que duplica el precio de estos dos por ahora no es la mejor elección. Tampoco nos hemos adentrado a comprar con marcas como la Sonos, Xiaomi, Acer o Asus, debido a su menor volumen de ventas y popularidad actualmente [34]. Por otra parte, el lanzamiento del altavoz inteligente con Cortana es muy reciente por esta razón no vamos a entrar a compararlo. Vamos a comparar el aspecto de reconocimiento de la voz entre estos dos.

En este caso compararemos el Google Home y el Amazon Echo. Estos dos altavoces inteligentes utilizan Google Assistant y Alexa respectivamente.

Por lo que se puede decir de Alexa es que es capaz de entender varios comandos simples, pero en este caso son bastante básicos. El motor de búsqueda predeterminado es Bing, y requiere que el usuario exprese de manera clara y particular para lo pueda llegar a entender, además el lenguaje que usa para responder e interactuar con el usuario no siempre es natural.

En el aspecto de búsquedas el altavoz de Google cuenta con la ventaja de que utiliza el mayor motor de búsqueda, Google. Por ejemplo, si le recitas una lista de compra este lo envía a tu dispositivo móvil y se guarda.

Google Assistant cuenta con conversación bidireccional que se basa en el propio algoritmo de procesamiento del lenguaje natural que utiliza Google. Esto produce que el asistente está dentro del contexto donde se está hablando, por lo que la conversación será más natural alejándose así de las respuestas robóticas, es decir al pedir información meteorológica sobre el día si le pides y mañana el asistente entiende que sigues hablando sobre el tiempo.

Ambos dispositivos admiten múltiples usuarios, se utiliza el reconocimiento de voz para que se pueda obtener información personalizada sobres aspectos como la ruta diaria y su propio calendario entre otras funcionalidades.

Con esta comparativa cerramos el capítulo de la explicación general de los diferentes sistemas en diversas plataformas que están en el mercado, ahora pasamos a explicar más detalladamente la API elegida que nos ha parecido más razonable como propuesta para profundizar, ya que es una plataforma que busca marcar unas pautas para seguir en un futuro en el desarrollo de aplicación para el reconocimiento de la voz.

2.2. Crítica al estado del arte

Se ha observado que, aunque en los últimos años este campo ha dado un salto de calidad y diversificación del uso bastante grande, no hay unas pautas principales a seguir, para así poder desarrollar aplicaciones compatibles en relación con los diversos dispositivos, desarrollando el código a partir de una API principal.

Desde W3C [35] sí que intenta esta unificación mediante la propuesta de una API para la creación de aplicaciones mediante JavaScript, compatibles con diversos navegadores, y que intenta que se generalice la forma del uso del reconocimiento de la voz mediante JavaScript. En su API pretende dar ejemplos de uso de los diferentes métodos y atributos que se pueden usar y propone un estándar a seguir para el desarrollo de este tipo de aplicativos de entrada y salida de voz.

Lo que se pretende con este proyecto, es dar un paso para poder de alguna manera unificar a grandes rasgos la manera de reconocer la voz en las aplicaciones de diferentes compañías. Se pretende elaborar una aplicación utilizando la Web Speech API [36] de W3C, que es totalmente de código abierto y explicar la funcionalidad de esta librería y los usos que puede acometer.

3. "Web Speech API" W3C

Después de exponer algunos de los sistemas que existen hoy en día en el campo del uso de la voz para el interfaz de aplicaciones y haber expuesto de forma básica qué herramientas usan para el funcionamiento, nos pareció lo más razonable incidir más en la API de reconocimiento de voz de la W3C, debido a que la API es la más extensa y portable de todas las que hemos podido investigar. Además, el uso de JavaScript como lenguaje para programar los métodos para llevar a cabo nuestro proyecto, ya que mediante HTML, CSS y JavaScript podemos ver la funcionalidad del aplicativo fácilmente, aparte de ser un lenguaje bastante popular en el uso de aplicaciones web. Otra de las características es el uso de código completamente abierto.

3.1 World Wide Web Consortium (W3C)

W3C, es un consorcio a nivel internacional con sede en el MIT (EEUU), en ERCIM (Francia) y la Universidad de Keio (Japón) entre otros, en España la sede la alberga la fundación CTIC con sede en el Parque Científico Tecnológico de Gijón (Principado de Asturias). Se genera documentación recomendada y propone unos estándares que pretenden asegurar el crecimiento y la expansión de la *World Wide Web,* así como la supervisión y la estandarización del desarrollo de las diferentes tecnologías sobre la que se basa la Web. Este consorcio entre otros lo que busca es la neutralidad e interoperabilidad de sus lenguajes para así hacer más fácil la construcción de la Web, creado y dirigido por Tim Berners-Lee creador de la primera comunicación entre un cliente y un servidor con el uso del protocolo HTTP, que junto a su equipo crearon la URL, el lenguaje HTML y el protocolo HTTP, que son las principales tecnologías sobre las que se basa la Web.

En el aspecto en que nos vamos a centrar es en su *Web Speech API*. Esta API tiene como objetivo permitir, de una manera neutral a los desarrolladores web, la opción de poder introducir funciones de entrada y salida de voz, convirtiendo tanto la voz a texto como el texto a voz. La API es independiente de la implementación de la síntesis, puede admitir el reconocimiento y la síntesis basados en el servidor y en el cliente. La API cuenta con las funciones de reconocer la entrada de voz de manera continuada, así como la entrada de voz de tiempo breve.

Debido a la reciente aparición de los sistemas de reconocimiento de la voz de forma más general, lo que esta API intenta agrupar, a modo informativo, unos estándares que se pueden aplicar para la construcción de métodos capaces de reconocer la voz. Esta API es una librería para JavaScript. Como bien dice en su repositorio [18], este subconjunto de métodos no es la versión oficial de la librería que se tiene que utilizar para el reconocimiento de la voz, pero se intenta mediante esta API que se defina una posible hoja de ruta para una futura estandarización.

3.2 Usos

Entre los casos de usos propuestos por el grupo de desarrolladores para investigar la compatibilidad [36] de esta API estarían:

I. Búsqueda por voz en la web, el usuario puede recitar una cuestión y así obtener un resultado.
II. Construcción de una interfaz de comandos de voz, que permitiría mediante el reconocimiento de la voz, tomar como argumentos ciertas palabras y transformarlas en un evento como pueda ser reproducir cierta canción o buscar cierta información.
III. Reutilización de gramáticas en frases anteriores, por ejemplo, al consultar una ciudad cuando en la frase posterior se pregunte los monumentos que hay, se tiene que interpretar como que se refiere a la ciudad.
IV. Reconocimiento de forma continua un dialogo abierto, se trata de recopilar información de forma libre muy útil en aplicaciones como los editores de texto que además de poder traducir la voz al texto el aplicativo también será capaz de identificar palabras pertenecientes a los formatos.
V. "Re-reconocimiento", es decir al decir una frase sobre quiero reservar dos billetes a Nueva York, la aplicación debe entender que lo que quiere el usuario es buscar vuelos.
VI. Un último caso que expondremos es el de selección de gramáticas específicas para llenar entradas de diferentes campos. Por ejemplo, en la frase: Quiero reservar una mesa para dos en el hotel pio a las 22:00. En este caso los desarrolladores plantearon este caso para ser capaz de que se llenara un formulario con el número de comensales el nombre hotel y la hora.

3.3 Interfaz

El funcionamiento de esta API, que, como se ha descrito en el anterior punto está escrita en JavaScript, consta de una interfaz creada con diferentes métodos y eventos, los cuales citaremos y explicaremos en los puntos 3.3.1 y 3.3.2, para controlar el reconocimiento de la voz. Primero hablaremos de la interfaz *SpeechRecognition*.

3.3.1 Interfaz SpeechRecognition

En este punto vamos a ver algunos de los atributos, eventos e interfaces dentro de esta. En un primer lugar tendríamos los atributos de reconocimiento de la voz o *atributos SpeechRecognition:*

- El atributo *grammars* es del tipo *SpeechGrammarList*, este atributo cumple la función de almacenar la colección de objetos *Speech Grammar* que representan las gramáticas que están activas para el reconocimiento en curso.

- El atributo *lang* es del tipo *DOMString*, este es el atributo que se usa para establecer el idioma en que se procederá al reconocimiento, utilizando la etiqueta BCP 47 si no se especifica este atributo, se utilizará el idioma determinado en el HTML. Este valor se calcula en la solicitud de entrada cuando se abre una conexión con el servicio de reconocimiento.
- El atributo *continuous,* este atributo del tipo booleano cuando está en falso no puede devolver más de un resultado final. En cambio, si se establece a verdadero devolverá cero o más resultados. Por valor predeterminado es falso, este atributo no afecta a los valores provisionales.
- El atributo *interimResults,* también de tipo booleano, es para dar permisos para devolver atributos provisionales, verdadero o falso según se requiera configurar. En falso de manera predeterminada no afectará a resultados finales.
- El atributo *maxAlternatives* del tipo *unsigned long*, para establecer el número máximo de *SpeechRecognitionAlternatives* por resultado. Por defecto a 1.
- El atributo *serviceURI*, del tipo DOMString, sirve para especificar la ubicación del servicio del reconocimiento de la voz que se va a utilizar. Si no se establece se utilizará el servicio de reconocimiento de voz predeterminado.

Seguidamente vamos a exponer los métodos de SpeechRecognition:

- Método *start()*: Cuando se llama al método *start()*, representa el momento concreto donde la aplicación web desea que se inicie el reconocimiento de la voz. Cuando la entrada de voz se transmite en directo a través del flujo de medios de entrada, esta llamada al método start() representa el momento en que el servicio de reconocimiento debe empezar a escuchar y tratar de hacer coincidir las gramáticas asociadas con la petición consecuente. Para tratar una situación en que se hace una llamada al evento en que ya estaría este en marcha, obviando errores, el agente de usuario deberá lanzar una *DOMExcepcion "InvalidStateError"* e ignorar la llamada.
- Método *stop():* Este método es la instrucción que sirve para detener la escucha de audio. Al utilizarse este método devolverá un resultado de estas escuchas. Podría configurarse este método para que cuando un usuario llegue al final del mensaje que quiera transmitir accione un botón y este pare el proceso de escucha. Una vez se ha llamado a este método el sistema debe parar de recopilar información vía audio, en este momento el servicio de voz debe devolver un resultado. Cabe la posibilidad de que cuando se llame a este método ya se este parando el servicio o que no haya sido inicializado, si ocurre esto se debe ignorar la llamada.
- Método *abort():* Este método es una petición para detener inmediatamente el servicio de escucha en ejecución. Cuando se hace una llamada a este método el sistema debe para de recibir información y no devolver ninguna información. Al igual que en el método stop() si se llama al método y el sistema no ha llamado a start(), se debe ignorar la llamada.

Llegados a este punto vamos a hablar de los *eventos de la SpeeechRecognition*. Para los eventos de reconocimiento de la voz se utiliza el DOM de nivel 2. Los métodos de la interfaz eventTarget son los utilizados para registrar a los oyentes del evento. Para todos los eventos se utilizará el atributo timeStamp que se ajustará con la mayor estimación posible al tiempo de cuando se efectuó el evento. A menos que se especifique en algunos

de los eventos, que se van a ver a continuación, en donde especifica que antes de que se dispare un evento se debe disparar otro antes que, para ejecutar un evento de una manera u otra, el orden en que se ejecuten los eventos será independiente.

- Evento *audiostart:* Este evento se activa cuando el usuario empieza a capturar audio.
- Evento *soundstart:* Este evento se activa en cuando recibe o detecta algún sonido, como pueda ser la voz. Para que se active este evento tiene que haberse activado antes el evento audiostart.
- Evento speechstart: Este evento se activa cuando se reconoce que el canal de reconocimiento de la voz está activado. Como los anteriores se activar cuando el evento audiostart haya sido activado
- Evento speeched: Se activa al finalizar el habla que se utiliza para el reconocimiento de voz. Antes de que se lance este evento tiene que haberse lanzado el speechstart, ya que si no puede funcionar este.
- Evento soundend: Este evento se activa cuando ya no se detecta ningún sonido. Este evento se disparará siempre y cuando el evento soundstart haya sido disparado previamente.
- Evento audioend: Este evento se activa cuando el agente de recepción ha concluido la captura de audio. Este evento se tendrá que disparar siempre antes de la finalización del audio.
- Evento result: Este evento se lanzará cuando el reconocedor de voz devuelve un resultado, se debe de utilizar la interfaz SpeechRecognitionEvent. Este evento se disparará si el evento audiostart ha sido lanzado previamente.
- Evento nomatch: Este evento se lanzará siempre que el reconocedor de voz devuelva un resultado final que no cumpla el umbral de confianza marcado. Como el evento result también utilizara la interfaz SpeechRecognitionEvent. Este evento no se lanzará si no se ha lanzado previamente el evento audiostart.
- Evento error: Se lanzará siempre y cuando se produzca un error en el reconocimiento de voz, usando la interfaz SpeechRecognitionErrorEvent.
- Evento start: Se lanzará cuando el servicio encargado del reconocimiento de voz ha comenzado a escuchar el audio para reconocerlo.
- Evento end: Se lanzará cuando el servicio se desconecte. El evento se generará siempre al finalizar la sesión, y no se verá afectado por el motivo de esta.

Para los eventos causados por errores en el reconocimiento de voz, se utiliza la interfaz *SpeechRecognitionErrorEvent*. El atributo error tiene estos diferentes errores.

- No-speech, cuando no se detectó ninguna voz que se pueda utilizar.
- *Aborted*, la entrada del reconocimiento de voz en algún momento puntual se aborta, puede que sea abortada por algún comportamiento especifico del usuario. Se podría usar mediante la interfaz de usuario para que según lo descrito se cancele la entrada de voz.
- *Audio-capture*, cuando falla la captura del audio.
- *Network*, fallo en la comunicación de red que se utilizan para completar algún reconocimiento.
- *Not-allowed*, el agente de usuario no permite el servicio de reconocimiento de voz ya sea por razones de seguridad, privacidad, entre otras.

- También estarían *bad-grammar y language-not-supported*, entre otros.
- El tratamiento de errores se puede especificar mediante el atributo *message* para que los desarrolladores directamente usen un mensaje propio de un error concreto.

En este caso vamos a hablar del SpeechRecognitionAlternative que es la interfaz que representará la respuesta del reconocimiento hecho. En este caso tiene dos atributos.

- Atributo transcript: este atributo es del tipo DOMString. Esta cadena de transcripción representa las palabras en bruto que son reconocidas. Para poder realizarse un reconocimiento continuo, además de las palabras reconocidas se deben introducir los espacios en blanco pertinentes. De este modo se podrá hacer mediante la concatenación de palabras y espacios resultantes en SpeechRecognitionResults produzcan una transcripción adecuada.
- Atributo confidence, es del tipo float, solo de lectura. La confianza representa la estimación numérica entre el rango de valores de 0 a 1, que transmite el nivel de seguridad que esta el sistema de reconocimiento de la voz de que esta ha sido correcto.

La interfaz SpeechRecognitionResult, es la representación de la coincidencia única cuando se lanza, puede ser como una pequeña parte de un reconocimiento continuo o como el resultado completo de un reconocimiento que no sea continuo. Entre los atributos estan:

- Atributo length, del tipo "unsigned long". Este atributo representa el número de alternativas representadas en el array de elementos.
- Atributo isFinal, de tipo booleano. Este atributo sirve para si está en verdadero, esta será la última vez que el servicio de reconocimiento devolverá el valor del índice particular, es decir el valor que será el definitivo. En cambio, si el valor está en falso, en este caso representará un valor provisional que se podría aún modificar.
- Buscador de índices: Este buscador de elementos devolverá un SpeechRecognitionAlternative del índice en una matriz de los n mejores valores. Si el índice es mayor o igual a la longitud devolverá nulo.

3.3.2 Interfaz SpeechSynthesis

Esta interfaz es la que se usa para la conversión de texto a voz, en este punto vamos a ver algunos de los métodos, eventos producidos e interfaces dentro de esta. A su vez contiene la SpeechSynthesisVoice, que contiene todas las voces disponibles en el navegador.

Algunos de los atributos de SpeechSynthesis:

- Atributo *pending*: Este atributo es de tipo booleano, devuelve verdadero si en la cola de la instancia global de SpeechSynthesis aún existen expresiones que aún no han sido reproducidas o habladas.

- Atributo *paused*, del tipo booleano, devolverá verdadero cuando la instancia global de SpeechSynthesis está en pausa, independiente de si hay algo en la cola.
- Atributo *speaking*. Este atributo que también es del tipo booleano devolverá cierto si se está reproduciendo una frase, es decir si al iniciar recitado de un enunciado aún está en transcurso y no ha terminado aún. Es independiente de la instancia global de pausado del SpeechSynthesis.

Algunos de los métodos de SpeechSyntehsis son:

- Método *speak(utterance)*: La particularidad de este método es la de añadir al final de la cola el objeto *SpeechSynthes.isUtterance* cuya interfaz hablaremos en los próximos puntos. Este método no cambia el estado de la instancia SpeechSynthesis si está en estado de pausa, si está en pausa seguirá en el mismo estado, ahora en el caso de no estar en pausa y no hay otros enunciados en la cola, entonces este enunciado iniciará su reproducción, si hay algún enunciado entrará en la cola para comenzar a hablar cuando se hayan pronunciado los enunciados anteriores a este. Si se modifica el objeto *SpeechSynthesisUtterance* antes de que se llame a este método, no está respaldado que el valor sea el correcto y su funcionamiento de esta manera podría traducirse en un error. El objeto SpeechSynthesis es propiedad exclusiva del objeto *SpeechSynthesisUtterance*, si lo pasáramos como un argumento *speak()* a otro objeto SpeechSynthesis debería de salir una excepción.
- Método *pause*(): Este método pondrá la instancia global del SpeechSynthesis. Si se está pronunciando una palabra, se detendrá a la mitad de esta.
- Método *cancel()*: Este método es el encargado de eliminar todas las expresiones en cola cuando un discurso es interrumpido inmediatamente. Este método no cambia el estado pausa de la instancia global del SpeechSynthesis.
- Método *resume():* Este método pone la instancia global del SpeechSynthesis en el estado no pausado. Si un enunciado estaba hablando este método lo reanuda siguiendo desde donde se había pausado, o empieza el siguiente enunciado en la cola.
- Método *getVoices():* Este método devuelve las voces disponibles. Dependerá del agente de usuario usado que voces están usadas. Si el listado de voces esta vacío o en el caso de que no se conozcan, en este caso el método devolverá un *SpeechSynthesisVoiceList* de tamaño igual a cero.

Entre los eventos *SpeechSynthesis*:

En este caso está el evento *voiceschanged,* se dispara cuando el contenido de SpeechSynthesisVoiceList, que es devuelto por el método getVoices(), ha cambiado. Esto incluye la síntesis del lado del servidor que se determina de forma asíncrona o cuando las voces del lado del cliente se instalan/desinstalan.

Entre los Atributos de SpeechSynthesisUtterance:

Esta es una interfaz que contienen diferentes servicios del reconocimiento e información sobre cómo se van a leer, vamos a verlos en detalle.

- Atributo *text*. Este atributo es del tipo DOMString y especificara el tipo de texto que se sintetizará y hablara en esa expresión. Los tipos de texto que pueden ser va desde un documento SSML bien formado, a texto plano. En el caso de los motores de voz que no son compatibles con SSML, o que solo admiten ciertas etiquetas, el agente de usuario deberá eliminar estas etiquetas no admitidas y recitar el texto. La longitud máxima del texto no superará los 32767 caracteres.
- Atributo *Lang*. También del tipo DOMString, este atributo especifica el idioma de la síntesis de voz para la expresión al igual que en otros atributos de lenguaje utiliza la etiqueta de idioma BCP 47. Si no se configura utilizara el lenguaje propiamente configurado en la página HTML.
- Atributo *SpeechSynthesisVoice*. Tipo nullable, este atributo especifica la voz de reconocimiento de la síntesis de la voz que la aplicación web desea utilizar. Cuando se crea un objeto SpeechSynthesisUtterance este atributo se inicializará a null. En cuanto al momento en que se hace una llamada a speak(), este atributo se ha establecido en uno de los objetos de SpeechSynthesisVoice que devuelve el método getVoice(). En este caso el usuario usará esta voz. Si por el contrario esta inicializado a null en el momento de la llamada a speak(), se usará la voz por defecto del agente de usuario, que debe soportar el idioma actual (lang).
- Atributo *volumen*. Es de tipo float, este atributo lo que indica será el volumen en que se reproducirá el discurso entre el rango de 0 a 1. A excepción de si se utiliza SSML que este valor, al igual que en el atributo rate y pitch que hablaremos a continuación, vendrá dado por elemento prosody, encargado de controlar tipo de interés y volumen del discurso.
- Atributo *rate*. Este atributo del tipo float, especifica la velocidad de la voz para la expresión. Este abarca desde el rango de 0,1 hasta 10 menos o mas no estarían permitidos, por ejemplo, si se utiliza como valor 3 sería el triple de la velocidad normal que seria 1. También hay que decir que según los motores de síntesis de la voz empleados pueden limitar más aún este rango. Como en el atributo volumen si se utiliza SSML se utilizarán los valores de la etiqueta prosody.
- Atributo *pitch*. Este atributo especifica el tono de voz para la expresión. Estará definido entre el rango de 0 a 2 donde 1 será el tono por defecto. Como también pasa en el atributo rate los motores de la síntesis de voz pueden limitar aún más el rango y como también se habló al utilizarse SSML, los valores vendrán dados por la etiqueta prosody.

De los eventos SpeechSynthesisUtterance no hablaremos de todos. Porque ya coinciden con los eventos hablados en el punto 4.3.1 en la interfaz de SpeechRecognition, estos eventos serian el start, end, error, pause, resume, etc. Vamos a exponer estos dos:

- Evento mark: Se dispara cuando la expresión que se va a leer alcanza una etiqueta denominada mark en el tipo SSML. El agente de usuario debe disparar este evento si el motor de síntesis de voz proporciona el evento.
- Evento boundary: Este evento se dispara si la expresión oral alcanza el límite de una palabra o frase, es decir con una palabra marcada con un atributo name, que es una palabra pactada donde si se reconoce esta se dispararía el evento pudiendo así tomarla como palabra de parada.

Los atributos que conforman la SpeechSynthesisEvent, que son estos tres.

- Atributo charIndex: Este atributo es del tipo unsigned long, Este atributo nos indica el índice de caracteres que más se aproximan a la posición del discurso actual, que usa el motor de voz en uso. Cabe decir que este atributo no os garantiza donde estará charIndex con respecto a una palabra, es decir no nos dirá si estamos en medio de una palabra, sino que lo que hace es decir en la posición donde está el índice en una frase. Así se indicará las palabras ya recitadas por las que quedan.
- Atributo elapsedTime: Este atributo es de tipo float y nos indica el tiempo, en segundos, el tiempo de tardanza en ponerse en marcha desde que se empezó a hablar.
- Atributo name: Este atributo de tipo DOMString, se usa para eventos mark, como el elemento de marca de un elemento como se indica en los tipos SSML. Para eventos boundary, este atributo indicara el tipo de limite causado en este caso por una palabra o frase.

Otros atributos que aparecen en la interfaz SpeechSynthesis serían los *SpeechSynthesisErrorEvent*, referentes a los errores que se puedan dar y los *SpeechSynthesisVoice*, que especificarán la voz suportada por el sistema, además de incluir información sobre el lenguaje, nombre, entre otros.

3.3.3 JSpeech Grammar

En este punto una vez que ya se ha comentado de la parte del reconocimiento de la voz y de la síntesis de la voz a partir de un texto, vamos a comentar el JSpeech Grammar que es una plataforma independiente que nos da la facilidad de mediante el uso del reconocimiento de voz, para poder despachar un evento. En general lo que hace es que el navegador está en continuo modo de reconocimiento, pero con el uso de JSpeech Grammar se pueden acotar las palabras a las que realmente le asignaremos funciones para disparar eventos.

En general tiene la función de poner reglas, es decir para ciertas palabras reconocidas este devolverá un texto o disparará un evento:

```
recognition.start();

        var myCommand = {
          siguiente : function(){
                //go to the next post

        this.writeCommand("siguiente");
          },
```

En este caso al reconocer la palabra siguiente devolverá por pantalla siguiente. En general para que la regla pueda ser legal tendrá que tener un identificador similar a lenguajes de programación, con la permitividad de añadir caracteres adicionales como puedan ser + - *: ; ,* = | / \ *( ) [ ]* @ # % *!* ^ & ~. Hay que remarcar que al usar el juego de caracteres de Unicode este sí que diferencia las mayúsculas de las minúsculas.

Entre otras de las características se encuentran los tokens, que son la parte de la gramática que pueden ser recitados por un usuario. Por ejemplo, la secuencia de caracteres (abrir ventana nueva), podría estar dentro de un token. También se puede agrupar varias palabras con significado, ya sean sinónimos o por ejemplo que pasa en la mayoría de los reconocedores que tienen soporte para diferentes idiomas.

```
<si> = si|yes|we;
```

Otra funcionalidad que presenta es que permite anidar las reglas de manera recursiva, donde la regla ser refiera a ella misma como la última parte de su propia definición. Podemos observar en el siguiente ejemplo una forma de recursión válida.

```
<comando> = <acción> | (<acción> y <comando>);

<acción> = parar | empezar | pausar | reanudar | terminar;
```

Por último, hablamos de los usos de <NULL>, este nombre está reservado por el sistema. Uno de los usos que tiene <NULL> es para mapear la estrella de Kleene y quedaría de la siguiente manera como se puede observar en este ejemplo.

```
<x> = <NULL> | a <x>;

<x> = a*;
```

3.3.4 Idiomas disponibles de la API en Navegadores

Después de explicar las dos principales interfaces la de reconocimiento de voz y la de conversión de texto a voz, hay que hablar un poco de las compatibilidades con el idioma que existen en esta API.

Si utilizamos Google Chrome este navegador tiene poco menos de 90 voces diferentes, para utilizar según el idioma que vayamos a utilizar. Hay que decir que desde la versión 33 de Chrome lleva soportando la síntesis y el reconocimiento de la voz.

Mediante un pequeño código [38] utilizando JavaScript, además de HTML hará que nos devuelva el texto, comprobamos las voces disponibles para habla hispana. En este caso en nuestro navegador y versión 70.0.3538.102 nos da tres tipos de voces de habla hispan:

- Microsoft Helena Desktop - Spanish (Spain)[es-ES]

- Google español[es-ES]

- Google español de Estados Unidos[es-US]

En el caso de Firefox con versión 63.0.3, tras varias pruebas se consigue el funcionamiento, pero de forma parcial además de que el número de voces para la síntesis es inferior al que se obtiene en Chrome. Se configuran los dos atributos del about:config de Firefox, *media.webspeech.recognition.enable y media.webspeech.synth.enabled*, pero la funcionalidad así como las voces es parcial, respecto a Chrome. Después de buscar información con la compatibilidad, se encuentra que, sí que es compatible con Firefox

[39], pero tiene algunas particularidades. Entre esas particularidades, por ahora Firefox no soporta la función de reconocimiento continuo.

En este caso podríamos decir que por parte del lado del navegador de Chrome la parte del reconocimiento de voz tiene más potencia de procesamiento que la que tiene Firefox.

3.4 Ejemplos básicos de uso de la Web Speech API.

Antes de iniciar el diseño de la aplicación final, se han realizado dos pequeñas aplicaciones separadas, entre el uso del reconocimiento y el uso de la síntesis. Estas aplicaciones han sido realizadas a partir de diseños que se han encontrado en documentos de la MDN de Mozilla [40], se han realizado pequeños cambios, pero el código JavaScript es muy similar. Vamos a detallar el funcionamiento de estas dos aplicaciones prueba.

3.4.1 Aplicación de reconocimiento de voz "Cambio de color"

A raíz de la aplicación ejemplo "Speech color changer" [40] se ha desarrollado una aplicación donde se cambiará el color de la cadena al recibir la cadena con los colores prestablecidos. En el ejemplo que nos hemos basado cambia el color al fondo de pantalla, pero las cadenas prestablecidas utilizan un array para almacenarlas y están en inglés y la manipulación de los datos del reconocimiento es también diferente al utilizado por nosotros a la hora de manipular los datos transcritos.

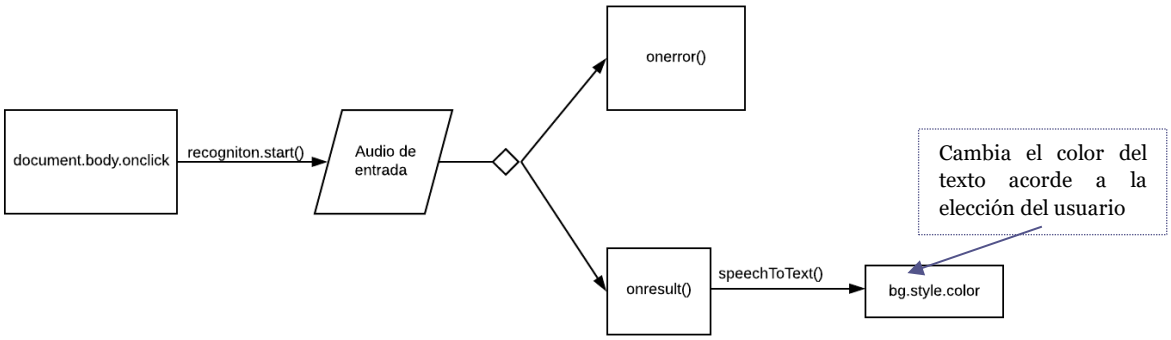

Figura 8: Flujo general de reconocimiento

En primer lugar, se recibe la voz a través de un dispositivo de entrada (en nuestro caso el micrófono de nuestro ordenador), al recibir el audio este es verificado por un servicio de reconocimiento de voz con una lista gramática que se ha configurado (en nuestro caso la hemos acotado a 5 colores), esta lista básicamente tiene la función para preconfigurar las cadenas de texto que al ser reconocidas se ejecutará una acción predeterminada como

resultado. En nuestro caso sería la acción de cambiar de color el texto que aparece en primera instancia.

En primer lugar, en el documento *JavaScript* se debe de asignar la interfaz que vamos a utilizar en este caso será la del *SpeechRecognition,* como podemos apreciar en la figura 8.

```
window.SpeechRecognition = window.webkitSpeechRecognition || window.SpeechRecognition;
```

Figura 9: Interfaz reconocimiento.

En la figura 9, se puede observar de forma simple el proceso de inicialización de la interfaz de reconocimento. En primer lugar, al accionar cualquier parte del documento HTML, se iniciará el proceso de recopilación de la voz mediante un dispositivo de entrada. El código constaría de una función que al detectar el *click* iniciara el proceso de captura con el recognition.start(), como se puede observar en la figura 10.

```
document.body.onclick = function () {
  dictate();
  esc.innerHTML = 'Preparado para escuchar un color...';
}

const dictate = () => {

  recognition.start();
```

Figura 10: Inicialización reconocimiento.

Tras iniciarse el reconocimiento como se expone la figura 10, el sistema de reconocimiento empieza a recibir audio desde el dispositivo de entrada, mostrando una cadena “Preparado para escuchar un color”. Al iniciar este proceso pueden aparecer diferentes errores como que no se reciba ninguna voz o la calidad de esta sea defectuosa. De ser este el caso devuelve una cadena con el error detecta que será detectado gracias a la propiedad “onerror” del Speech Recognition, devolverá el tipo de error que se detecte.

Si no se detecta ningún error el audio que se recibe se transformará y se almacena en la constante “speechToText” con la utilización de la propiedad “transcript”. Esta propiedad de lectura devuelve un “String” con la transcripción del audio recibido. Que la función que hace es tras ejecutarse el evento “SpeechRecognitionEvent” devuelve una lista de objectos “SpeechRecognitionResult”. Esta lista contiene los objetos que son accedidos mediante un array.

Una vez tenemos la cadena transformada en la constante “speechToText”, se procede a comparar esta con las cadenas preconfiguradas, para si se da el caso que coincide dispare el evento creado, como se puede apreciar en la figura 5.

```
const speechToText = event.results[0][0].transcript;

if (event.results[0].isFinal) {
  if (speechToText.includes('azul')) {
    bg.style.color = "blue";
    limpiar();
  }
```

Figura 11: Constante SpeechToText.

Como se puede apreciar en la Figura 11, primero comprueba que el resultado este completo con el "isFinal", seguidamente compara la cadena resultante mediante el "includes" con una cadena que hemos prestablecido en el ejemplo de la Figura 11 'azul'. Si coincide cambiará el color del texto mediante el "bg.style.color = "blue";". El "bg" es una variable que extrae del html el elemento con id = "hc1", que será el que contiene la cadena a cambiar de color.

En la figura 12 se puede ver ejemplos de diferentes estados de la aplicación, el estado inicial, cuando se inicia el sistema de reconocimiento y cuando se realiza el cambió del título a azul al coincidir el texto reconocido con la cadena prestablecida. Si no es ejecuta un click el sistema de reconocimiento no empezará el proceso.

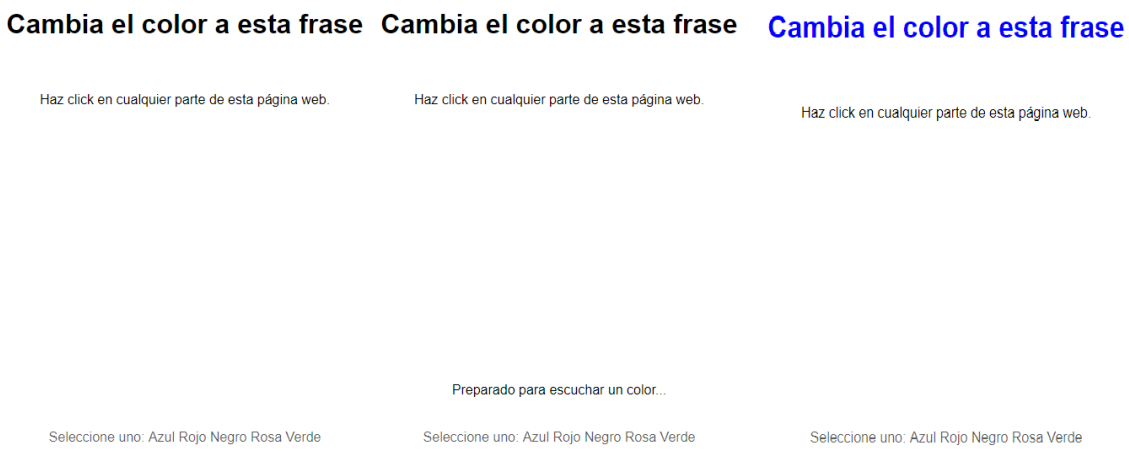

Figura 12: Captura aplicación reconocimiento.

3.4.2 Aplicación de síntesis de voz "Síntesis de voz simple"

A raíz de la aplicación del ejemplo [40] *Speech synthesiser* en que nos basamos, realizamos esta aplicación con la tarea de reproducir un texto que escribimos en un text-box, la aplicación original en que nos hemos basado aparte de hacer esto le añade un

método para obtener las voces y poder seleccionar una, también para poder modificar la velocidad y la tesitura de la reproducción.

En primer lugar, el usuario introducirá un texto deseado dentro de un cuadro de texto y tras pulsar el botón reproducir, empezará a reproducir el texto introducido mediante el dispositivo de salida.

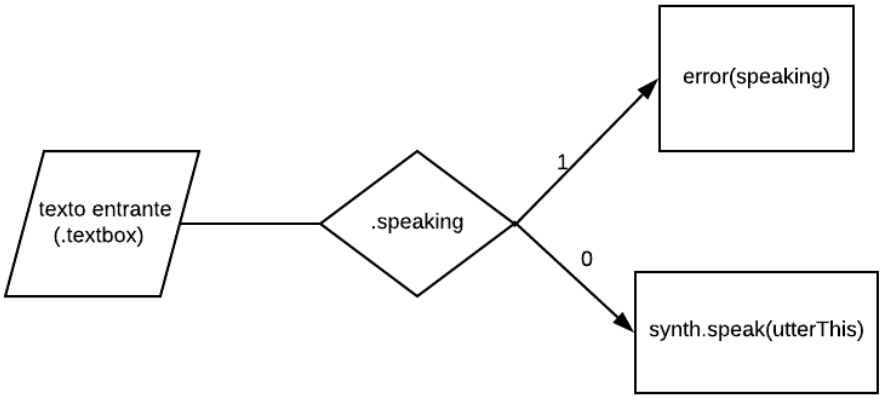

Figura 13: Flujo aplicación síntesis.

En primer lugar, en el documento *JavaScript* se debe asignar la interfaz que vamos a utilizar en este caso será la del *SpeechRecognition,* como podemos apreciar en la figura 14.

```
const synth = window.speechSynthesis;
```

Figura 14: Constante de síntesis.

En la figura 7 se puede observar de forma general los procesos que se ejecutan en esta aplicación. En primer lugar, se debe introducir el texto deseado en el input del tipo *text*, una vez que se ha introducido la cadena, se pulsará el botón de reproducir o se pulsa *intro*. Tras detectar el *onsubmit,* ejecutará la función *speak* y se ejecutará el método *speaking* para comprobar si ya se está ejecutado alguna reproducción de ser así devuelve *true* y no reproduce hasta que, termina la reproducción en curso, como se observa en la figura 9.

```
function speak(){
    if(synth.speaking){ //Devuelve true si ja se est
        console.error('speechSynthesis.speaking');
        return;
    }
```

Figura 15: Función speak.

En caso de devolver el método *speaking el valor false,* continuaremos con la síntesis, del texto entrante previa comprobación que no esté vacío. En este caso hemos configurado un solo lenguaje para su reproducción en castellano con el atributo *lang* y seguidamente se disparará el evento *speak,* como se puede apreciar en la figura 16.

```
utterThis.lang = 'es-ES';
synth.speak(utterThis);
```

Figura 16: Evento speak.

Tras la ejecución del *speak*, se empezará a reproducir la cadena introducida. En la figura 17 se puede apreciar la interfaz de esta aplicación

Figura 17: Interfaz de la aplicación de síntesis.

Como se observa en la figura 17, al introducir un texto y accionar el botón de reproducir, cambiara la cadena inicial "Inserta caracteres en la caja de texto para reproducir", por Leyendo más la cadena introducida. Esto ha sido posible mediante modificación de la variable del documento HTML, que contiene la cadena un vez se ejecuta el evento *speak() como se observa en la figura 18*. Una vez se detecta que la reproducción ha terminado mediante el *utterThis.onend()*, evento que dispara la interfaz de síntesis una vez se termina la reproducción, volveremos a dejara la cadena con su contenido inicial.

Figura 18: Interfaz de la aplicación de síntesis en reproducción.

4. Diseño del aplicativo

La aplicación a desarrollar constituye una aplicación web dinámica. En este sentido, desde el lado del cliente o *frontend,* el usuario interactuará con el sistema de reconocimiento, para poder ejecutar las diversas funciones con las que se le va a dotar.

4.1 Idea de la aplicación

Los resultados vistos en el capítulo 3 son solo el principio de este desarrollo. En este capítulo vamos a explicitar las funcionalidades que debería mostrar este tipo de aplicación para poder ser comparado con un asistente de los que hay en el mercado y que se han expuesto anteriormente. A partir de estas aplicaciones iniciales se pensaron diversas funciones que en se podrían realizar mediante JavaScript y así poder ejecutarlas con los diferentes métodos de la Web Speech API.

4.2 Casos de uso.

Los primeros casos que puede ejecutar el usuario son los de seleccionar la voz en que nos responderá la síntesis, la velocidad y la tesitura de respuesta.

A continuación, se van a enumerar los casos de uso realizables por el usuario, al accionar el icono del inicio del reconocimiento.

1. Abrir navegador, mediante la pronuncia de la cadena “abrir navegador”.
2. Abrir YouTube, mediante la pronuncia de la cadena “abrir YouTube”.
3. Abrir Facebook, mediante la pronuncia de la cadena “abrir Facebook”.
4. Abrir Twitter, mediante la pronuncia de la cadena “abrir Twitter”.
5. Abrir Galería, mediante la pronuncia de la cadena “abrir galería”, acción que nos llevará a una ventana secundaria donde se encuentra el visor.
6. Saber la hora actual, mediante la pronuncia de la cadena “que hora es”, esta acción iniciará la síntesis con la hora actual.
7. Saber la fecha actual, mediante la pronuncia de la cadena “que fecha es”, esta acción iniciará la síntesis con la fecha actual.
8. Saber la condición climática actual, mediante la pronuncia de la cadena “que tiempo hace en + ciudad”, esta acción iniciará la síntesis con la climatología actual con la temperatura actual más la máxima y mínima.
9. Saber la presión y humedad atmosférica, mediante la pronuncia de la cadena “cuál es la presión y humedad en + ciudad”, esta acción iniciará la síntesis con la presión y la humedad actual.
10. Saber la velocidad del viento, mediante la pronuncia de la cadena “cuál es la velocidad del viento en + ciudad”, esta acción iniciará la síntesis la velocidad del viento actualmente.

11. Saber la geolocalización actual, mediante la pronuncia de la cadena “cuál es mi posición”, esta acción devolverá las coordenadas de la posición actual donde se encuentre nuestra IP.
12. Pedir que nos cuente un chiste, mediante la pronuncia de la cadena “cuéntame un chiste”.
13. Consultar la clasificación de la liga Santander, mediante la pronuncia de la cadena “consulta clasificación liga Santander”, nos llevará a la página del ranking de la liga Santander.
14. Consulta la clasificación de goleadores, mediante la pronuncia de la cadena “máximos goleadores en la Liga” nos llevará a la página del ranking de goleadores de la liga Santander.

En la parte del visor de imágenes, el documento HTML secundario vamos a enumerar los casos usos que puede realizar el usuario, cuando se pulsa el icono de ejecución de la síntesis.

1. Pasar a la siguiente imagen, mediante la pronuncia de la cadena ”siguiente imagen” o “siguiente”.
2. Volver a la imagen anterior, mediante la pronuncia de la cadena ”anterior imagen” o “anterior”.
3. Ir directamente a un numero de imagen, mediante la pronuncia de la cadena “(primera, segunda…) imagen” o simplemente “(primera, segunda…)”.
4. Volver a la página principal, mediante la pronuncia de la cadena “volver a la página principal”

4.3 Prototipado previo del aplicativo.

Antes de realizar el desarrollo de la aplicación llevamos a cabo algunos diseños prototipados de la idea general que queríamos que tuviera nuestra aplicación.

En la ventana principal se tenía una idea donde estuviera el peso de la aplicación con un botón central para iniciar el reconocimiento y otros botones secundarios para en caso de fallido del sistema poder realizarlo manualmente. También se decide añadir otros botones del tipo desplegable que se accionarán de manera manual para añadir configuraciones como las voces disponibles, la tesitura de la voz, la velocidad de la voz y una lista de los comandos que se han preconfigurado. En la parte final se introduce un *label text* a modo de guía para saber la cadena que ha reconocido y el intervalo de confianza del reconocimiento de esta cadena. En la figura 18 podemos observar el prototipo de la ventana principal.

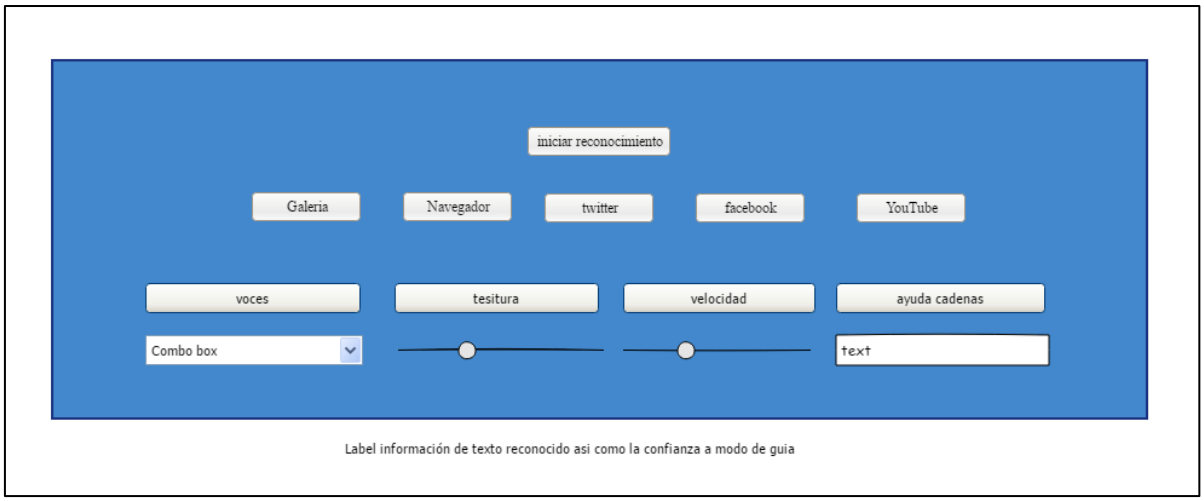

Figura 18: Prototipado de ventana principal.

En la ventana secundaria, la idea que se tenía es la de hacer un visor de imágenes con un botón principal para iniciar el reconocimiento, así como en el caso de la ventana principal por si falla el reconocimiento botones manuales de imagen anterior, imagen siguiente, así como uno para volver a la ventana anterior.

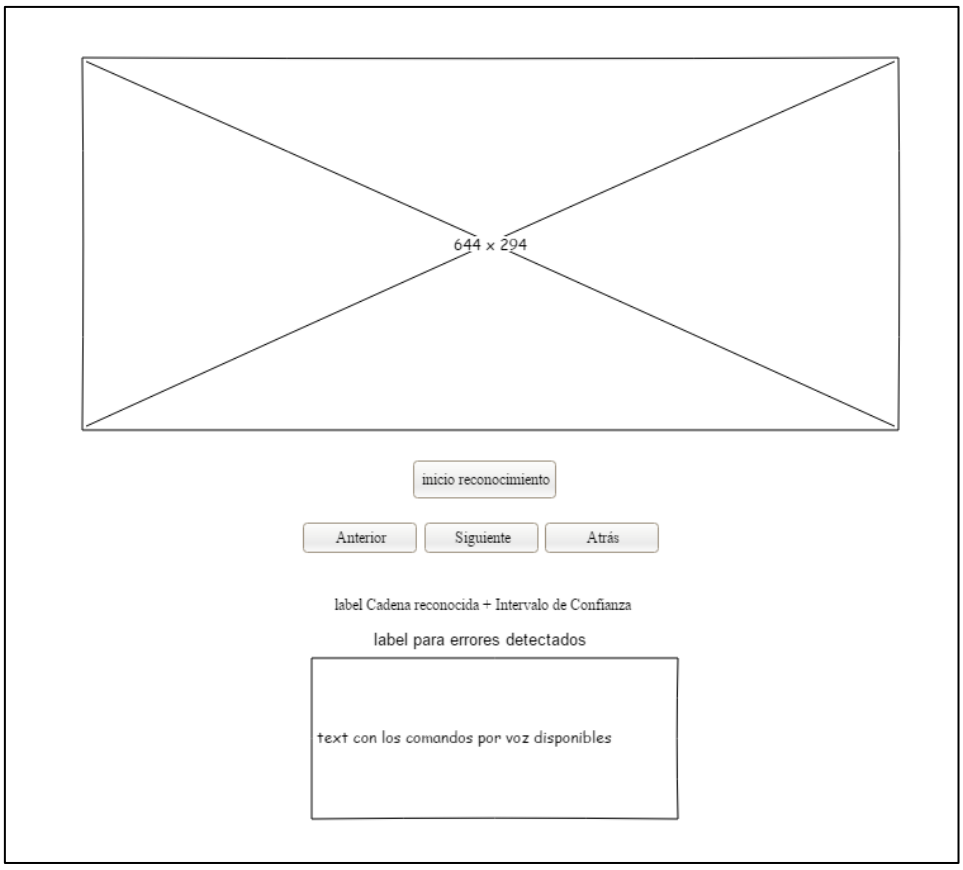

Figura 19: Prototipado de la ventana secundaria.

Se diseña con dos *label text* el primero como en la ventana principal a modo de guía para saber la cadena que ha reconocido y el intervalo de confianza del reconocimiento de esta

cadena y el otro *label text* para que nos devolviese la cadena con el error. Además de insertar una caja de texto donde indicar las cadenas por voz disponibles. En la figura 19 podemos observar el prototipo de esta ventana secundaria.

4.4 Arquitectura del aplicativo.

En el proyecto, se ha utilizado el propio navegador para realizar la visualización del aplicativo, donde mediante un web Server ejecutamos un archivo HTML principal, que está respaldado por otros dos archivos CSS y JavaScript, que a su vez invoca a un archivo HTML secundario, que también estará respaldado por otros dos archivos CSS y JavaScript propios.

A continuación, en la figura 20 podemos observar de forma general las tres capas que tendremos en nuestra aplicación.

<table>
<tr><td>3</td><td colspan="3">Aplicación</td></tr>
<tr><td>2</td><td colspan="3">Datos de audio</td></tr>
<tr><td>1</td><td>Dispositivo entrada audio</td><td>Dispositivo de salida de audio</td><td>Dispositivo salida texto</td></tr>
</table>

Figura 20: Capas de la aplicación.

En la primera capa se detalla los dos dispositivos utilizados para capturar la entrada y efectuar la salida del audio. En este caso se utilizan un micrófono con la respectiva tarjeta de audio del equipo al igual que los altavoces integrados en el equipo para la salida respaldados también por la tarjeta gráfica. Además, encontramos la propia pantalla del equipo para poder visualizar el texto informativo de las cadenas transcritas, así como las imágenes del visor y la propia interfaz de la pantalla principal.

En la segunda capa encontramos los datos de audio que serán generados a partir de la recepción de la voz mediante el dispositivo de audio de entrada y que serán gestionados por la aplicación. También se encuentra los datos de salida que serán reproducido la parte de la síntesis de la cadena resultante.

La última capa la de la aplicación es la encargada de pasar a texto el audio correspondiente y una vez se ejecute los diferentes métodos que se apliquen se podrá pasar de texto a audio con la respuesta correcta. También es la encargada de que las diferentes acciones como pulsar botones, pasar imágenes como configurar las voces disponibles o la tesitura… Mediante la utilización del documento JavaScript podremos configurar todas estas funciones en consonancia con el documento HTML, estas configuraciones la veremos en el capítulo 5.

4.5 Tecnología utilizada

Para el desarrollo de este proyecto se emplean diferentes tecnologías que no has permitido moldear la aplicación debido a que al utilizar la Web Speech API, se ha tenido que utilizar estas tecnologías. A continuación, procedemos a comentar cada una de ellas.

4.5.1 HTML

El código HTML [40], es un lenguaje de marcas de hipertexto, cuyo primer desarrollador fue Tim Bernes-Lee, pero debido al crecimiento mayúsculo de la web, surgió la necesidad de crear un estándar para que los navegadores y los desarrolladores pudieran basarse en unas mismas normas para escribir HTML. Actualmente lo desarrolla el *Wide Web Consortium* (W3C). En el HTML se contempla el lenguaje establecido para la creación de páginas web. Este lenguaje define las estructuras básicas y las marca o etiquetas, para definir el contenido de una página web, ya sea el texto, imágenes, vídeos entre otros. Considerado como el lenguaje web más importante es el lenguaje empleado por todos los navegadores.

Entre funcionalidades que encontramos en este lenguaje, al añadir un elemento externo a la página (por ejemplo, una imagen), este no se incrustará directamente en el código de la página, al contrario que en otros lenguajes, este utiliza directamente una referencia a la ruta donde se encuentra el elemento. Con esto la página web aparecerá solo texto y el navegador utilizado será el encargado de interpretar el código para poder visualizar el resultado final de la página, de aquí la búsqueda de la estandarización del código para que en los diferentes navegadores se visualice el mismo contenido.

A lo largo de la historia se han ido desarrollando diferentes versiones del lenguaje HTML, desde la versión HTML en 1991 pasando por la 2.0, 3.3, 4.01, XHTML hasta la HTML5 donde se han ido incorporando y suprimiendo diferentes características, para hacer posible una mayor eficiencia, facilitando así el desarrollo de las páginas web. Con estos cambios los navegadores tienen que ir actualizándose con los cambios efectuados, es por esta razón que algunos navegadores soportan versiones de HTML anteriores y, en algunos casos, la visualización en diferentes navegadores es diferente al interpretar una misma página web.

4.5.2 CSS

El código CSS (las siglas equivalen a Cascading Style Sheets) que en español son “Hojas de estilo en cascada” es un lenguaje que se emplea en el estilo/presentación. Muy usado para la configuración del diseño visual de los documentos web, e interfaces escritas y aplicaciones en HTML entre otros. Uno de los usos que tiene es al aplicarse junto con HTML y JavaScript, CSS es empleado por diversos sitios web para crear páginas con

diseños más atractivos, así como interfaces de usuario para aplicaciones web y diversas aplicaciones móviles.

Básicamente el funcionamiento que tiene es el de separar el contenido del documento en que se utilizaría el HTML del diseño de la forma de presentación de este, donde se aplicarían los tamaños y ubicaciones de las capas, cambios de colores, tipografías, tamaños entre otros. Esto ayuda a que la estructuración de los diferentes aspectos de la web sea más clara, reduciendo así la complejidad y la repetición en la estructura de código.

Este lenguaje es llamado estilo en cascada porque si hay duplicidad se aplican la propiedad configurada del elemento más externo, aplicando los atributos a los elementos que lo contienen.

4.5.3 JavaScript

Es un lenguaje de scripting que nos permite mejorar la iteración de la página web con el usuario. Nos permite controlar el contenido nuevo y dinámico, como el control de archivos multimedia y, en particular en nuestras capas gestionar la síntesis y reconocimiento de la voz. Otro de los usos que tiene este lenguaje es en el desarrollo de juegos y de aplicaciones para smartphones. Se puede diferenciar del lenguaje de programación Java en que Java tiene un tipado fuerte y es secuencial entre otras mientras que JavaScript es más permisivo y tiene un tipado débil además de asíncrono.

4.5.4 jQuery

Es una librería de gran utilización en JavaScript. Esta librería permite simplificar la manipulación y manejo de eventos en los documentos HTML, facilitando el uso de JavaScript en el sitio web. Simplifica funciones que en JavaScript nos costaría muchas más líneas de código logrando grandes resultados con menos código. Una de las ventajas que obtenemos al utilizar la jQuery es que nos permite de una forma más sencilla seleccionar elementos del HTML, de esta forma la capacidad de interactuar con este documento será más simple, así como el desarrollo de aplicaciones *AJAX*, como puntos fuertes. Para una mejor explicación, a continuación se ejemplifica al clicar un botón preconfigurado ocultara todos los párrafos del HTML, mediante el atributo hide().

$("p").hide();

La librería JQuery nos permite mediante el símbolo “$”, seleccionar todos los párrafos o el elemento que sea y realizar la función que deseemos de una manera más fácil entre otras funciones disponibles.

5. Desarrollo del aplicativo

En esta parte de centrar en el desarrollo de la aplicación de reconocimiento de la voz, ya en la parte anterior se ha detallado el flujo del funcionamiento del aplicativo. Este sería el esquema de los documentos creados para su implementación. La aplicación cuenta con un primer documento HTML "principal.html" donde se encuentra la estructura de la ventana principal de la aplicación, tiene su propio documento de configuración de la apariencia "estilo.css" así como para la configuración de las acciones en el documento JavaScript "principal.js". Este primer documento HTML invocará a un segundo HTML secundario. En este esta almacenado el diseño de la ventana secundaria "secundaria.html". Este documento HTML tiene también su propio documento CSS de configuración de la apariencia "estilo_secundaria.css" así como también un documento JavaScript "secundaria.js" para la configuración de las acciones en el HTML, en la figura 21 se observa la estructura.

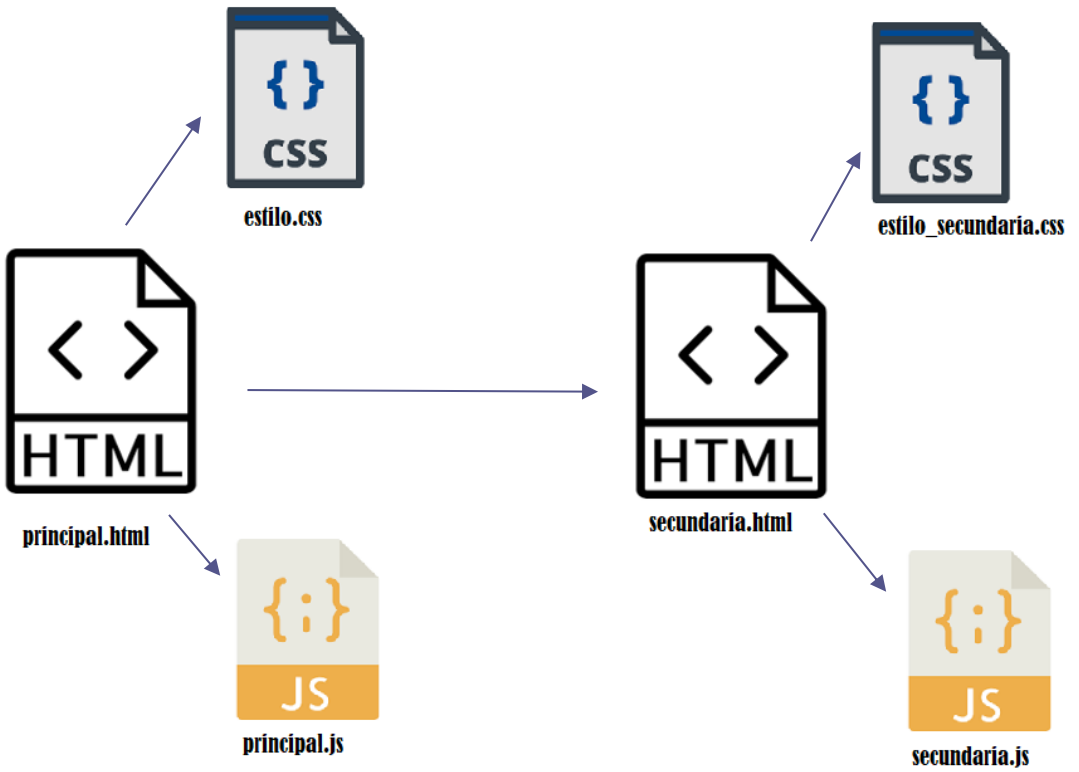

Figura 21: Esquema documentos creados para la aplicación.

Una vez detallada la estructura de los documentos creados, vamos a detallar el funcionamiento de cada uno, como en caso de esta aplicación ha sido creada para la utilización y prueba de métodos y eventos de la *Web Speech API*, nos centraremos más detalladamente en los documentos JavaScript que es donde realmente se encuentra la

utilización de esta librería. En el caso de los documentos de HTML y CSS hablaremos de ellos, pero de una forma más general.

Principal.html

Para empezar esta aplicación se ha creado una estructura principal sobre código HTML, en este se han configurado diferentes elementos para poder visualizar lo que teníamos en mente en el prototipado inicial. Como hablamos en el punto 4.1 de esta memoria para realizar esta estructura nos hemos basado en la w3.css de W3Schools. Es un tipo de estructura HTML/CSS de la cual hemos utilizado diferentes elementos.

Para englobar la aplicación se crea un container general:

** **

En este container se encuentra toda la interfaz, a excepción del *text-box* final, que contendrá diferentes clases.

Como se describió en el punto 4.5 de la memoria para los iconos que se han creado se ha utilizado diferentes tipos de iconos de la clase <i>, como el icono que nos permite la inicialización del reconocimiento:

<i class="fa fa-microphone fa-4x micro">

De esta forma para los iconos secundarios que hemos configurado dentro de una clase iconos, también se han configurado de la misma forma que el icono micrófono, añadiendo el atributo <a href>, para así poder indicar la ruta que se abrirá.

Para la configuración de la columna con las etiquetas, se ha utilizado la clase *w3-row* de w3.css configurando cada etiqueta con las opciones que se han hablado en el punto 4.5 (voces disponibles, tesitura, velocidad y comandos por voz disponibles), en este caso el código utilizado que podemos observar en la figura 22 donde el primer seria la configuración de la etiqueta y el segundo seria la configuración del container de la etiqueta. Esto nos permite tener la función al pulsar en una de las etiquetas aparecerá su contenido, ocultando el contenido del resto de las etiquetas.

```
<div class="w3-row">
<a href="javascript:void(0)" onclick="abrirSeleccion(event, 'Voces');">
    <div class="w3-third tablink w3-bottombar w3-hover-light-grey w3-padding">Voces disponibles</div>
</a>

<div id="Voces" class="w3-container seleccion" style="display:none">

   <select>

    </select>
</div>
```

Figura 22: Caputra configuración etiquetas.

En la segunda y tercera etiqueta correspondientes a las etiquetas de tesitura y velocidad, se han configurado mediante un *<label for>* en la figura 23 se puede apreciar su configuración.

```
<div id="Tesitura" class="w3-container seleccion" style="display:none">

   <label for="pitch"></label><input type="range" min="0.1" max="2" value="1" step="0.1" id="tesitura">
   <div class="tes-valor">1</div>
   </div>
```

Figura 23: Captura configuración etiquetas.

Por último, configuramos una clase de tipo audio para que cuando se accione el botón de iniciar el reconocimiento se ejecute una campana de advertencia:

<audio class="sound" src="chime.mp3"></audio>

Secundaria.html

Como en la aplicación principal para realizar este visor de imágenes también hemos utilizado elementos de la w3.css. En este caso empezamos por definir la clase donde se visualizan las imágenes:

** **

La clase de w3.css que utilizamos es la "w3-content", este contenedor incluirá las siete imágenes que hemos seleccionado para este visor. Cada imagen estará definida por una clase *img*.

Después del contenedor de las imágenes se ha definido los iconos como en el caso de la ventana principal y de la misma forma hemos definido el icono del micrófono. A continuación, se han definido tres botones de la clase de w3.css *<button class=" w3-button w3-light-grey">* para los botones de anterior, siguiente y atrás. De la misma forma se han definido los siete botones en la parte de abajo con cada número de imagen.

Por último, se han definido dos "text-box", como se menciona en el punto 4.5 de la memoria, para la gestión de la cadena y confianza del reconocimiento realizado, así como los errores, que se detecten y al final se ha definido un panel propio de la w3.css que contiene la tabla con la información de las cadenas prestablecidas.

Hojas de estilo (estilo.css y secundaria_estilo.css)

En estos dos archivos se han configurado todas las estructuras, colores, tipografías... entre otras características.

Por citar algunas configuraciones más generales, ya que no vamos a centrarnos en el aspecto visual de la aplicación encontramos una configuración de la clase "general", que

es la clase referida al container principal. En la figura 25, se puede observar configuraciones como background que es la configuración del fondo del container, donde hemos optado por un color con gradiente o los márgenes que se han seleccionado para esta ventana, así como la tipografía que se emplea.

```css
.general{
    border: 1px solid #f2efe2;
    background: linear-gradient(to bottom, #0066ff 0%, #ff99cc 100%);
    padding-top: 20px;
    padding-right: 40px;
    padding-bottom: 60px;
    padding-left: 100px;
    margin-top: 100px;
    margin-right: 90px;
    margin-left: 90px;
    font-family: 'Saira Stencil One', cursive;
    font-size: 20px;
}
```

Figura 24: Captura configuración .CSS.

JavaScript "principal.js"

En este apartado vamos a detallar el funcionamiento del documento JavaScript de la ventana principal que será el que nos servirá para explicar el tratamiento que tiene el audio al interaccionar con la aplicación. En este documento es donde se efectúan todas las acciones que hemos utilizado mediante la utilización de la *Web Speech API*.

```
window.SpeechRecognition = window.webkitSpeechRecognition || window.SpeechRecognition;
```

Figura 25: Captura código inicialización recognition.

En la figura 25 se inicializa la interfaz responsable del reconocimiento en nuestro navegador que nos permitirá iniciar el proceso de voz a texto. El *window.SpeechRecognition* es la interfaz encargada de activar el proceso en Firefox mientras que la *webkitSpeechRecognition* es la interfaz encargada de activar el reconocimiento en Chrome.

Una vez se carga la página por primera vez una de las funciones que primero se ejecuta es la función listavocesnavegador(). Esta función nos devolverá en <select> definido en la etiqueta de voces disponibles de la ventana principal. Esta función utiliza diferentes métodos de la *Web Speech API*.

- synth.getVoices(): el método getVoices() de la interfaz SpeechSynthesis que nos devuelve una lista de objetos SpeechSynthesisVoice. Con esta función almacenamos en un array previamente definida en este caso “voces”, donde se guardan estos objetos.
- *sort():* Mediante diferentes funciones propias como son el método *sort()* que lo que nos permite es ordenar el array de los objetos descritos anteriormente. Se crea un elemento del tipo *option* que nos permite mediante el *setAttribute* atribuirle a la variable el ‘dat-name’ y ‘data-lang’ de las voces.

```
var voces = [];
function listavocesnavegador() {
  voces = synth.getVoices().sort(function (a, b) {
    const nombre_a = a.name.toUpperCase(), nombre_b = b.name.toUpperCase();
    if (nombre_a < nombre_b) return -1;
    else if (nombre_a == nombre_b) return 0;
    else return +1;
  });
  var selectedIndex = sel_voz.selectedIndex < 0 ? 0 : sel_voz.selectedIndex;
  sel_voz.innerHTML = '';
  for (i = 0; i < voces.length; i++) {
    var option = document.createElement('option');
    option.textContent = voces[i].name + ' (' + voces[i].lang + ')';

    if (voces[i].default) {
      option.textContent += ' -- DEFAULT';
    }


    option.setAttribute('data-name', voces[i].name);
    option.setAttribute('data-lang', voces[i].lang);
    sel_voz.appendChild(option);
  }
  sel_voz.selectedIndex = selectedIndex;
}
```

Figura 26: Captura función listavocesnavedor.

Para el reconocimiento se define una constante synth: **const synth = window.speechSynthesis;** y para la síntesis se define la constante recognition: const **recognition = new SpeechRecognition();**.

Una vez se ha clicado el icono micrófono se ejecutará el sonido que habíamos configurado en el HTML y se empezará el reconocimiento con la ejecución de dictate(), que es el código que se aprecia en la figura 27.

```
icon.addEventListener('click', () => {
  sound.play();
  dictate();
});
```

Figura 27: Captura código inicialización.

Una vez se ejecuta *dictate()* se empieza el procesos de reconocimiento.

- *start():* Este método del *Web Speech API*, es el que nos permite iniciar el reconocimiento. En nuestro caso se inicializa la constante *recognition.start()*. Una vez se inicie se ha configurado una función para que cambie el icono y así tener una guía de cuando se inicia el reconocimiento.
- *onerror:* Esta propiedad del *Web Speech API* es la que nos permite comprobar si al iniciarse el reconocimiento se ha producido algunos errores. En nuestro caso se ha definido cadenas a reproducir mediante la síntesis de los tipos de errores *'no-speech', 'audio-capture' y 'network'*. El primero se dispara si no recibe ninguna voz, el segundo es para cuando no recibe audio y el tercero para cuando falla la conectividad mientras se estaba realizando las tareas de reconocimiento. En la figura 28 podemos observar cómo se ha definido estas funciones.
- onend(): Esta evento se dispara cuando se termina el sistema de reconocimiento de voz. Este evento se ha utilizado para cuando termine el reconocimiento que el elemento del micrófono vuelva a su estado inicial, indicara que se mantiene a la espera.

```
recognition.onerror = function (event) {
  /** speak(error_en);
}*/

  if (event.error == 'no-speech') {
    speak(error_en);
  }
  if (event.error == 'audio-capture') {
    speak(error_cap);
  }
  if (event.error == 'network') {
    speak(netwo_error);
  }


};
  const error_en = () => {
    return `No ha sido posible el reconocimiento de ninguna
     cadena o se ha producido un error, pruebe de nuevo.`
  };
```

Figura 28. Captura código onerror.

Si no se produce ningún error seguimos con los métodos de la API empleados, en este caso llegamos hasta la utilización de la propiedad *recognition.**onresult...*** esta propiedad representa el evento que se ejecuta cuando el servicio de reconocimiento devuelve un resultado. Aquí aparecen dos constantes que hemos definido que nos servirán la primera para almacenar la cadena una vez transcrita que será la constante speechToText, cuyo funcionamiento ya ha sido explicado en el punto 4.2 de la memoria seria definido de esta forma:

const speechToText = event.results[0][0].transcript;

La segunda constante seria la que nos permite averigurar la calidad del reconocimiento de la voz. Es la propiedad de la interfaz *SpeechRecognition* "*confidence*". Con el uso de la confianza que nos devuelve un valor de cero a uno, nos permite evaluar si la calidad es la correcta pudiendo hasta filtrar para que se siga con los métodos o se emita un error de que no tiene suficiente calidad. En nuestro caso la usamos tanto como método informativo como para que nos devuelva una advertencia si no supera un umbral mínimo fijado en 0,5. Esta propiedad se ha definido de la siguiente forma:

const confid = event.results[0][0].confidence;

A continuación, en la figura 29 se puede observar un flujo general del proceso descrito.

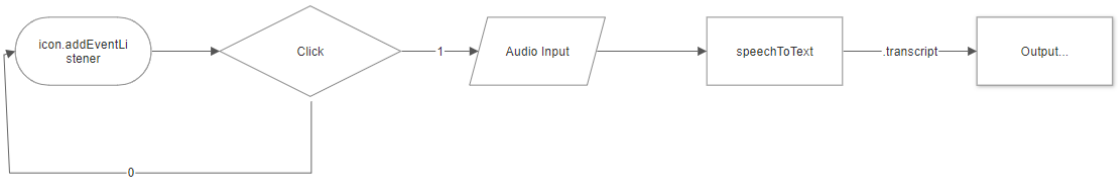

Figura 29: Flujo general.

Siguiendo con los métodos y propiedades de la *web speech API* llegamos a la parte de la síntesis. En esta parte vamos a empezar por el método speak() que se ha definido. Al iniciarse se define una variable del tipo *SpeechSynthesisUtterance*. A continuación se consulta la voz que se ha consultado, recordemos que se consultan en listavocesnavegador(), también se consulta la tesitura y velocidad y una vez se tiene toda la información se invoca el método speak() en la constante creada anteriormente para la síntesis "*synth*".

synth.speak(utterThis);

Si seguimos analizando la parte de la síntesis, encontramos las propiedades pitch y rate que en este caso equivalen a la tesitura o tono de la voz y la velocidad ambas en la reproducción final. La configuración que se ha empleado consiste en la utilización del método onchange el valor de la variable creada para la tesitura y la velocidad, en que se observará si el valor ha sido modificado. Si se ha modificado el valor, entonces dentro del método *sepak()* como se observa en la figura 30, cambiaremos el valor del pitch y el rate, de la variable de síntesis creada "*utterThis*".

```
const speak = (action) => {
  utterThis = new SpeechSynthesisUtterance(action());
  var selectedOption = sel_voz.selectedOptions[0].getAttribute('data-name');
    for(i = 0; i < voces.length ; i++) {
      if(voces[i].name === selectedOption) {
        utterThis.voice = voces[i];
        break;
      }
    }
    utterThis.pitch = pitch.value;
    utterThis.rate = rate.value;
  synth.speak(utterThis);
};
```

Figura 30: Captura del método speak.

Por lo que respecta a las cadenas predefinidas para realizar acciones y disparar eventos, ya sean de respuesta por síntesis o de manejo de la interfaz, se han definido una estructura tipo *switch* donde evaluamos si la cadena almacenada en la constante *speechToText* equivale a una cadena predefinida, de coincidir se ejecutara el método correspondiente. En la figura 31 aparece un ejemplo con una captura del código realizado.

```
case (speechToText.includes('abrir youtube')):
    abrirenNavegador('http://www.youtube.com');
    break;
```

Figura 31: Captura switch.

A parte de la utilización de la interfaz de síntesis se ha definido diferentes constantes y funciones que nos permiten realizar acciones. Se ha definido un método *getTime* para cuando se encuentre la cadena "que hora es" se reproduzca una cadena con la hora actual. Otro de los métodos que se han definido es el *getDate* que es similar al *getTime* pero que en este caso y tras detectar la cadena "que día es hoy" reproducirá una cadena con la fecha del día actual. También se define la función "abrirenNavegador", que nos permite realizar la acción de abrir en una pestaña nueva la url deseada, tras detectar una cadena predefinida.

Otro de los métodos que se han desarrollado es el que nos devuelve la condición climatológica de una ciudad. En este caso mediante el uso del API meteorológico de *openweathermap* y la utilización del objecto promesa *fetch()*, su funcionamiento consta de que toma la ruta que en este caso le hemos puesto la del openweathermap y la posición de la array donde esta almacenada la ciudad. Es entonces cuando se devuelve un objeto Promise, que contine la respuesta HTTP, a continuación para extraer el contenido en JSON, utilizaremos el método json(). A partir de ahora podremos leer los datos en json y incluirlos en la cadena de la síntesis de respuesta.

JavaScript “secundario.js”

Por último, vamos a explicar el desarrollo del documento JavaScript de la ventana secundaria en la cual se ha configurado un visor de imágenes controlado por la voz. Al tener métodos coincidentes con la ventana principal vamos a explicar algún método y realizar alguna explicación de aclaración con respecto a las diferencias entre el documento de la ventana principal y este.

La gestión de los errores en este documento llevará una programación igual a la ventana principal, aunque en el tema del output de la información en esta ventana se llevará a cabo mediante un text-box.

En esta venta no se define una constante para la síntesis ya que no se ha utilizado, si que se define la constante *recognition* de la misma forma que se definió en la ventana principal.

const recognition = new SpeechRecognition();

En este caso también se ha seguido una estructura *switch* para la comparativa de las cadenas prestablecidas y con esto poder ejecutar las funciones pertinentes, como podemos observar en la figura 32.

```
switch (event.results[0].isFinal) {
  case (speechToText.includes('siguiente imagen')) || (speechToText.includes('siguiente')):
    siguienteImg();
    break;
  case (speechToText.includes('anterior imagen')) || (speechToText.includes('anterior')):
    anteriorImg();
    break;
  case (speechToText.includes('primera imagen')) || (speechToText.includes('primera')):
    priImg();
    break;
```

Figura 32: Captura de código con include.

Para concluir se han creado diversas funciones que al coincidir con la cadena prestablecida ejecutar los botones creados, como se puede apreciar en la figura 33. En este caso la función tiene un funcionamiento simple, se han creado diferentes variables en este caso “sig” que contiene el elemento del botón siguiente. Una vez se llama esta función accionará el botón con el *click()* y realizara la acción pertinente.

```
function siguienteImg() {
  // se simula el cklick del boton en el html
  sig.click();
}
function anteriorImg() {
  // se simula el cklick del boton en el html
  ant.click();
}
```

Figura 33: Captura funciones diversas.

5.1 Manual de usuario

Para la explicación de esta aplicación vamos a utilizar capturas de la aplicación para explicar el funcionamiento de los elementos configurados en esta.

5.1.1 Ventana principal

En primer lugar, al cargar la aplicación aparece la ventana principal. En esta ventana se encuentran diferentes elementos como el icono de micrófono que es el que nos permitirá iniciar el reconocimiento, que podemos observar en la captura de la aplicación en la figura 15.

Figura 34: Captura de los iconos ventana principal.

A continuación, como se observa en la figura 21, encontramos una serie de iconos que se han introducido por si ocurre un fallo en el sistema de reconocimiento, poder accionarlo manualmente. En este caso se han programado diferentes funciones, el icono primero será el de la galería que nos abrirá la ventana secundaria, el segundo icono nos abrirá una nueva pestaña en el navegador, el tercer icono nos llevará a la página de Twitter, el cuarto a la página de YouTube y por último el quinto que nos llevará a la página de Facebook. Estas funciones se iniciarán cuando el sistema de reconocimiento reconozca la cadena prestablecida.

En la parte inferior de la interfaz encontramos una sección que se divide en 4 opciones donde al accionar cada una se desplegarán diferentes elementos.

Figura 35: Captura con voces disponibles.

La primera opción es la correspondiente a las voces disponibles que tiene el navegador, es del tipo *select tag* donde aparecerá una lista de las voces disponibles con el título, como se observa en la figura 22. Nos permitirá seleccionar la voz de contestación para la síntesis.

En la segunda opción de la Tesitura al desplegarla encontraremos en su contenido un *label for* comprendido entre los valores 0.1 y 2. Esta opción nos permitirá ajustar el valor de la tesitura de la voz para la síntesis, cuando menor sea el valor más grave será la voz si por el contrario aumentamos el valor será más aguda, en la figura 36 podemos observar una captura de esta opción.

Figura 36: Captura de label for.

En la tercera opción la función es prácticamente igual que para la opción de la tesitura, pero en este caso será para la velocidad de reproducción de la síntesis. Como en la anterior opción cuando disminuyamos el valor del *label for* más lento será la reproducción de la síntesis, en cambio cuando augmentemos este valor la velocidad de reproducción se incrementará.

Por última a modo de guía hemos creado una etiqueta complementaria con el botón o función disponible y el comando por voz que se tiene que recitar para activar esta

función. Simplemente el contenido de la etiqueta es el de una tabla con las diferentes funciones, se puede observar en la figura 37.

Figura 37: Captura diferentes funciones.

Estos dos grupos de elementos que se han descrito forman parte de un container general que los agrupa. Finalmente, fuera de este container hemos configurado un *text-box* con la cadena reconocida entre comillas más el intervalo de confianza que se obtiene al convertir le audio a cadena. En la ventana principal los errores no aparecen en el text-box debido a que si se genera algún error de falta de audio u otro tipo será el sistema de síntesis el que nos advierta mediante la voz que se ha visto en el capítulo 5. Finalmente podemos observar en la figura 38, la captura de la ventana principal en su estado inicial.

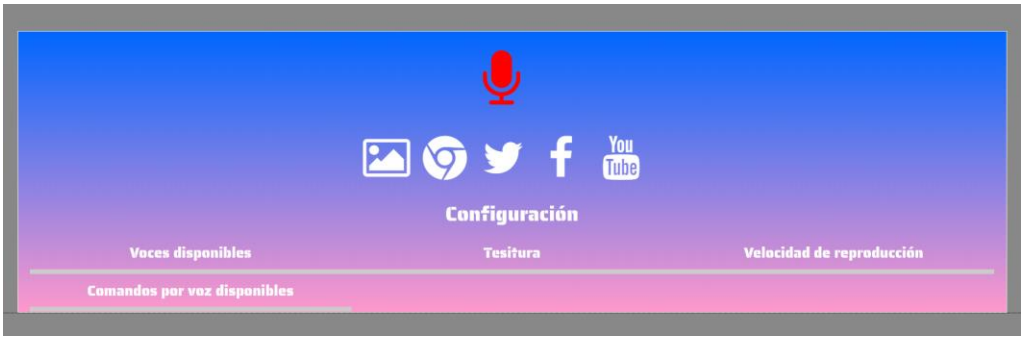

Figura 38: Captura ventana principal.

5.1.2 Ventana secundaria

Una vez se ha explicado el contenido de la ventana principal, vamos a detallar que las particularidades de este visor de imágenes. Esta ventana comprende un visor de imágenes básico la función de este será la de poder utilizar los comandos de voz para ir cambiando de ciertas imágenes que hemos prestablecido.

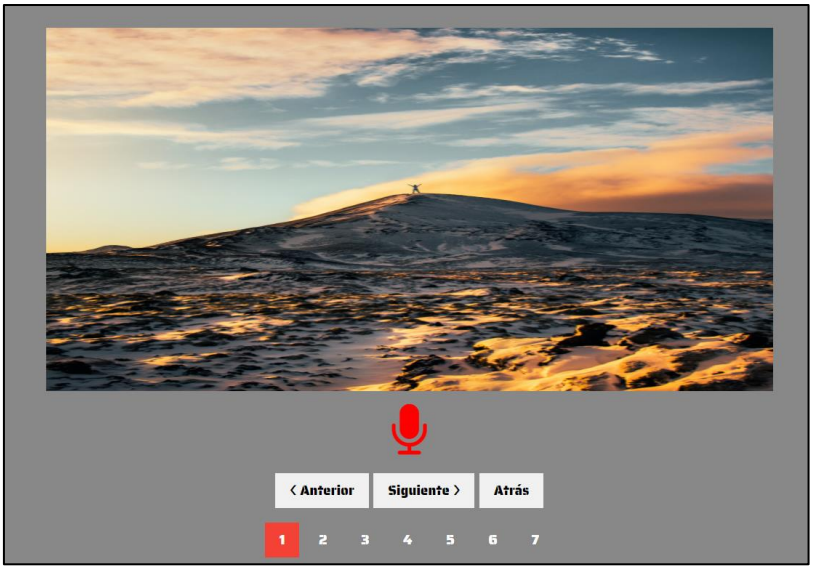

Figura 39: Captura ventana secundaria.

En primer lugar, esta ventana se cargará una vez en la ventana principal se ejecute la acción de abrir la galería. Una vez se carga la ventana secundaria nos aparecerá en la parte superior la imagen que se está visualizando, para esta aplicación hemos configurado siete imágenes. A continuación, se han configurado un *icon* de tipo micrófono para que una vez se pulse inicie el reconocimiento. Como en la ventana principal se han configurado a continuación de este *icon* tres botones por si falla el reconocimiento que se pueda accionar de manera manual, el primero nos permitirá la función de ir a la imagen anterior a la que en ese momento se está visualizando, el segundo nos permitirá ir a la imagen siguiente a la que se está visualizando, por último se configura el botón atrás que nos permitirá volver a la ventana principal del aplicativo. Después de estos tres botones se han configurado siete botones con el número de cada imagen, que nos permitirán en caso de fallo en el sistema de reconocimiento de voz ir directamente a la imagen, en la figura 39 podemos observar una captura de esta configuración.

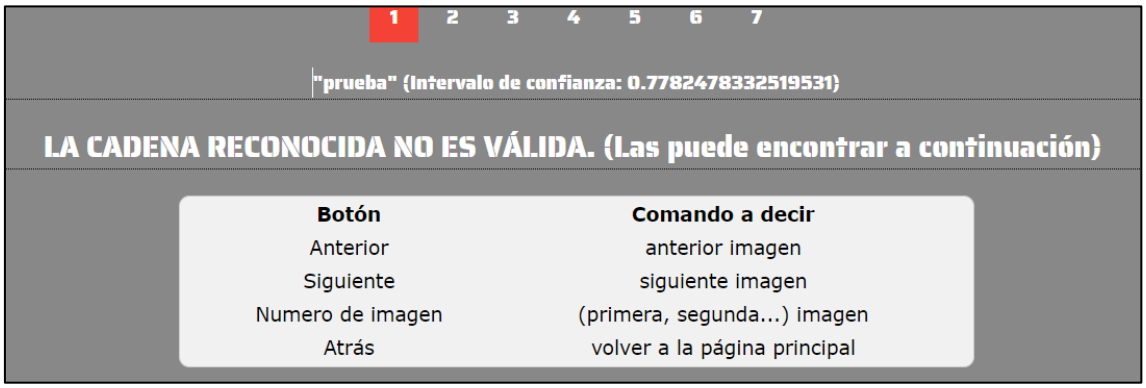

Figura 40: Captura de elementos inferiores.

Después de esta primera configuración se ha configurado dos *text-box*, uno como en la ventana principal para visualizar la cadena que se reconoce entre comillas a modo de

prueba de fallo más la confianza del reconocimiento y luego tenemos el otro *text-box* que nos servirá para visualizar si se produce algún tipo de fallo y la cadena que se ha prestablecido que devuelva. Por último, a modo de ayuda hemos configurado un panel tipo tabla con los comandos por voz disponibles en esta ventana. En la figura 40 podemos visualizar el contenido de esta configuración.

6. Pruebas

Para las pruebas que se realizan utilizamos como herramienta principal el navegador Chrome, mediante la utilización de un web server que nos permite visualizar la aplicación mediante el servidor local.

En primer lugar, vamos a especificar que navegadores soportan la *web speech API* en la actualidad. Tras varios intentos con en el navegador Firefox con la última versión disponible, donde se habilitó en la configuración de los banners *("media.webspeech.recognition.enable" y "media.webspeech.synth.enabled")*, se observa que al ejecutar la aplicación no realizaba ningún método de las interfaces de la *Web Recognition API*. Se realiza una búsqueda para solventar el problema y se encuentra información online donde se va actualizando las disponibilidades de la interfaz de reconocimiento en la actualidad en los navegadores más utilizados. En lo relativo a la nula funcionalidad en la actualidad del Firefox encontramos que es debido a un tema de permisos con el soporte, actualmente a la espera de aprobación. Se puede consultar de forma actualizada los navegadores que soportan de forma completa estas funcionalidades. [19].

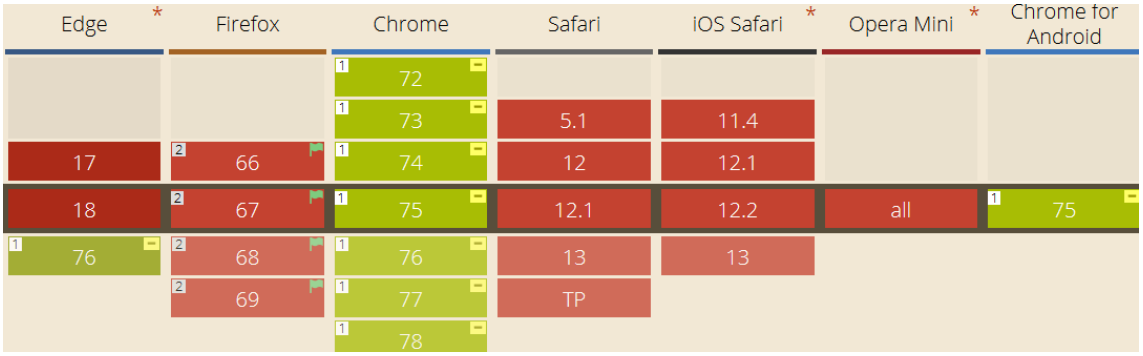

Figura 41: Disponibilidad de interfaz en navegadores.

Como se puede observar actualmente el único navegador compatible es Chrome, ya que la versión 76 de Edge aún no está disponible. De esta forma las pruebas se van a realizar con el uso del navegador Chrome, ya que en Firefox el paquete de voces para la síntesis es menor y la únicas que obtenemos son en inglés y como la configuración de las cadenas ejemplo ha sido en castellano nos parece los más indicado utilizar el Chrome que si tiene estas voces.

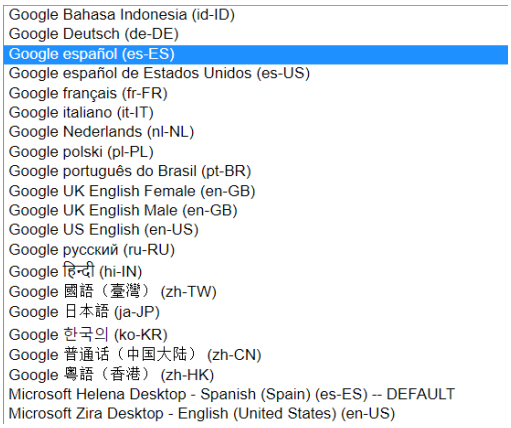

Figura 42: Voces Disponibles.

Antes de seguir con las pruebas vamos a especificar el catálogo de voces en los diferentes idiomas disponibles para este navegador, que se pueden observar en la figura 42, que se extrae de la propia aplicación. En total son 21 voces disponibles, destacar los tipos de voz tanto femenina como masculina en el Ingles Británico (en-GB).

Tras unas primeras pruebas realizadas con una velocidad intermedia a la hora de decir la frase, se observa que el reconocimiento es bastante exacto, al hacer preguntas nos la acentuación en la diferenciación en palabras tónicas y átonas monosilábicas se ejecuta de manera bastante correcta, los signos de exclamación e interrogación no se transmiten.

Cabe indicar que el idioma de entrada establecido es el castellano (es-ES).

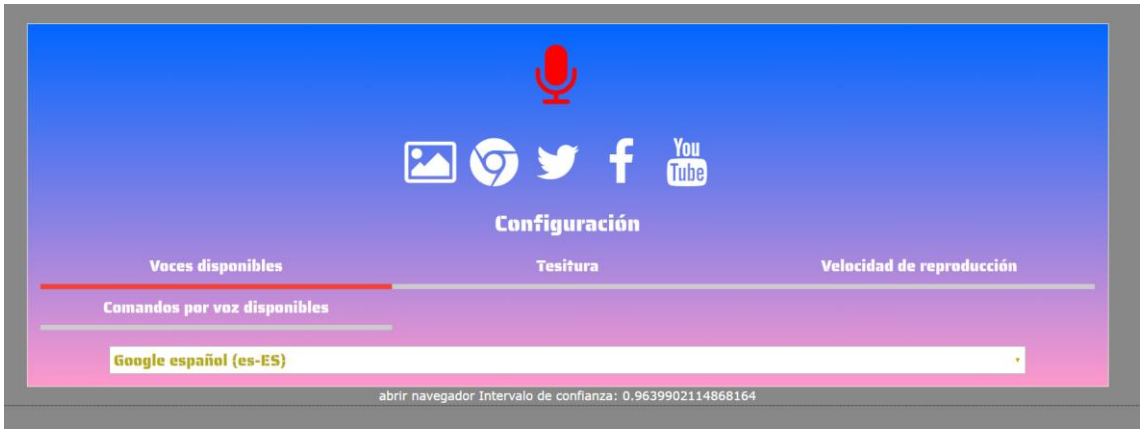

Figura 43: Cadena "abrir navegador".

Tras el reconocimiento de la cadena prueba “abrir navegador”, la síntesis ha ejecutado correctamente el sintetizado de la cadena que se había prestablecido y de esta forma se ha abierto una nueva ventana de navegador con una confianza del 0,96, como se captura en la figura 43.

La segunda prueba que ser realiza es para observar cómo se comporta el sistema de reconocimiento cuando la cadena que se captura tiene alguna palabra en ingles del algún producto internacional, como es el caso de Facebook, YouTube y Twitter entre otros. Al recitar la cadena “abrir Facebook”, no se encuentra ningún problema en el reconocimiento de esta palabra originaria del inglés, es decir el sistema de reconocimiento en su diccionario léxico/gramático tiene excepciones para poder reconocer cadenas de marcas frecuentemente utilizadas y no tener problemas. Se puede observar en la figura 44, donde la confianza del reconocimiento es superior al 0,96.

Figura 44: Captura tras "abrir Facebook".

Las siguientes pruebas, se han realizado en la ventana secundaria o visor de imágenes. En este caso vamos a comprobar el correcto funcionamiento cuando se captura las diferentes cadenas para realizar acciones sobre el visor.

En esta primera captura que la podemos encontrar en la figura 45, observamos el estado inicial de la ventana.

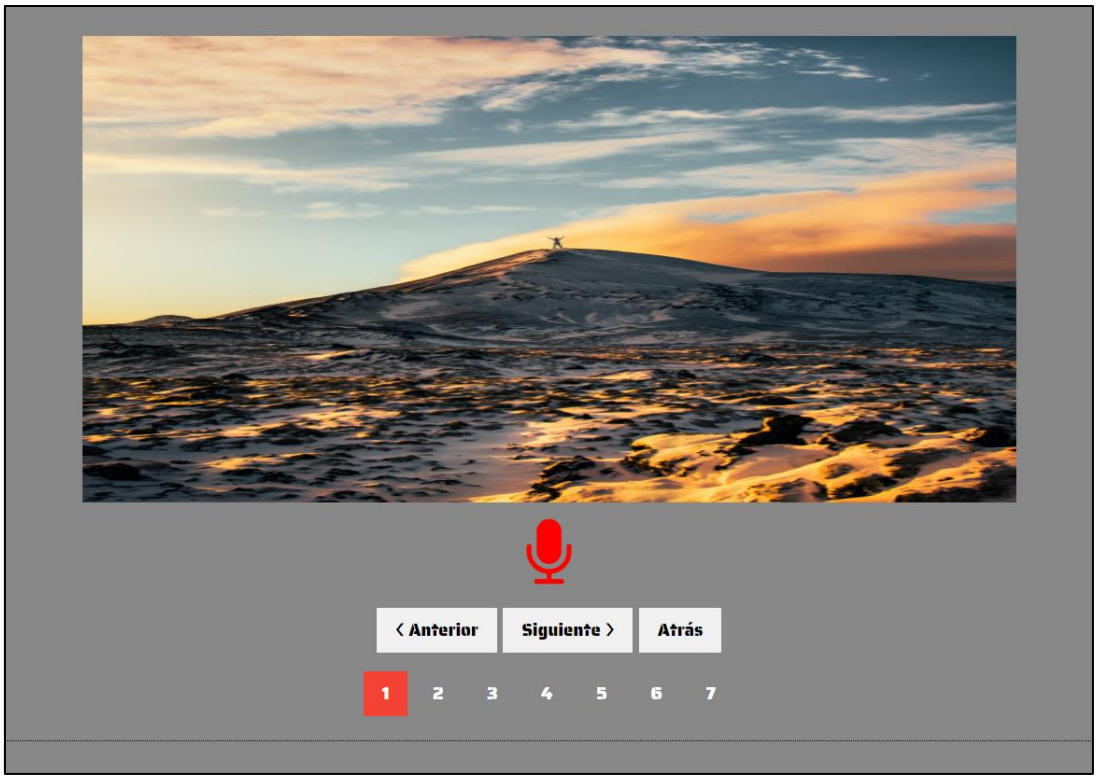

Figura 45: Ventana secundaria inicial.

En primer lugar, vamos a realizar prueba de errores. El primer error que vamos a detectar es cuando no recibe ningún audio que se pueda transformar a una cadena, como se observa en la figura 46.

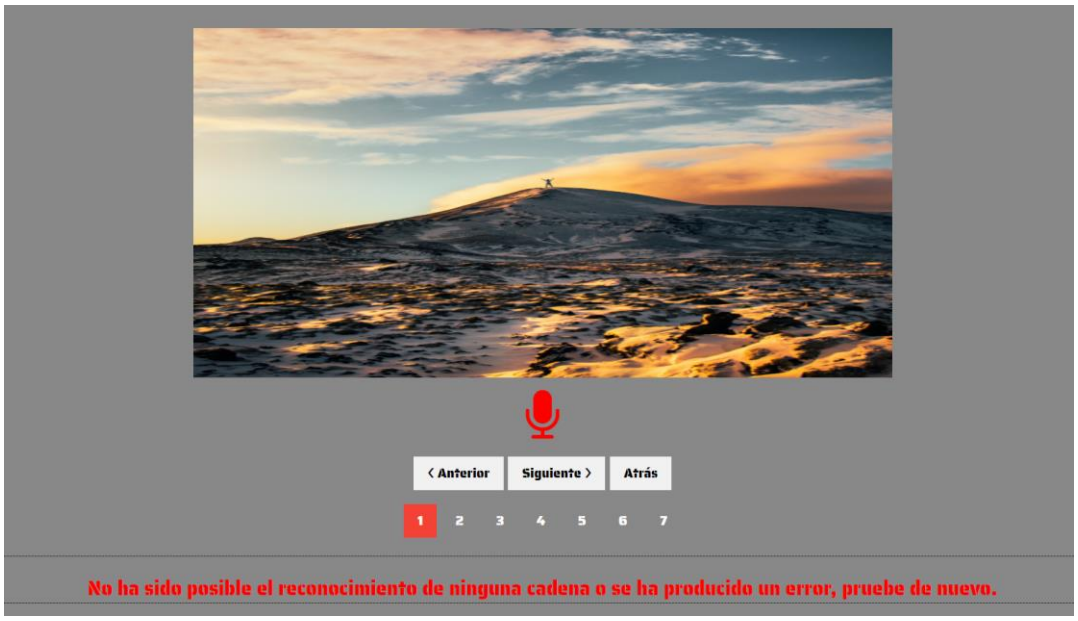

Figura 46: error no-speech.

El segundo caso de error que estudiamos será cuando la cadena que se detecta no coincide con las prestablecidas anteriormente. En este caso al no coincidir con ninguna cadena de las que están prestablecidas, nos devolverá la cadena que se observa en la figura 47.

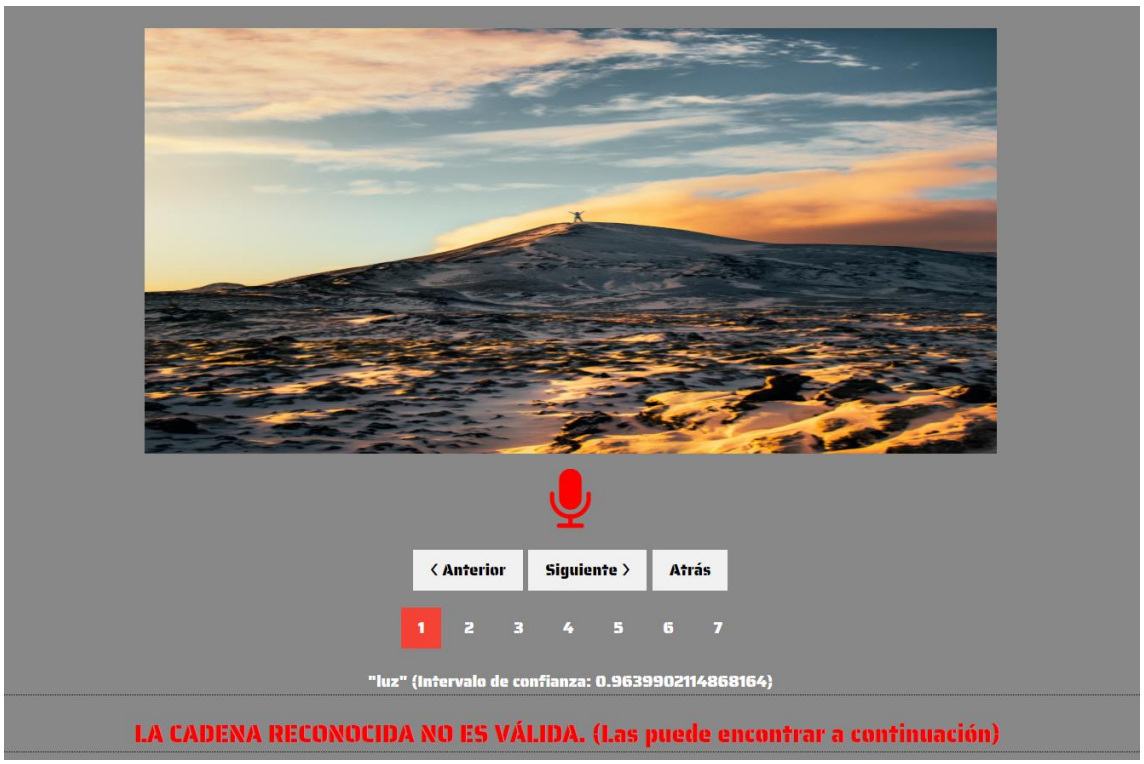

Figura 47: Captura cadena reconocida no válida.

Ahora se van a realizar pruebas para la comprobación del botones anterior, siguiente y los botones numéricos de imágenes que se han establecido. Para esta prueba se han realizado pruebas con las cadenas “anterior imagen”, “siguiente imagen” y (primera, segunda…) + imagen. En este caso la confianza que nos devuelve la captura de estas cadenas tanto a velocidad de habla normal como si aumentamos la velocidad está por encima del 0,8, es por esto que observamos que el sistema de reconocimiento en cadenas pequeñas tiene un funcionamiento bastante acertado, en la figura 48 podemos observar los diferentes cambios de pantalla.

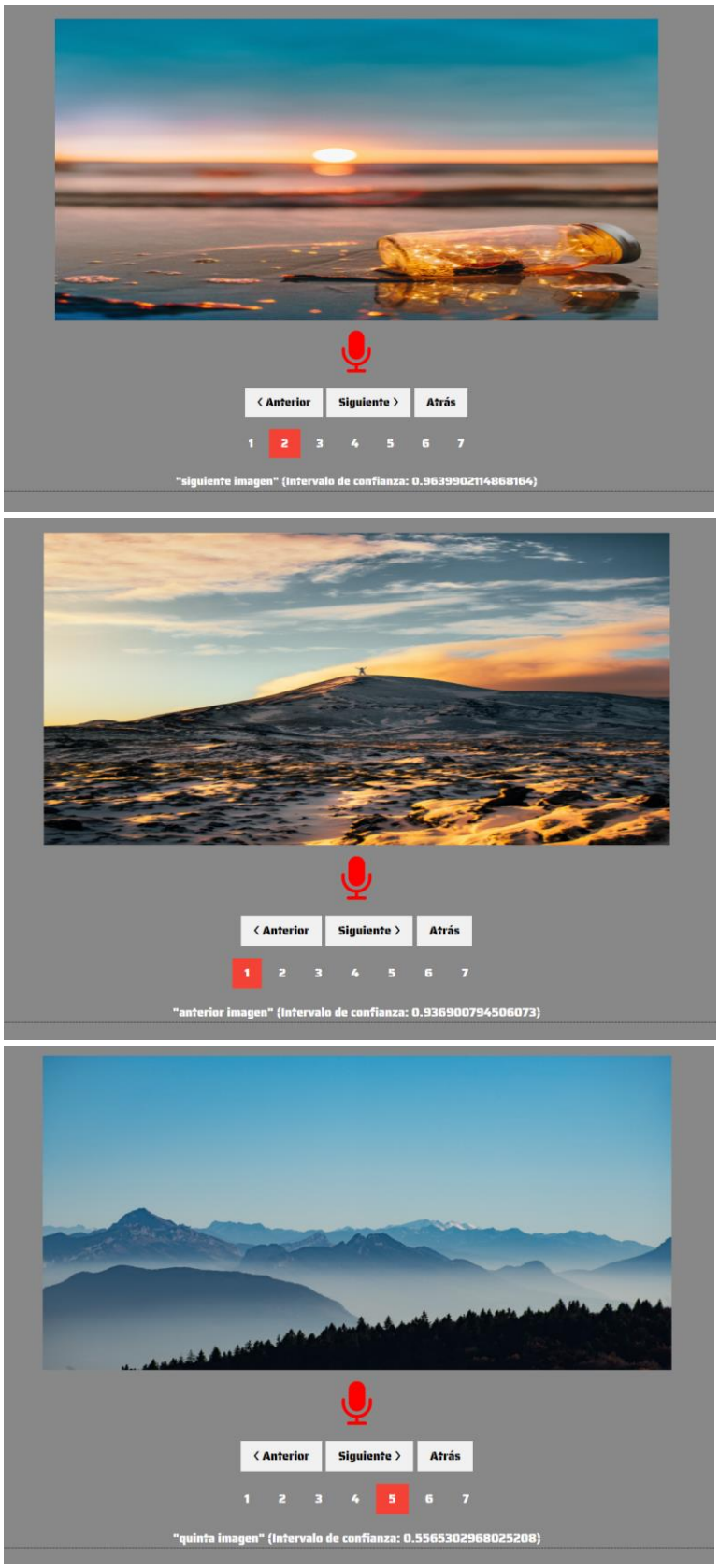

Figura 48: Capturas funciones varias.

Por último, la última prueba que se realiza es en el aspecto de la síntesis. En primer lugar, se configurarán estos valores de inicio:

- Tesitura con valor a 1.5
- Velocidad con valor a 1.5
- Idioma Google Español (es-ES)

Para estos valores observamos que la voz aún es entendible, a partir de esto valores con el aumento sobre todo de la velocidad el entendimiento se vuelve más complexo.

Por lo que se trata a otros idiomas se hicieron pruebas con cadenas en Ingles y uno de los aspectos a destacar del sistema de síntesis, en este idioma podemos seleccionar entre voz masculina y femenina.

En prueba realizadas para el reconocimiento con cadenas que no tienen ninguna función determinada en nuestra aplicación se observa que cuando se aumenta la velocidad con la que se habla, aunque la confianza que nos devuelve es buena para cadenas largas el sistema no de transcribir correctamente ciertas palabras.

Por último, se realizan pruebas con ruido de fondo. En este caso en particular ya con una velocidad intermedia del habla se producen errores producidos por este ruido, el sistema como con los casos de velocidades altas en condiciones normales, no llega a asimilar las cadenas o solamente parte de ellas.

7. Conclusiones

En este proyecto se han utilizado tres de los lenguajes más utilizados para el desarrollo web de aplicaciones. Como es el HTML, CSS y JavaScript, este último es el que nos ha permitido poder utilizar los métodos disponibles en la *Web Speech API*. Por tanto, he profundizado en el desarrollo de diferentes utilidades en métodos y librerías de JavaScript.

Al no estar oficializada esta librería, el tema de compatibilidades con los diferentes navegadores, así con otros tipos de sistema ya sean móviles como otros tipos de sistemas de reconocimiento de voz no son compatibles por ahora. Este aspecto nos ha limitado la utilización de esta tecnología al realizar pruebas a solamente un navegador.

Se han encontrado dificultades a la hora de utilizar las cadenas reconocidas por el sistema de reconocimiento. Entre ellas encontramos que diferencia las minúsculas de las mayúsculas, pero de manera aleatoria, sin seguir un patrón natural. En el caso de los números los reconoce mediante cadena *String* su manipulación para comparar con otros enteros es compleja, teniendo que pasarlos de manera manual a variables, llegando a ser muy costoso para un gran número de enteros a pasar. Se observa que, para completar campos simples, así como navegar y activar enlaces, botones, insertar texto en elementos, funciona correctamente.

Como diferencias que encontramos más significativas de esta de esta API respecto a los sistemas de las grandes compañías, es la eficacia. Este sistema aún está en desarrollo y el problema de compatibilidades y la velocidad de respuesta aún este lejos de estas compañías. Se observa que aún está en fase experimental y que el soporte que nos puede aportar es limitado. Pero después de las pruebas realizadas y los diferentes eventos y métodos que contiene, se observa que lleva un buen camino para consolidarse como la pauta a seguir.

7.1 Relación del proyecto con las asignaturas cursadas

Sin entrar en aspectos básicos vistos en las asignaturas de programación, como sentencias, *Arrays* entre otros, estas es la asignatura que más relacionada esta es Desarrollo web. En esta asignatura que se cursó en tercer curso dentro de la rama "Tecnologías de la información", es donde se pudo aprender sobre las tecnologías web utilizadas, como son el HTML, CSS, JavaScript entre otras. En esta asignatura se introdujo la utilización de estas tecnologías en conjunto, para poder implementar pequeñas aplicaciones web. Mediante el aprendizaje de la manipulación de diferentes elementos de HTML mediante la utilización de JavaScript. En este aspecto y al a ver utilizado diferentes librerías como la *JQuery*, la introducción de la *Web Speech API* nos ha sido más sencilla.

7.2 Trabajos futuros.

Uno de los aspectos que no se incluye en este proyecto es el tema de la seguridad, en las diferentes sistemas y aplicaciones creados por las grandes compañías comentados en el punto 2 de esta memoria.

En los últimos años y con la salida al mercado de estos sistemas como el "Amazon Echo" o el "Google Home" en cuanto a altavoces inteligentes se refiere y en el campo de la telefonía móvil los sistemas Siri por parte de Apple o el Google Assistant de Android. Al ser sistemas que están en espera de que el usuario pronuncie las palabras pactadas para su inicialización, no puede dar a pensar en que durante el tiempo que está en pausa, toda la información que se recopila mientras esta en la escucha constante, puede ser usada por las compañías a la hora de recopilar información sobre los usuarios.

En estos dispositivos y ya sin entrar en el aspecto de lo que puedan hacer con nuestra información las empresas implicadas en el desarrollo de estos sistemas, se podría investigar que sistemas de seguridad usan estos sistemas para impedir ataques de hackers ya que podrían ser capaces de manipular los sistemas asociados, como la configuración de la calefacción, configuración de luces entre otras funcionalidades.

Finalmente, y relacionado con la *Web Speech API* un problema de seguridad que se ha observado es que si la página se encuentra alojada en un https solo se pregunta una vez si permites el acceso al micrófono. Esto parece una vulnerabilidad de seguridad en el sentido de que una aplicación puede grabar un audio de una página alojada en un https una vez que el usuario lo ha autorizado. La API de Chrome interactúa con la API del reconocimiento de voz de Google, por lo que los datos van directamente a Google y cualquier otra persona que este escuchando. Esto debería de ser un tema para solucionarse mediante protocolos de alerta que nos indiquen que nos están grabando continuamente.

8. Referencias

[1] Paul Adenot, Raymond Toy. Web audio API.
Disponible: https://www.w3.org/TR/webaudio/

[2] Gordon Donnelly. 33 Voice Search Statistics to Prepare You for the Voice Search Revolution.
Disponible: https://www.wordstream.com/blog/ws/2018/04/10/voice-search-statistics-2018

[3] Robert Watts. 6 Best Voice Recognition Software 2019.
Disponible: https://fitsmallbusiness.com/best-dictation-voice-recognition-software/

[4] *Voicebot.ai. Voice Assistant Consumer Adoption Report 2018.*
Disponible: https://voicebot.ai/voice-assistant-consumer-adoption-report-2018/

[5] *Voicebot.ai. In-Car Voice Assistant Consumer Adoption Report 2019.*
Disponible: https://voicebot.ai/in-car-voice-assistant-consumer-adoption-report-2019/

[6] *Clark Boyd. Speech Recognition Technology: The Past, Present, and Future.*
Disponible: https://medium.com/swlh/the-past-present-and-future-of-speech-recognition-technology-cf13c179aaf

[7] *Jose Antonio. Control por voz en casa ¿presentan los asistentes virtuales por voz más ventajas o inconvenientes?*
Disponible: https://www.xatakahome.com/domotica/control-por-voz-en-casa-presentan-los-asistentes-virtuales-por-voz-mas-ventajas-o-inconvenientes

[8] *Ayn de Jesus. AI for Speech Recognition – Current Companies, Technology, and Trends.*
Disponible: https://emerj.com/ai-sector-overviews/ai-for-speech-recognition/

[9] *Microsoft. Preguntar a Cortana en Microsoft Edge.*
Disponible: *https://support.microsoft.com/es-cl/help/4026809/windows-ask-cortana-in-microsoft-edge*

[10] *Microsoft Azure. Key concepts for building a bot for Cortana skills using Node.js.*
Disponible: *https://docs.microsoft.com/en-us/azure/bot-service/nodejs/cortana-skill-concepts?view=azure-bot-service-3.0*

[11] *Microsodft Azure. Documentación de Azure Cognitive Services.*
Disponible: *https://docs.microsoft.com/es-es/azure/cognitive-services/*

[12] *Alan W Black. How to install Festival from source in a new location and customize that installation.*
Disponible: *http://www.festvox.org/docs/manual-2.4.0/festival_6.html#Requirements*

[13]Alan W Balck. On-line synthesis demol.
Disponible: http://festvox.org/voicedemos.html

[14] CMUSphinx.Building an application with sphinx4.
Disponible: https://cmusphinx.github.io/wiki/tutorialsphinx4/

[15] CMUSphinx.PocketSphinx 5prealpha.
Disponible: https://github.com/cmusphinx/pocketsphinx

[16] Dialogflow. Google Assistant
Disponible: https://dialogflow.com/docs/integrations/google-assistant

[17] Google Cloud. Documentación de Dialogflow.
Disponible: https://cloud.google.com/dialogflow/docs/

[18] TalAter. A tiny javascript SpeechRecognition library.
Disponible: https://github.com/TalAter/annyang

[19] TalAter. Speech KITT
Disponible: https://github.com/TalAter/SpeechKITT

[20] Carlos Delgado. Top 7: Best voice commands and speech recognition related javascript libraries.
Disponible: https://ourcodeworld.com/articles/read/163/top-7-best-voice-commands-and-speech-recognition-related-javascript-libraries

[21] Eric Enge. Mobile voice usage trends 2019.
Disponible: https://www.perficientdigital.com/insights/our-research/voice-usage-trends

[22] Google assistant SDK. Librería de Google Assistant.
*Disponible:*https://developers.google.com/assistant/sdk/guides/service/python/embed/install-sample

[23] Hipertextual. Apple compra Siri, el asistente personal para iPhone.
Disponible: https://hipertextual.com/archivo/2010/04/apple-compra-siri-el-asistente-personal-para-iphone/

[24] Developer Apple. Speech
Disponible: https://developer.apple.com/documentation/speech

[25] *Eva Rodriguez de Luis. Google Home, Amazon Echo o HomePod: qué altavoz inteligente comprar en función de necesidades y presupuesto.*
Disponible: https://www.xataka.com/seleccion/google-home-amazon-echo-homepod-que-altavoz-inteligente-comprar-funcion-necesidades-presupuesto

[26] Google assistant SDK. Librería de Google Assistant.
*Disponible:*https://developers.google.com/assistant/sdk/guides/service/python/embed/install-sample

[27] Alba Mora. Review de HomePod: Precio, disponibilidad y prestaciones
Disponible: https://www.macworld.es/review/hardware/homepod-3683891/

[28] Mvelegon. Overview of the AVS Device SDK
Disponible: https://github.com/alexa/avs-device-sdk

[29] Alistair Barr. Amazon buys text-to-speech software company Ivona.
Disponible: https://www.reuters.com/article/us-amazon-ivona/amazon-buys-text-to-speech-software-company-ivona-idUSBRE90N0T020130124

[30] Amazon. Amazon Echo.
Disponible: https://www.amazon.es/Amazon-Echo-Altavoz-Inteligente-Alexa/dp/B079PFKDZC

[31] Francisco Ruiz. Así funciona el asistente de Amazon Alexa en mi Android
Disponible: https://www.androidsis.com/descargar-e-instalar-amazon-alexa-en-android/

[32] Pedro Santamaria. Sin manos: ya puedes invocar Alexa con un comando de voz en cualquier PC con Windows 10
Disponible: https://eloutput.com/noticias/aplicaciones/alexa-soporta-manos-libres-windows-10/

[33] Manu Contreras. Harman Kardon Invoke: el altavoz inteligente de Microsoft con Cortana
Disponible: *https://clipset.20minutos.es/harman-kardon-invoke-altavoz-microsoft-cortana/*

[34] Jorge Sanz Fernández. Los altavoces inteligentes aumentan sus ventas un 50%, con una Apple irrelevante a la cola.
Disponible:https://cincodias.elpais.com/cincodias/2019/05/20/gadgets/1558382700_083165.html

[35] Paul Adenot, Raymond Toy. *Introductionde la Web Speech API.*
Disponible: https://w3c.github.io/speech-api/#introduction

*[36]*Paul Adenot, Raymond Toy. *Casos de uso de la Web Specch API*
Disponible: https://www.w3.org/2005/Incubator/htmlspeech/XGR-htmlspeech-20111206/#use-cases

*[37]*Paul Adenot, Raymond Toy. *Web Speech API*
Disponible: https://w3c.github.io/speech-api/

[38] Juan Antonio Jiménez Torres. Voces habla hispana:
Disponible: https://jsfiddle.net/jjimenezt/esfx9naL/

[39] Chris Mills. Firefox and the Web Speech API.
Disponible: https://hacks.mozilla.org/2016/01/firefox-and-the-web-speech-api/

[40] MDN web docs. Using the Web Speech APIiones
Disponible: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API

[41] MDN web docs. HTML.
Disponible: https://developer.mozilla.org/es/docs/Web/HTML

[42] Caniuse. Method to provide speech input in a web browser
Disponible: https://caniuse.com/#feat=speech-recognition