Facultad de Ciencias

Extensión de una app para transportistas con servicio de voz y reconocimiento de actividad

(Extension of a carrier app with voice service and activity recognition service)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Diego Revuelta Hoz

Directora: Marta Elena Zorrilla Pantaleón

Co-Director: Pablo Rodríguez Solís

Julio - 2020

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Agradecimientos

Me gustaría agradecer en primer lugar a mi familia y amigos por todo el apoyo que me han dado durante mis años en la Universidad de Cantabria, ayudándome en todo lo posible.

Agradecer a todos los profesores que me he encontrado en estos años en el grado de Ingeniería Informática y en especial a Marta Zorrilla por ayudarme y guiarme en la realización de este proyecto.

Y a Fieldeas por permitirme realizar el proyecto de fin de grado en su empresa rodeado de profesionales del mundo del software. Agradecer a todos mis compañeros de Fieldeas por el gran ambiente de trabajo y la ayuda para la realización de este proyecto.

Resumen

La empresa Fieldeas dispone de diversas soluciones de software para la gestión de procesos de trabajo. Uno de los principales productos es el servicio de Track & Trace, el cual proporciona una solución para la gestión del transporte y la logística.

Fieldeas me solicitó implementar para su aplicación móvil Android diversas funcionalidades que mejoraran la experiencia de usuario de sus clientes. Al tratarse de un software que utilizan principalmente los transportistas las funcionalidades pedidas fueron: **Reconocimiento de voz**, esto es, permitir que los usuarios interactúen con la aplicación sin necesidad de desatender la conducción; **Generación de voz**, el sistema pueda interactuar con el usuario de manera que no sea necesario que este mire el dispositivo; y, **Reconocimiento de actividad,** el sistema sea capaz de detectar si un usuario llega a una parada o finaliza una ruta.

Este proyecto describe todas las fases del ciclo del software desarrolladas para la implementación de los plug-ins que recogen esta funcionalidad pedida, así como también incluye un estudio de diversas APIs que fueron analizadas como paso previo a la construcción del software.

Palabras clave: Reconocimiento de voz, Generación de voz, Reconocimiento de actividad, Cordova, Android

Abstract

Fieldeas has several software solutions for the management of work processes. One of the main products is the Track & Trace service, which provides a solution for transport management and logistics.

Fieldeas asked me to implement various features for its Android mobile application that would improve the user experience of its customers. As it is a software used mainly by carriers, the requested functionalities were: **Voice recognition**, that is, allow users to interact with the app without losing sight of driving. **Voice generation**, the system can interact with the user so that the user does not need to look at the device. **Activity recognition**, the system is able to detect whether a user arrives at a stop or ends a route.

This project describes all the phases of the software cycle developed for the implementation of plug-ins that collect this requested functionality, as well as a study of various APIs that were analyzed as a pre-construction step of the software.

Keywords: Speech Recognition, Voice Generation, Activity Recognition, Cordova, Android

1. Introducción

En este capítulo se contextualiza el trabajo de fin de grado realizado. En primer lugar, se explica el objeto del mismo y en entorno en el que se ha desarrollado; y a continuación, se detalla la metodología y plan de trabajo que se ha seguido para su desarrollo.

1.1 Antecedentes y Objetivo

En los últimos 10 años el uso de los *Smartphones* y *Tablets* se ha incrementado exponencialmente y gracias a la inclusión de nuevas capacidades como los reconocedores de voz han logrado mejorar la experiencia de los usuarios. Basta ver a las grandes empresas como Amazon, Google o Apple que tiene su propio asistente virtual, además proporcionan herramientas para poder implementar sus tecnologías de reconocimiento de voz en aplicaciones de terceros gracias a sus APIs. En el ámbito profesional, cada vez son más las empresas que están apostando por estas nuevas tecnologías para incrementar la productividad y poder monitorizar los procesos empresariales.

Este es el caso de la industria de la logística y del transporte en el cual Fieldeas es un referente nacional a la hora de proveer soluciones a medida para la gestión del transporte de productos, seguimiento de vehículos, manejo de cargas, descargas y productos. Para este sector Fieldeas ofrece un producto software denominado *Fieldeas Track & Trace* que aglutina todas las operaciones de los servicios de logística y transporte. Este producto puede ser personalizado para cada compañía y consiste en un conjunto de aplicaciones móviles y webs. Este TFG está orientado al producto móvil en el cual los principales destinatarios son los conductores de camiones o furgonetas, cuya interacción con la aplicación no puede realizarse mientras están conduciendo ya que puede provocar distracciones al volante y accidentes.

El objetivo de este proyecto por ello es dotar a esta aplicación móvil de un sistema de control y asistencia por voz que permita a los trabajadores interactuar con la aplicación de gestión de rutas. Además, sería muy útil reconocer el estado en el cual se encuentra el conductor (conduciendo, descargando o en su periodo de descanso) para no resultar invasivo.

Fieldeas desarrolló esta aplicación móvil utilizando Cordova, una tecnología para la realización de aplicaciones móviles híbridas utilizando tecnologías web. Por ello, este proyecto implementará el reconocimiento de voz, el sintetizador, y el reconocimiento de actividades sobre esta aplicación base.

El desarrollo será realizado únicamente sobre Android. Se desarrollarán tres plugins sobre el *Core* de Fieldeas para que los desarrolladores Front-End puedan utilizar estas tecnologías en las soluciones a medida de los clientes. Las APIs a utilizar para el reconocimiento y sintetizador de voz fueron seleccionadas previamente por la

empresa que eligieron las APIs nativas de Android. No obstante, dentro de este TFG se realizó una breve investigación y desarrollo adicional de un prototipo para el reconocimiento de voz en busca de tecnologías mejores para la solución que se planteaba. Por otra parte, para el reconocimiento de actividades fue necesario buscar una herramienta que se adaptase a los requisitos del proyecto como se detalla en el apartado 2.3. Las tecnologías finalmente utilizadas para el desarrollo de este TFG son:

- Android Speech-to-Text para la implementación del reconocimiento de voz.
- Android Text-to-Speech para el desarrollo del sintetizador de voz.
- Google Activity Recognition API utilizada para el reconocimiento de actividad.

Estas tecnologías y otras descartadas porque no se ajustaban a los requisitos se explican con detalle en el apartado *2 Análisis de Tecnologías.*

1.2 Metodología y Plan de Trabajo

Este proyecto se puede considerar como un proyecto de I+D en principio para la parte de reconocimiento de actividad y posteriormente para el reconocimiento de voz, ya que fue necesaria una investigación exhaustiva en busca de las herramientas que mejor se adaptaban al producto de la empresa. El proyecto se realizó sin conocimientos iniciales sobre la tecnología de Cordova en Android, ni conocimiento de las APIs que se utilizarían. El proyecto fue realizado en su totalidad por el autor de este TFG, creando desde cero los plugins e incluyendo el mayor número de funciones posibles, estando además en contacto con programadores Front-End más experimentados que hicieron alguna recomendación en cuanto al paso de información a los plugins.

Al inicio del proyecto se definieron las tareas a realizar junto con su estimación de tiempo. El proyecto se debía realizarse en unas 500 horas, correspondientes a 3 meses a jornada completa.

Periódicamente, normalmente cada semana, se revisaban los tiempos y se ajustaban para controlar el tiempo de realización del proyecto. Para controlar los tiempos de las tareas e imputar los tiempos se usaron dos herramientas que la empresa utiliza para gestionar sus proyectos internos, estas son Jira y Mopa, (ver sección 3.4.4 Mopa y 3.4.5 Jira). Se programaron reuniones periódicas, cada 2-3 semanas, con el gerente de soluciones móviles de la empresa con el fin de refinar los requisitos y solucionar dudas. Además, hubo reuniones cortas con otros programadores expertos tanto en Front-End como en Android para resolver dudas o problemas con el desarrollo.

A continuación, se muestra el estado del diagrama de Gantt del proyecto, observando las tareas recogidas a lo largo del proyecto y su estimación en horas o días.

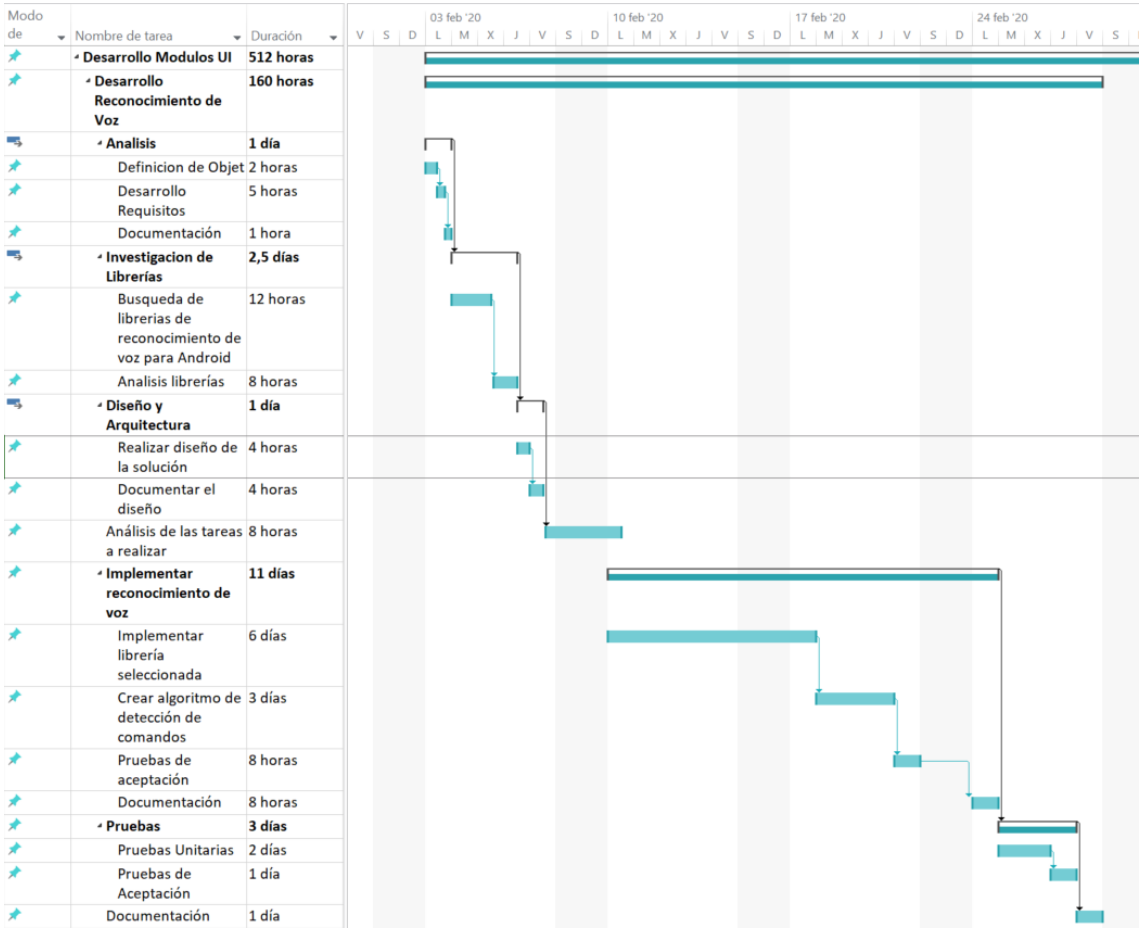

Figura 1.1 Diagrama de Gantt Parte 1

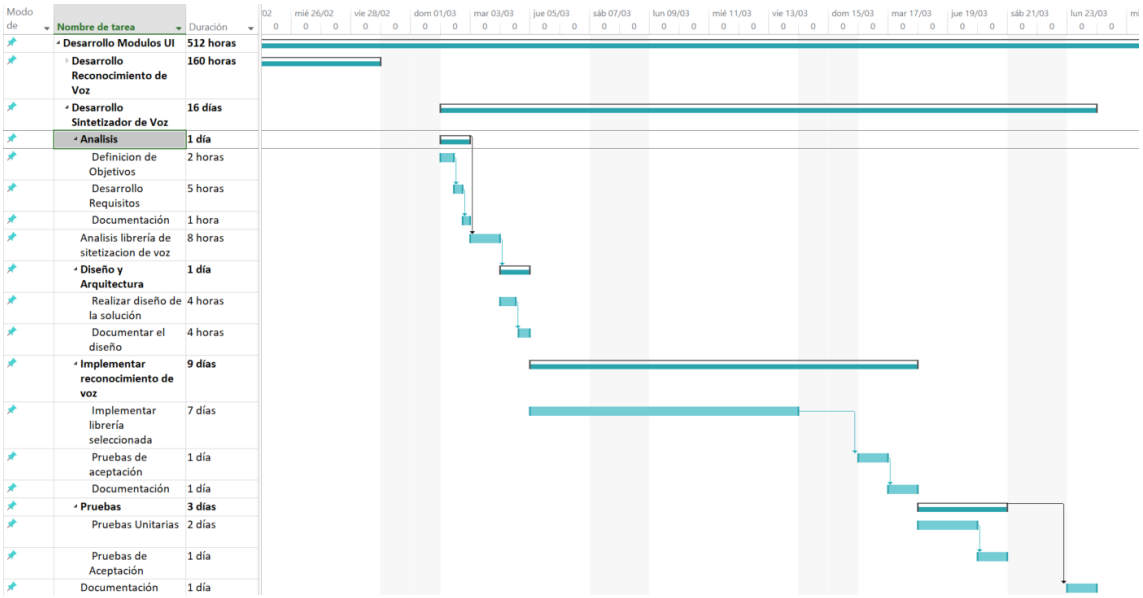

Figura 1.2 Diagrama de Gantt Parte 2

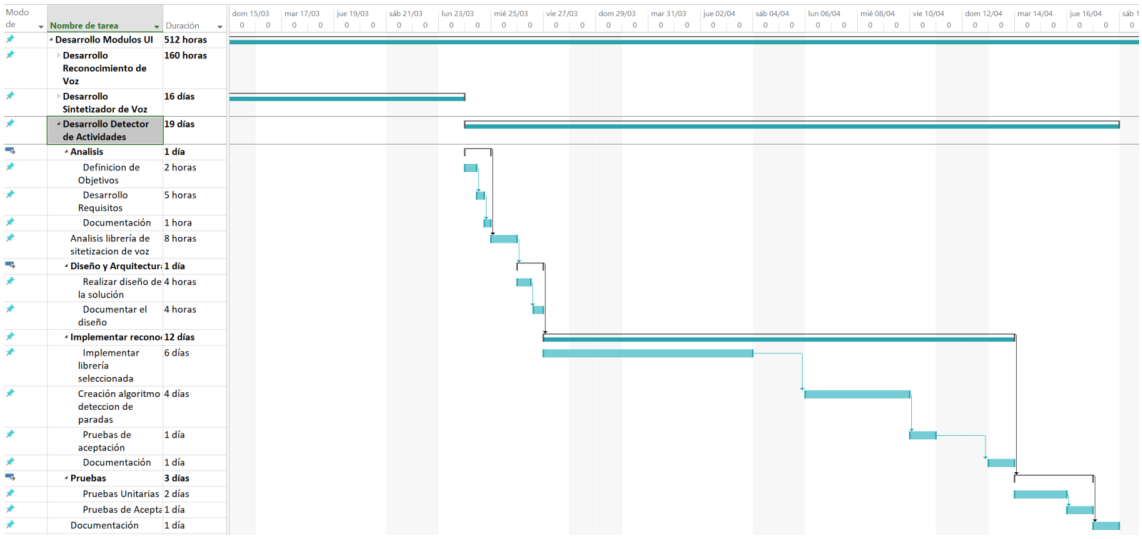

Figura 1.3 Diagrama de Gantt Parte 3

Figura 1.4 Diagrama de Gantt Parte 4

2. Análisis de Tecnologías

Este apartado recoge el proceso de selección e investigación de las distintas tecnologías tanto de reconocimiento de voz como de reconocimiento de actividad, así como un análisis de la tecnología de síntesis de voz que ya había sido seleccionada por Fieldeas. Además, se incluye una explicación en detalle de la tecnología seleccionada y los posibles problemas que se han encontrado a la hora de implantar la tecnología en el proyecto.

2.1 Análisis de Tecnologías de Reconocimiento de Voz

La tecnología de reconocimiento de voz se comienza a desarrollar en 1952 por los laboratorios Bell, una máquina que solo era capaz de reconocer diez dígitos dictados por una sola voz. Desde 1952 a la actualidad han surgido numerosas mejoras que han permitido pasar de 10 dígitos a millones gracias al desarrollo y mejora de microprocesadores que permiten mejorar los tiempos de respuesta [1][2]. En los últimos años gracias al desarrollo del reconocimiento de voz, empresas como Google o Apple han desarrollado asistentes virtuales que permiten interactuar con el sistema mediante el uso de la voz.

En el gráfico adjunto **Figura 2.1** se puede apreciar la evolución de la cantidad de palabras detectadas por una máquina, comenzando por los diez dígitos comentados anteriormente y terminando por los cerca de diez millones. No solo eso, sino que además se ha mejorado considerablemente el grado de precisión de las palabras reconocidas llegando a alrededor del 90% [13].

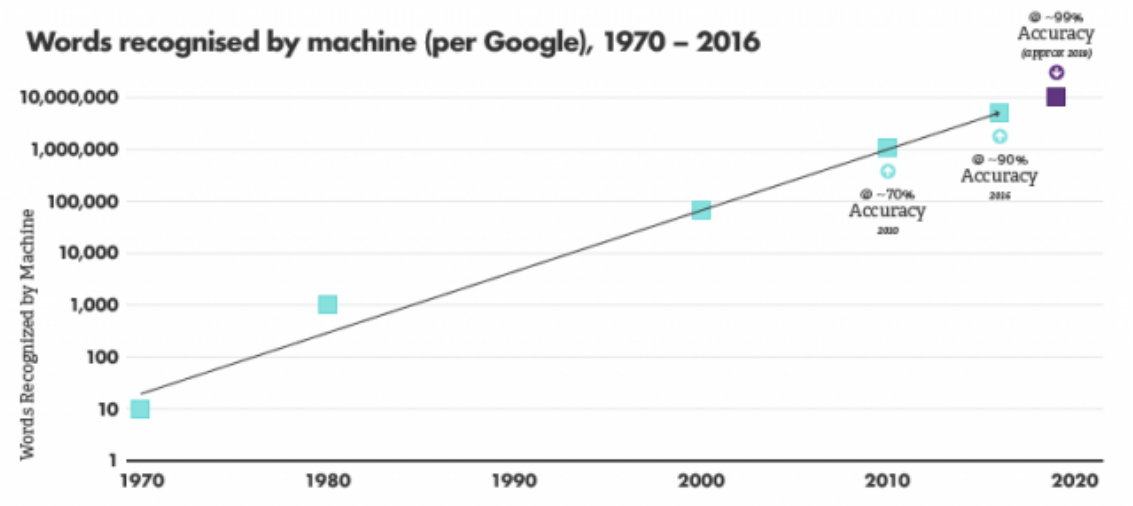

Figura 2.1 Grafico Evolución de Palabras Reconocidas por una Maquina

Pero, ¿cómo funciona realmente el reconocimiento de voz? Para saber cómo funciona el reconocimiento de voz debemos entender que cuando hablamos, lo que se crea son vibraciones en el aire, lo cual se puede representar mediante una señal analógica, pero

esto no es entendido por un computador. Para que un ordenador entienda esto es necesario convertir la señal analógica en digital. Una vez obtenida la señal digital se divide en partes más pequeñas, el sistema compara estas partes con fonemas conocidos del idioma seleccionado. Por último, utiliza un algoritmo estadístico que busca las posibles palabras/frases que se han construido en función de los fonemas encontrados [3][4].

A medida que el uso de los smartphones se ha ido extendiendo, también lo han hecho las librerías existentes para el desarrollo de aplicaciones cada vez más potentes y con más utilidades. Este es el caso del reconocimiento de voz que ha visto como sus librerías se han incrementado en los últimos años.

En un principio la tecnología a utilizar vino dada en los requisitos del proyecto, pero tras un primer análisis de la API nativa de Android se encontró un problema referente al tiempo de escucha de la API, la API únicamente escuchaba durante 10 segundos y se terminaba, lo que hacía imposible un reconocimiento de comandos constante. Esto hizo que fuese necesario una investigación en busca de una tecnología más adecuada.

Tras una búsqueda inicial de librerías de reconocimiento de voz se encontró numerosas APIs que funcionan sobre el navegador web, pero estas no servían para el proyecto ya que, aunque la aplicación se ejecute sobre una especie de navegador web, el WebView de Android, este no soporta la ejecución de estas librerías, por estar más orientadas a navegadores como Chrome o Firefox. Por tanto, la búsqueda tuvo que centrarse en APIs con soporte para Android.

A continuación, se muestra la tabla 2.1 con la comparativa entre las librerías estudiadas. Se ha de señalar que la búsqueda se limitó teniendo en cuenta los siguientes requisitos:

- La tecnología permite el reconocimiento de voz en, al menos, los siguientes idiomas, inglés y español. El soporte para más idiomas es un plus a tener en cuenta, pero no es prioritario.
- La tecnología es compatible con Android y se puede ejecutar como plugin para Cordova.
- La tecnología debe tener el menor coste posible siendo en la medida de lo posible totalmente gratuita.
- La tecnología debe tener soporte actualizado no solo en la actualidad, sino con previsión a futuro también.
- La tecnología permite personalizar los comandos introducidos por el usuario.

Tecnología	Inglés/Español	Pago	Android	Soporte	Personalización
Google Cloud Speech-to-Text	Si	Si	Si	Si	Si
Amazon Transcribe	Si	Si	Si	Si	Si
CMUSphinx	Si	No	Si	No	Si
Android Speech	Si	No	Si	Si	No
Android Voice Service	Si	No	Si	Si	No
Microsoft Azure Speech-to-Text	Si	Si	Si	Si	Si
IBM Watson	Si	Si	Si	Si	Si
TensorFlow	Si	No	Si	Si	Si

Tabla 2.1 Comparativa Tecnologías de Reconocimiento de Voz

- **Google Cloud Speech-to-Text.** Es un Servicio de Google en la nube, el cual permite transcribir voz en texto a través de una sencilla API. Utiliza una serie de redes neuronales capaces de detectar voces en 120 idiomas, eliminar ruido o filtrar el contenido. Permite la transcripción tanto en tiempo real como de un fichero de audio grabado. La mayor ventaja de esta API es la posibilidad de incluir palabras que pueden ser utilizadas durante el reconocimiento de voz. Esta API tiene un coste en función de su uso (entre 0,004 y 0,009 USD por cada 15 segundos en función de las opciones escogidas) tiene soporte actualizado para Android [5].
- **Amazon Transcribe.** Se trata de un servicio por parte de Amazon de transcripción de audio en texto, al igual que la API de Google permite convertir la voz en tiempo real o través de fichero de audio, aunque tiene alguna característica distinta como la generación de marcas temporales por palabra, la API es muy similar a la de Google. También es de pago (aproximadamente 0,006 USD por cada 15 segundos de reconocimiento de voz), tiene soporte para Android, que al tratarse de una API de *Amazon Web Services,* el soporte presente y futuro está garantizado [6] .
- **CMUSphinx.** Librería de código abierto para el reconocimiento de voz desarrollada por la Universidad Carnegie Mellon. Utiliza una serie de modelos entrenados para poder reconocer voz en distintos idiomas, entre ellos inglés y español, se pueden entrenar modelos a medida en función de las necesidades. La API es totalmente offline, no necesita conectarse a un servidor como las otras librerías analizadas. Existen distintas versiones que pueden ser usadas en distintos entornos, como sistemas empotrados, es posible integrarlo dentro de Android. La librería es gratuita pero la última versión estable es de 2015 [7][8].
- **Android Speech.** Es una librería interna de Android que provee un servicio de reconocimiento de voz, la API reconoce voz durante aproximadamente diez segundos desde que se inicia. Esta API no permite modificar los modelos entrenados, por lo que no se pueden añadir nombres propios, por ejemplo. La

API necesita conexión para conectarse a los servidores para que puedan generar el texto. Esta librería está integrada en Android por lo que no tiene ningún coste asociado a su uso, tiene el soporte de Android para todas sus librerías [9].

- **Android Voice Service.** Al igual que Android Speech estamos ante una librería integrada en Android. Esta librería permite utilizar un servicio de Android que está pendiente de una *hotword* como puede ser "Ok, Google" en el asistente de Google, permite personalizar la *hotword* que se quiera. La API no tiene ningún coste asociado, Android provee el soporte necesario [10].
- **Microsoft Azure Speech-to-Text.** Solución por parte de Microsoft al reconocimiento de voz, similar tanto a la solución de Google como a la de Amazon, todos basados en la nube. Permite configurar el vocabulario, tratamiento en tiempo real, etc. Al igual que estas dos librerías estamos ante un servicio de pago (entre 1 USD y 2.10 USD por hora de transcripción en función de las características seleccionadas) [11].
- **IBM Watson.** Librería de IBM para el reconocimiento de voz basado en la nube al igual que Google, Amazon y Microsoft, además tiene soporte para Android. Con características similares, reconocimiento en tiempo real, terminología personalizada para ampliar el vocabulario, utilización de *machine learning* para mejorar los resultados, etc. Se trata de un servicio de pago (entre 0.01 USD y 0.03 USD por minuto en función del uso) [12 .
- **TensorFlow.** No se trata de una librería de reconocimiento de voz, sino que es una librería de Google creada para el desarrollo de soluciones de *machine learning.* Con esta librería es posible la implementación de una solución *Deep learning* mediante redes neuronales que aprendan el lenguaje mediante un sistema de entrenamiento. Permite el uso en dispositivos móviles y de IoT mediante la librería TensorFlow Lite una vez sea entrenado el modelo (red neuronal). Provee además de una librería de alto nivel llamada Keras totalmente integrada en TensorFlow que permite la creación de estas redes neuronales de una manera más sencilla sin necesidad de conocer toda la funcionalidad de las capas de neuronas.

Una vez realizado un análisis de las herramientas disponibles se realizó una selección de la solución que mejor se adapta al sistema y a los requisitos del proyecto. En análisis únicamente se recogen las librerías que proporcionan servicio sobre Android, ya que es la plataforma donde se va a desarrollar el proyecto.

Uno de los principales requisitos del proyecto era que el coste de la solución sería cero o mínimo. Por esta razón las librerías de Google, Amazon, Microsoft e IBM se descartaron, debido a su coste por uso, aunque todas ellas proporcionan unas horas de servicio gratuito. Al tener un coste asociado, todas estas librerías proporcionan más funcionalidades que las restantes, por lo que, aunque inicialmente el proyecto se realizara con las herramientas gratuitas en un futuro se valorara la inclusión de alguna de estas librerías si el sistema lo necesitase.

Nos quedaban cuatro librerías de las cuales la librería de Android Speech fue la que inicialmente se encontraba en los requisitos del sistema, pero el límite de máximo 10

segundos de uso continuado hacía que no fuese la mejor opción. Por ello se decidió que la mejor alternativa era CMUSphinx, que tenía la particularidad de ser utiliza de manera offline, y que, aunque no era un requisito inicial del proyecto se considera una ventaja frente al uso de servicios en la nube (posteriormente se añadió a los requisitos).

Tras realizar las correspondientes pruebas con esta librería no se consiguió hacer funcionar con un modelo de reconocimiento de voz en español, y en cambio funcionaba perfectamente en inglés. Esto unido a la escasa información disponible de la librería y que la última versión estable era del 2015, nos llevó a descartarla y retomar la opción inicial, esto es, la API de Android Speech.

2.1.1 La Tecnología Utilizada: Android Speech-to-Text

La librería Android Speech está integrada dentro de Android por lo que la implementación dentro de un plugin se haría de manera sencilla. Se trata de una librería gratuita que se encarga de transcribir voz en texto. Para realizar la traducción se conecta a los servidores de Google (API online).

Después de desarrollar un prototipo, se comprobó que la librería funcionaba correctamente, reconocía el texto introducido mediante la voz y, mediante un algoritmo de comprobación de resultados, era capaz de detectar si se había dicho un comando o no.

No obstante, también hubo problemas con esta API. El principal problema a la hora de implementar la librería de Android Speech fue el límite de tiempo de escucha, no permite nativamente un reconocimiento de voz continuo a la espera de una entrada de audio/voz, si no que tras diez segundos aproximadamente devuelve un error y finaliza la escucha. Por este motivo se modificó la implementación para realizar un servicio en Android que ejecutase el reconocimiento de voz de manera continuada (se explica en detalle en sección 6.1.1 Implementación/Capa de Negocio/Speech-to-Text).

Viendo los problemas con la librería de Android Speech, se intentó hacer uso del Android Service Voice el cual permite un reconocimiento de palabras clave similar a CMUSphinx actuando como un servicio en segundo plano. Aunque parecía que esta podía ser la mejor solución posible para el proyecto, tras construir un prototipo, se descubrió que esta librería crea una aplicación asistente, y para su funcionamiento es necesario desactivar el asistente de Google integrado en Android u otro asistente que el usuario tuviera instalado, por lo que fue descartada.

Finalmente, la tecnología utiliza es Android Speech, la cual se expuso como un servicio para que pudiera estar funcionando en segundo plano.

Tras una revisión del desarrollo se decidió realizar una investigación acerca de la posibilidad de implementar el reconocimiento de voz de una manera Offline. Por ello se estudió más profundamente acerca del *Deep Learning*, se desarrollaron varios

prototipos de redes neuronales, utilizando redes neuronales convolucionales y redes neuronales LSTM, pero siempre apareció el problema del procesado de datos que van a ser entrenados. La distinta longitud de los ficheros de audio y texto hacen que los datos deban ser redimensionados ya que los *inputs* de la red neuronal deben tener el mismo tamaño tanto de entrada como de salida. Por estas razones, finalmente se decidió descartar la utilización de una red neuronal propia.

2.2 Análisis de Tecnologías de Síntesis de Voz

Como ya se ha indicado este proyecto se integra dentro de la aplicación de transporte de Fieldeas, por lo que un módulo que transforme el texto en audio es esencial con objeto de que el conductor no necesite mirar a la pantalla del dispositivo y evite así, cualquier distracción al volante.

La síntesis de voz consiste en la reproducción artificial del habla de una persona. A lo largo de la historia han sido muchos los intentos de simular el habla humana, los primeros intentos datan del año 1779, año en el que Christian Kratzenstein (científico danés) crea un simulador del habla que es capaz de reproducir las cinco vocales. Durante los siguientes años se desarrollaron mejoras que permitieron incrementar el número de sonidos reproducidos. Los laboratorios Bell, que también participaron en la creación y desarrollo del reconocimiento de voz, desarrollaron una máquina que era capaz de analizar el habla, llamada Vocoder. A partir de esta máquina se crearon los posteriores sintetizadores, como el Voder, una máquina creada por Homer Dudley considerado el primer sintetizador de voz [14][15].

En la **Figura 2.2** se puede observar la estructura del Voder. Los sistemas de síntesis de voz actuales siguen la estructura básica e idea del Voder. Aunque los sistemas actuales incluyen algoritmos y métodos mucho más complejos, como el modelo oculto de Markov o las redes neuronales.

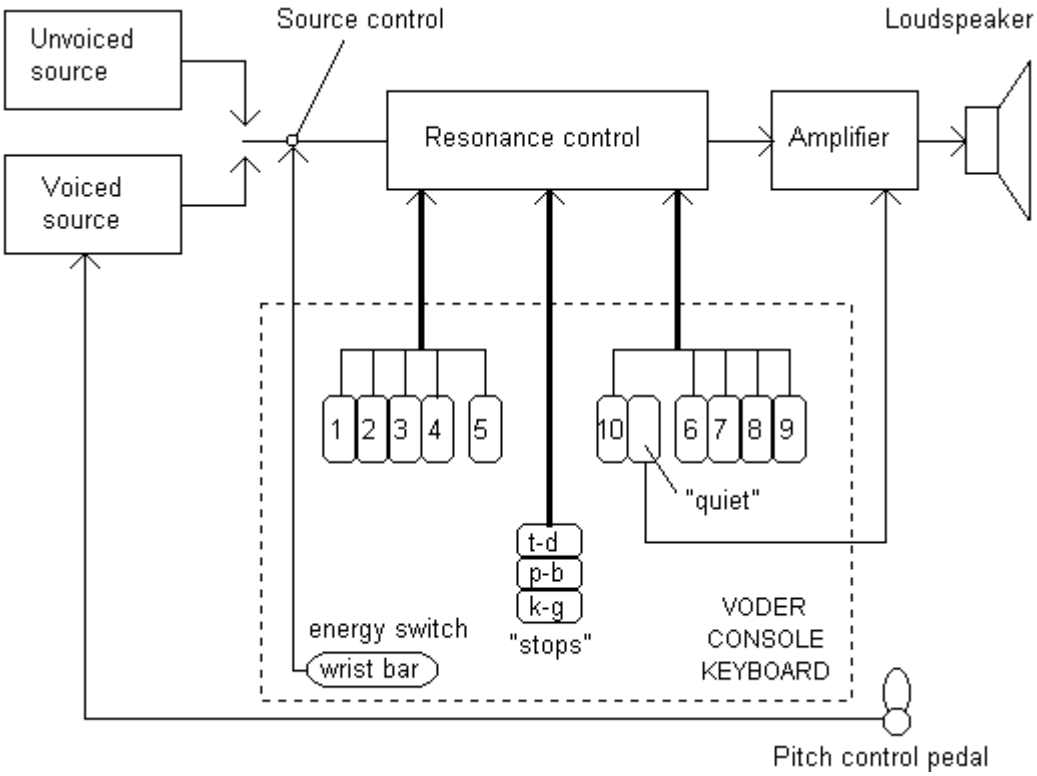

Figura 2.2 Sintetizador de Voz Voder

Para realizar la síntesis de voz a través de una máquina, se reproducen sonidos previamente grabados y almacenados en la base de datos. Estos sonidos pueden ser fonemas o palabras completas. El objetivo de un sistema de síntesis de voz es que el habla sea lo más natural posible encadenando estos fonemas y palabras de forma correcta. Para realizar esta síntesis existen distintas maneras, pero nos vamos a centrar en las dos más utilizadas, Síntesis Concatenativa y Síntesis de Formantes [15][16][17].

- **Síntesis Concatenativa**. La síntesis Concatenativa utiliza la unión de segmentos de voz previamente grabados para la generación del habla. Dentro de la síntesis Concatenativa se pueden encontrar tres métodos distintos para la concatenación de sonidos, Síntesis por Selección de Unidades, Síntesis por Difonemas y Síntesis Especifica.
    - ***Síntesis por Selección de Unidades.*** Esta síntesis utiliza una base de datos en la que se pueden almacenar sonidos tanto de fonemas, letras, silabas, palabras, frase u oraciones. Para la división se suele utilizar un sistema de reconocimiento de voz con un procesado del lenguaje modificado para poder dividir en segmentos más pequeños el texto, de manera que se introduzca una frase larga y se puede dividir en segmentos más pequeños sin necesidad de introducir estos uno a uno. Este método produce la voz más natural, pero el tamaño de las bases de datos puede ser demasiado grande y tardar demasiado tiempo en generar el resultado.
    - ***Síntesis por Difonemas***. Este método utiliza una base de datos de tamaño reducido, en ella se almacenan todos los difonemas del lenguaje, en el castellano hay aproximadamente 800 difonemas. El

resultado es una voz muy poco parecida al habla natural, por eso este método está prácticamente en desuso.

  - ***Síntesis Especifica.*** Se especifica un dominio de palabras reducido, acorde a las necesidades de cada solución y se graban en una base de datos las grabaciones de cada frase del dominio. Esto permite limitar el tamaño de la base de datos y utilizar una voz muy similar al habla natural, por la contra el dominio de frase a reproducir es limitado.

- **Síntesis de Formantes.** A diferencia de los métodos anteriores, la síntesis de formantes no utiliza grabaciones del habla en tiempo de ejecución, si no que utiliza modelos acústicos que crean una onda que simula el habla. El resultado de la síntesis es un sonido robótico fácilmente diferenciable del habla natural, la principal ventaja de este método es la no utilización de grabaciones en base de datos, haciendo así un programa mucho más ligero.

Para la síntesis de voz, la selección de la librería fue hecha por la propia empresa, siendo ésta la librería de Android Text-to-Speech. Los sistemas TTS actuales dividen el texto en fonemas y convierten estos fonemas a una onda que puede ser representada como un sonido, como se ha indicado previamente sería una síntesis por selección de unidades.

2.2.1 La Tecnología Utilizada: Android Text-to-Speech

Android Text-to-Speech es un API interna de Android gratuita, uno de los requisitos principales del proyecto. Esta API es capaz de sintetizar un texto pasado a audio/voz para ser reproducido en el mismo momento o para crear un fichero de audio y que pueda ser guardado. Para este proyecto únicamente será necesaria la reproducción de la voz. La librería permite la reproducción de voz en multitud de idiomas esto hace que la aplicación de la empresa sea mucho más versátil. Además, permite la modificación de voces, cambiando no solo su idioma sino su calidad haciendo que los datos de red consumidos sean menores o mayores en función de la calidad indicada. Se permite el uso de la tecnología tanto en online como offline. Para el uso offline es necesario que el dispositivo Android tenga descargado el paquete de idioma correspondiente, normalmente ya vienen instalados tanto el inglés como el español [18].

Una vez integrada la herramienta, se observa que la calidad de la voz no es muy buena, suena demasiado robótica. Google ofrece un servicio en la nube en el que se pueden elegir más de 180 voces distintas en 30 idiomas, utiliza para ello potentes redes neuronales, pero se trata de un servicio de pago por lo que la empresa lo descartó inicialmente quedando la de Android por el momento, en un futuro puede ser actualizada a la Api de Google si la empresa lo considera oportuno.

2.3 Análisis de Tecnologías de Reconocimiento de Actividad

El reconocimiento de actividad consiste en conocer el estado en el que se encuentra un usuario. Estos estados pueden ser caminando, conduciendo, parado, etc. Hoy en día tanto Android como iOS conocen en todo momento en qué estado se encuentran los usuarios, para ello hacen uso de los sensores existentes en el dispositivo móvil.

Los sensores más utilizados para este reconocimiento son:

- **Acelerómetro**. Mide la aceleración, los cambios de movimiento, etc.
- **Giroscopio**. Detecta los giros del dispositivo.
- **Orientación.** Indica la dirección a la que apunta el dispositivo.
- **Proximidad**. Detecta si hay un objeto a menos de 5cm.
- **Campo Magnético**. Función de brújula y detección de campos magnéticos.
- **Presión Atmosférica**. Función de altímetro y barómetro.
- **Gravedad**. Mide la aceleración de la gravedad.
- **Acelerómetro Lineal**. Mide la aceleración descontando la gravedad.
- **Vector de Rotación**. Detecta giros, uso en conjunto con el giroscopio.
- **WiFi.** Detección de redes WiFi.

Con todos los datos obtenidos de estos sensores se aplican técnicas de minería de datos y aprendizaje automático con el fin de conocer la actividad del usuario. El principal dificultad a la que se enfrenta esta tecnología es el gran rango de valores que toman en función de la persona que realiza la actividad , por ejemplo, una persona mayor no camina a la misma velocidad que un joven [22][23].

2.3.1 La Tecnología utilizada: Google Activity Recognition API

La tecnología utilizada para este propósito fue la API de Google llamada Google Activity Recognition API. Esta API nos permite conocer el estado actual de un usuario, pero también nos permite recibir actualizaciones cada vez que el usuario cambie de actividad a través de un servicio Android. Esta última funcionalidad es la que se necesita para cumplir los requisitos del sistema. Las actividades captadas por la API son:

- **IN_VEHICLE:** El dispositivo está en un vehículo.
- **ON_BICYCLE.** El dispositivo está en una bicicleta.
- **ON_FOOT.** El dispositivo está en un usuario andando o corriendo.
- **RUNNING.** El dispositivo está en un usuario corriendo.
- **STILL.** El dispositivo está sin movimiento.
- **TILTING.** El ángulo del dispositivo ha cambiado significativamente.
- **UNKNOWN.** Imposible detectar una actividad.
- **WALKING.** El dispositivo está en un usuario andando.

Como se indicó en el análisis de la tecnología existe el problema de la diferencia de datos entre distintas personas, es por ello que Google ha recopilado una gran cantidad de datos de muchas personas diferentes a las que piden que etiqueten sus actividades

en caminar, correr, en bicicleta, parado, en vehículo, etc. Con estas etiquetas Google es capaz de entrenar una red neuronal en la que los datos de entrada son los datos periódicos recogidos de los sensores y la salida es la etiqueta que los usuarios han indicado.

De esta manera cuando en un dispositivo cualquiera se recogen los datos de los sensores se pueden pasar estos datos por el modelo entrenado por Google y generar una predicción.

Recientemente Google ha liberado esta API para desarrolladores de manera que cualquiera puede utilizar esta API para sus aplicaciones [24][25][26].

3. Tecnologías y Herramientas Usadas

En esta sección se enumera y describe las tecnologías utilizadas para el desarrollo de la solución, así como las herramientas que han sido necesarias a lo largo del proyecto.

3.1 Lenguajes de Programación

Para el desarrollo de los módulos requeridos se han utilizado diferentes lenguajes de programación. A continuación, se describen brevemente.

- **Java**. Es el lenguaje que utiliza Android para el desarrollo de aplicaciones móviles que son ejecutadas sobre su sistema operativo. Se ha utilizado para el desarrollo de los plugin de Cordova. Estos plugin están programados de manera nativa ya que acceden a las librerías internas de Android, sensores del dispositivo, etc.
- **JavaScript JS**. Es el lenguaje de alto nivel que se utiliza para el desarrollo web junto con HTML. Las aplicaciones híbridas de Cordova son renderizadas sobre un WebView por lo que el desarrollo es similar al que se realiza para páginas web o aplicaciones web.
- **HTML**. lenguaje de marcas utilizado para la construcción de páginas webs. HTML define la estructura de la página, pero no aplica la funcionalidad, esto se desarrolla utilizando JavaScript.
- **SASS**. Es un lenguaje de script utilizado para proveer de estilo a un documento HTML. Indica cómo se visualizará el contenido al usuario mediante una serie de reglas. Este lenguaje es traducido a CSS3.

3.2 Tecnologías

Para el desarrollo de la solución se han utilizado modernas tecnologías de desarrollo móvil, a continuación, se describe cada tecnología usada.

3.2.1 Android

Android es un sistema operativo móvil propiedad de Google. Está basado en el núcleo de Linux, diseñado en primera instancia para *smartphones* y tabletas; posteriormente fue ampliado a *smartwatches*, televisores, coches, etc. La plataforma se divide en seis capas. [19] [20]

- **Kernel Linux**. En lo más profundo de la plataforma se encuentra el kernel de Linux, que es la base de la plataforma. La utilización del kernel de Linux permite a los fabricantes desarrollar controladores más fácilmente.
- **Capa de Abstracción de Hardware.** Capa intermedia entre el hardware y la API de Java que permite el acceso a los sensores, cámara, etc. por parte de las aplicaciones.

- **Android Runtime**. Entorno de ejecución para aplicaciones Android. Lleva a cabo la traducción de la aplicación en lenguaje máquina.
- **Bibliotecas C/C++ Nativas.** Proveen la mayor parte de las funcionalidades del Android Runtime y de la Capa de Abstracción de Hardware en código nativo.
- **Framework API Java.** Conjunto de funcionalidades del sistema operativo Android. Este es en este marco donde se crean las aplicaciones Android.
- **Aplicaciones del Sistema.** Aplicaciones proporcionadas por Android.

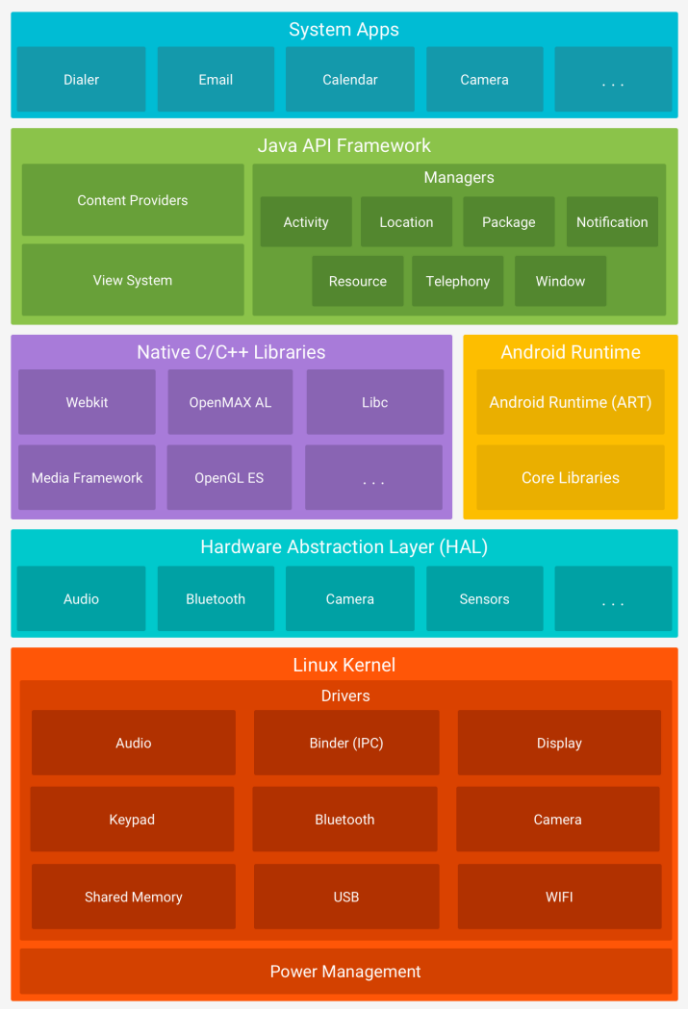

Figura 3.1 Arquitectura Android

3.2.2 Cordova

Apache Cordova es un framework utilizado para el desarrollo de aplicaciones móviles multiplataforma utilizando tecnologías web. Esto nos permite desarrollar una aplicación una sola vez y hacerla funcionar en Android, iOS o Windows Phone. Para que una aplicación pueda funcionar en todas las plataformas Cordova utiliza un WebView generado nativamente en cada plataforma de manera que se pueda

renderizar el código generado en HTML, SASS (CSS) y JavaScript como si fuese una web común sobre este componente.

Para que las aplicaciones tengan acceso a los sensores, librerías, servicios, etc. que provee cada sistema operativo se utilizan lo que Cordova llama plugin, esto es, una parte de código desarrollada de forma nativa que puede ser ejecutada desde el JavaScript mediante una llamada "exec ()". Esta se encuentra en la librería de Cordova y es necesaria para que las aplicaciones funcionen correctamente en los dispositivos móviles. Por defecto, Cordova tiene varios plugin ya desarrollados para el acceso a la cámara, botón de retroceso, información de red, etc. Pero ninguno de los plugins por defecto, realiza la funcionalidad que se requiere en este proyecto [21].

3.2.2 Vue JS

Vue JS es un framework de código abierto Front-end de JavaScript utilizado para el desarrollo de aplicaciones web o móviles. Con este framework es posible construir interfaces de una manera sencilla mediante componentes. Un componente es un elemento en el que se incluye código que puede ser reutilizable. Dentro de un componente nos encontramos código HTML, SASS (CSS) y JavaScript.

Vue es un framework reactivo, esto quiere decir que el sistema puede cambiar en función de eventos internos y mostrar cambios en la vista. Un cambio en una variable puede mostrar un componente u otro sin necesidad de recargar la página.

3.2.4 Framework 7

Es un framework de código abierto utilizado para crear aplicaciones móviles y web con aspecto nativo a través del uso de componentes reutilizables. Es utilizado en conjunto con Vue JS, pero también da soporte a React, Angular, Svelte, etc.

3.2.5 WebPack

WebPack es un sistema de empaquetado de módulos de código abierto para aplicaciones web. Permite la gestión de los recursos necesarios para ejecutar una aplicación mediante el uso de ECMAScript 6.

3.3 Herramientas de desarrollo

La implementación de la solución se ha desarrollado utilizando las siguientes herramientas.

3.3.1 Android Studio IDE

Android Studio es el entorno de desarrollo oficial de Google para la construcción de aplicaciones sobre su sistema operativo (Android) basado en el IDE (Integrated Development Environment) de IntelliJ IDEA, producto de JetBrains. El IDE provee de una serie de herramientas imprescindible para el desarrollo en Android, autocompletado, debugger, refactoring, linker, etc. Además, permite integrar Cordova para el desarrollo de aplicaciones y de plugin. También incluye una serie de emuladores con los cuales se puede probar las aplicaciones desarrolladas sin necesidad de un dispositivo físico, muy útil para probar distintas versiones de Android.

3.3.2 Visual Studio Code

Visual Studio Code es un editor de código desarrollado por Microsoft, contiene multitud de plugin para poder desarrollar en casi cualquier tecnología. Contiene herramientas de depuración, manejo de Git, bases de datos, etc. Para Vue JS tiene distintas extensiones, Vetur es una de las más utilizadas y la que se utilizará para desarrollar una aplicación que implemente las funcionalidades desarrolladas en Android.

3.3.3 Chrome DevTools

Chrome DevTools es un conjunto de herramientas utilizadas para el desarrollo y depuración de páginas o aplicaciones web. Chrome DevTools está integrado dentro del navegador Google Chrome. Se ha utilizado principalmente para depurar la aplicación web desarrollada en Vue JS y la ejecución de plugin, ya que la herramienta de debugging de Android Studio no es suficiente para aplicaciones de Cordova. Este solo depura el plugin, pero no el contenido del WebView. En ocasiones fue necesario depurar con las dos herramientas al mismo tiempo.

3.4 Herramientas para la gestión de la configuración

Para el registro y actualización de la información del proyecto se han utilizado las siguientes herramientas.

3.4.1 Git

Git es un sistema de control de versiones para manejar los cambios en el código fuente de un proyecto que se está desarrollando. Permite a varias personas trabajar en el mismo proyecto de manera que cada versión incluya los cambios correspondientes.

3.4.2 BitBucket

Es el cliente de Git utilizado en la empresa, permite tener los repositorios en el servidor de Atlassian y además se integra con otras herramientas como Jira o Sourcetree.

3.4.3 Sourcetree

Sourcetree es una aplicación de Atlassian que permite el manejo de repositorios Git de una manera más visual, si necesidad de utilizar la línea de comandos.

3.4.4 Mopa

Mopa es una aplicación web utilizada por la empresa que permite la gestión de tareas dentro de un parte de trabajo así como su seguimiento (horas dedicadas a cada proyecto y tarea, días de baja, permisos especiales, etc.).

3.4.5 Jira

Jira es otra aplicación utilizada por la empresa para la gestión de proyectos. Esta permite la creación de proyectos, asignación de tareas, tiempos estimados y reales.

4. Análisis de Requisitos

En esta sección se recogen el proceso de captura de requisitos, así como el detalle de los requisitos tanto funcionales como no funcionales. A lo largo del proyecto la lista de requisitos fue modificada, en concreto, los relativos al módulo de reconocimiento de actividad y al de reconocimiento de voz ya que dependían de las posibilidades que ofrecía la tecnología elegida.

4.1 Captura de Requisitos

El proceso de captura de requisitos comenzó con el planteamiento de la idea que se quería implantar en el producto de Fieldeas. Los requisitos iniciales fueron seleccionados a muy alto nivel, durante las reuniones posteriores se detallaron y ampliaron los requisitos de los tres módulos. Las listas con los requisitos funcionales y no funcionales nunca estuvieron cerradas del todo, hubo propuestas de mejora sobre los requisitos iniciales.

El resultado final de la captura de requisitos funcionales se ha dividido en tres listas, una por módulo a desarrollar. Dentro de cada módulo, existe una lista con los requisitos globales y otra lista con los requisitos (funcionalidades) que debían proporcionarse a los desarrolladores Front-End. Se describen en primer lugar los requisitos funcionales y a continuación, los requisitos no funcionales.

4.2 Requisitos Funcionales

Los requisitos funcionales detallan el comportamiento del sistema. En las tablas 4.1, 4.3 y 4.5 se recogen las funcionalidades generales del sistema, las tablas 4.2, 4.4 y 4.6 recogen las funcionalidades que los desarrolladores de la capa visual requieren ejecutar.

Reconocimiento de Voz	
Identificador	**Descripción**
RF01	Permitirá reconocer la voz del usuario sin necesidad de que este realice ninguna acción.
RF02	Reconocerá comandos de una lista de comandos posibles totalmente personalizables, utilizando lenguaje natural.

Tabla 4.1 Requisitos Funcionales Reconocimiento de Voz

A continuación, se muestra la descripción de las funcionalidades con las que se proveerá a los desarrolladores Front-End para el módulo de reconocimiento de voz.

Identificador	Descripción
RF03	Iniciar el reconocimiento de voz cuando lo necesite.
RF04	Indicar los comandos a ser reconocidos, tanto al inicio como en tiempo de ejecución.
RF05	El sistema enviará un evento cuando haya una coincidencia con algún comando de los solicitados.
RF06	El sistema devolverá en el evento el comando reconocido.
RF07	Seleccionar el idioma que se quiere escuchar tanto al inicio como en tiempo de ejecución.
RF08	Obtener los comandos actuales que se están escuchando.
RF09	Eliminar los comandos que se quieran en tiempo de ejecución.
RF10	Parar el reconocimiento de voz cuando sea requerido.
RF11	Conocer los lenguajes disponibles del módulo de reconocimiento.
RF12	Conocer el lenguaje actual que está siendo reconocido por el sistema.
RF13	Se podrá comprobar si el dispositivo del usuario tiene los permisos requeridos para que el reconocimiento de voz pueda ser llevado a cabo.
RF14	Se podrán solicitar los permisos para iniciar el servicio al usuario.
RF15	Saber si el servicio está escuchando o está parado.

Tabla 4.2 Requisitos Funcionales Plugin Reconocimiento de Voz

Sintetizador de Voz

Identificador	Descripción
RF16	Permitirá traducir cualquier texto suministrado en audio.
RF17	No mostrará ningún mensaje visual.

Tabla 4.3 Requisitos Funcionales Sintetizador de Voz

La siguiente tabla corresponde con la descripción de los requisitos para las funcionalidades con las que se proveerá a los desarrolladores Front-End para el módulo de síntesis de voz.

Identificador	Descripción
RF18	Iniciar una conversión de texto en audio en el idioma seleccionado.
RF19	Parar una reproducción de voz.
RF20	Conocer los lenguajes disponibles para la sinterización de voz.
RF21	Cambiar el idioma por defecto del sintetizador.
RF22	Comprobar si se está ejecutando una sintetización de voz.

Tabla 4.4 Requisitos Funcionales Plugin Sintetizador de Voz

Reconocimiento de Actividad	
Identificador	**Descripción**
RF23	Permitirá conocer el estado de un usuario en todo momento a través de los sensores del dispositivo.
RF24	El sistema almacenará la última actividad detectada hasta que se cierre la App.

Tabla 4.5 Requisitos Funcionales Reconocimiento de Actividad

La descripción de las funcionalidades de las que se proveerá a los desarrolladores Front-End son las siguientes.

Identificador	Descripción
RF25	Iniciar el reconocimiento de actividades.
RF26	El sistema lanzará un evento cuando se detecte un cambio de actividad en el dispositivo.
RF27	El sistema devolverá la actividad detectada y el grado de confianza de la detección.
RF28	Detener el reconocimiento de actividades.
RF29	Obtener la última actividad reconocida por el sistema.

Tabla 4.6 Requisitos Funcionales Plugin Reconocimiento de Actividad

4.3 Requisitos No Funcionales

En cuanto a los requisitos no funcionales, estos especifican restricciones o condiciones de calidad que impone el cliente al producto (rendimiento, disponibilidad, estabilidad, seguridad, etc.). La tabla 4.7 detalla los requisitos no funcionales que se han definido para el proyecto.

Identificador	Descripción	Tipo
RNF01	Los módulos serán desarrollados como plugin para Cordova (las aplicaciones de Fieldeas son desarrolladas como aplicaciones híbridas).	**Software**
RNF02	Los módulos funcionarán sobre Android, siendo la API mínima la 21 (Android 5.0 Lollipop)	**Compatibilidad**
RNF03	Los módulos no serán ejecutados por defecto, sino que se arrancarán desde la parte de Front-End.	**Portabilidad**
RNF04	El desarrollo debe ser los más reutilizable posible.	**Usabilidad**
RNF05	Los módulos serán integrados en la aplicación *Track and Trace* de Fieldeas.	**Portabilidad**
RNF06	Inicialmente las librerías utilizadas para el reconocimiento y sintetizador de voz serán las que provee Google para Android.	**Software**
RNF07	El sistema permitirá que la interacción entre el usuario y el dispositivo sea íntegramente por voz, el usuario no interactuará táctilmente con el dispositivo.	**Usabilidad**
RNF08	El desarrollo seguirá una estructura en tres capas.	**Software**
RNF09	La transferencia de datos se realizará utilizando JSON.	**Portabilidad**
RNF10	El idioma por defecto en el reconocimiento de voz será español de España.	**Interfaz**
RNF11	Se permitirá el reconocimiento de voz tanto si el usuario tiene conexión a internet como si se encuentra sin conexión.	**Disponibilidad**

Tabla 4.7 Requisitos No Funcionales

5. Diseño de la Aplicación

Durante la etapa previa a la implementación de los módulos requeridos, se realizó el diseño tanto arquitectónico como detallado del sistema. A continuación, se describen ambos diseños.

5.1 Diseño arquitectónico

En cuanto al diseño arquitectónico, este ha sido diseñado siguiendo el patrón en tres capas *Model-View-ViewModel* utilizado para el desarrollo de aplicaciones basadas Vue JS. La principal premisa que se debía cumplir era que los módulos fuesen totalmente reutilizables en cualquier aplicación de la empresa, por ello se decidió realizar los módulos como plugin de Cordova en Android, y que estos pudiesen ser utilizados en función de las necesidades del desarrollador Front-End. Para el desarrollo de los módulos solo fue necesario modificar parte de la capa *ViewModel*, no obstante, se desarrolló también una aplicación de prueba modificando también la capa de presentación.

La descripción de las capas se detalla a continuación:

- **Capa de presentación o capa *View***. Este componente se encarga de interactuar con el usuario. Para este proyecto se ha desarrollado una interfaz sencilla ya que el objetivo de este proyecto era añadir las funcionalidades en el *backend* para su reutilización en otros proyectos de la empresa.
- **Lógica de negocio o *ViewModel***. En este componente se recogen las distintas funcionalidades implementadas, se ha dividido en tres boques desarrollados en Java para Android, un bloque por módulo que se detallan a continuación:
    - **Speech Recognizer.** Este módulo se encargar de iniciar todas las funcionalidades del reconocimiento de voz a la parte Front-End, se encarga también del manejo del servicio y de analizar la respuesta en busca de posibles comandos de voz detectados.
    - **Text to Speech.** Este módulo provee de las funcionalidades de síntesis de voz al sistema.
    - **Activity Recognition.** Módulo con las funcionalidades para conocer el estado de un usuario, manejando el servicio y la comprobación de posibles actualizaciones.
- **Capa de datos o *Model*.** En este componente se recoge el acceso a los datos. En un principio este componente no se verá afectado por el desarrollo ya que no será necesario ningún almacenamiento ni obtención de datos. Para un futuro es posible que se requiera el guardado de actividad de un usuario por eso se mantiene la capa de persistencia.

La Figura 5.1 describe la arquitectura de un sistema Cordova. En ella se puede observar como la aplicación web construida se ejecuta sobre un *WebView,* y utiliza los plugin que proporciona Cordova para conectarse con la parte nativa del sistema operativo. A través de los plugin de Cordova podemos tener acceso al micrófono, altavoz, GPS y demás sensores que podamos necesitar. Además de poder ejecutar servicios sobre el sistema operativo que puedan ejecutarse en segundo plano sin que afecten a la experiencia de usuario. De ahí que la implementación se realice como plugin de Cordova ya que es imprescindible el acceso a los sensores y servicio. Así mismo, uno de los principales requisitos del proyecto es que fuera lo más reutilizable posible, un plugin de Cordova puede ser instalado en cualquier aplicación y ser utilizado cuando sea necesario desde el *WebView*.

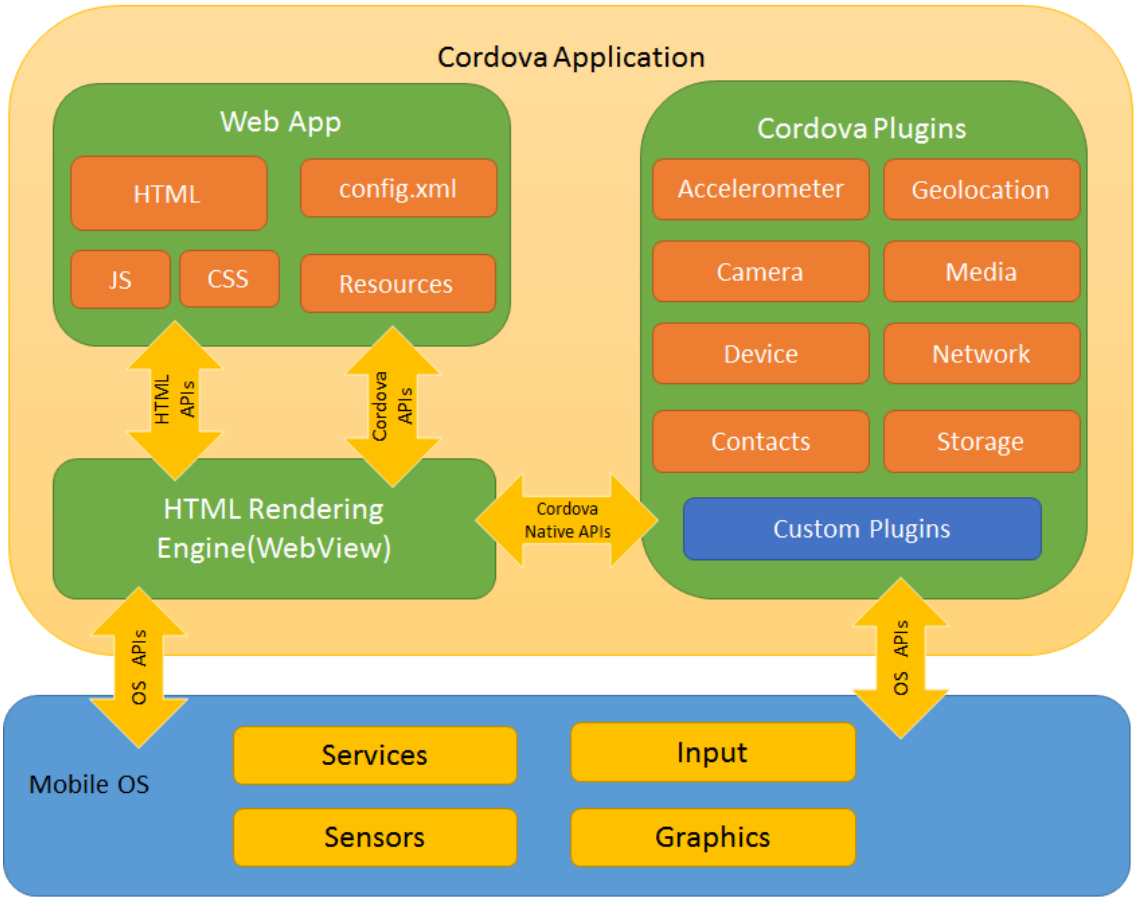

Figura 5.1 Arquitectura de una Aplicación Cordova

5.2 Diseño Detallado

Este apartado muestra el diagrama de despliegue (Figura 5.2) del sistema. El sistema se ejecuta sobre dispositivos móviles Android a través del componente WebView.

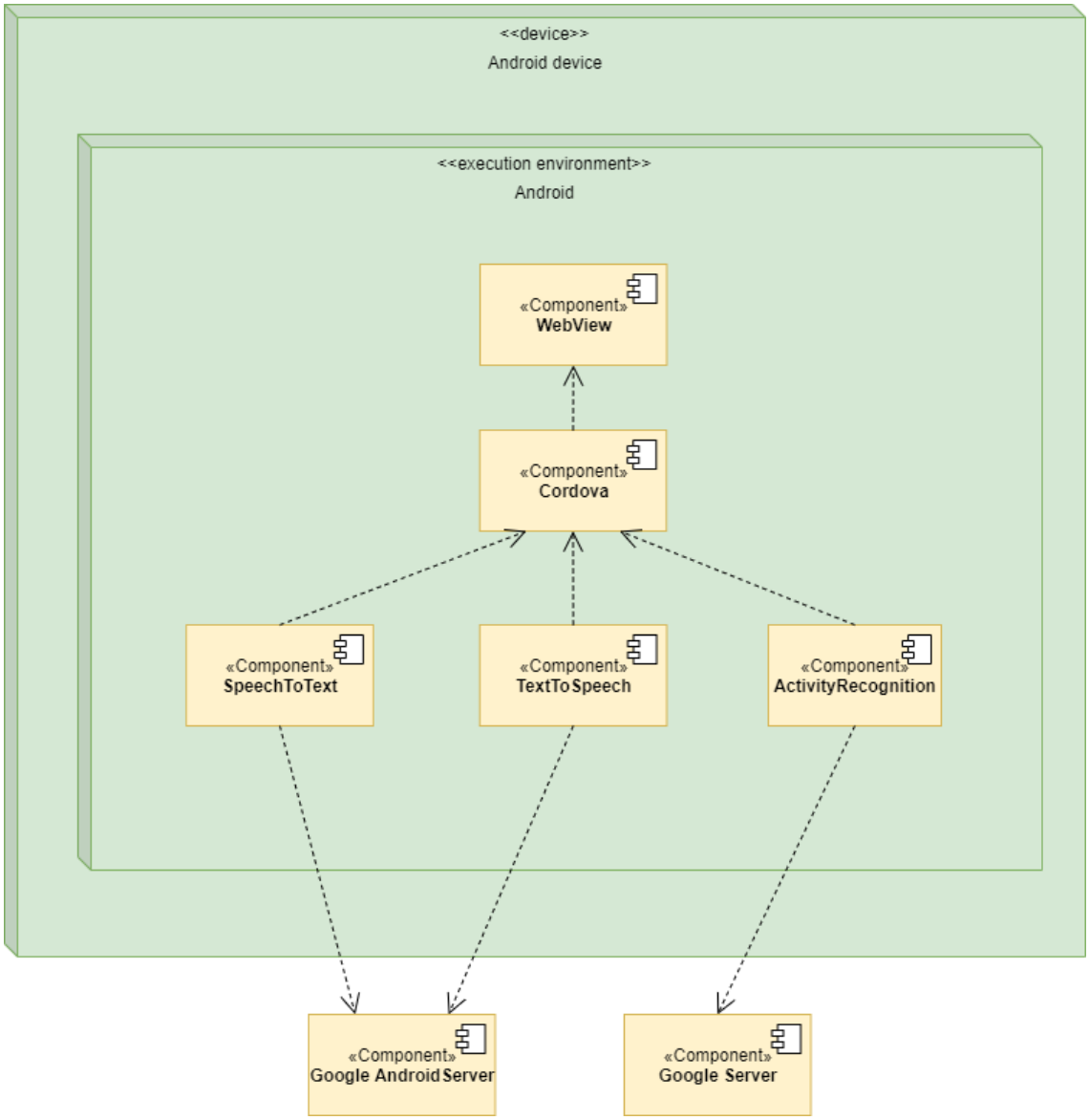

Figura 5.2 Diseño Detallado del Sistema

6. Implementación

Tras la realización del diseño del sistema, en este apartado se explicará el proceso seguido en la implementación de los módulos requeridos.

Antes de comentar cada módulo en profundidad se mostrará la estructura de la aplicación móvil, tanto de la aplicación híbrida, como de la parte nativa de Android.

En la Figura 6.1 se puede ver la disposición de un proyecto Cordova con Vue y soporte para Android. A continuación, se explica el contenido de cada carpeta.

- **Config**. Archivos de configuración de WebPack.
- **Dist**. Archivos compilados de la aplicación web.
- **Hooks**. Scripts especiales de Cordova.
- **Node Modules**. Dependencias de la aplicación web. Se descargan al ejecutar npm install sobre el proyecto.
- **Platforms**. En esta carpeta se encuentran las aplicaciones nativas de todas las plataformas (Android, iOS, etc.) que requiera el proyecto. Inicialmente esta carpeta se encuentra vacía, para este proyecto se ha creado una aplicación Android utilizando el framework Cordova.
- **Plugins**. Carpeta con los plugin a desarrollar para cada plataforma (Android, iOS, etc.).
- **Res**. Carpeta con recursos para las aplicaciones.
- **Src**. Carpeta con los ficheros fuente de la aplicación web.
- **WebPack**. Contiene los ficheros de configuración para la aplicación web. A diferencia de la carpeta Config, esta configuración es específica para la aplicación web.
- **Www**. Carpeta que contiene los ficheros que cargar a la aplicación Android para renderizar en el WebView.

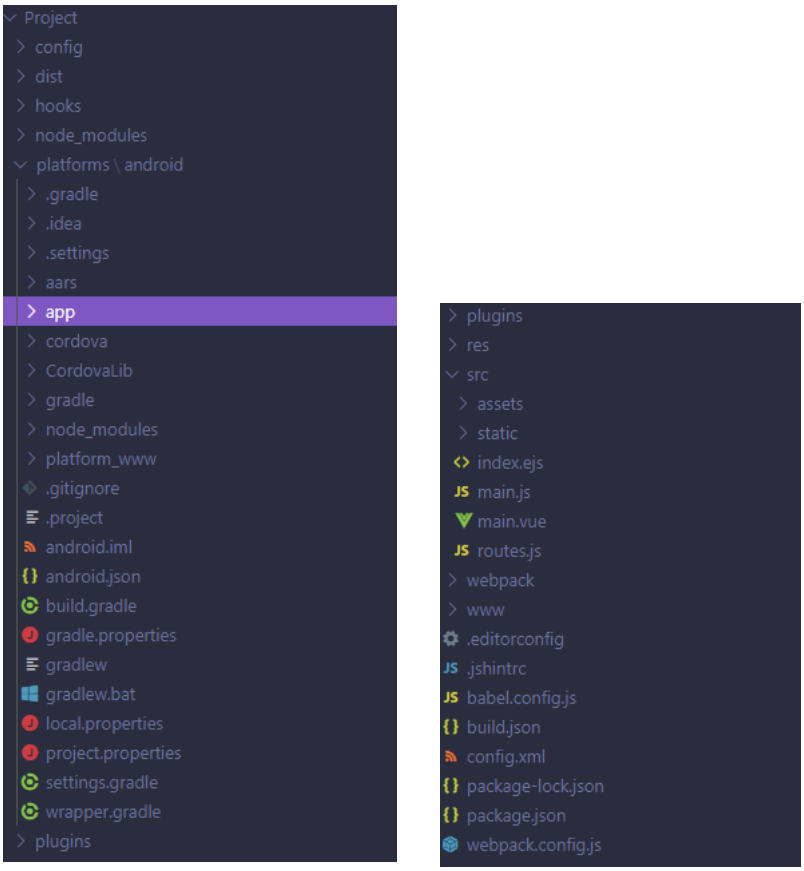

Figura 6.1 Estructura de Proyecto Cordova/Vue

6.1 Lógica de Negocio

La lógica de negocio recoge las peticiones realizadas por el desarrollador Front-End, ejecuta los procesos que deben cumplirse y generar el resultado correspondiente en caso de ser necesario.

En este caso, para implementar la lógica de negocio se han desarrollado tres plugin de Cordova para ser ejecutados sobre Android utilizando Java.

Para desarrollar un plugin en Cordova es necesario crear una *Activity* base que herede de la clase *CordovaPlugin* sobrescribiendo el método *execute*() que recibe la acción que el plugin debe realizar, así como sus argumentos y contexto. A partir de ahí puede ser ejecutado desde el código JavaScript de nuestra aplicación en cualquier punto.

6.1.1 Speech-to-Text

El módulo de Speech-to-Text aborda la implementación del reconocimiento de voz.

En la Figura 6.2 se puede ver la estructura de un plugin que puede ser instalado en cualquier aplicación Cordova que lo necesite. Está formado por los archivos Java necesarios para desarrollar el plugin. La carpeta www contiene un fichero JavaScript que contiene los métodos que pueden ser llamados desde la aplicación web y se utiliza como capa de abstracción entre la aplicación web y la parte nativa. Este fichero no es necesario para que el plugin funcione, pero se añade para la comodidad del desarrollo desde la parte Front-End. El fichero *plugin.xml* contiene el nombre del plugin, sus permisos para ser utilizado y un direccionamiento de los archivos fuente en la aplicación final. En resumen, es el archivo de configuración del plugin. Por último, el fichero *package.json* contiene los parámetros del plugin para ser instalado en una plataforma soportada, versión del plugin, licencia, etc.

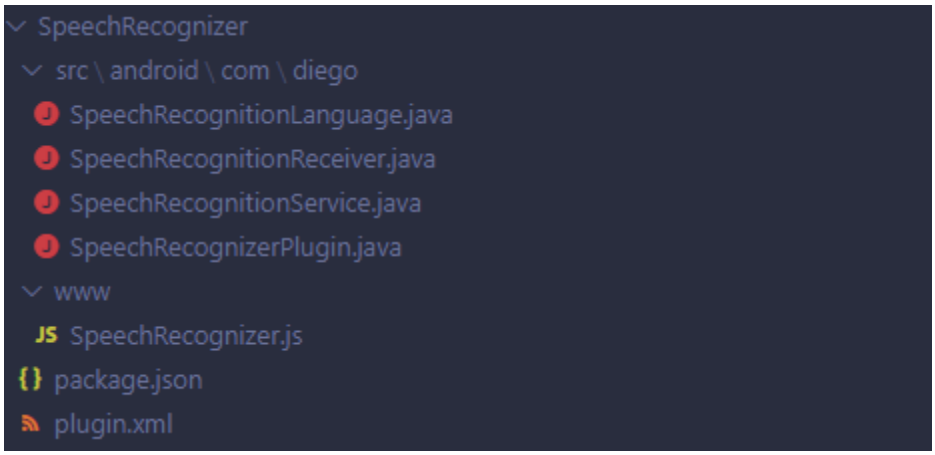

Figura 6.2 Estructura Plugin Reconocimiento de Voz

Este plugin se implementa como un servicio en Android que se queda escuchando a la espera de que el usuario diga algún comando. Actúa de manera transparente al usuario, eliminando los *popup* típicos del reconocimiento de voz de Google. Además, la librería de Android para el reconocimiento de voz no permite quedarse a la escucha en espera de la voz por lo que al implementarlo como servicio podemos cubrir esa carencia reiniciando el reconocimiento de voz cada vez que termine sin encontrar una entrada de audio. Esto pierde alrededor de 100ms en el reinicio por lo que puede perder alguna entrada de audio. El plugin contiene cuatro clases Java.

- **SpeechRecognizerPlugin**. Clase principal del plugin. Extiende de CordovaPlugin, necesario para poder utilizarlo desde la aplicación web. En esta clase se reciben las peticiones por parte del Front, permite las siguientes operaciones:
    - **startListening**. Comienza el reconocimiento de voz. Permite que se envíen los comandos a escuchar y el idioma que se va a escuchar. Si no

se envían comandos y no se han incluido previamente devuelve un error, si no se envía un idioma se pone por defecto español de España.
    - **stopListening**. Para el reconocimiento de voz.
    - **getAvailableLanguages.** Retorna los idiomas disponibles para el reconocimiento de voz.
    - **getLanguage.** Retorna el lenguaje actual que se está escuchando o aquel que se escucharía si no se indicase ningún idioma al iniciar.
    - **setLanguage**. Cambia el idioma del reconocimiento antes de comenzar. Si ya está iniciado en un idioma requiere que se pare para cambiar el idioma. Recibe un idioma como cadena de caracteres con formato "ee-EE".
    - **setCommands**. Permite introducir nuevos comandos a reconocer por el sistema, la actualización se puede hacer en tiempo de ejecución. No es necesario parar y reiniciar el plugin. Se crean dos estructuras de datos: una, si el comando requiere información adicional como puede ser un número y otra, para comandos solo de texto. Recibe una lista de comandos.
    - **addCommand**. Permite introducir un comando en tiempo de ejecución. Recibe un comando como cadena de caracteres.
    - **removeCommand.** Elimina el comando de la lista correspondiente. Recibe el comando a eliminar como cadena de caracteres.
    - **getCommands.** Retorna los comandos que se están escuchando o se escucharan cuando inicie el reconocimiento.
    - **hasPermission.** Comprueba si el dispositivo tiene permiso para acceder al micrófono.
    - **requestPermission.** Solicita al usuario permisos para acceder al micrófono si este no los ha dado ya.
    - **isListening.** Comprueba si el sistema de reconocimiento de voz está funcionando.
- **SpeechRecognitionService.** Esta clase provee el servicio de reconocimiento de voz. Su implementación consiste en un servicio de Android que se ejecuta de forma transparente al usuario. Este servicio crea una instancia de Android Speech Recognizer con su correspondiente listener. El funcionamiento del listener es devolver un error a la clase *SpeechRecognitionReceiver* si se produce cualquier tipo de error excepto un error de audio, error de ninguna coincidencia o error de tiempo sin hablar; en estos casos se reinicia el Recognizer y se vuelve a escuchar al usuario. Si se devuelve un resultado se envían los resultados al receiver y se reinicia el servicio para seguir escuchando. El servicio se para si la aplicación entra en segundo plano ya que esos comandos no se van a poder tratar desde la aplicación web.
- **SpeechRecognitionReceiver.** Esta clase extiende de *BroadcastReceiver* lo cual permite que reciba los resultados del servicio. Esta clase es la encargada de procesar los resultados ya sean errores o resultados. En el caso de ser errores se devuelve el error como respuesta del plugin. Si son resultados, se busca alguna coincidencia con los comandos a reconocer. Si se reconoce un comando, se envía un evento de Cordova que debe ser capturado en el Front-End. Este

resultado contiene el ID del comando encontrado y en caso de tener parámetros adicionales se envían en formato JSON.

- **SpeechRecognitionLanguage.** Clase auxiliar que recibe los resultados de la consulta realizada getAvailableLanguages.

Se añade además el fichero SpeechRecognizer.js que contiene una API para acceder a todas las funciones que pueden ser llamadas desde el Front-End.

6.1.2 Text-to-Speech

El módulo Text-to-Speech corresponde con la implementación del sintetizador de voz en el sistema.

En la **Figura 6.3** se puede ver la única clase que ha sido necesaria para el sintetizador de voz, aunque se podría haber divido en dos clases una para el plugin y otra para el listener de la librería de Android Text-to-Speech. Se decidió una solo clase ya que no iba a ser reutilizable ninguna de las dos clases por si sola y no afectaba al entendimiento del código.

- **TextToSpeech**. Clase que contiene el plugin y toda la funcionalidad del sintetizador de voz- A continuación, se explican las distintas funcionalidades que proveerá el plugin a los desarrolladores Front-End:
    - **speak.** Es el método principal del sistema, convierte un texto pasado como cadena de caracteres a una voz en el idioma que se indica. Si no se indica ningún texto devuelve un error, si no se indica el idioma se selecciona por defecto español de España.
    - **stopSpeak.** Detiene la reproducción de un texto por completo en el caso de que haya alguna sintetización en curso.
    - **checkLanguage.** Retorna un array de JSON con los idiomas disponibles para el sintetizador.
    - **setLanguage.** Asigna un idioma al sintetizador de manera que la próxima vez que se quiera reproducir un texto y no se indique ningún idioma se ejecute el método utilizando este idioma.
    - **getLanguage.** Retorna el idioma seleccionado.
    - **isSpeaking.** Comprueba si el sintetizador está ejecutando alguna conversión de texto en voz.

El fichero TextToSpeech.js contiene una API de acceso a estas funcionalidades desde el Front de la aplicación.

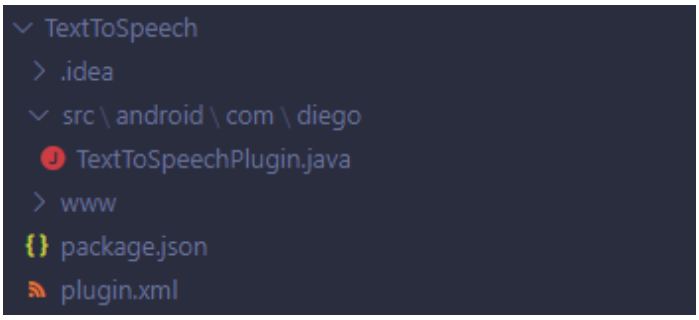

Figura 6.3 Estructura Plugin Sintetizador de Voz

6.1.3 Reconocimiento de Actividad

El módulo de reconocimiento de actividad es capaz de enviar un evento de Cordova cada vez que haya una actualización de estado del usuario; si la API de Google detecta dos veces el mismo estado no se envía nada. La parte de negocio del reconocimiento de actividad se ha realizado en Android. Además de la parte de Android este módulo tiene un componente de Vue con la parte de negocio específica para cada aplicación de Fieldeas. Este componente se debe a que la parte nativa de cada app no sabe los datos que están siendo tratados, por ejemplo, las paradas o los pedidos de una ruta, en esta parte es necesario acceder a detalles de las paradas para saber que está haciendo el usuario en un momento específico y poder reaccionar al estado.

La implementación del reconocimiento de actividad ha constado de tres clases (ver **Figura 6.4**):

- **ActivityRecognitionPlugin**. Esta clase contiene la funcionalidad básica de los plugin de Cordova, las funcionalidades que se usan desde el Front y la conexión a la API de Google. Este plugin únicamente tiene tres funcionalidades para el Front:
    - **activityRecognition.** Inicia el servicio de reconocimiento de actividad y registras las posibles actualizaciones de estado en la API de Google. Pueden seleccionarse solo los estados que se desee reconocer, en este caso se han seleccionado todos. Por último, inicia el receptor de resultados.
    - **disableActivityRecognition.** Deshabilita el servicio de reconocimiento de actividad y cancela las actualizaciones de la API que se han añadido al iniciar el servicio.
    - **currentActivity**. Retorna la última actividad detectada por el servicio.
- **ActivityRecognitionService**. Servicio de Android que comprueba en segundo plano los sensores del dispositivo y obtiene el estado en que se encuentra el usuario. El resultado es una lista de posibles estados con una confianza entre 0 y 100. El resultado que más confianza tenga se envía al receptor creado previamente junto con la confianza.
- **ActivityReceiver**. Esta clase se encarga de recibir el resultado del servicio, comprobar que el resultado recibido es válido y enviar un evento de Cordova al WebView de la aplicación.

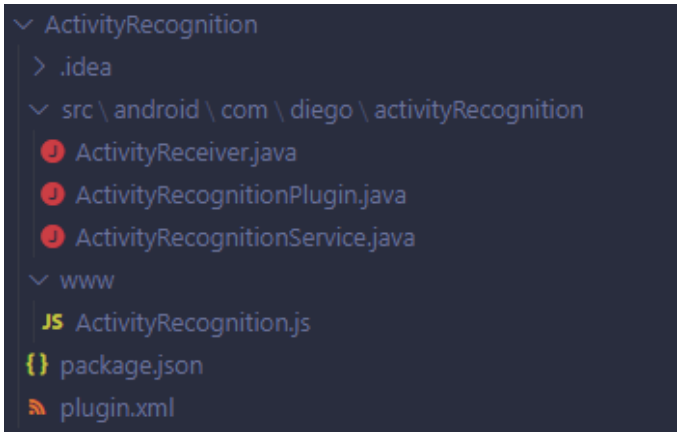

Figura 6.4 Estructura Plugin Reconocimiento de Actividad Android

Al igual que los otros dos plugin, se ha desarrollado una interfaz que permite acceder a las funcionalidades del plugin de una manera más sencilla y transparente para el desarrollador Front-End

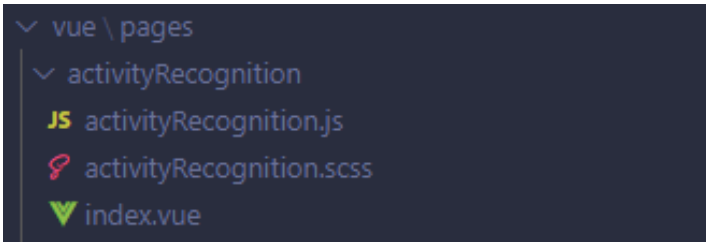

Figura 6.5 Estructura Componente de Reconocimiento de Voz Vue

Esta es la estructura básica de un componente de Vue, es similar a una página web con la particularidad de que el archivo *index.vue* que contiene la estructura de la vista en HTML utiliza *templates* y no utiliza las etiquetas *head* o *body* de un archivo HTML común. Se ha decidido implementar como un componente para que sea lo más reutilizable posible ya que era uno de los requisitos principales del sistema.

La funcionalidad implementada en este componente en relación al detector de actividades consiste en conocer si un usuario se encuentra en una parada o ha finalizado una ruta. Para ello se ha desarrollado un algoritmo que comprueba que un usuario que estaba conduciendo ha pasado a un estado “a pie”, si este estado no cambia durante un tiempo “x”, en este caso se indica un minuto (pero es totalmente personalizable) se le pregunta al usuario si ha llegado a una parada, al final de la ruta, si no se obtiene la respuesta se puede activar el GPS y se compara con los datos de las paradas si los hubiere.

6.2 Capa de Presentación

Para la capa de presentación se ha desarrollado una aplicación sencilla similar a las utilizadas en producto *Track & Trace* de Fieldeas que utiliza todas las funcionalidades previamente descritas. Para los tres módulos no es necesaria ninguna interfaz de usuario, no obstante, se ha realizado una sencilla interfaz que se muestra a continuación. Esta capa se ha desarrollado como una aplicación de prueba.

En la figura 6.6 se puede ver la página principal de la aplicación Demo. Esta aplicación es totalmente navegable a través de comandos de voz, te responde con voz y detecta la actividad en la que está el usuario. Se trata de una aplicación sencilla para comprobar el correcto funcionamiento de todos los módulos implementados.

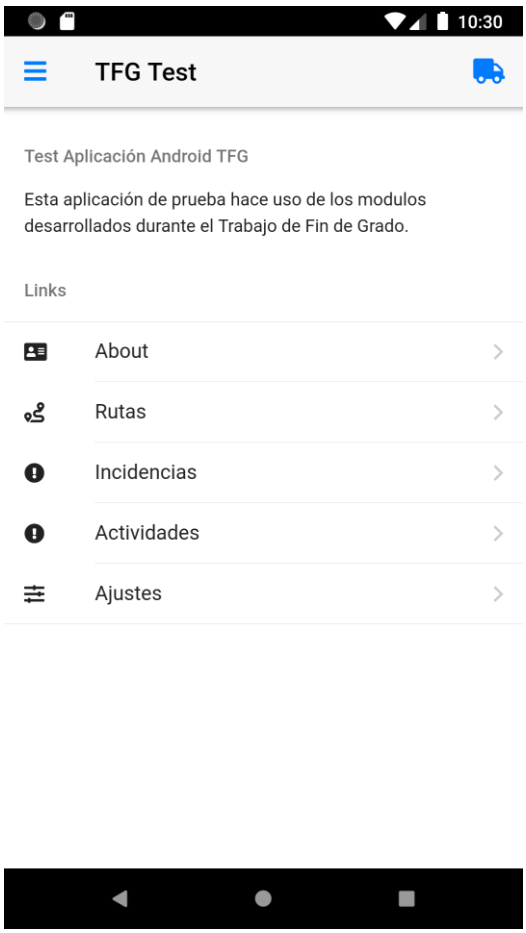

Figura 6.6 Interfaz Aplicación de Prueba

6.3 Gestión de la Configuración

Para la gestión de los cambios en el código, algo vital en cualquier empresa del ámbito del desarrollo software, para un correcto mantenimiento, una gestión de los cambios correcta, un trabajo colaborativo eficiente y una gestión de las versiones de los productos, se ha utilizado Bitbucket como cliente Git con Sourcetree para un uso más sencillo de los repositorios. Ambos productos son de la empresa Atlassian y son los usados por la compañía para la gestión de la configuración.

7. Integración en Fieldeas

La aplicación *Fieldeas* es la aplicación base de la empresa, sobre ella se pueden realizar modificaciones a medida para cada solución. La aplicación se encuentra tanto en Android como en iOS, aunque este proyecto se ha realizado sobre la plataforma Android. Para integrar los módulos requeridos dentro de la aplicación es necesario instalar los plugins desarrollados sobre la propia aplicación, la cual ya tiene otros plugins instalados para otros procesos de negocio de la compañía.

La integración se realiza directamente sobre el *core* de la aplicación, de esta manera cualquier aplicación derivada de esta pude tener acceso a los plugins desarrollados. En la instalación de un plugin sobre la plataforma es necesario añadir las clases Java desarrolladas en la carpeta plugin del proyecto, además de modificar la configuración tanto de la aplicación global como en la configuración de los plugins de Cordova. La modificación de la configuración global se realiza sobre el *AndroidManifest.xml*. En este fichero es necesario añadir los servicios incluidos, los receptores de resultados y todos los permisos que se necesitan para los plugins funcionen correctamente.

Por último, es necesaria una modificación de la API de *Fieldeas* para el desarrollo móvil. Esta API provee acceso a todas las funcionalidades internas del dispositivo. Para la modificación de la API es necesario incluir los métodos desarrollados en los archivos *ActivityRecognition.js, SpeechRecognizer.js* y *TextToSpeech.js.* El componente desarrollado para el reconocimiento de actividad se subió al repositorio de la compañía y puede ser usado por cualquiera incluyéndolo en su proyecto.

8. Pruebas

Cualquier producto software es susceptible de tener errores, para ello ha sido necesaria la creación de una batería de pruebas con el fin de que todo el sistema funcione correctamente acorde con los requisitos del sistema.

El proceso de pruebas se divide en tres grupos:

- **Pruebas unitarias.** Cada módulo se prueba de manera independiente.
- **Pruebas de integración**. Se integran todos los módulos en el sistema final y se prueba el correcto funcionamiento.
- **Pruebas de aceptación**. Se prueba que el sistema tenga la funcionalidad requerida acorde con los requisitos capturados.

Las pruebas han sido realizadas en dos dispositivos Android distintos y en dos emuladores distintos para comprobar el funcionamiento en diferentes dispositivos.

8.1 Pruebas Unitarias

Por la peculiaridad de este proyecto, no es posible automatizar las pruebas unitarias. No se puede automatizar el input del reconocimiento de voz, en la síntesis de voz no se puede saber si se ha realizado correctamente de forma automática con un framework como Junit y el reconocimiento de actividad depende de datos de los sensores. No he encontrado la manera de poder automatizar la modificación de estos datos de manera correcta.

Por estos motivos las pruebas se han realizado de manera manual por el propio desarrollador. Para realizar las pruebas se creó una aplicación Android sobre la que corre Cordova y una aplicación que posteriormente serviría como Demo del proyecto.

La mayoría de las pruebas se realizaron desde la consola del inspector de Chrome. A través de los comandos es posible llamar a las funciones de Cordova y a partir de ahí a las funciones desarrolladas.

8.2 Pruebas de Integración

Con objeto de probar cómo se incluyen las funcionalidades desarrolladas en un aplicativo, así como estas en sí mismas se desarrolló una aplicación web con Vue JS a modo de prototipo. Primero se ejecutó sin la integración en la aplicación de Fieldeas.

La mayoría de las funcionalidades fueron llamadas desde la propia aplicación, no obstante, algunas funcionalidades se llamaron desde la consola de Chrome Devtools

Una vez todos los módulos funcionaban correctamente juntos se integró en la aplicación de Fieldeas y se probó con la misma aplicación. Lo único que cambia es la base sobre la que corre la aplicación.

El resultado fue la aplicación mostrada en el apartado **6.2 Capa de Presentación**, una app que funcionaba mediante comandos de voz, era capaz de responderte al cambiar de vista y contenía una pantalla en la que se mostraba la actividad en la que se encontraba el dispositivo.

Una vez que estas pruebas fueron satisfactorias, solo quedaba validar que cumplía todos los requisitos del sistema.

8.3 Pruebas de Aceptación

Las pruebas de aceptación se realizaron sobre la aplicación Demo, para ello se comprobó que cumplía todos los requisitos capturados a lo largo del proyecto. Por último, se hizo una demostración al tutor del proyecto con el fin de que una persona externa al desarrollador verificara que el sistema cumplía con lo especificado.

9. Conclusión y Trabajos Futuros

En este capítulo se va a valorar de forma general el desarrollo del trabajo de fin de grado y los posibles cambios, bajo mi punto de vista, que pueden llevarse a cabo en el futuro.

9.1 Conclusiones

El objetivo de este proyecto era la implementación de las nuevas tecnologías de reconocimiento de voz y síntesis de voz, así como del reconocimiento humano de actividad para mejorar la usabilidad de una aplicación dirigida a los profesionales del transporte. Estos profesionales necesitan distintos métodos de interacción con los dispositivos que utilizan para gestionar su trabajo ya que no se pueden permitir distracciones.

Lo primero que se ha de señalar es que se han cumplido todos los requisitos especificados al inicio, excepto el uso offline del reconocimiento de voz que, aunque no era un requisito inicial se añadió durante el desarrollo del proyecto.

Este proyecto se ha divido en tres partes, cada una dedicada a un módulo de la aplicación. En la primera, se realizó el estudio de las distintas tecnologías existentes de reconocimiento de voz y se realizaron varias implementaciones hasta encontrar la que mejor se adaptaba a los requisitos. A continuación, se abordó el sintetizado y reconocimiento de voz, cuya API fue especificada por la empresa y cumplía los requisitos por lo que no fue necesaria más que su integración. Por último, el reconocimiento de actividad se llevó a cabo mediante la Api de Google, la cual fue proporcionada en los requisitos del proyecto.

Este proyecto me ha permitido conocer en profundidad el entorno de Cordova desde el lado nativo, en este caso Android y trabajar con las nuevas tecnologías de desarrollo de aplicaciones móviles multiplataforma como es Vue JS. Además, me ha permitido conocer cómo funcionan los sistemas de reconocimiento de voz y actividad en profundidad, llegando a ver técnicas de *Deep Learning* para el reconocimiento de actividad.

La posibilidad de realizar el proyecto dentro de un entorno profesional me ha ayudado a conocer como es el trabajo en equipo, como se estructuran los proyectos software en una empresa y cuáles son las tecnologías que se utilizan actualmente en el mundo profesional. Además, el disponer de compañeros profesionales me ha permitido acceder a consejos y conocimiento que no hubiese sido posible si no hubiese realizado el proyecto en una empresa, por lo que estoy muy agradecido a la empresa Fieldeas por permitirme realzar este proyecto con ellos.

Para finalizar, Fieldeas incluyó el módulo de reconocimiento de voz en una demostración con un posible futuro cliente.

9.2 Trabajos Futuros

El proyecto que se ha llevado a cabo ha cumplido con todos los requisitos excepto con uno, el cual es el principal objetivo de futuro.

- Utilizar CMUSphinx con un modelo en castellano para conseguir un reconocimiento de voz offline y en el idioma principal del sistema.
- En caso de no conseguir un modelo en castellano para esta librería, utilizar *Deep Learning* para entrenar una red neuronal con el *dataset* de Mozilla, este *dataset* tiene más de 200 horas de audio en castellano y su transcripción para poder entrenar modelos.

La primera propuesta se ha intentado desarrollar durante el proyecto, pero el resultado no fue satisfactorio como se explicó en el apartado 2.1. Como consecuencia de no disponer de un conjunto de datos de entrenamiento adecuado.

Como trabajos futuros fuera de los requisitos del sistema se pueden encontrar los siguientes:

- **Mejorar el rendimiento**. Los tiempos de conexión a las APIs, en especial el reconocimiento de voz puede ser algo lento en condiciones con poca conexión a internet. Esto mejoraría con las técnicas para reconocimiento de voz offline como se ha explicado anteriormente.
- **Eliminar sonido al inicio/fin del reconocimiento de voz**. Al iniciar y terminar el reconocimiento de voz, la API genera un sonido para que el usuario sepa que se inicia un reconocimiento de voz. Este sonido está bien cuando es un único reconocimiento, pero en la implementación proporcionada este sonido es molesto ya que se repite cada 10 segundos (aproximadamente). Este sonido se suprime de forma manual en el plugin, pero a costa de bajar el volumen multimedia de todo el dispositivo. La solución pasaría por suprimir el sonido desde que se llama a la API, desconozco si es posible.
- **Mejorar comandos reconocidos**. Por ahora el reconocimiento de voz reconoce comandos exactos y comandos con algún número en su interior. Mejorar esta detección para reconocer nombres propios personalizables, por ejemplo, el comando “ir a” + una ciudad podría abrir el mapa por defecto del dispositivo y generar la ruta a esa ciudad.

Referencias

[1] *History of Voice Recognition (19/02/2018).* **Visto** (05/02/2020)
https://www.happyscribe.co/blog/history-voice-recognition/
[2] *Introduction to Speech Recogntion Software*. **Visto** (05/02/2020)
https://www.voicerecognition.com.au/pages/introduction-to-speech-recognition-software
[3] *How Speech Recognition Works.* **Visto** (06/02/2020)
https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition1.htm
[4] *What is Speech Recognition Software?.* **Visto** (06/02/2020)
https://www.callrail.com/blog/speech-recognition-software/
[5] *Transcripcion de Voz de Cloud.* **Visto** (07/02/2020)
https://cloud.google.com/speech-to-text
[6] *Amazon Transcribe*. **Visto** (10/02/2020)
https://aws.amazon.com/es/transcribe/faqs/?nc=sn&loc=5
[7] *Open Source Speech Recognition Toolkit.* **Visto** (10/02/2020)
https://cmusphinx.github.io/
[8] *CMUSphinx.* **Visto** (10/02/2020)
https://en.wikipedia.org/wiki/CMU_Sphinx
[9] *Documentation Android Speech.* **Visto** (12/02/2020)
https://developer.android.com/reference/android/speech/package-summary
[10] *Documentation Android Service Voice.* **Visto** (13/02/2020)
https://developer.android.com/reference/android/service/voice/package-summary
[11] *Microsoft Azure Speech to Text.* **Visto** (13/02/2020)
https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/
[12] *IBM Watson Speech to Text*. **Visto** (13/02/2020)
https://www.ibm.com/cloud/watson-speech-to-text
[13] *Sistemas de reconocimiento de voz, ¿la correcta evolución a la hora de crear contenidos?.* **Visto** (17/02/2020)
https://www.marketingdirecto.com/digital-general/digital/sistemas-reconocimiento-voz-correcta-evolucion-hora-crear-contenidos
[14] *History and Development of Speech Synthesis.* **Visto** (24/02/2020)
http://research.spa.aalto.fi/publications/theses/lemmetty_mst/chap2.html
[15] *Speech Synthesis.* **Visto** (24/02/2020)
https://en.wikipedia.org/wiki/Speech_synthesis
[16] *Síntesis de Voz.* **Visto** (24/02/2020)
https://www.monografias.com/trabajos89/sintesis-voz/sintesis-voz.shtml
[17] *Reconocimiento y Síntesis de voz.* **Visto** (27/02/2020)
http://recursostic.educacion.es/observatorio/web/es/software/software-general/689-reconocimiento-y-sintesis-de-voz
[18] *An Introduction to Text-To-Speech in Android.* **Visto** (02/03/2020)
https://android-developers.googleblog.com/2009/09/introduction-to-text-to-speech-in.html
[19] *Arquitectura de la Plataforma Android.* **Visto** (05/03/2020)
https://developer.android.com/guide/platform
[20] *Android Runtime.* **Visto** (05/03/2020)
https://en.wikipedia.org/wiki/Android_Runtime

[21] *Cordova Framework.* **Visto** (05/03/2020)
https://cordova.apache.org/docs/en/latest/
[22] *Activity Recognition.* **Visto** (27/03/2020)
https://en.wikipedia.org/wiki/Activity_recognition
[23] *Deep Learning Models for Human Activity Recognition.* **Visto** (30/03/2020)
https://machinelearningmastery.com/deep-learning-models-for-human-activity-recognition/
[24] *Responding to User Activity with the Activity Recognition API.* **Visto** (30/03/2020)
https://www.androidauthority.com/using-the-activity-recognition-api-829339/
[25] *Activity Recogntion.* **Visto** (06/04/2020)
https://www.cs.dartmouth.edu/campbell/cs65/lecture22/lecture22_2018.html
[26] *Activity Recogntion API.* **Visto** (07/04/2020)
https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi