Departamento de Ingeniería Informática

PROYECTO FIN DE CARRERA

DESARROLLO DE UNA APLICACIÓN MÓVIL EN EL ÁMBITO DE LA LOGOPEDIA

Autor: Alfonso García-Vaquero Aguilera

Tutor: Dr. David Griol Barres

Leganés, enero de 2016

Título: DESARROLLO DE UNA APLICACIÓN MÓVIL EN EL ÁMBITO DE LA LOGOPEDIA
Autor: Alfonso García-Vaquero Aguilera
Director: David Griol Barres

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Agradezco en primer lugar al tutor de este PFC, David Griol Barres la paciencia y la dedicación que ha tenido conmigo desde el principio. También agradezco la ayuda y ánimos que me han dado mis amigos y compañeros de la universidad a lo largo del proceso.

Quiero agradecer especialmente a toda mi familia, que siempre me ha dado todo su apoyo y me han ayudado a superar los malos momentos. En especial quería agradecer a mi abuela Pura, cuyo esfuerzo y dedicación para superar sus problemas de afasia, han motivado en gran parte la elaboración de este proyecto.

Por último quiero acordarme de mi padre, que aunque no esté con nosotros físicamente, siempre ha estado presente en todo momento en nuestros corazones y pensamientos, y siempre ha sido y será un referente en mi forma de actuar. También a mi hermano Antonio, que siempre ha estado ahí, compartiendo tanto los buenos momentos de mi vida, como los no tan buenos y los que nos quedarán por compartir, comportándose como el mejor de los hermanos. Y por supuesto agradecer a mi madre, sin cuyo esfuerzo y cariño no habría podido superar los escollos que nos ha puesto la vida y que siempre confió en mi incondicionalmente.

Solo me queda añadir, ¡Muchas gracias a todos!!Muchas gracias mamá!

Resumen

Para este proyecto de fin de carrera se ha optado por desarrollar una aplicación móvil en el ámbito de la salud y la logopedia, cuyo objetivo es ayudar a tratar a personas con enfermedades del habla o problemas que deriven de alguna complicación en este campo, mediante el uso de las plataformas que proporcionan los smartphones.

En la aplicación desarrollada llamada LOGOPEDIA, se implementa una combinación de herramientas útiles y ejercicios de rehabilitación, cuyo objetivo es facilitar a estas personas la interacción con el resto de las personas, teniendo en cuenta el estado en el que se encuentran actualmente. A su vez se tratará de que el usuario pueda mejorar en sus afecciones a través de su interacción con la aplicación .Para lograr el objetivo propuesto se ha decidido usar el sistema operativo Android, ya que tiene un alto índice de penetración y permitirá llegar al mayor número de usuarios, además de contar con los sintetizadores de texto a voz y de voz a texto, los cuales se han usado para construir las herramientas presentes en la aplicación.

A lo largo de esta memoria se pormenorizan las distintas afecciones del habla que se han decidido tratar, los ejercicios y herramientas propuestos y la forma de interactuar con la aplicación de LOGOPEDIA, para que el usuario consiga los efectos deseados. A su vez, también se tratarán los aspectos técnicos, desglosando las opciones de implementación que se han barajado y se explicará cómo esta implementado el sistema operativo Android y como desarrollar una aplicación para dicho sistema operativo

Por último se arrojarán las conclusiones finales y se propondrán las líneas a seguir para posibles desarrollos futuros de la aplicación.

Palabras clave: Android, e-health, enfermedades del habla, logopedia, interacción oral.

Abstract

Throughout this report it will be detailed the different speech disabilities that have been possible to treat, the exercise and tools proposed and the way to interact with the LOGOPEDIA application, so the user can get all the expected benefits. At the same time, it will be described the technical aspects, detailing the different implementation options that have been evaluated, explaining how the Android Operating system is implemented and how to develop an application in this environment.
Lastly, it will be presented the final conclusions of the report and will propose the guidelines for future enhancements of the application.

Keywords: Android, e-health, speech disabilities, spoken interaction.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1
Introducción y objetivos

El presente capítulo de la memoria pretende exponer la motivación que ha llevado a la elaboración del PFC, así como ilustrar los objetivos que pretenden conseguirse con la elaboración del mismo. Además de esto, en este capítulo se describirá la estructura de la memoria y se incluirá el presupuesto y los recursos empleados para la elaboración del mismo.

1.1 Motivación

Actualmente el mercado de las aplicaciones es un mercado en alza. La aparición de numerosos soportes tecnológicos (Smartphones,PCs,tablets,etc) han permitido que las aplicaciones lleguen a gran cantidad de usuarios de todas las edades de una forma muy barata y efectiva.

A su vez se detecta que la mayoría de las aplicaciones existentes, están orientadas a un público joven y con un objetivo lúdico. Dentro del amplio abanico de aplicaciones, este proyecto se ha centrado en las aplicaciones cuyo objetivo es ayudar a la gente a tener mejor calidad de vida.

Este proyecto se planteó a su vez, como un proyecto en el que se podría dar una utilidad a las herramientas de reconocimiento de voz y síntesis de texto a voz en el campo de las aplicaciones móviles

Posteriormente se inició la búsqueda de los usuarios que podrían beneficiarse de una aplicación que hiciese uso de estos elementos. Investigando sobre todo tipo de enfermedades, se detectó que hay unas determinadas afecciones, las afecciones del habla, que afectan a un amplio espectro de población de todas las edades. Estas enfermedades tienen un amplio margen de mejora si se sigue un tratamiento continuado con logopedas.

El problema radica en la dificultad y el coste asociado que implica tener a un logopeda dedicado al tratamiento de forma continuada. Este problema se pretende atajar con la aplicación LOGOPEDIA, ya que proporcionará a los usuarios un proceso de rehabilitación y unas herramientas que les permitan enfrentarse a la vida cotidiana y practicar en el día a día contando a su vez con la supervisión de un profesional si lo estiman oportuno.

1.2 Objetivos

El presente PFC tiene como objetivo principal, la elaboración de una aplicación móvil Android en el ámbito de las aplicaciones médicas, que cubra la carencia actual de aplicaciones dedicadas al campo de la logopedia en el Android Store y que use para conseguir este fin las herramientas de síntesis de texto a voz (Text To Speech) y reconocimiento de voz (Speech Recognition), que se implementan en el API de Android. Para ello se persigue cumplir con los siguientes puntos:

- Mostar como desarrollar una aplicación móvil Android, describiendo para ello la estructura del sistema operativo, las diferentes herramientas que se deben usar para la planificación y desarrollo de la misma. Este punto pretende clarificar a su vez, los elementos que componen una aplicación Android y cómo interactúan ente ellos.
- Esta memoria pretende también exponer el funcionamiento de las herramientas de reconocimiento de voz y de síntesis de texto a voz, explicando sus características, usos y funcionamiento. Se debe mostrar a su vez el API que posee Android para implementar dichas herramientas y como estas se pueden integrar en una aplicación.
- Estudiar las enfermedades del habla, el espectro de población a la que afecta y los tratamientos que se deben seguir para que el paciente experimente una mejoría en dichas afecciones. Además debe concretarse que parte del tratamiento puede extrapolarse al ámbito de las aplicaciones móviles, para conseguir que el efecto de la aplicación sea beneficioso para los pacientes.
- Como último punto se pretende que esta aplicación pueda servir como punto de partida para desarrollos futuros, o inspire al desarrollo de otras aplicaciones en el ámbito de la logopedia y la medicina.

1.3 Estructura de la memoria

La memoria de este PFC se ha estructurado en siete capítulos más glosario y bibliografía. A continuación detallamos el contenido de estos capítulos:

- **Capítulo 1. Introducción:** En este primer capítulo se describe la motivación que ha llevado a la realización del mismo, se definen los objetivos que se pretenden conseguir mediante la realización del PFC y se describe la estructura de la memoria desarrollada. Este capítulo incluye también un detalle de los recursos que han sido necesarios para la elaboración del proyecto y el coste en el que se ha incurrido en la elaboración del proyecto y la memoria resultante del mismo.

- **Capitulo2. Estado del arte:** Este segundo capítulo está destinado a sintetizar toda la información recabada para la elaboración de este proyecto y mostrarla de manera didáctica y sencilla. Este capítulo será la base del desarrollo de los capítulos posteriores, ya que en la información contenida en el mismo es en la que se han basado todas las decisiones tomadas para la elaboración del proyecto.

  En este capítulo se estudiarán los dispositivos móviles, sus aplicaciones y los factores a tener en cuenta a la hora de desarrollarlas, así como un estudio de los principales sistemas operativos justificando la elección de Android como SO para desarrollar la aplicación LOGOPEDIA. Se estudiarán a su vez las enfermedades del habla, centrándose en dos: afasia y dislexia. Por último se realizará un estudio sobre las aplicaciones que hay ya creadas en este ámbito y se arrojarán conclusiones sobre las mismas.

- **Capítulo 3. Estructura de aplicaciones Android:** En este capítulo se describirá la estructura del sistema operativo Android. Se explicará además la estructura que tiene una aplicación Android, y los elementos que la componen.

- **Capítulo 4. Análisis y diseño de la aplicación:** En este capítulo se analizan los requisitos que se deben tener en cuenta a la hora de implementar la aplicación LOGOPEDIA, para adecuarla a los usuarios a los que va dirigida. Posteriormente se muestra el diseño elegido para el desarrollo, mostrando y describiendo todas las pantallas que se mostrarán al usuario durante la ejecución de la aplicación.

- **Capítulo 5. Desarrollo aplicación LOGOPEDIA:** En este capítulo de describen las herramientas necesarias para crear la aplicación LOGOPEDIA. Después se muestra del diagrama de clases Java resultante de la elaboración del proyecto. Posteriormente se describirán las clases y funciones implementadas para conseguir las funcionalidades planteadas en el diseño.

- **Capítulo 6 Evaluación:** En este capítulo se explica el proceso de evaluación de la aplicación, llevado a cabo por el instituto CCEE AIDEMAR

- **Capítulo 7. Conclusiones y desarrollo futuro:** En este último capítulo se explicarán las conclusiones a las que se ha llegado al finalizar la memoria y se propondrán las posibles líneas a seguir para desarrollos futuros.

1.4 Planificación Temporal

Para la planificación temporal del proyecto, se ha subdivido el proyecto en las principales tareas a desarrollar. A continuación se detallan estas tareas:

- **Planificación:** Esta tarea tiene como objetivo estructurar todo el trabajo a llevar a cabo para el desarrollo del PFC, elaborar el diagrama de Gantt y obtener los recursos necesarios.
- **Estudio Android:** Tarea correspondiente al estudio del sistema operativo Android y bibliografía recomendada para desarrollo de aplicación para este sistema operativo
- **Estudio Enfermedades del habla:** Tarea cuyo objetivo es estudiar las distintas enfermedades del habla y tratamientos recomendados
- **Diseño:** Teniendo en cuenta los estudios realizados en las anteriores tareas, en esta tarea se aborda el diseño de la aplicación, acorde a las especificación desprendidas de las anteriores tareas.
- **Desarrollo de la aplicación:** Esta tarea engloba el tiempo destinado a desarrollar el código de la aplicación LOGOPEDIA
- **Fase de corrección de errores:** Tarea dedicada al testeo de la aplicación y posterior corrección de errores. Para ello se distribuye la aplicación a la asociación CCIE Aidemar y se recibe su feedback.
- **Elaboración de la memoria:** Tarea correspondiente a la elaboración de esta memoria de proyecto de fin de carrera. Esto incluye la recopilación de toda la documentación, redacción de la memoria y corrección de errores.
- **Elaboración presentación:** Tarea correspondiente a la elaboración y preparación de la presentación a defender ante el tribunal.

El proyecto tiene una duración total de 171 días desde su hito de inicio de proyecto hasta su hito de fin de proyecto. Con el fin de detallar la fecha de inicio y final de cada tarea se ha realizado un diagrama de Gantt. Para la realización del diagrama de Gantt se ha usado el software GanttProject.[1].GanttProject es una programa con licencia GPL elaborado por la universidad Marne la Vallée en Francia, basado en java, que permite la realización de diagramas de Gantt. La Figura 1 muestra el diagrama de Gantt resultante de la planificación del proyecto.

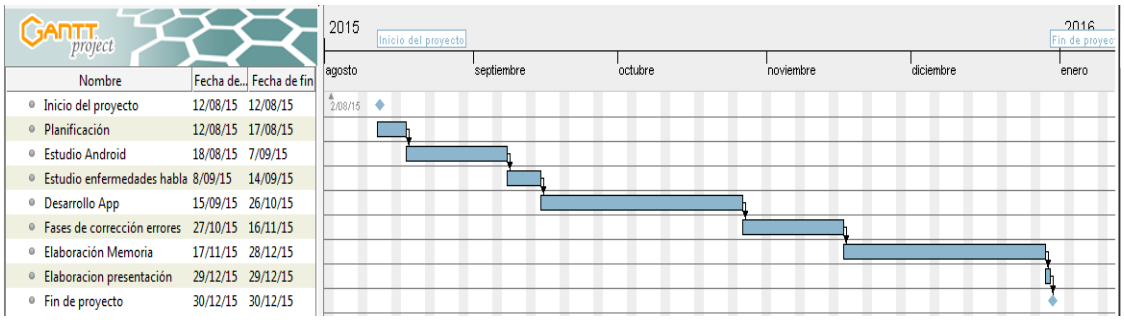

Figura 1. Diagrama Gantt PFC Aplicación Logopedia

1.5 Recursos

Para la realización de este PFC se han tenido que utilizar recursos tanto hardware como software que se detallan a continuación:

Hardware

- Ordenador portátil
- Teléfono móvil Samsung trend plus
- Cable USB
- Cable de red

Software

- Entorno desarrollo Eclipse Juno
- Plugin ADT (Android Development Kit)
- SDK (Sofware Development Kit)
- GanttProject
- Google Chrome
- Paquete Microsoft Office

El coste de estos recursos se detalla en el siguiente apartado de presupuesto.

1.6 Presupuesto

La obtención del presupuesto del PFC se ha hecho basada en las indicaciones que contiene la plantilla para la realización del PFC proporcionada por la Universidad Carlos III de Madrid [2] y cuyas partidas desglosamos a continuación.

1.6.1 Costes de personal

Los costes de personal fijados han sido calculados teniendo en cuenta una dedicación de 3 horas trabajadas por día de duración del proyecto. A continuación se desglosan los datos:

Duración total en días: 171
Horas diarias: 3
Coste diario de un ingeniero: 2694,39/22 = 122,47 €
Coste por hora de un ingeniero: 122,47/8 = 15,31 €

Total: 15,31*3*171 = **7854,03 €**

1.6.2 Costes de equipo

Hardware

- Ordenador portátil→750€
- Teléfono móvil Samsung trend plus→200€
- Cable USB→5€
- Cable de red→5€

Software

- Entorno desarrollo Eclipse Juno→0€
- Plugin ADT (Android Development Kit)→0€
- SDK (Sofware Develpment Kit)→0€
- GanttProject→0€
- Google Chrome→0€
- Paquete Microsoft Office 2010 →150€

Basándose en el cálculo de amortización de equipos indicados en la plantilla de la memoria se desprende las cifras mostradas en la Tabla 1.

Descripción	Coste(€)	%Uso dedicado	Dedicación (meses)	Periodo de depreciación (meses)	Coste imputable
Ordenador	750	100	5,7	60	71,25
Smatphone	200	100	1,5	60	5
Cable USB	5	100	1,5	60	0,125
Cable de red	5	100	5,7	60	0,475
Office 2010	150	100	2	60	5

Tabla 1. Amortización recursos de equipo

De esta tabla se desprende que el **coste total imputable a los recursos de equipo** es de **81,85€.**

1.6.3 Coste total

A los costes anteriores que son costes sin IVA, solo queda añadirles los costes indirectos, que se han calculado como un 20% del total. Resultando el siguiente presupuesto:

Descripción	Coste(€)
Costes personal	7.854,03
Costes equipo	81,85
Costes indirectos	1.587,176
Total sin IVA	9.523,056
IVA	1.999,84
Total con IVA	11.522,9

Tabla 2. Coste total PFC

El presupuesto total del proyecto asciende a la cantidad de: ONCE MIL QUINIENTOS VENTIDOS CON NUEVE EUROS

Madrid a 20 de Enero de 2016 Fdo. Alfonso Gª-Vaquero Aguilera

Capítulo 2
Estado del arte

En este capítulo se explicará la influencia que tienen los dispositivos móviles y las aplicaciones relacionadas con los mismos. Se hará una labor de investigación sobre los principales SO desarrollados para estas plataformas, justificando la elección de Android sobre todos ellos para desarrollar la aplicación LOGOPEDIA.

Posteriormente se explicará que son los módulos de reconocimiento de voz y de síntesis de texto a voz, los cuales serán la base de este proyecto, describiendo detalladamente como están realizados y que ventajas e inconvenientes presentan a la hora de realizar aplicaciones.

Por último, se realizará una descripción de las distintas afecciones del habla, donde se indicará a qué tipo de personas afecta, que tratamientos existen a día de hoy para tratar dichas afecciones y se hará una exposición de algunas de las aplicaciones de logopedia que hay desarrolladas actualmente en el Android Market.

2.1 La importancia de los dispositivos móviles y sus aplicaciones

La tecnología evoluciona rápidamente en la industria del hardware y el software. En la década de los 90, solo era común el uso de los ordenadores personales con un número limitado de usuarios, ya que los precios no eran asequibles para la mayoría de usuarios potenciales. Esa tendencia se dio la vuelta rápidamente en las décadas siguientes con la aparición de dispositivos cada vez más pequeños y potentes que permitían abaratar costes, haciendo los precios más competitivos y acercándolo a todo tipo de personas.

Ahora tenemos a nuestra disposición los conocidos como dispositivos móviles (tablets,smartphones, dispositivos wearables) a precios asequibles para el consumidor medio. Esta evolución ha provocado que el índice de penetración de los dispositivos móviles en la sociedad haya crecido enormemente. Así podemos ver datos como los que se muestran a continuación sobre la penetración de smartphones en nuestra sociedad [3]:

- En 2013 las ventas de smartphones alcanzaron 968 millones de unidades en todo el mundo. Ello supuso un incremento del 43% con respecto a las ventas en 2012
- La mayor penetración se da en Europa central y del este con una penetración del 151%
- En España hay 36 millones de personas que poseen un teléfono móvil

- La mayor penetración se da en usuarios con edades comprendidas entre 18-25 años, donde un 91% de los españoles ya poseen un Smartphone, seguida de la franja de 26-35 años con un 87%
- En España el móvil ya se ha situado como principal medio de acceso a internet por encima de los ordenadores portátiles y los PC de sobremesa.

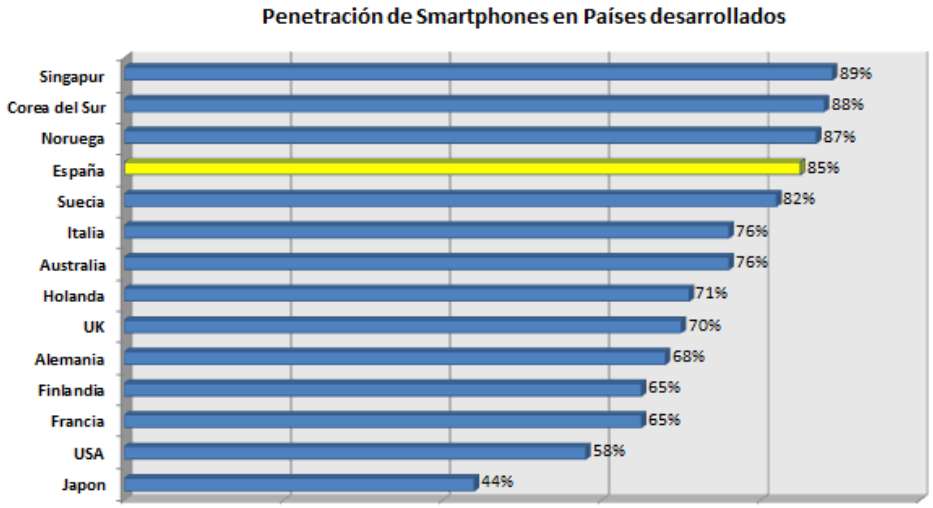

Figura 2. Penetración de smartphones en países desarrollados (cifras a finales de 2014)

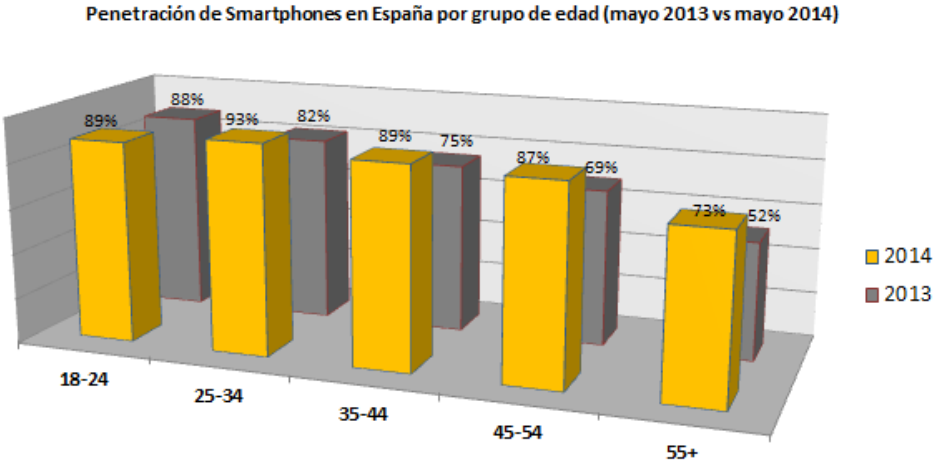

Figura 3. Penetración de smartphones en España por rango de edad 2013/2014

Con los datos anteriormente mostrados, resulta evidente que la industria del desarrollo software de aplicaciones para estas plataformas ha debido tener un crecimiento enorme en paralelo a la industria del hardware.

A lo largo de los años las app store han pasado por varios hitos, cien mil millones de aplicaciones descargadas en 2013, dos millones de aplicaciones disponibles en Octubre de 2014 y la primera aplicación en llegar a mil millones de descargas en un app store en Mayo de 2014 [4].

Según los últimos informes las app de tipo comercial han sufrido un frenazo en las ventas, ya que los usuarios ya no pagan aplicaciones al mismo ritmo que lo hacían cuando empezaron a surgir en los App markets.

Además de las aplicaciones comerciales existen las aplicaciones denominadas e-health. Estas aplicaciones en las cuales se enmarcará nuestro proyecto, son aplicaciones que pretenden facilitar las relaciones entre médico y paciente.

En un reciente congreso sobre aplicaciones destinadas a e-health, el presidente del congreso Charles Lowe aseguraba que la forma en la que más puede contribuir este campo a la medicina, es en la salud mental ya que el medico puede actuar con el paciente con un solo teléfono.

La Doctora Mireia Sans Corrales de la sección de e-Salud del colegio oficial de médicos de Barcelona, asegura que pese a que los beneficios de estas aplicaciones no están demostrados científicamente, ya son positivas solo por el hecho de promover hábitos saludables

Pese a todo, existen muchos profesionales que aunque reconocen que la sanidad autogestionada guiada por profesionales será el futuro, indican que debería crearse un marco regulador para controlar dichas aplicaciones y asegurarse que en ningún caso puedan ser nocivas.

Tras constatar el gran índice de penetración que hay de los dispositivos móviles en nuestra sociedad y teniendo en cuenta que en la sociedad actual ha sufrido un cambio en sus hábitos de consumo en cuestión de aplicaciones, se puede deducir que las aplicaciones médicas es una gran línea de negocio a desarrollar, ya que las personas cada vez tienden a vivir y a querer vivir más y mejor.

Este punto combinado con que el espectro social que ahora queda más excluido del uso de tecnologías móviles se irá renovando por personas que ya están familiarizadas con el uso de las mismas y que demandarán en gran medida el uso de estas aplicaciones, hace de las aplicaciones e-health o aplicaciones médicas una de las grandes líneas a tener en cuenta en el futuro de las aplicaciones móviles

2.2 Desarrollo aplicaciones móviles

Desarrollar aplicaciones no es una tarea sencilla. Actualmente en el mundo del desarrollo de aplicaciones existe una alta competencia, ya que existen desarrolladores muy experimentados en todo el mundo trabajando a la vez para lanzar al mercado aplicaciones relativas a todos los ámbitos comerciales. Como se desprende del estudio realizado en el punto anterior, las aplicaciones están dejando de ser rentables y se calcula que para 2018 solo un 0,01% de las aplicaciones existentes lo será.

Para que las aplicaciones sean efectivas, el desarrollador debe plantearse que pretende conseguir con su aplicación y cuál es la mejor manera de desarrollar para ello. Lo primero que hay que tener claro, es el tipo de aplicación que se quiere desarrollar A continuación se detallan los tipos de aplicaciones móviles existentes según la forma de desarrollarlas [5].

2.2.1 Aplicaciones Nativas

Se denominan aplicaciones nativas, a las aplicaciones que están desarrolladas para un sistema operativo específico, bien sea Android, IOS o cualquiera de los sistemas operativos de uso menos común. Para que estas aplicaciones puedan ser desarrolladas, estos sistemas operativos, deben de disponer de la herramienta denominada SDK (Software Development Kit).

Un SDK es un sistema de herramientas para desarrollar software, que ya posee código realizado, orientado para el desarrollo del mismo. Para realizar una aplicación en los distintos sistemas operativos, se necesita desarrollar un código en el lenguaje propio del sistema operativo. Si no se dispusiese de SDK, habría que implementar absolutamente todas las herramientas desde cero basándose solo en el lenguaje de programación y eso repercutiría fatalmente en el tiempo de desarrollo y rentabilización de la aplicación desarrollada.

Cada una de las plataformas, Android, iOS o Windows Phone, tiene un sistema diferente, por lo que para que una app esté disponible en todas las plataformas, se deberán de crear varias apps con el lenguaje del sistema operativo seleccionado.

- **Objective-C** es el lenguaje que se utilizado para las aplicaciones desarrolladas en entornos **IOS**. Objective-C es un lenguaje orientado a objetos que se implementa como suplerclase de C.

- **Java** se utiliza para desarrollar las aplicaciones en entornos **Android. Java** es probablemente el lenguaje orientado a objetos más extendido y utilizado por los desarrolladores. Necesita la máquina virtual de Java para ejecutarse aunque como veremos a lo largo del transcurso de la memoria, Android dispone de su propia máquina virtual denominada run time,el runetime de Android.

- El lenguaje **.Net** es el lenguaje elegido por **Windows** para el desarrollo de sus aplicaciones. .Net es el lenguaje utilizado por Windows para el desarrollo de la mayoría de sus aplicaciones, así pues hay varias extensiones del lenguaje como el propio Visual Basic de Microsoft o Visual C++ que implementan métodos compatibles para el acceso a la plataforma .Net

Cuando hablamos de desarrollo móvil, casi siempre nos estamos refiriendo a aplicaciones nativas. La principal ventaja con respecto a los otros dos tipos, es la posibilidad de acceder a todas las características del hardware del móvil: cámara, GPS, agenda, dispositivos de almacenamiento y otras muchas. Esto hace que la experiencia del usuario sea mucho más positiva que con otro tipo de apps.

Las aplicaciones nativas, no necesitan necesariamente de internet para su funcionamiento, aunque si pueden necesitarlo parcialmente para algún requisito del diseño que el desarrollador deba implementar.
La comercialización e instalación de estas aplicaciones, normalmente se realiza a través de las tiendas diseñadas por los proveedores de los sistemas operativos, aunque hay ocasiones en las que como en el caso de Android, se puede generar una archivo con

extensión .apk e instalarlo directamente sobre el dispositivo. La Tabla 3 muestra las principales ventajas e inconvenientes de las App.

Ventajas	**Inconvenientes**
Acceso completo al dispositivo	Diferente desarrollo para cada plataforma
Mejor experiencia del usuario	Más caras de desarrollar
Visibilidad en APP Store	Código de cliente no reutilizable entre plataformas
Envío de notificaciones o avisos a usuarios	
La actualización de la app es constante	

Tabla 3. Ventajas e inconvenientes aplicaciones nativas

2.2.2 Aplicaciones Web

Las aplicaciones web son aplicaciones elaboradas en un lenguaje de programación orientado a la web, como pueden ser HTML, Javascript,CSS etc. Estas aplicaciones se ejecutan a través de una URL y son independientes del SO en el que se desee visualizar la misma.

Esto implica que a diferencia de las aplicaciones nativas, donde se debe desarrollar una aplicación para cada sistema operativo, para una aplicación web solo sería necesaria una única aplicación que sería común para todas las plataformas. Esta aplicación se podría visualizar con el navegador web usado para cada sistema operativo, por ejemplo Google Chrome para Android o Safari para IOS.

Las desventajas que presenta a efectos de programación es que la aplicación web, al ejecutarse en un servidor externo en vez de en el mismo terminal, no puede acceder o puede acceder de forma muy limitada al hardware del equipo. Otra gran desventaja de este tipo de aplicaciones, es que necesitan que el usuario esté conectado a internet para poder ejecutarlas

Una aplicación web no se distribuye por ningún market si no que se encuentra siempre accesible para el usuario en la URL que apunte hacia el servidor donde se encuentra alojada.

Estas aplicaciones son ideales para hacer la réplica de páginas web para adaptar a los terminales móviles, o para aplicaciones relacionadas con el almacenamiento en la nube o icloud, ya que de por si estas herramientas ya necesitan una acceso a internet para su utilización. La Tabla 4 resume las ventajas y desventajas que presentan las aplicaciones de tipo web.

Ventajas	Inconvenientes
Mismo código para todas las plataformas	Requiere de conexión a internet
Desarrollo más sencillo	Acceso muy limitado al hardware
No necesitan aprobación externa para publicarse	Interacción con la aplicación más costosa para el usuario
El usuario siempre tiene la última versión	Requiere más esfuerzo de promoción
Pueden reutilizarse sitios ya diseñados	

Tabla 4. Ventajas e inconvenientes del desarrollo de aplicaciones Web

2.2.3 Aplicaciones híbridas

Las aplicaciones hibridas son aplicaciones desarrolladas con frameworks, y como su propio nombre indica, son una combinación de las características de los dos tipos de aplicaciones a la vez.

Estas aplicaciones están desarrolladas en los mismos lenguajes de programación que las de las aplicaciones web, pero a la vez el Framework agrupa los códigos de tal manera que estas se comporten como aplicaciones nativas en los diferentes SO y se puedan distribuir a través de los Markets.

Esta forma de implementación, permite a la aplicación tener una acceso más completo a las funciones hardware de los dispositivos móviles. Los principales frameworks utilizados para la realización de este tipo de aplicaciones son:

- **PhoneGap:** Es un framework de código abierto para el desarrollo de aplicaciones móviles hibridas, producido por Nitobi y comprado posteriormente por Adobe Sistems.
- **Apache Cordova:** Como en el caso de PhoneGap también es un framework destinado a la realización de aplicaciones hibridas y desarrollado por Adobe Sistems.

La Tabla 5 resume las principales ventajas y desventajas de estas aplicaciones.

Ventajas	Inconvenientes
Posibilidad distribución App Market	Experiencia de usuario más cercana a web que a app
Instalación nativa pero construida con HTML,CSS,Java Script…	Diseño visual no siempre relacionado con el sistema operativo en que se muestra
Mismo código base para múltiples plataformas	
Acceso a parte del hardware del dispositivo	
Pueden reutilizarse sitios ya diseñados	

Tabla 5. Ventajas e inconvenientes desarrollo aplicaciones hibridas

¿Cuál es la mejor elección?

Una vez explicados los tipos de aplicaciones existentes según la forma de desarrollarla, vamos a proceder a explicar que factores determinantes debe tener en cuenta un desarrollador a la hora de decantarse por un tipo de desarrollo u otro. Los factores más determinantes a tener en cuenta son los siguientes:

- **Limitación de presupuesto para desarrollar:** Al plantearse desarrollar una aplicación, el desarrollador siempre parte de un presupuesto inicial. En caso de tener un presupuesto muy limitado, la opción a tener más en cuenta debería ser la opción web, ya que solo necesita desarrollar una vez el código y no necesita tener una licencia para poder poner en servicio la aplicación. Para aplicaciones nativas podría tener que pagar unos costes asociados como es en el caso de los desarrolladores de IOS, y también aumentarían los costes al aumentar el tiempo empleado para el desarrollo al tener que desarrollar distinto código para varias plataformas.

- **Necesidad de usar hardware del dispositivo:** Si se necesita tener acceso a las funciones hardware de los dispositivos, la mejor opción sería decantarse por el desarrollo de una aplicación nativa, ya que estas tienen un acceso mucho más completo al API del sistema operativo

- Público al que va dirigida la app: El usuario al que va dirigido también es un factor determinante. No es lo mismo desarrollar una aplicación para personas que estén constantemente en contacto con internet y que vayan a requerir un servicio exclusivo de consulta, que a personas que necesitan tener acceso a las funcionalidades estando fuera de línea.

- **Complejidad del diseño gráfico:** Si el diseño gráfico es muy complejo, una aplicación web podría suponer un problema, ya que este tipo de aplicaciones poseen más restricciones a la hora de implementar la interfaz

- **Requerimiento de notificaciones al usuario:** Si el desarrollador necesita gestionar notificaciones hacia el usuario, habría que decantarse por una aplicación nativa o hibrida, ya que las aplicaciones web, al no estar instaladas en el terminal del usuario, no dispondrían de acceso a esta funcionalidad.

- **Actualizaciones futuras de la app:** La forma más sencilla de implementar las actualizaciones es una aplicación web, ya que solo haría falta actualizar el sitio web y los usuarios cuando accediesen ya verían el contenido actualizado , sin necesidad de instalar o descargar software como sería el caso de las aplicaciones hibridas y nativas

Para el caso que aplica a este proyecto, será necesario usar el hardware del teléfono para poder usar el sintetizador de texto a voz y el reconocedor de voz, con lo cual se ha optado por desarrollar una aplicación nativa.

En el siguiente punto se describe las características de los sistemas operativos existentes para smartphones y se justifica la elección del sistema operativo Android para desarrollar la aplicación de logopedia, objeto de este proyecto.

2.3 Sistemas operativos móviles

Un **sistema operativo móvil** o **SO móvil** es un sistema operativo que controla un dispositivo móvil, igual que los ordenadores utilizan Windows, Linux o Mac OS entre otros.

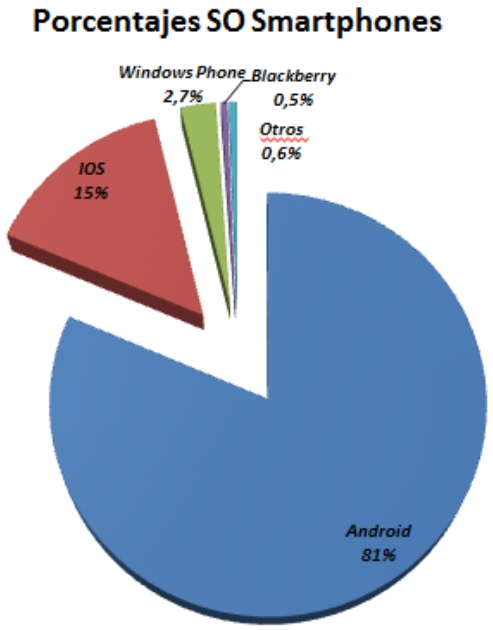

Figura 4. Porcentajes de utilización de los diferentes SO en smartphones

Sin embargo, los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos.

Los Sistemas Operativos para teléfonos móviles son cada día más importantes, pues la tecnología avanza y en materia de comunicaciones aún más. La telefonía móvil cada vez se convierte más en una parte importante de nuestras vidas, y en una sociedad que exige más y más, es importante diseñar sistemas que soporten las aplicaciones que se demandan, que sean fluidos, fáciles, accesibles y hasta divertidos.

Es por eso que las compañías móviles han desarrollado una competencia bastante reñida en cuanto al desarrollo de SO se refiere, desde los inicios en los años 90 con las versiones de EPOC32 para PDAs hasta los más avanzados y sofisticados como Android, IOS, Windows Phone, que además de ser eficientes y estables son multiplataforma, lo

que hace que cualquier persona tenga acceso a ellos desde un teléfono móvil básico hasta un Smartphone.

La proliferación de desarrolladores externos, ha contribuido en gran medida al desarrollo y mejora de estos sistemas operativos, ya no solo por la mejora del mismo Kernel en proyectos de desarrollo de código abierto, si no poniendo de manifiesto las carencias y limitaciones que pueden presentar los sistemas operativos para cubrir las necesidades de los usuarios.

2.3.1 Elementos de un sistema operativo móvil

Capas Como en los sistemas operativos tradicionales, los sistemas operativos móviles también están formados por capas:

Figura 5. Capas de un SO móvil

Kernel

Una de las más importantes y esenciales piezas que componen cualquier sistema operativo, sea el de nuestro dispositivo móvil, o el del PC, es el denominado núcleo o Kernel, el cual es la capa de software que permite el acceso a los diferentes elementos de hardware que conforman el dispositivo.

También es el encargado de brindar diferentes servicios a las capas superiores como los controladores de hardware, gestión de procesos, sistemas de archivos, además del acceso y administración de la memoria del sistema.

Los kernel de los sistemas operativos suelen estar basados en Unix, en el caso de Android está basado en Linux y en el caso de IOS está basado en el SO de Mac, el cual tiene un núcleo Darwin muy parecido a los Linux. Al ser kernels destinados a dispositivos móviles, han sido reducidos y optimizados al máximo para que cumplan con los requerimientos de funcionalidades de estos dispositivos, ocupando el mínimo espacio. En el caso de Android y el de RIM, ambos sistemas operativos presentan la particularidad de contar con un motor Java en el desarrollo de sus núcleos

Middleware

Esta capa es el conjunto de módulos que permite que las aplicaciones diseñadas y escritas para tales plataformas, puedan ser ejecutadas. Su funcionamiento es totalmente trasparente para el usuario, no debiendo realizar ni configurar ninguna acción para su correcto desenvolvimiento. El Middleware actúa como soporte para servicios de la jerarquía superior permitiendo a estos ejecutarse

Entre los servicios que presta esta capa podemos citar los motores de comunicaciones y mensajería, funciones de seguridad, servicios para la gestión de diferentes aspectos del dispositivo, servicios claves como el motor de mensajera y comunicaciones, codecs multimedia, intérpretes de páginas Web y servicios WAP, además de soporte para una gran variedad de servicios concernientes al apartado multimedia que es capaz de ejecutar el dispositivo. Estos servicios, como mencionamos, son vitales para el normal funcionamiento de la estructura del sistema operativo de nuestro dispositivo.

Entorno de ejecución de aplicaciones

Esta capa provee de todos los elementos necesarios para la creación y desarrollo de software a los programadores. Proporciona los APIs de programación para el desarrollo de aplicaciones nativas, además de servir de soporte a estas aplicaciones para funciones como servicios de localización, telefonía, gestión de aplicaciones y otros muchos.

Interfaz de usuario

En esta capa se presenta al usuario la interfaz gráfica que utilizan tanto las aplicaciones nativas como las interfaces de las aplicaciones propias del sistema operativo Esta interfaz facilita el acceso elementos gráficos que harán posible el uso cómodo y sencillo del móvil: botones, menús, pantallas y listas, entre otros.

En esta capa se alojarán las aplicaciones creadas por los desarrolladores, que recurrirán a soportes de la capa inferior para realizar las funciones antes mencionadas en el apartado capa de aplicaciones.

Una de las particularidades más importantes incorporadas a la interfaz gráfica de usuario en los últimos años ha sido, sin lugar a dudas, la posibilidad de utilizar todas las funciones del teléfono mediante el uso de los dedos de la mano, desplazando a los botones a un segundo plano en cuanto al manejo de opciones y funciones en el móvil.
Esta tendencia se convertirá paulatinamente en un estándar de la industria.

También una gran ventaja son los teclados del tipo virtual, que eventualmente reemplazarán a los tradicionales dispositivos de entrada, permitiendo de esta manera un mejor y más liviano diseño del dispositivo móvil. Otro punto muy interesante a tener en cuenta es la capacidad de personalización que permite la interfaz del usuario de nuestro sistema operativo.

2.3.2 Características de los principales sistemas operativos

IOS

IOS[6] es el sistema operativo con el que cuentan todos los dispositivos móviles desarrollados por la compañía **Apple.** Como ya se ha mencionado en este capítulo, IOS es una versión reducida del sistema operativo MAC OS, el cual tiene un núcleo Darwin muy similar al de Linux. IOS ha ido evolucionando con el paso del tiempo, adecuando sus características a los requerimientos de los usuarios y a la evolución del mercado de los dispositivos móviles. A continuación vamos a analizar la evolución cronológica de las distintas versiones de IOS:

2007: En este año Apple lanza al mercado su primer smartPhone el iPhone.Este dispositivo revolucionaría para siempre el concepto de telefonía móvil, ya que implementaba aplicaciones propias de los PCs La primera versión del software fue denominada por Apple como **iPhone OS 1**. Esta primera versión se caracterizaba por su pantalla táctil y una parrilla de iconos que se pudiese manipular con los dedos, ya que según la filosofía de la compañía, el puntero más útil con la que cuenta el ser humano son los propios dedos de la mano.

Esta parrilla fue denominada Springboard, la cual disponía de unos accesos directos a las aplicaciones que el usuario empleaba con más asiduidad, estos accesos formaban parte de una barra fue denominada Dock. En cuanto a las aplicaciones que contenía el sistema operativo, cabe indicar que eran bastante limitadas ya que la primera versión de IOS solo permitía utilizar las apps que venían preinstaladas en dicho SO. Entre las aplicaciones de las que ya disponía este sistema operativo, se encontraba el navegador Safari, que pasaría a estar disponible en el resto de versiones implementadas por Apple. Otras aplicaciones destacables con las que contaba el iPhone eran Google Maps y Youtube.

2008: Con el lanzamiento al mercado su nuevo smarthphone, el Iphone 3G, se desarrolló la nueva versión de su SO el **IOS 2.**Esta nueva versión permitía a desarrolladores externos a la compañía desarrollar aplicaciones para Iphone y compartirlas con el resto de usuarios, ya fuese de forma gratuita o remunerada. Para conseguir que los desarrolladores externos a Apple pudiesen compartir dichas apps, crearon su Market, denominado iTunes App Store.

Este nuevo modelo de negocio abrió un nuevo mercado en el que Apple se beneficiaba de tener a millones de programadores potenciales en todo el mundo mejorando su producto de forma gratuita, a la vez que dichos programadores se beneficiaban de tener acceso a los clientes de los que ya disponía Apple. En esta versión se incluyó a su vez un sistema de visualización de los correos Microsoft.

2009: La tercera versión de Iphone (**IOS3**), introdujo elementos destinados a facilitar el manejo cotidiano del dispositivo por parte del usuario, introduciendo nuevos elementos de búsqueda por voz dentro de las aplicaciones o facilitar accesos directos a dichas aplicaciones con comandos de voz. Otras actualizaciones de esta tercera versión de IOS, en concreto la versión 3.2 , surgió tras el lanzamiento al mercado del IPad, su primera Tablet, la cual como en el caso del iphone abrió un nuevo mercado que luego seguirían

el resto de compañías. Además en esta versión se incluyeron las opciones de copiar y pegar texto, así como un servicio de notificaciones de entrada de correo electrónico.

2010: Apple lanzó en Iphone4, su nuevo Smartphone. Este nuevo terminal usaba el sistema operativo **IOS4**, el cual supuso un gran salto con respecto al anterior, gracias en gran parte a que introduce el concepto de multitarea, en el cual el usuario puede estar ejecutando varias tareas en paralelo e ir saltando de una en otra según sus necesidades. Para esta versión también se añadieron nuevas librerías al API de programación que se ponía a disposición de los desarrolladores externos, haciendo proliferar así nuevas y variadas aplicaciones. En esta versión Apple también añadió un centro de juegos y una red social para que el usuario agregase a otros usuarios y compitiesen entre si

2011: Apple presento el nuevo Iphone4S, que poseía el **IOS5** como nuevo sistema operativo. Las primera evolución del sistema operativo que se introdujo fue una barra desplegable para acceso a notificaciones, idea que se importó del SO Android que ya utilizaba este sistema con anterioridad. En el apartado de la interacción por voz, la principal aportación de Apple para esta nueva versión de su sistema operativo fue **SIRI**. SIRI es un mecanismo que permite interactuar al usuario con el Smartphone con un modo de comunicación mucho más natural e inteligente.

2012: La llegada del sistema operativo I**OS6** puso de manifiesto la necesidad de Apple de independizarse de las aplicaciones de Google. Así pues llego a un acuerdo con TomTom para desarrollar un sistema de mapas y geolocalización y retiró google maps de su parrilla de aplicaciones. Se añadió además una integración de la plataforma Facebook en su sistema operativo.

2013: IOS 7 lanzado en 2013, no supuso un gran cambio en la parte operativa sino más bien se enfocó a hacer un nuevo rediseño a la interfaz gráfica que le proporcionase un aspecto más moderno y atractivo, aunque si incluye alguna aportación como un sistema biométrico de acceso al terminal mediante huella. También se incluyó en esta versión iTunes radio, un servicio de radio streaming que pretende competir con otros como existentes como Spotify

2014: **iOS 8**, y los modelos iPhone 6 y iPhone 6 Plus, llegamos al sistema operativo vigente desde septiembre de 2014 hasta hoy. De este último podemos destacar la funcionalidad que han denominado Handoff, consistente en la continuidad de acciones entre dispositivos iPhone, iPad y Mac, gracias a la cuál puede por ejemplo comenzar a redactarse un correo desde cualquiera de ellos para continuar más tarde desde otro, recuperando el punto exacto de la redacción dónde se había dejado.

De igual manera, es factible recibir y contestar una llamada telefónica, o redactar un mensaje de texto desde un Mac. El único requisito para utilizar esta interoperabilidad es que todos los dispositivos estén registrados con la misma cuenta de iCloud, que es el sistema de almacenamiento en la nube de Apple.

ANDROID

Android es un sistema operativo desarrollado por Google y otras empresas basado en Linux y utilizando para desarrollarlo la misma filosofía de Open Source, que se utiliza para el desarrollo de este. Android es utilizado por un gran número de fabricantes de

dispositivos móviles para gestionar sus necesidades operativas. Estos fabricantes suelen personalizar el sistema operativo, pero existe una versión genérica denominada Android Open Souce Proyect sin añadidos de los fabricantes.

La primera versión de Android fue una versión beta que se lanzó en 2007, aunque la primera versión puesta en servicio en un dispositivo comercial no surgió hasta 2008.
A continuación se va a proceder a repasar las distintas versiones por las que ha ido pasando el SO a través de su evolución desde su creación [7].

Esta primera versión del SO Android, fue la versión conocida como **Apple Pie**. Este SO presentaba algunas novedades con respecto a los sistemas operativos presentados hasta la fecha por Apple y suponían una mejora en la experiencia en la experiencia de usuario. Tal es así que Apple fue introduciendo algunas de estas características en posteriores actualizaciones de software. Incluía también aplicaciones relacionadas con la compañía Google, de gran demanda entre los usuarios como pueda ser el buscador google o el software de geolocalización de Google Maps.

Otra gran aportación fue la creación del Android Market, el cual permitía a desarrolladores de todo el mundo crear nuevas aplicaciones y distribuirlas entre los usuarios Android.

Posteriormente, hacia **2009**, Android lanzó una actualización de su primera versión llamada **Banana Bread**, en la que mejoraban algunas de sus funcionalidades como la mensajería multimedia y la pantalla en modos manos libres, pero por el contrario no suponía un gran cambio en la interfaz de usuario.

Esta no interfaz no sufrió grandes cambios hasta **Abril de 2009** cuando Android lanzó la versión **Cupcake**, donde se incluyó la posibilidad para el usuario de utilizar widgets, así como transiciones entre pantallas más atractivas visualmente.

A nivel de funcionalidades esta versión incorporó un elemento fundamental para el desarrollo de este PFC, el API de reconocimiento de voz. Pero no fue hasta meses más tarde cuando se incorporó el API de **text to speech** en la versión **1.6 llamada Donut** y la cual también tiene gran importancia en el desarrollo de este PFC.

En noviembre de **2009** salió al mercado la versión **Eclair**, la cual sorprendió por su integración y facilidad para el manejo de redes sociales. Pero esta actualización no solo quedó en eso, sino que además trajo multitud de aplicaciones para la cámara u otras mejoras como el teclado multitouch.

Ante la salida al mercado de IOS4, Android lanzó la versión **Froyo** para competir con esta. Froyo aportaba la posibilidad de crear una red WiFi a partir de la conexión de datos 3G del teléfono. Además incluía el motor Java V8 que ofrecía a los usuarios un aumento de velocidad gracias a su compilador. Otra gran aportación de Froyo fue el soporte Adobe flash tanto para contenido web como contenido multimedia que los usuarios habían estado esperando mucho tiempo.

El diciembre de **2010** Android saco la versión **GingerBread**. Esta versión traía consigo varias novedades importantes, como soporte para el protocolo SIP de telefonía VoIP, soporte para la tecnología inalámbrica NFC, nuevos sensores como giroscopio y

barómetro, nuevo gestor de descargas optimizado para transferencia de archivos grandes, soporte multicámara y nuevos codecs multimedia como WebM/VP8 y AAC, mejoras en la precisión del teclado y entrada por voz. Cuenta además con un diseño de interfaz más simple y eficiente en cuanto a consumo de recursos.

Las versiones **Honeycomb** de Android surgen en **febrero de 2011** con el propósito lanzar una versión optimizada de Android para tablets, por lo que no está disponible para ningún teléfono móvil. Incluye un teclado optimizado para pantallas grandes, aplicaciones de correo, contactos y navegador adaptados, barra de notificaciones en la parte inferior de la pantalla, nueva interfaz para copiar-pegar texto, ajuste de tamaño de widgets y una serie de mejoras en la interfaz para hacerla más intuitiva.

La versión Ice **Cream Sandwich** de Android en octubre de **2011** se incorpora el kernel 3.0.1 de Linux. Constituye una versión fundamental al introducir compatibilidad con tabletas y teléfonos inteligentes, que será conservada en todas las versiones posteriores hasta la fecha.

Incorporó otras características como capturas de pantalla, acceso a aplicaciones desde la propia pantalla de bloqueo, versión del navegador Google Chrome para Android con sincronización de marcadores y un límite de quince pestañas simultáneas, creación de carpetas mediante arrastrar y soltar, dictado de voz mejorado, rediseño de la galería de imágenes, así como las habituales mejoras de rendimiento y estabilidad.

Jelly Bean es presentado el **9 de julio de 2012** con el propósito de superar los problemas de rendimiento y batería de su versión predecesora. Google adoptará una serie de medidas dentro del que denominó "Proyecto Butter" para conseguir una experiencia de usuario suave y fluida.

En esta versión se inaugura, como parte de Google Search, el asistente personal inteligente que recopila nuestros hábitos, gustos y búsquedas recientes para presentarnos información de forma anticipada en forma de tarjetas, denominado Google Now. Integra además una nueva versión de Voice Search con el añadido de "OK Google", función por la que se pueden realizar búsquedas de voz sin pulsar si quiera el icono del micrófono, bastan con pronunciar esas dos palabras para que el sistema interprete las que precedan como los términos o indicaciones de búsqueda.

Otras funciones destacables son la impresión inalámbrica, nueva entorno de ejecución denominado ART, una evolución experimental de Dalvik, y soporte con el dispositivo Google Chromecast

El **31 de octubre de 2013** se presenta la versión **KitKat**, con importantes características como mayor rendimiento del sistema incluso para dispositivos con recursos limitados, mejora de la compatibilidad de ART y de la multitarea, uso de Chrome en las vistas web de las aplicaciones, evolución de la aplicación de descargas y correo electrónico, modo inmersivo (se ocultan las barras de estado y navegación) en aplicaciones stock, así como nuevos efectos visuales.

La versión más reciente de Android es la versión **LolliPop**. La interfaz gráfica ha sido rediseñada para conseguir un aspecto más sencillo y plano. El nuevo patrón de diseño

no sólo se limita al sistema y las aplicaciones preinstaladas, sino que Google pretende sea el nuevo patrón de diseño para todos los desarrolladores.

Otra novedad es el renovado sistema de notificaciones con una nueva plantilla que permite hasta seis elementos de control de notificaciones con funciones como responder mensajes desde la pantalla de bloqueo, clasificación inteligente, etc. Se obtienen también importantes las mejoras de rendimiento gracias al entorno de ejecución ART que ahora es el utilizado por defecto, soporte para plataformas de 64 bits, así como un sistema de gestión de la batería más inteligente que promete extender hasta 90 minutos la autonomía y las frecuentes mejoras de estabilidad.

El 9 de marzo de 2015 Google lanzó la versión 5.1 con mejoras de rendimiento, soporte multi-sim y sistema de protección de terminales permitiendo bloquearlos en caso de robo o extravío. Como último apunte sobre Android, existe una marcada fragmentación de versiones en activo.

WINDOWS PHONE

Windows Phone (WP) es el sistema operativo móvil desarrollado por Microsoft, anteriormente llamado Windows Mobile. Su núcleo se basa en Windows CE 6.0 en las versiones tempranas para posteriormente evolucionar a un núcleo Windows NT.

La novedosa interfaz diseñada por Microsoft, denominada Modern UI (anteriormente Metro), se basa en una pantalla de inicio compuesta de unos "azulejos" geométricos de colores básicos denominados Live Tiles. En esencia, son iconos anclados en la pantalla principal que representan enlaces a aplicaciones, música, contactos, el tiempo, etc. Poseen además la capacidad de representar información en tiempo real con animaciones.

Destaca el uso de Bing como motor de búsqueda, en un primer momento reemplazable por Google pero suprimida la posibilidad en versiones posteriores, e Internet Explorer como navegador predefinido.

La primera versión, denominada **WP 7.0**, ve la luz en **octubre de 2010** con déficits a nivel de rendimiento y funcionalidad del sistema. Pasados unos meses **(marzo de 2011)** se realiza el lanzamiento de **WP 7.1**, con importantes mejoras de rendimiento y optimización del tiempo de arranque. Las siguientes versiones denominadas WP 7.5.0 y WP 7.5.1 añaden multitud de mejoras, algunas tan importantes como la multitarea y una rebaja de los requisitos mínimos de hardware necesarios para instalar WP (a petición de Nokia). La versión 7.8, que sería la última con esta numeración, supone una transición hacia una interfaz de usuario similar a la que tendría la versión 8.

Windows Phone 8.0 supone una evolución en el núcleo del sistema, comenzando a utilizar Windows NT. Desde finales de 2012, pasará por tres subversiones hasta llegar a la 8.0.3, añadiendo gradualmente en cada una de ellas nuevas mejoras, como NFC, Internet Explorer 10, Skype, WiFi en suspensión, radio FM, optimización de HTML5, modo conducción, soporte para procesadores de cuatro núcleos, etc.

En **abril de 2014** se produce el lanzamiento de **WP 8.1**, versión que perdura hasta la fecha de realización de este proyecto. Supone un cambio de núcleo del sistema pasando a utilizar Windows. En ella, se incluyen una serie de mejoras para intentar rivalizar con sus inmediatos competidores, Android e iOS, los cuales siguen ocupando el primer y segundo puesto respectivamente en cuanto a cuota de mercado. A continuación se muestran las más importantes:

- Cortana.

Es el asistente virtual que debuta con esta versión para competir con los asistentes Siri y Google Now. El usuario puede interactuar con Cortana utilizando lenguaje natural, sin necesidad de pronunciar comandos concretos, gracias al uso de los servicios proporcionados por el buscador Bing. A los ya habituales usos de gestión de llamadas y alarmas, búsqueda de información en la web, etc., se une una característica novedosa denominada Cuaderno de Cortana. Esta función recopila información personal del usuario, como sus intereses, gustos y acciones que suele solicitar, permitiendo editar o eliminar cada una de las entradas. Con el paso del tiempo, escapa de combinar la información guardada en el Cuaderno con otra relevante para poder realizar sugerencias proactivas.

- Centro de notificaciones.

Desde esta versión es posible, al estilo Android o iOS, deslizar desde la parte superior de la pantalla para acceder al centro de notificaciones. Desde él se pueden consultar los eventos susceptibles de notificación o acceder a opciones de configuración rápida como habilitar/deshabilitar el WiFi o Bluetooth.

-Personalización pantalla de inicio.

Permite la elección de fondo de pantalla y una tercera columna de iconos o Live Tiles.

- Sensores WiFi, datos, almacenamiento y batería.

Se tratan de aplicaciones destinadas a la monitorización y optimización de los recursos.

- Nuevas aplicaciones preinstaladas.

Comida y Bebida, Viajes y Mapas, Salud y Ejercicios, y Mapa, que incluye una licencia gratuita con todos los mapas mundiales gracias al acuerdo con HERE Drive+. La distribución de aplicaciones se realiza a través de su tienda de nominada Windows Phone Store.

- Word Flow.

Es el nuevo teclado por defecto para WP cuya principal novedad es la posibilidad de escribir deslizando el dedo por la pantalla seleccionando las letras que conforman la palabra a obtener.

La distribución de aplicaciones para terminales con Windows Phone se realiza a través de la Store. Esta tienda está gestionada por Microsoft, quien se encarga de examinar y

validar el contenido adecuado de las aplicaciones, juegos, vídeo y música subidos. En enero de 2015 dispone de más de 560.000 aplicaciones, con un crecimiento mensual cercano a 15.000 aplicaciones nuevas. En cuanto a las actualizaciones de software, se sirven de Microsoft Update al igual que el resto de sistemas Windows.

De cara al desarrollador, el acceso a los entornos de desarrollo necesarios para realizar aplicaciones compatibles es gratuito, sin embargo, la publicación en el Store tiene un coste asociado de 19 dólares al año para desarrolladores individuales o bien 99 dólares anuales para compañías.

2.3.3 Justificación elección sistema operativo

Tras realizar un estudio de la estructura de un SO, y de la evolución de los principales SO existentes en el mercados, se van a exponer los distintos factores que han hecho elegir Android como sistema operativo.

- **Software Libre**

El concepto de software libre implementado por Android para la realización de su API y sistemas operativos, hace que sea mucho menos complejo el desarrollo de aplicaciones en este entorno, teniendo un mayor control de todo el hardware de los dispositivos móviles.

- **Índice de penetración del sistema operativo entre usuarios de dispositivos móviles**

De acuerdo a un informe publicado por la consultora IDC, Android y iOS cuentan con un impresionante 96.3% de toda la distribución de smartphones en el 2014, confirmando que Google y Apple tienen un aplastante dominio. Este porcentaje representa un incremento en relación al año anterior que hubo un 93.8% de dominio de ambos sistemas operativos.

Android es líder por un amplio margen al obtener un 81.5% del mercado en 2014, superando las mil millones de unidades distribuidas en el año que sin duda es un logro sorprendente. Estos resultados fueron un incremento importante ya que en 2013 tuvieron un 78.7% y 802 millones de unidades distribuidas.

Por su parte, iOS logró un 14.8% del mercado en 2014, que aunque es poco en comparación a Android sigue siendo un gran porcentaje.

El resto de sistemas operativos móviles se reparten solamente el 3.7% de mercado, con Windows Phone logrando un 2.7% y BlackBerry un 0.4%, este último con una dramática caída de -69.8% en la comparación anual.

- **Librerías sintetizador texto a voz y voz a texto de Android**

Otro punto fuerte para la elección de Android como el SO a usar, han sido las librerías que posee para realizar las funciones de sintetizador de texto a voz y sintetizador de voz a texto. A continuación en este capítulo de Estado del Arte, se indica en qué consisten estos procesos y cómo funcionan las librerías de Android dedicadas a esta finalidad.

2.4 Reconocimiento de voz y síntesis texto a voz

En este apartado se van a explicar los conceptos de reconocimiento de voz y síntesis de texto a voz, explicando cómo funcionan y las principales aplicaciones que tienen actualmente. Después de la explicación, se procederá a explicar el API que Android ha desarrollado para implementar estas herramientas.

2.4.1 Reconocimiento Automático del habla (RAH)

El reconocimiento automático del habla (RAH) o reconocimiento automático de voz son procesos que tienen como objetivo establecer una comunicación oral con el procesador.

Un sistema de reconocimiento de voz, es un algoritmo programado para procesar la señal de voz emitida por el ser humano y reconocer la información contenida en ésta, convirtiéndola en texto o emitiendo órdenes que actúan sobre un proceso. En su desarrollo intervienen diversas disciplinas, tales como: la fisiología, la acústica, la lingüística, el procesamiento de señales, la inteligencia artificial y la ciencia de la computación.

Los sistemas de reconocimiento de voz dependen de diversos factores que determinan el diseño final del mismo. Depende en gran medida del usuario final al que vaya dirigido ya que se implementará de manera distinta si debe aprender de la forma de hablar de un usuario en particular, o debe ser implementado para múltiples usuarios independiente de la forma en que se expresen.

Otro factor que influye en el diseño final del RAH, es la necesidad de si el sistema debe interpretar el habla de forma continuada o puede haber pausas entre palabra y palabra, lo cual simplifica el diseño. El tamaño del lenguaje a interpretar es un factor determinante a la hora de diseñar el sistema RAH, ya que un sistema de palabras reducidas es mucho más fácil de parametrizar que un idioma completo.

En la actualidad existen números usos para este tipo de sistema tanto a nivel industrial como a nivel privado. Este sistema es utilizado por sintetizadores de texto en el campo académico, contestadores automáticos de atención al cliente, control por comandos de sistemas móviles etc.

La implementación de un sistema de reconocimiento automático del habla, conlleva el tratamiento de la señal de voz, pasando por un sistema formado por diferentes bloques, los cuales extraen los parámetros fundamentales de la señal, para posteriormente interpretarlos. El proceso puede resumirse en los siguientes pasos:

1. El primer paso consiste en captar la señal de voz y extraer sus parámetros fundamentales, para ello se utiliza un micrófono con el cual se captaría la señal de voz, para posteriormente pasar la información eléctrica obtenida por el micrófono a un convertidor analógico/digital para poder obtener los parámetros fundamentales de la señal. Este proceso es crítico ya que se puede producir una pérdida irreparable de la información de la señal que posteriormente pueda desencadenar errores de interpretación.
2. La siguiente etapa es la segmentación y el etiquetado, aquí el sistema intenta encontrar las regiones estables donde las características son constantes. Uno de las técnicas más utilizadas es la utilización de solapamiento entre el enventanado, para evitar dejar parte de señal sin analizar. En este nivel además se suelen aplicar filtros de normalización y pre-énfasis, con los cuales se prepara la señal para ser procesada.
3. En tercer lugar se realiza el cálculo de parámetros, lo que proporciona una representación espectral de las características de la señal de voz que podemos utilizar para entrenar el sistema de reconocimiento (HMM, Redes neuronales, entre otros). Los métodos más comunes en esta etapa son el análisis de banco de filtros y LPC. Para el cálculo de coeficientes que caracterizan la señal se sigue un patrón de bloques estandarizado por la ETSI.

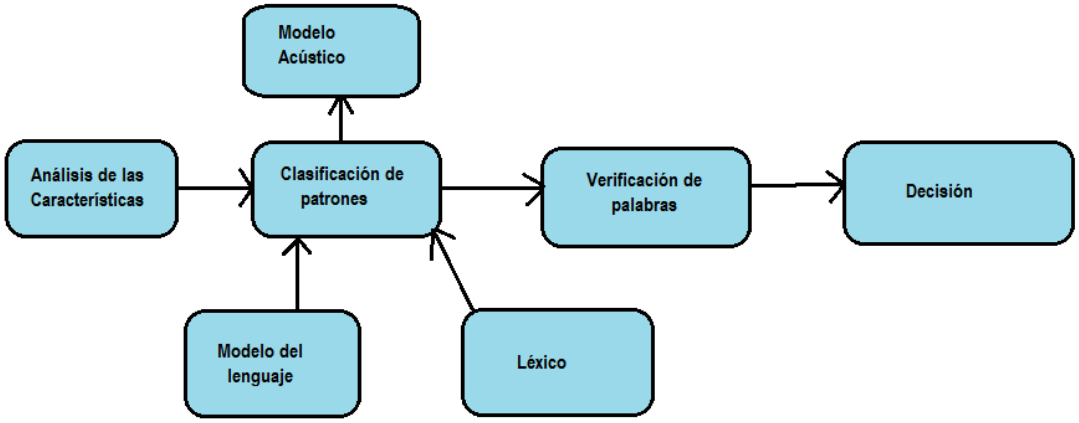

Figura 6. Esquema bloques reconocedor de voz

Una vez el sistema ha procesado toda la señal, se ha pasado al dominio de la frecuencia y ha comparado patrones con los parámetros que tiene en su base de datos, toma una decisión basada en la opción más probable que se haya dicho y la devuelve como salida del sistema.

Paquete SpeechRecognizer Android

Speech reconigzer es el API que Android ha desarrollado para poder interactuar con su software de reconocimiento del habla. Dicha API establece una conexión de audio entre el dispositivo en el cual se está ejecutando el API con un servidor remoto, en el cual se encuentra alojado el reconocedor automático del habla de Google. Este servidor es donde se interpreta está señal de audio y se devuelve una transcripción en texto.

Al estar el reconocedor del habla en un servidor remoto, esta API no está diseñada para grandes conversaciones ya que consumiría mucha batería y ancho de banda en el dispositivo empleado.

Las funciones que se implementan en esta API vienen descritas en la Tabla 6 [8].

Metodo implementado en el API	Descripción
void cancel()	Cancela el reconocimiento del habla
static SpeechRecognizer createSpeechRecognizer (Context context, ComponentName serviceComponent)	Constructor de la clase
static SpeechRecognizer createSpeechRecognizer (Context context)	Constructor de la clase
void destroy()	Elimina el elemento de la clase
static boolean isRecognitionAvailable (Context context)	Comprueba si es posible hacer un reconocimiento del habla
void setRecognitionListener (RecognitionListener listener)	Selecciona el elemento en el que se guardará la escucha
void startListening (Intent recognizerIntent)	Inicia el intento de reconocimiento
void stopListening ()	detiene el intento de reconocimiento

Tabla 6. API SpeechRecognizer

Vamos a proceder a ilustrar un ejemplo de configuración de la clase de reconocimiento de voz que implementa Android, para ello debemos seguir los siguientes pasos:

- Importar los paquetes del API SpeechRecognizer con los siguientes comandos:

```
import android.speech.RecognizerIntent;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;
```

- En la definición de atributos de la clase debemos añadir el siguiente parámetro correspondiente al reconocedor de voz:

```
private static final int VOICE_RECOGNITION_REQUEST_CODE =
1234;
```

- Posteriormente se va a crear un método inicializarReconocimiento() que se pondrá en el método OnCreate() de la activity, para que este se ejecute cuando llamen a la activity y sé que el reconocedor listo para actuar. Habilitamos un botón para que se inicie el reconocimiento cuando lo pulsemos

```
private void inicializarReconocimiento(){
PackageManager pm = getPackageManager();
List<ResolveInfo>activities=pm.queryIntentActivities(new
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH), 0);
if (activities.size() != 0) {
escuchar.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
startVoiceRecognitionActivity();
}}
);}}
```

- Por último creamos un método que se implementara cuando hayan pulsado el botón iniciando el reconocimiento:

```
private void startVoiceRecognitionActivity() {
Intent intent = new
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "");
startActivityForResult(intent,
VOICE_RECOGNITION_REQUEST_CODE);}
```

2.4.2 Síntesis de texto a voz (TTS)

Muchos sistemas de laboratorio y dispositivos comerciales realizan la conversión automática de un texto a voz sintetizada. El progreso en este área ha sido posible debido a los avances en la teoría lingüística, en el modelo de caracterización acústica-fonética de los sonidos, en el modelado matemático de generar voz, en la programación estructurada y en el diseño hardware de los ordenadores. Los pasos seguidos en todo proceso de síntesis son: primero, un conjunto de módulos analiza el texto de entrada para determinar la estructura de la sentencia y la composición fonética de cada palabra y un segundo conjunto de módulos transforma esta representación lingüística abstracta en voz.

Los métodos para sintetizar voz han cambiado mucho a lo largo de los años, desde los primeros sistemas que utilizaban dispositivos eléctricos y mecánicos resonantes hasta los sintetizadores modernos que emplean ordenadores o circuitos digitales de diseño específico. En la actualidad, la clasificación más común de los sistemas de síntesis de voz es atendiendo a las reglas que se siguen para la reconstrucción de la voz, distinguiéndose los siguientes cuatro sistemas:

- **Sintetizadores articulatorios**.

En éstos se realiza una analogía entre parámetros relativos a los órganos articulatorios y sus movimientos con parámetros circuitales. Pueden proporcionar una calidad altísima, pero es muy difícil obtener y controlar parámetros para un sintetizador de este tipo.

- **Sintetizadores por formantes**.

Son una serie de filtros que modelan el tracto vocal, excitados por fuentes que simulan las cuerdas vocales. Gozan de gran difusión.

- **Sintetizadores derivados de las técnicas de predicción lineal (LPC).**

Son sintetizadores de análisis-síntesis, en los que los parámetros que controlan la función de transferencia del filtro que simula el tracto vocal son parámetros LPC.

- **Sintetizadores por concatenación de forma de onda.**

Concatenan unidades pregrabadas para generar nuevas frases, con lo que intentan aumentar la calidad de la señal generada minimizando el ruido de codificación. La complejidad es alta, pero la calidad obtenida es muy buena.

API TTS Android

El API desarrollado para desarrollar su sintetizador de texto a voz de Android, es la clase **TextToSpeech.** Este API permite a los desarrolladores convertir un texto en un sonido o un archivo de audio.
Las funciones que se implementan en este API vienen descritas en la Tabla 7 [9].

Metodo implementado en el API	Descripción
int addSpeech (String text, String filename)	Comprueba si es posible hacer un reconocimiento del habla
Set < Locale > getAvailableLanguages ()	Consultar al motor sobre idiomas disponibles
String getDefaultEngine ()	Obtiene el nombre del motor de síntesis por defecto
List < TextToSpeech.EngineInfo > getEngines ()	Obtiene una lista de todos los motores instalados
static int getMaxSpeechInputLength ()	Limite longitud de la cadena de entrada
Voice getVoice ()	Devuelve una instancia de voz que describe la voz que utiliza el TTS
Set < Voice > getVoices ()	Consultar al motor sobre el cojunto de voces disponible
int isLanguageAvailable (Locale loc)	Comprueba si un idoma está disponible en el motor
boolean isSpeaking ()	Comprueba si el motor está ocupado hablando
int playSilentUtterance (long durationInMs, int queueMode, String utteranceId)	Reproduce silencio la cantidad de tiempo indicada
int setAudioAttributes (AudioAttributes audioAttributes)	Modificar los atributos del audio
int setLanguage (Locale loc)	Establece el idioma indicado
int setPitch (float pitch)	Establece el tono de voz utilizado
int setSpeechRate (float speechRate)	Establece la velocidad de la voz
int setVoice (Voice voice)	Establece la voz
void shutdown ()	Libera los recursos utilizados por le motor
int speak (CharSequence text, int queueMode, Bundle params, String utteranceId)	El motor habla por el altavoz el texto indicado
int stop ()	El motor inturrumpe lo que esta diciendo y descarta lo que tiene en cola
int synthesizeToFile (CharSequence text, Bundle params, File file, String utteranceId)	Sintetiza el texto propuesto a un archivo de audio

Tabla 7. API TextToSpeech

Esta herramienta no necesita de una conexión a internet para su ejecución ya que almacena toda la información necesaria para la síntesis en sus librerías La herramienta te permite seleccionar los parámetros de la voz. Como el idioma, el tipo de voz, el tono y la velocidad en que se expresará. Para utilizar las funciones se debe importar la librería en las clases java desde las que se vaya a ejecutar

Se va a proceder a ilustrar como se debería usar las funciones mostradas en este API para implementar el TTS correctamente en una clase Java englobada en un proyecto Android:

- Lo primero que debemos hacer es importar el paquete TTS y el OnInitListener de la clase, para ello usaremos los siguientes comandos antes de la definición de la clase:

```
import android.speech.tts.TextToSpeech;
import android.speech.tts.TextToSpeech.OnInitListener;
```

- En la definición de parámetros que vamos a utilizar en la clase debemos definir un objeto TTS con el siguiente comando:

```
private TextToSpeech tts;
```

- Posteriormente vamos a proceder a crear un método para inicializar el TTS que se debe añadir en el método OnCreate() de la clase a implementar, con esto conseguiremos que se inicialice el TTS cuando instancien la clase. El método sería el siguiente:

```
private void inicializar()
   {
tts = new TextToSpeech(this, (OnInitListener) this); // Creamos el
objeto tts tts.isLanguageAvailable(new Locale("spa"));// Comprobamos
que el tts está en español
  //setOnClickListener es cuando indicamos al botón que empiece a
estar atento a si le pulsan
//onClick es donde le decimos al botón cuando le han pulsado la acción
que debe realziar
decir.setOnClickListener(new OnClickListener(){
public void onClick(View v) {
speakOut(texto);//Llamamos a la función del API TTS que dice el
mensaje}});
```

- Para modificar los parámetros del TTS, solo hace falta invocar a los métodos mostrados en el API con el objeto de tipo TTS que hemos generado

2.5 Enfermedades del habla

Las primeras clasificaciones incluían bajo la denominación de trastornos del lenguaje únicamente las discapacidades referentes a la función motora de los órganos vocales, excluyendo así perturbaciones centrales, como la afasia. En otro extremo se sitúan los que incluyen bajo el concepto todas las discapacidades físicas y mentales que obstaculizan la comunicación verbal, incluyendo la esquizofrenia, la condición de sordo o hipoacúsico, lo paladar hendido o alteraciones en la lectura y en la escritura, como la dislexia y la disortografía.

Una posición más moderada consideraría como trastorno del lenguaje las perturbaciones referidas específicamente a la producción y a la recepción del habla, condiciones que excluyen las anomalías del lenguaje que son consecuencia de otros desórdenes, como las características de la esquizofrenia. También se excluirán los que son más propios de trastornos de la voz (disfonías y fenopatías), aunque pueden ser incluidas aquellas alteraciones que, a pesar de ser consecuencia de otros síndromes clínicos, pueden ser recuperables, o aquellos que son en parte consecuencia de trastornos propios del lenguaje.

Dentro de todas los trastornos del habla en este PFC vamos a centrarnos en la sintomatología y tratamiento de dos enfermedades concretas, la afasia y la dislexia, las cuales presentan distintos síntomas pero que presentan una parte del proceso de rehabilitación extrapolables al uso de las herramientas de reconocimiento de voz y síntesis de texto a voz como explicaremos a continuación.

2.5.1 Afasia

La afasia es un trastorno a consecuencia de una lesión a las partes del cerebro responsables del lenguaje, y puede causar problemas con cualquiera o todas las siguientes destrezas: la expresión, la comprensión, la lectura y la escritura.

Las lesiones al hemisferio izquierdo del cerebro causan afasia para la mayor parte de las personas diestras y alrededor de la mitad de las personas zurdas. Las personas que experimentan daños al hemisferio derecho del cerebro pueden tener otras dificultades además del habla y el lenguaje.

Algunas personas con afasia tienen problemas en el empleo de las palabras y las oraciones (afasia expresiva). Algunas tienen problemas en entender a los demás (afasia receptiva). Otras personas con afasia tienen problemas tanto de expresión como de comprensión (afasia global).

La afasia puede causar problemas con el lenguaje oral (expresión y comprensión) y con el lenguaje escrito (lectura y escritura). Por lo general existen mayores dificultades con la lectura y escritura que con la expresión y comprensión orales.

La afasia puede ser leve o grave. La gravedad de los problemas de comunicación depende de la cantidad y ubicación del daño cerebral.

Un paciente con afasia leve, puede mantener una conversación normal en muchas circunstancias, aunque puede pasar por problemas para comunicarse si sube el nivel de complejidad de la conversación y en ocasiones esto se expresa con el problema que a todos nos ha pasado alguna vez que es tener la sensación de que estas a punto de encontrar la palabra que quieres usar pero no llegamos a encontrarla. En los casos más graves de la afasia, el paciente no puede comunicarse ni entender nada y puede expresar solo algunas expresiones sencillas.

Proceso de rehabilitación

El proceso de rehabilitación para personas con problemas de afasia es un proceso largo y laborioso para el paciente y se divide en las siguientes fases:

1) Fase preliminar.

Esta fase trata de fijar la atención del paciente en los distintos sonidos y sirve también para determinar su grado de afectación. En esta fase no es conveniente intentar que el paciente intente expresarse.

Los ejercicios recomendados en esta fase son ejercicios relacionados con la reordenación de secuencias lógicas. Esta fase es una fase dura para el paciente ya que suele tener mermadas sus capacidades de comunicación y se debe realizar con la presencia de un profesional cualificado.

2) Fase dirigida a la rehabilitación del sistema motor del habla.

En esta segunda fase se tratará de hacer que el paciente trabaje y estimule el sistema fonador. Para ello es necesario que el paciente realice ejercicios físicos, tanto gestuales como tracto vocales, para ejercitar los músculos empleados en la producción de sonidos. Los ejercicios que se deben realizar en esta fase, son ejercicios de respiración, fonación y bucofaciales.

3) Fase de rehabilitación del lenguaje

Esta fase es en la que se centrará la aplicación de logopedia objeto de este PFC. En ella se tratara de que el paciente relacione objetos de tipo cotidiano con su representación escrita y se vuelva a habituar a la forma de expresar fonéticamente el nombre de dichos objetos.

Dentro de esta fase y una vez el paciente vaya cogiendo confianza con los objetos cotidianos es necesario que coja soltura con el alfabeto.Para ello se pueden emplear sopas de letras y ejercicios similares para que el paciente tenga que discriminar la forma grafológica de las palabras asociadas a los objetos que ha aprendido de las otras letras que se representan en el alfabeto.

Por último se debe proponer al paciente unos ejercicios de cálculo relacionados con el alfabeto. Por ejemplo indicarle al paciente que cuente las veces que aparece una letra determinada dentro de un conjunto de letras.

Esta fase puede proporcionar una gran mejoría en el vocabulario y la forma de expresarse del paciente y no requiere de la presencia física de un profesional para llevarse a cabo.

2.5.2 Dislexia

La dislexia es una dificultad de aprendizaje en la que la capacidad de una persona para leer o escribir está por debajo de su nivel de inteligencia. Se tiende a usar este término de manera amplia ante cualquier problema de lectura. Hablando con propiedad, la dislexia es la dificultad para leer causada por un impedimento cerebral relacionado con la capacidad de visualización de las palabras. En lenguaje médico se llama ceguera congénita de las palabras; otras personas la suelen denominar impedimento para leer.

Las personas con dislexia suelen invertir las letras cuando trata de escribir una palabra aunque sepan deletrearla. También suelen escribir algunas letras al revés o invertidas. La lectura es difícil porque no pueden distinguir determinadas letras o las invierten mentalmente. Aunque hasta hace poco se calificaba a las personas con dislexia como "incapacitados para aprender", la mayoría pueden aprender y sus problemas no están relacionados con la inteligencia. De hecho, muchos disléxicos son muy inteligentes y algunos alcanzan un éxito extraordinario.

No hay ninguna seguridad sobre la causa de la dislexia. Se cree que el origen es una falta del sistema nervioso central en su habilidad para organizar símbolos gráficos.
Los niños y adultos con dislexia pueden presentar algunos de estos síntomas:

- Invierten las palabras de manera total o parcial, por ejemplo casa por saca.
- invierten las letras, por ejemplo p por b, o d por b.
- Dificultad para leer oraciones o palabras sencillas. Suelen presentarse problema frecuentes con palabras cortas como del o por.
- Escriben la misma palabra de distintas maneras.
- Cometen errores de ortografía raros, como merc por comer.
- Copian las palabras mal aunque están mirando cómo se escriben.
- Conocen una palabra pero usan otra, como gato por casa.
- Tienen dificultades para distinguir la izquierda de la derecha.

Las personas con dislexia pueden beneficiarse de la terapia del habla para desarrollar lo que se llama “conciencia fonémica”. La conciencia fonémica es la habilidad para oír y usar los sonidos de las letras y las combinaciones de letras.

Tratamiento

Para el tratamiento de la dislexia no existe una terapia específica pero si algunos ejercicios que recomiendan los profesionales, como los que se presentan a continuación:

- Segmentación silábica: Pedirle que nos diga cuántas sílabas tiene una palabra. Ejemplo: ¿Cuántas sílabas tiene la palabra espirales? “Es-pi-ra-les” = 4

- Omisión de sílabas: Pedirle que omita una determinada sílaba. Ejemplo: ¿Qué quedaría si a la palabra "espirales" le quitamos la 2º sílaba. "Esrales"
- Sustitución de sílabas: Pedirle que sustituya una determinada sílaba de la palabra por otra que le demos. Ejemplo: Sustituye la 2º sílaba de la palabra "Espirales" por la sílaba "bu". "Esburales"
- Encontrar sílabas ocultas oralmente. Ejemplo: Le pedimos que nos indique la sílaba oculta o trocito que falta en la palabra "Fri-rifico" y tendría que responder "go"
- Identificar que sílaba se repite en dos palabras distintas. Ejemplo: ¿Qué trocito suena igual en explanada y plano? "pla"
- Juegos tipo veo-veo o cadena de palabras a través de sílabas. Ejemplo Veo-veo una cosita que comienza por el trocito "pla" o cadenas de palabras tipo "escayola, lazo, zorro, ropa, paloma…"
- Ejercicios de ordenar sílabas para formar palabras: Ordena las sílabas para formar una palabra, "lla – tor – ti" / "tortilla".
- Ejercicios de completar palabras con sílabas. Ejemplo: Cara__lo, tendría que escribir "me".

2.5.3 Relajación

Las terapias de relajación, son un complemento indispensable a las terapias del habla. Hay que tener en cuenta que dichos pacientes intentan expresarse con normalidad y a menudo se ven frustrados durante las sesiones al no conseguirlo. Esta situación puede provocar una sensación y malestar del paciente después de la sesión de terapia del habla lo cual podría acabar por el abandono del tratamiento.

Con una buena sesión de relajación posterior a la terapia del habla, podemos conseguir mitigar los efectos del estrés del paciente, consiguiendo así una sensación de bienestar para el mismo.

Para una buena sesión de relajación es necesario combinar los siguientes elementos:

- **Técnicas de respiración:** relajación y respiración van unidas, de la misma manera que la ansiedad va unida a una alteración en la forma de respirar. Basta con observar lo diferente que es la respiración de una persona cuando está tranquila o cuando está nerviosa.De la misma manera que el estrés modifica la respiración, modificar la respiración puede aliviar el estrés.

- **Técnicas de relajación:** Una técnica de relajación o relax es cualquier método, procedimiento o actividad que ayudan a una persona a reducir su tensión física o mental. Generalmente permiten que el individuo alcance un mayor nivel de calma, reduciendo sus niveles de estrés, ansiedad o ira. La relajación física y mental está íntimamente relacionada con la alegría, la calma y el bienestar personal del individuo.

- **Técnicas de visualización:** Las técnicas de visualización aportan imágenes positivas y agradables que nos permitan vivir en relajación experiencias orgánicas de dicha, son quizás de los ejercicios más sencillos y atractivos de realizar. Siempre tras los ejercicios de relajación podemos comprobar como parecemos estar envueltos en una nube de positividad, calma y alegría. Por ejemplo una técnica de visualización seria decir al paciente que imagine que esta tumbado en una playa viendo atardecer con un mar en calma.

2.6 Aplicaciones Android relacionadas con la Logopedia

Este último apartado del capítulo se centra en la investigación de las principales aplicaciones que hay implementadas en el campo de la logopedia. Revisando el Market de Android se ha comprobado que existe un vacío en este campo. No se han encontrado aplicaciones que cumplan con un proceso de logopedia, pero si hemos encontrado aplicaciones que implementan determinadas actividades de voz que pueden ser beneficiosas en el tratamiento que este PFC contempla, las más importantes de ellas se mencionan a continuación:

Baby try to speak

'Baby try to speak' es una aplicación disponible en varios idiomas que pretende enseñar a los niños a decir los nombres de animales mostrándoles imágenes de estos. Esta aplicación usa el sistema de reconocimiento del habla de Android.

Figura7. Imagen de la aplicación Baby try to speak

Esta aplicación viene bien para niños pero también puede ayudar a personas que han tenido alguna afectación en el habla por algún accidente cerebro vascular. No es una aplicación diseñada en el campo de las aplicaciones médicas, por tanto no tiene un sistema de seguimiento del paciente ni plantea ningún ejercicio de relajación posterior a la sesión. Se podría concluir que la aplicación tiene un objetivo lúdico y didáctico

Talking pocoyo

'Talking pocoyó' es una herramienta para que los niños aprendan jugando. En esta aplicación se propone una serie de juegos en los que el personaje de animación pocoyo interactúa con los niños mediante el sistema de reconocimiento del habla.
Una interfaz sencilla y unos juegos didácticos es la mejor manera de que los niños aprendan divirtiéndose y ejerciten su vocabulario.

Figura 8. Interfaz aplicación Talking Pocoyo

PhotoQuiz: Preguntas con fotos

Esta aplicación trata de que el usuario responda preguntas acerca de una imagen. Tiene varias categorías y es un ejercicio ideal para personas con problemas de afasia.

Figura 9. Interfaz aplicación Preguntas con fotos

Las letras y yo

Esta aplicación es un cuento ilustrado para entender qué es la dislexia, explicado a través de los sentimientos de una niña que, por mucho que se esfuerza, no aprende las letras como sus compañeros. Incluye audio para facilitar la lectura del niño, y orientaciones para padres y educadores. También está indicado para logopedas, psicopedagogos, psicólogos y maestros.

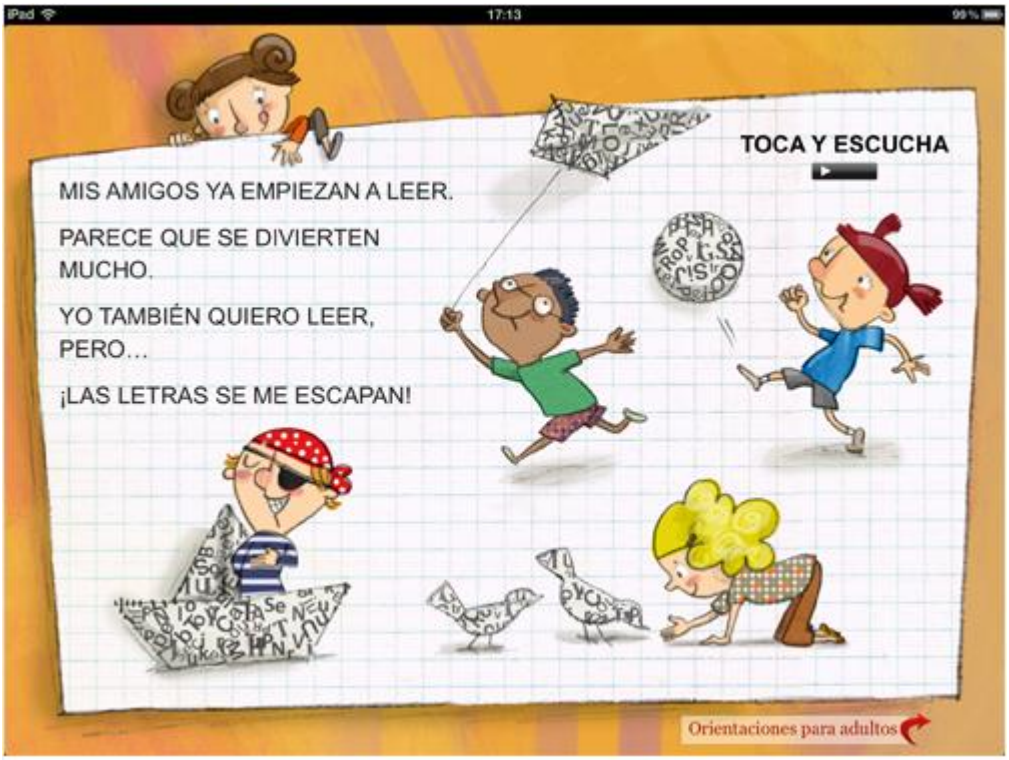

Figura 10. Interfaz de la aplicación las letras y yo

En las primeras etapas educativas, las dificultades de aprendizaje pueden llegar a ser angustiantes y desconcertantes para los niños que las sufren y para los padres y educadores. Las autoras de este cuento -la madre de una niña disléxica y una logopeda- quieren aportar a través del relato sus vivencias y conocimientos para ayudar a los niños con dislexia y a sus familias.

AMPDA

Tanto para personas con déficits en la capacidad auditiva u oral, o para personas del entorno, es necesario que se comuniquen con el método de signos para discapacitados auditivos, AMPDA, implementa una serie de herramientas encaminadas a cubrir estas necesidades

Incluye un abecedario dactilológico con imágenes, sonidos y letras, y una base de datos con palabras pre-definidas, agrupadas en diferentes grupos: animales, colores, hogar…etc. Además incluye otra sección denominada “Lector de Palabras” que no está disponible por el momento pero anuncia su estreno en la llegada de la versión 2.0, servirá de ayuda para poner sonido a las palabras escritas por el usuario.

CAPITULO 3
Estructura de Android y sus aplicaciones

En este capítulo, se va a explicar cómo funciona Android y para qué sirve cada una de las capas que lo componen, además se mostrará cual es la estructura que tiene un proyecto de desarrollo de una aplicación Android, detallando los elementos que forman la misma.

3.1 Estructura del SO Android

Como se ha indicado a lo largo del transcurso de esta memoria, Android es un sistema operativo basado en Linux. De hecho no todas las funcionalidades que están disponibles para Linux, están disponibles también para Android, ya que este último es una versión reducida de Linux, optimizada y adaptada para cumplir con las especificaciones de los dispositivos móviles para los que fue creado.

Cuando se explicaron los sistemas operativos, en el capítulo del estado del arte, vimos que un sistema operativo está formado por distintas capas, las cuales proveen a este SO de sus funcionalidades.

A continuación se va a proceder a explicar cómo funciona cada una de estas capas (Figura 11).

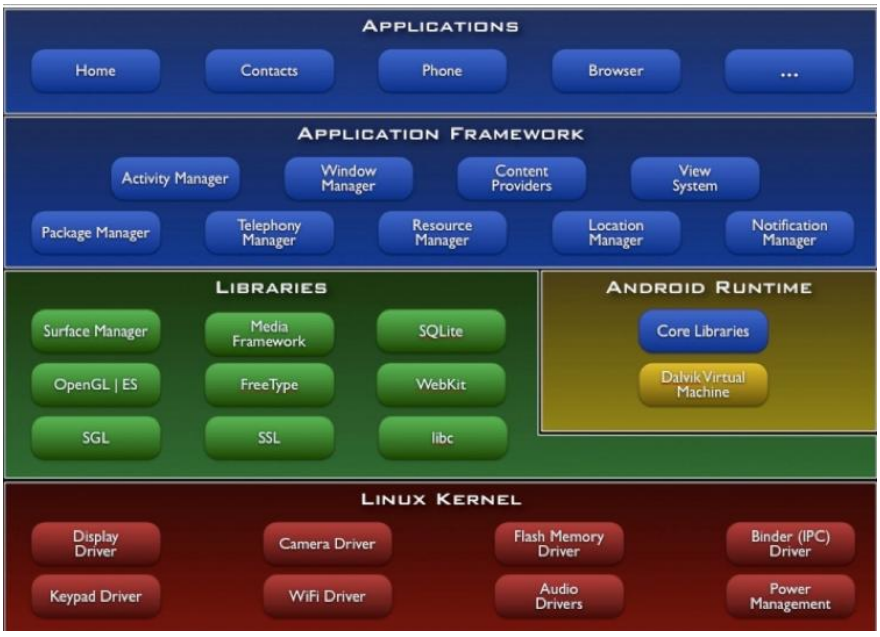

Figura 11. Capas que forman el SO Android

Kernel

El Kernel que utiliza Android es el Kernel de Linux reducido, pero introduce algunos elementos propios para los dispositivos móviles.. Esta capa sirve a su vez de nexo entre el hardware y el resto de capas de la pila .A su vez, esta capa proporciona otras funcionalidades como seguridad, gestión de memoria, control de memoria o soporte de drivers para dispositivos.

Se explican algunas de las funcionalidades propias a continuación:

- **Binder** provee a Android de un sistema de comunicación entre procesos.
- **Ashmem:** Permite manejar la memoria compartida.
- **Wavelocks:** Permite gestionar la energía del sistema

Como ya se ha expuesto anteriormente en esta memoria, Android está creado usando el lenguaje de programación Java. Este lenguaje de programación necesita disponer de una máquina virtual Java, pero la máquina virtual Java ocupaba demasiado para los dispositivos móviles. Para solventar este problema, los desarrolladores de Android crearon el **Runtime** (máquina virtual de Android para interpretar el lenguaje Java).

Librerías

La siguiente capa son las librerías nativas en C/C++ usadas en varios componentes de Android, las cuales están compiladas en el código nativo del procesador. Muchas de las librerías utilizan proyectos de código abierto.

- **WebKit:** Proporciona a Android el servicio de navegador web
- **System C Library:** Una derivación de las librerías BSD de C estándar, adaptados para dispositivos embebidos basados en Linux.
- **Media Framework:** Librería basada en PacketVideo's OpenCORE; soporta codecs de reproducción y grabación de multitud de formatos de audio, video e imágenes MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.
- **Surface Manager:** Permite al sistema acceder al motor de gráficos en 2D y 3D.
- **SGL:** Motor de gráficos 2D.
- **Librería 3D:** Implementación basada en OpenGL ES 10.0 AP. Las librerías utilizan el acelerador de hardware 3D si está disponible, o el software altamente optimizado de proyección 3D.
- **FreeType:**Fuentes en bitmap y renderizado vectorial.
- **SQLite:** SQlite provee acceso y gestión a las bases de datos SQL utilizadas por las aplicaciones Android
- **SSL:** Proporciona servicios de encriptación Secure Socket Layer.

Capa de aplicación

La capa de aplicación es la capa que permite a los desarrolladores crear aplicaciones, utilizando para ello las funcionalidades implementadas en el resto de la pila. Esta capa ha sido diseñada para simplificar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas. Este mismo mecanismo permite a los usuarios reemplazar componentes.

Una de las mayores fortalezas del entorno de aplicación de Android es que se aprovecha el lenguaje de programación JAVA. El SDK de Android no acaba de ofrecer todo lo disponible para su estándar del entorno de ejecución JAVA (JRE), pero es compatible con una fracción muy significativa del sistema.

Capa de aplicaciones

El nivel de Aplicaciones está formado por el conjunto de aplicaciones instaladas en una máquina virtual Android. Todas las aplicaciones han de ser ejecutadas en la máquina virtual de Dalvik para garantizar la seguridad del sistema.

Normalmente las aplicaciones Android están escritas en JAVA. Para desarrollar aplicaciones JAVA podemos utilizar el Android SDK, aunque también existe otra opción, que consiste en desarrollar las aplicaciones utilizando C/C++. Para esta opción podemos utilizar el Android NDK (Native Development Kit).

3.2 Estructura de un proyecto Android

Toda aplicación creada para Android sigue una misma estructura básica, que se compone del código fuente en sí, archivos de recursos y vistas, librerías de código y el android manifest.

Estas estructuras pueden gestionarse facilmente con algunos entornos de progrmación como el elegido para la realización de este PFC, el software **eclipse Juno**. Estos entornos proporcionan un sistema de ventanas en el que se facilita mucho la gestión de todos los elementos de un proyecto.

Eclipse permite crear proyectos, exportarlos, modifcarlos y gestionarlos, lo cual lo convierte en una herramienta muy potente e indispensable para el desarrollo de Aplicaciones Android.

La Figura 12 muestra como se estructuran los distintos elementos de un proyecto dentro del software eclispse

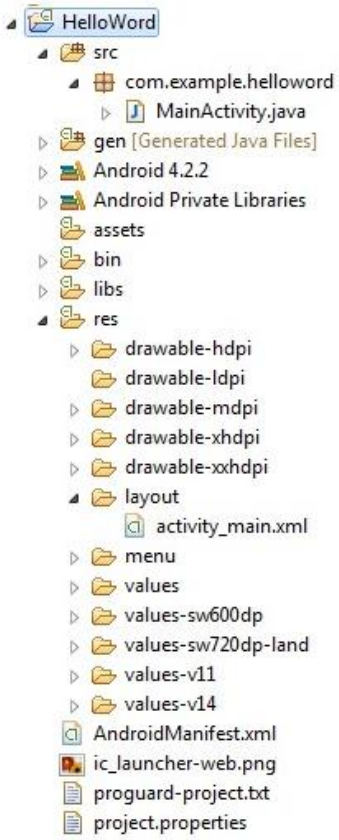

Figura 12. Estructura de un proyecto en eclipse

Como se puede observar en la figura anterior, un proyecto Android está compuesto por distintas librerías y carpetas. A continuación se va a detallar la función que cumple cada una de ellas dentro de un proyecto.

- **Directorio SRC:** Se encuentra toda la lógica de aplicación, todas las clases programadas en JAVA. Dentro de ella se definir distintos paquetes, donde puedes dividir en partes la estructura de nuestra aplicación. En la aplicación LOGOPEDIA desarrollada en el proyecto, se guardarán aquí todas las clases .java detalladas en el capítulo de desarrollo.

- **Android Library:** Aquí se encuentran todas las librerías propias del SDK de Android, dependiendo la versión elegida al crear el proyecto tendrá una versión u otra.

- **Directorios RES:** Se encuentran todos los archivos con los recursos que usan la aplicación. Las imágenes, archivos de idiomas, estilos, etc.

Este directorio a su vez se divide en otros subdirectorios específicos, donde se almacenará los diferentes recursos teniendo en cuenta su finalidad. A continuación explicamos algunos de ellos.

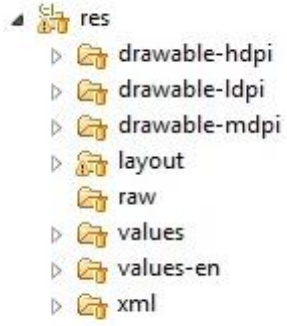

Figura 13. Estructura subdirectorios capeta res

- **Drawable:** Carpeta con todas las imágenes de la app. Se subdivide en múltiples carpetas, que contienen las imágenes en distintas resoluciones y tamaños que se usarán dependiendo el dispositivo usado. En la aplicación LOGOPEDIA, aquí se almacenan todas las imágenes mostradas en las distintas pantallas y actividades

- **Layout:** Aquí se encuentran las distintas "pantallas" de la aplicación, es decir, los archivos XML con las interfaces visual asociadas a las activities. En este directorio están guardados todos los archivos con extensión XML, de la aplicación LOGOPEDIA, nombrados y explicados en los capítulos de diseño y desarrollo de la aplicación presentes más adelante en esta memoria

- **Values:** Carpeta con los XML de contenido de la app. En esta carpeta se definen las constantes de la aplicación.

- **Directorio BIN:** Aquí se encuentran todos los archivos generados por la propia app.

  También está el ejecutable de la aplicación "apk", sería el equivalente a los "exe" de windows. Es el archivo que deberías instalar en cualquier teléfono Android para poder ejecutar la aplicación .En el caso de la aplicación LOGOPEDIA sería **Logopedia.apk** y se puede generar directamente con las opciones que aporta el software eclipse.

- **Raw:** En esta carpeta se guardan los recursos multimedia a usar en la aplicación, en el caso de la aplicación LOGOPEDIA, aquí se han almacenado las dos sesiones de relajación.

- **Directorio GEN:** En esta carpeta está el archivo R.class, éste contiene lo identificadores los recursos usados por tu proyecto: imágenes, layout, etc.

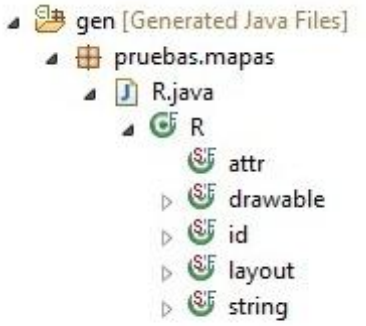

Figura 14 .Estructura del directorio gen del proyecto Android

En el archivo R.java, se generan instancias con identificadores de todos los elementos usados en la aplicación. Luego se pueden asociar estos elementos a objetos dentro de las distintas clases para actuar sobre los elementos a los que se refiere su ID

```
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int chincheta=0x7f020000;
        public static final int icon=0x7f020001;
    }
    public static final class id {
        public static final int descripcion=0x7f050001;
        public static final int mapview=0x7f050002;
        public static final int nombre=0x7f050000;
    }
    public static final class layout {
        public static final int detalles=0x7f030000;
        public static final int main=0x7f030001;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

Figura 15. Ejemplo de código generado en la clase R.java

- **Directorio Assests:** Carpeta donde se encuentran los archivos auxiliares de la aplicación: imágenes, audios, vídeos... la diferencia con los que se encuentran con la carpeta "RES", es que los archivos incluidos aquí no generarán un identificador dentro del archivo R.class anteriormente descrito. Para usar estos archivos, en vez de referenciarlos por un ID, habría que usar la ruta física como cualquier otro archivo

- **Directorio LIB:** Aquí irán las librerías externas que se deseen usar en la aplicación.

- **Android Manifiest:** Situado en la raíz de las aplicaciones como AndroidManifest.xml, es un archivo de configuración donde podemos aplicar las configuraciones básicas de la aplicación. Su configuración puede realizarse a través de una interfaz gráfica, pero es recomendable conocer la sintaxis ya que en muchas ocasiones será más fácil y rápido hacerlo desde el propio xml.

3.3 Elementos de una aplicación

Tras mostrar la estructura que tiene un proyecto desarrollado en Android, vamos a ver los distintos elementos que conforman y dan las distintas funcionalidades a una aplicación Android.

- **Activity**[10]**:** Una activity se podría definir como una pantalla que creamos para realizar una actividad concreta. Por lo tanto la activity tendrá una parte lógica que será un archivo **.java** asociada a un layout que será un archivo**.xml.**

  Cuando se ejecuta una aplicación Android, lo primero que se muestra al usuario es la ventana definida por la activity que esté marcada en el AndroidManifest.xml como principal. Las activity se gestionan como una pila, así que desde una activity se puede llamar a otra, y cuando esta finaliza se retorna a la actividad inicial.

  Una activity puede estar ejecutándose, en pausa o detenida. A continuación se muestra el ciclo de vida de una activity.

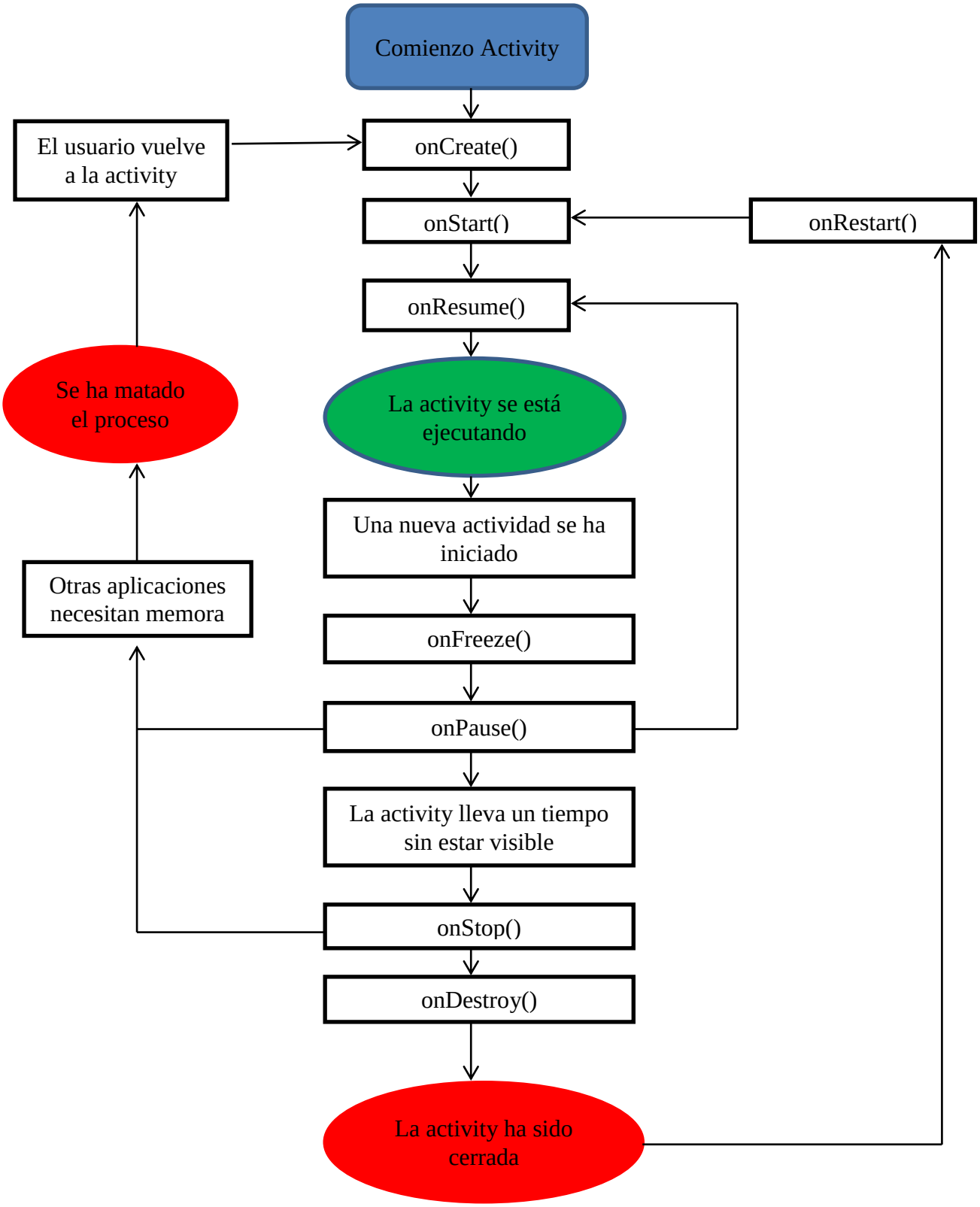

Figura 16. Ciclo de vida de un activity

Una vez vista la figura del ciclo de vida de un activity, vamos a explicar las principales funciones que intervienen en el proceso:

- **onCreate().** Este método se usa para crear nuestro activity. Solo se llama a este método la primera vez que se llama a una activity, o bien cuando se llama después de que el sistema haya tenido que eliminarla por falta de recursos. En el método OnCreate() se asocia el layout correspondiente a la activity.

- **onStart():** Es el método al que llamamos cuando queremos que la actividad se muestre al usuario. Es decir, la primera vez que se muestra, y las veces que en las que vuelve a aparecer tras haber estado oculta.

- **onStop():** Es el método que se usa para suspender una actividad sin matar el proceso.

- **onRestart()**: Es el método al que se llama para reactivar una actividad que ha sido suspendida con el método onStop().

- **onFreeze():** Este método se utiliza cuando otra activity ha sido creada pero no queremos que la actividad sea destruida, se volvería a activar el activity con el método onResume() .

- **onPause():** Al igual que ocurre con el método onFreeze, son llamadas secuencialmente cuando otra actividad va a pasar en encargarse de la interacción con el usuario. Tras onPause() la actividad permanece en un estado de espera en el que puede ocurrir que la aplicación sea destruida, por lo que estos eventos se usan para consolidar la información que no queremos que se pierda. Si la actividad no se destruye volverá al primer plano con el evento onResume().

- **OnResume:** Con este método recuperamos el hilo del activivity que ha sido suspendido temporalmente con los métodos onPause() y onResume().

La Figura 17 muestra un ejemplo de ejecución de la definición de un activity y su método onCreate().

```
import android.os.Bundle;
import android.app.Activity;


public class MainActivity extends Activity {

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
        }



}
```

Figura 17. Definición de un activity y su método onCreate()

View

Una view[11] es un objeto cuya clase es **android.view.View**. Es una estructura de datos cuyas propiedades contienen los datos de la capa, la información específica del área rectangular de la pantalla y permite establecer el layout. Una view tiene: layout, drawing, focus change, scrolling, etc.

La clase view es útil como clase base para los widgets, que son unas subclases ya implementadas que dibujan los elementos en la pantalla.

Content Provider

Es la forma que tiene Android de compartir datos entre aplicaciones. Es una forma sencilla de compartir datos sin la necesidad de dar de detalles sobre su almacenamiento.

Intent

Como se ha explicado, las activity son pantallas generadas para una actividad concreta, pues bien, cuando queremos pasar de una activity a otra utilizamos el elemento de tipo Intent.
Intent es una clase del API de Android creada para conseguir establecer este nexo entre actvities. En el intent a su vez, se pueden transferir parámetros de una activiy a otra con la función **putExtra().**La activity que ha sido llamada en el intent a su vez podrá acceder a dichos parámetro con la función **getExtra().**Ambas funciones tienen que ser llamadas desde un objeto de la clase Intent. Un ejemplo de estos comandos sería el siguiente:

```
Intent i = new Intent(this, Activity2.class);
 i.putExtra("nombre",nombre );
i.putExtra("edad", edad);
```

De esta forma pasaríamos del activity en el que nos encontrásemos a un segundo activity, pasando a este segundo activity los atributos de nombre y edad. Para que ese segundo activity recibiera esos parámetros deberíamos llamar a las siguientes funciones:

```
public void getParametrosIntent(){
        Intent i = this.getIntent();
  if (i == null)
      return ;
 Bundle b = i.getExtras();
  if (b == null)
      return ;
      nombre=b.getString("nombre");
      edad = b.getInt("edad");}
```

Con estas funciones ya habríamos conseguido pasar de una activity a otra y transferir datos entre ellas.

Broadcast Receiver

Este componente se encarga de detectar y reaccionar a los eventos generales como pueden ser batería baja, entrada de llamada, Sms recibido… Es decir es el encargado de recibir aquellos intent que son enviados a cualquier aplicación que esté "escuchando".

CAPITULO 4 Análisis y diseño de la aplicación

En este capítulo se analizarán los requerimientos que los usuarios necesitan obtener de la aplicación, teniendo en cuenta para ello objetivo y los usuarios a los que va dirigida la aplicación, que se han fijado previamente en esta memoria. En función de este análisis se realizará un diseño que cubra dichos objetivos.

4.1 Introducción a aplicación LOGOPEDIA

La aplicación Android LOGOPEDIA, que se desarrolla en este PFC, es una aplicación cuyo objetivo, es cubrir la carencia existente en el mercado actual, sobre aplicaciones destinadas a ayudar a personas con enfermedades del habla. Esta aplicación pretende proporcionar una herramienta a estos pacientes y sus familiares para mejorar sus habilidades comunicativas, pudiendo ser controladas por un profesional en el caso de considerarse oportuno por parte de los mismos.

A su vez, este proyecto pretende ilustrar las capacidades del SO operativo Android y el desarrollo de aplicaciones en este entorno, para que más desarrolladores puedan seguir ampliando este proyecto o inducirles a que creen otras aplicaciones en el campo de las aplicaciones médicas o e-health.

4.2 Escenario propuesto

Una vez introducido el concepto de la aplicación, se va a proceder a concretar el escenario que se plantea para la utilización de la misma.

Esta aplicación de logopedia, está pensada para que los pacientes puedan disponer de una herramienta en un entorno doméstico, que con la supervisión de familiares les permita desarrollar unas actividades de rehabilitación acordes a sus enfermedades.

Se pretende también que la aplicación les proporcione unas herramientas que les puedan ayudar a suplir sus carencias a la hora de comunicarse con otras personas.

La aplicación también debe contener una herramienta para interactuar con profesionales en la medida de que el profesional y el paciente lo consideren oportuno.

4.3 Requisitos de usuario

A continuación se va a proceder a definir los requerimientos que se deben implementar para que el usuario pueda interactuar con la aplicación de una forma eficiente, para lograr una evolución positiva en su enfermedad del habla:

- Registro de usuario en el que se contemplen la edad, el nombre y la enfermedad del paciente.
- Creación de base de datos para almacenar los datos de las sesiones realizadas por el paciente.
- Pantalla de diagnóstico donde recomendar al paciente los ejercicios más beneficiosos teniendo en cuenta su enfermedad.
- Interfaz de uso sencilla.
- Implementación de ejercicios específicos para las afecciones tratadas
- Herramientas para facilitar al paciente las labores de comunicación.
- Posibilidad de interactuación por voz entre paciente y aplicación.
- Introducción de ejercicios de relajación para aliviar estrés del paciente.

4.4 Diseño de la interfaz de usuario de la aplicación LOGOPEDIA

Como ya se ha indicado a lo largo del transcurso de esta memoria, la aplicación LOGOPEDIA está destinada a personas con enfermedades del habla. Estas personas pueden padecer problemas cerebrales, y el rango de edad en el que se sitúan es muy amplio, pudiendo ir desde niños hasta ancianos. Por lo tanto, la interfaz de usuario de la aplicación debe tener un diseño sencillo, que permita al usuario interactuar con ella sin dificultad, a la vez que una imagen genérica y clara debido al espectro tan amplio de población al que va destinada.

A continuación se van a mostrar las distintas pantallas con las que el usuario interactuará dentro de la aplicación Logopedia

Pantalla de registro

La pantalla de registro es la primera pantalla con la que interactuará el usuario al acceder a la aplicación. En ella se le solicitará que introduzca su nombre y edad y a continuación que indique dentro del selector de enfermedades del habla si tiene alguna enfermedad dentro de las siguientes opciones:

- Ninguna
- Afasia
- Dislexia

En la Figura 18 se puede observar el diseño de la interfaz de la pantalla de registro.

Figura 18. Pantalla de registro de la aplicación

Una vez el usuario haya completado los datos de registro y haya pulsado el botón comenzar, se crea una instancia en la base de datos con los datos del paciente para guardar los datos de la sesión realizada.

Pantalla de diagnóstico

Tras haber introducido los datos del paciente, se accede a la pantalla de diagnóstico. En esta segunda pantalla, se tendrán en cuenta los datos introducidos por el paciente en el registro y se le hará una recomendación sobre los ejercicios que más le ayudarán en la rehabilitación de su enfermedad. En el caso de haber seleccionado en el selector de enfermedades la opción "ninguna" simplemente se le dará la bienvenida al usuario.

Figura 19. Pantalla de diagnóstico de la aplicación

Una vez leída la recomendación por parte del usuario debe pulsar sobre el botón comenzar para acceder a la siguiente pantalla.

Pantalla del menú principal

En esta pantalla el usuario ya puede seleccionar a que opción de la aplicación quiere acceder. En la siguiente figura se muestra la interfaz de este menú y a continuación se indica a donde lleva cada una de las opciones.

Figura 20. Pantalla del menú principal de la aplicación LOGOPEDIA

- **Opción ejercicios**

Dentro de esta opción del menú, se accede a un submenú donde se muestra una serie de ejercicios, que se han implementado siguiendo las recomendaciones en los procesos de rehabilitación que se han mostrado en esta memoria.

El usuario podrá elegir dentro de este menú el ejercicio que desea realizar. En la siguiente figura se muestra la interfaz del menú de actividades.

Figura 21. Pantalla del menú de actividades de la aplicación LOGOPEDIA

A continuación se describe cada una de las actividades

- **Actividad Ahorcado**

Para esta actividad se ha planteado el juego clásico del ahorcado. Al usuario se le plantearan un número de espacios que corresponden a letras y mediante el reconocedor de voz, este dirá las letras que pueden corresponder a cada espacio.

Si el usuario acierta la letra, el espacio será sustituido por la letra correspondiente, en caso contrario se sumará una pieza a la figura del hombre ahorcado.

Si se completa la figura del ahorcado se dará paso a una pantalla en la que se indicará al usuario que ha perdido, en caso contrario se dará paso a una pantalla en la que se indicará que el usuario ha ganado.

En la pantalla de resultado se dará la opción al usuario de volver a intentarlo o volver al menú de actividades y se contabilizará el resultado en la base de datos.

A continuación mostramos las interfaces de las pantallas que intervienen en esta actividad.

Figura 22. Pantalla correspondiente a los layouts mostrados en la actividad ahorcado

- **Actividad sopa de letras**

Esta actividad consiste en mostrarle al usuario sopas de letras. El usuario mediante el reconocedor de voz, debe decir las palabras completas que distinga entre las letras. Cuando acierte la palabra se señalará en azul. Una vez completadas todas las palabras pasará a una pantalla de resultados, donde se le indicará que ha completado la sopa de letras, se actualizará la base de datos y se le dará al usuario la opción de volver a intentarlo o salir al menú principal. La Figura 23 muestra un ejemplo de ejecución de esta actividad.

Figura 23. Layouts correspondientes a la ejecución de la actividad sopa de letras

- **Actividad frases desordenadas**

En esta aplicación se mostrará al usuario una frase desordenada. El usuario, mediante el reconocedor de voz, deberá indicar cuál cree que es la forma correcta de decir la frase. En caso de acertar, se actualizará la base de datos contabilizando el acierto y se le mostrará una pantalla de resultado, indicando al usuario que ha acertado. En caso de fallo, se le mostrará al usuario una pantalla con la respuesta correcta y se actualizará la base de datos. La Figura 24 muestra un ejemplo de ejecución de esta actividad.

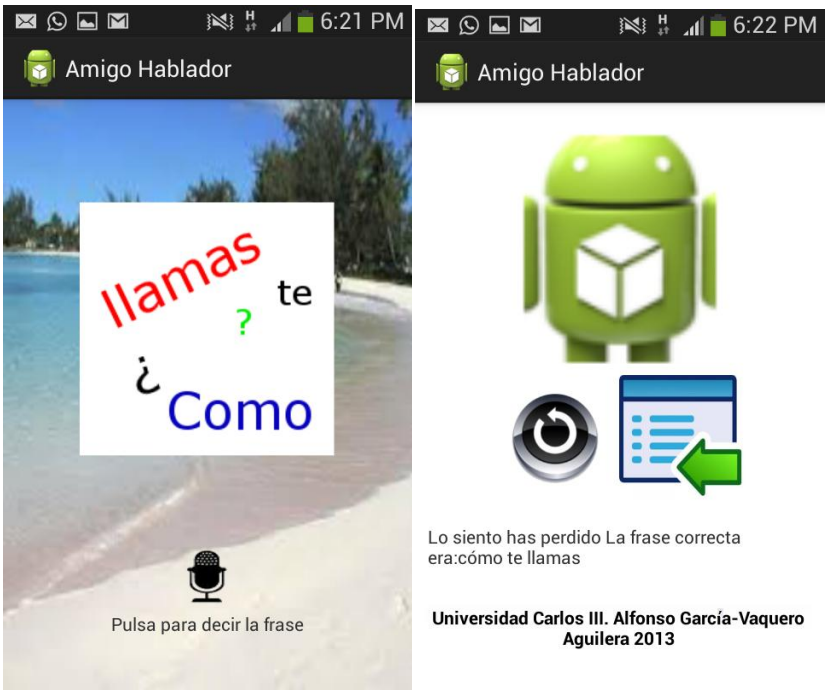

Figura 24. Layouts correspondientes a la ejecución de la actividad frases desordenadas

- **Actividad Matriz de letra**

Esta actividad, consiste en mostrar al usuario una matriz de letras del alfabeto desordenadas. La aplicación pide al usuario que cuente cuantas veces aparece una letra determinada. Mediante el reconocedor de voz el usuario indica el número a la aplicación, y esta le muestra una pantalla indicando si ha acertado o ha fallado y el resultado correcto. La Figura 25 muestra un ejemplo de ejecución de esta actividad.

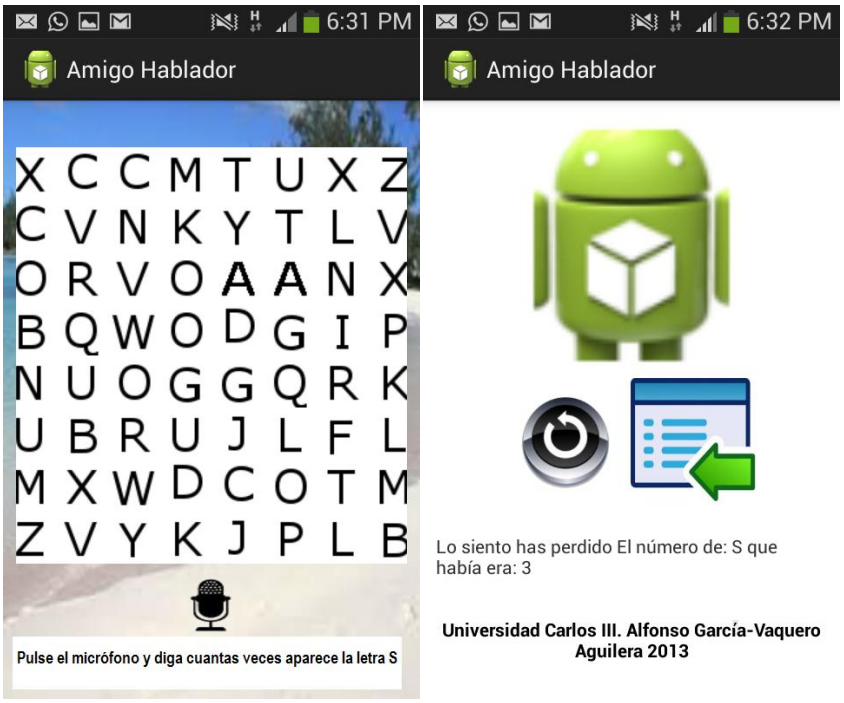

Figura 25. Layouts correspondientes a la ejecución de la actividad matriz de letras

- **Aplicación Safari**

Esta aplicación pretende mejorar la capacidad psicomotriz del paciente, a la vez de seguir ayudándole a mejor su vocabulario. En ella, se ha planteado un juego en 2D, en el que aparece una serie de animales en la pantalla, sobre los que si el usuario pulsa con el dedo, la aplicación dice el nombre del animal por medio de la librería TTS.

Además de la característica antes mencionada, la pantalla se va llenando de unos cazadores que se mueven por toda la pantalla, y el usuario tiene que ir eliminándolos pulsando sobre ellos con el dedo. Si el número de cazadores supera un umbral, el usuario pierde, pero si por el contrario el usuario elimina a todos los cazadores habrá ganado.

En este juego se han implementado 3 niveles de dificultad, y según se va aumentando de nivel, los cazadores aparecen más rápido y se mueven por la pantalla a mayor velocidad dificultando así su eliminación.

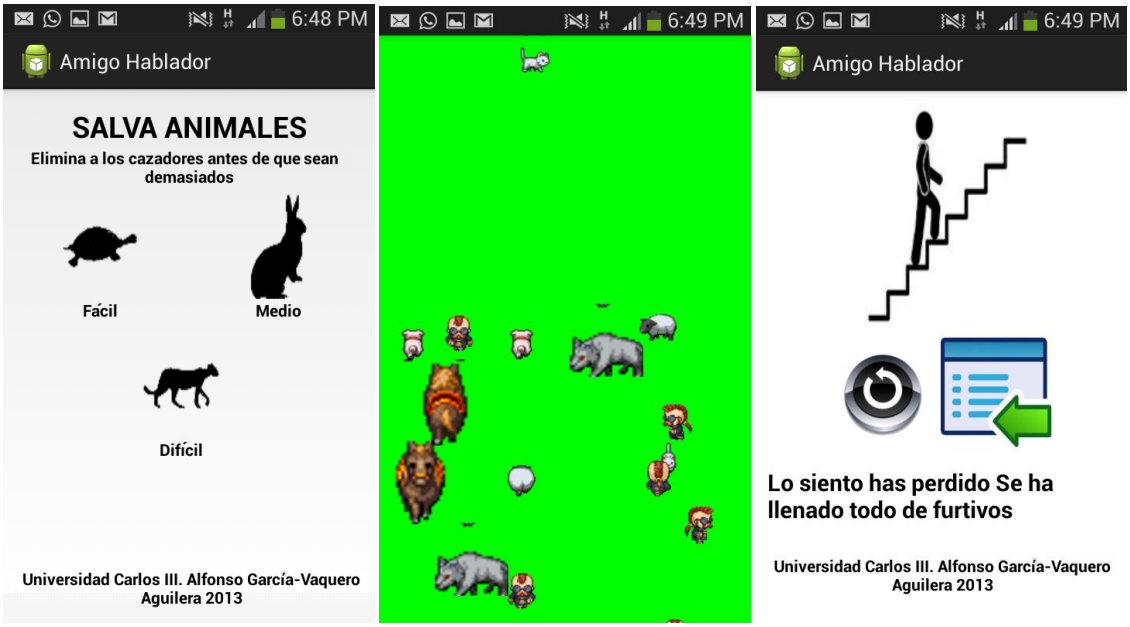

Figura 26. Layouts correspondientes a la ejecución de la actividad animales

- **Asistente de conversación**

El asistente de conversación, es una herramienta que ponemos a disposición del usuario para facilitarle la comunicación con otras personas. El asistente se compone de dos partes, una primera parte donde mediante la librería TTS el usuario puede escribir en un recuadro la frase que quiere decir, y pulsando el micrófono el móvil lo reproduce por el altavoz.

La segunda parte consiste en usar el reconocedor de voz, para que el usuario pueda entender lo que le está diciendo el interlocutor. Para ello pulsaría el botón de escuchar y la aplicación escribiría lo que ha entendido en un cuadro de texto destinado a tal efecto.

Figura 27. Layout correspondiente a la herramienta asistente de conversación

- **Diccionario de imágenes**

El diccionario de imágenes, es un apartado de la aplicación de logopedia en el que se muestran al usuario una serie de objetos o imágenes de uso cotidiano. El usuario pincha sobre el micrófono y mediante la libre TTS se reproduce la transcripción fonética de dicha imágenes.

Las imágenes están englobadas dentro de categorías. El menú de categorías dispone de las siguientes categorías:

- En casa: colección de objetos cotidianos del hogar
- Naturaleza: colección de objetos en el ámbito de naturaleza
- Ropa: colección de prendas de ropa de uso cotidiano
- Haciendo deporte: colección de imágenes representativas de los deportes más conocidos
- Los colores: Muestra al usuario los colores más conocidos

La Figura 28 muestra la interfaz de las categorías del diccionario.

Figura 28. Layout correspondiente al menú del diccionario de imágenes

La Figura 29 muestra una galería de objetos dentro de las categorías.

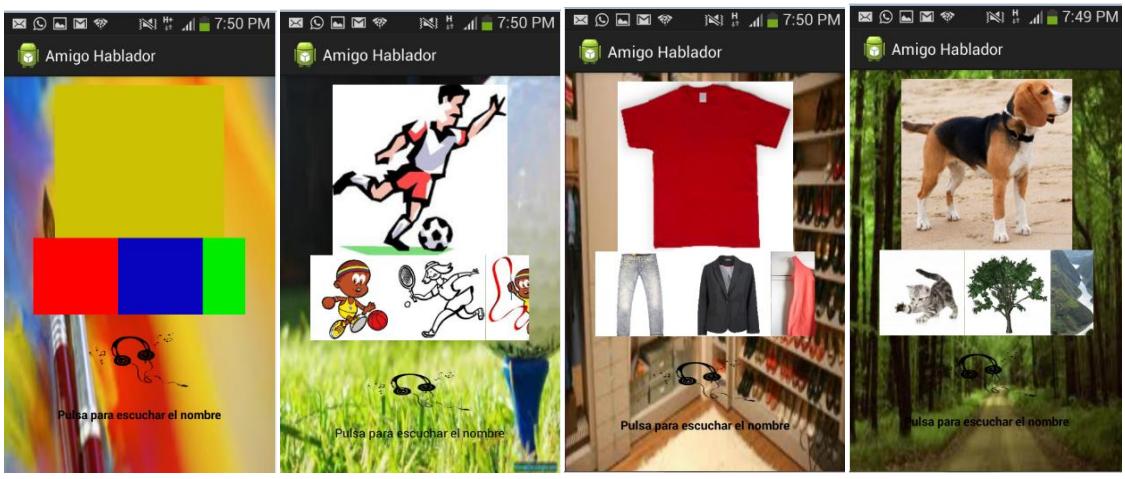

Figura 29. Layouts correspondites a las distintas categorías del diccionario de imagenes

Como se puede observar en la figura, cada categoria dispone de una imagen principal, la cual será la nombrada al pulsar el botón y un slider donde se pueden seleccionar el resto de imágenes para oir su transcripción.

- **Ejercicios de relajación**

Como se ha hablado a lo largo de este PFC, la realización de los ejercicios propuetos para una persona sin efermedades del habla no supondría ningún esfuerzo, pero para las personas que padecen estas enfermedades supone un reto que puede acarrearles un alto indice de estrés y frustración.

Para mitigar está situación se han añadido dos sesiones de relajación que el paciente puede realizar al finalizar la sesión de ejercicios. Hemos elegido añadir dos sesiones distintas y de distintas duraciones para que el usuario decida cual le funciona mejor y pueda alternarlas si lo desea. En la Figura 30 se muestra la interfaz de la actividad de relajación.

Figura 30. Layout correspondiente a la herramienta de relajación de la aplicación

Como se puede observar el la figura, en cada una de las 2 sesiones se indica la duración y posee un botón de play y otro de stop para que el usuario puede iniciar la sesión y pararla cuando estime oportuno.

Enviar datos sesión

En los requerimientos de usuario, se incluyó proporcionar una herramienta para que el paciente pudiese enviar los datos de la sesión, que se han ido guardando en la base de datos interna

Este requerimiento se ve solventado por la pestaña enviar datos de sesión. Cuando el usuario actúe sobre está pestaña, la aplicación le solicitará un correo electrónico al que enviar la información.

Automáticamente generará un correo para la dirección de email que tenga vinculado el dispositivo Android, cuyo cuerpo será generado automáticamente con los datos de la sesión y con dirección de destino la que se ha incluido en la pregunta inicial.

La Figura 31 muestra como al pulsar la pestaña se le solicita al usuario la dirección de correo del destinatario, y la figura x1 muestra el contenido del cuerpo del correo recibido por el destinatario.

Figura 31. Layout correspondiente a la herramienta de envío de email de la aplicación

Datos sesión: Alfonso Garcia - Edad: 28
Enfermedad: Afasia
Ahorcado: Número de intentos:1 - Número aciertos:1
Frases: Número de intentos:1 - Número aciertos:1
Matriz: Número de intentos:1 - Número aciertos:0
Safari: Número de intentos:1 - Número aciertos:0
Crucigramas Completados: 0
Sesión de relajación:SI

Figura 32. Cuerpo del mensaje recibido por el destinatario del email de control de sesión

CAPITULO 5 Desarrollo aplicación LOGOPEDIA

En este capítulo se mostrarán las herramientas empleadas para desarrollar la aplicación Android de Logopedia. También se explicará cómo se ha implementado el software y como funciona cada una de las clases que intervienen en el mismo.

5.1 Eclipse, SDK y ADT

El software eclipse, es un entorno de programación multiplataforma basado en Java. Esto quiere decir que soporta el desarrollo de software para múltiples lenguajes de programación en función de los complementos que le instalemos.

Para poder desarrollar la aplicación objeto de este PFC en Android, es necesario instalarle el complemento SDK(Software Development Kit) con su plugin ADT(Android Development Tool).

Figura 33. Imagen corporativa de la herramienta ADT

Estos complementos proveerán de librerías propias de Android, así como de un emulador del sistema operativo Android que se puede utilizar para visualizar y depurar la aplicación Android.

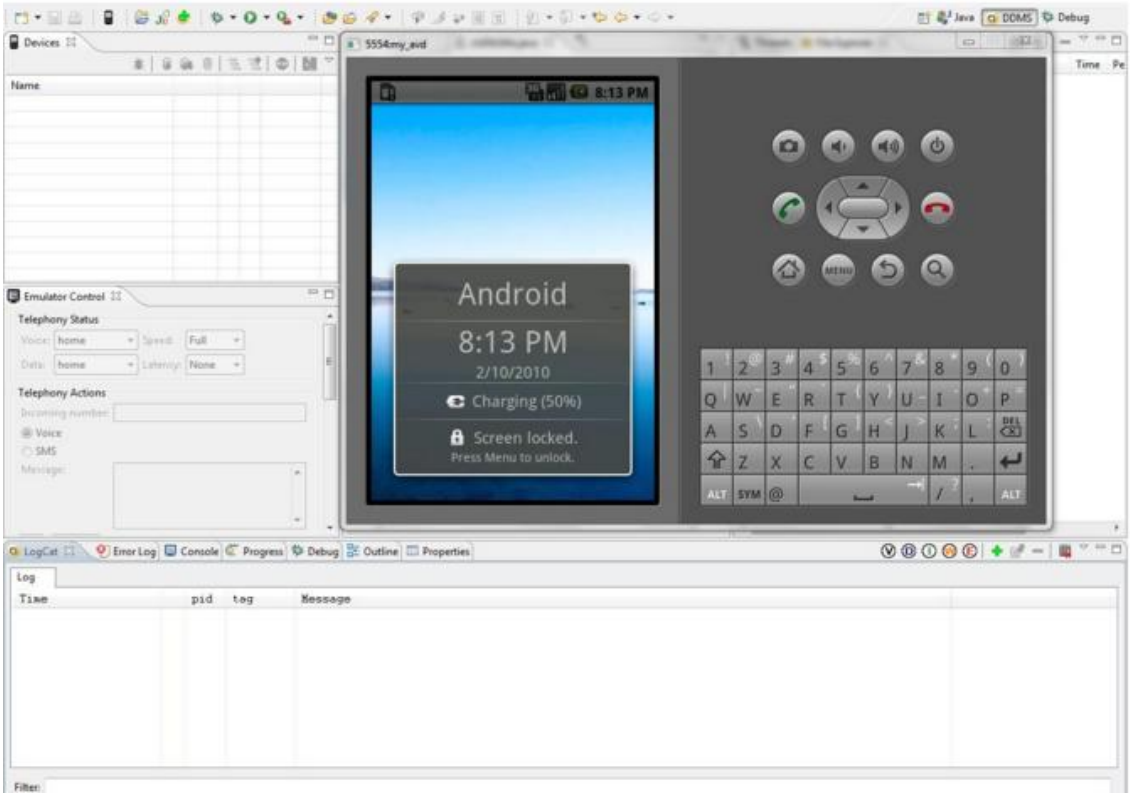

Figura 34. Herramienta AVD eclipse

El emulador Android es conocido como AVD (Android Virtual Device), y es una herramienta ideal para no tener que cargar todas las versiones del software directamente sobre el dispositivo físico, sin saber que comportamiento tendrán. El AVD puede simular la mayoría del hardware y funcionalidades de un terminal físico, aunque cabe indicar que tiene algunas limitaciones, en cuyo caso solo cabría la posibilidad de testear directamente sobre el terminal

Eclipse nos permite gestionar el proyecto y segmentarlo fácilmente y provee de herramientas gráficas para la configuración de los layouts, lo cual lo convierte en una herramienta muy potente para el desarrollo de aplicaciones Android.

5.2 Desarrollo de la lógica del programa

Para desarrollar la lógica y funcionalidades de la aplicación LOGOPEDIA, se han desarrollado una serie de clases y métodos que interactúan entre sí, proporcionando las funcionalidades requeridas para los usuarios finales.

La Figura 35 muestra el diagrama de clases resultante. En esta figura, las líneas rojas corresponden a elementos de tipo Intent, que hay creados de forma bidireccional entre clases, y que como se ha indicado a lo largo de la memoria son los encargados de transferir el hilo del programa de una clase a otra. La figura representa las clases principales, pero existen otras clases de apoyo a algunos procesos que serán expuestas en este mismo capítulo junto con el resto de las clases.

5.2.1 Clases y métodos implementados

A continuación se va a proceder a detallar las clases y métodos que intervienen en el desarrollo software de la aplicación Logopedia

MainActivity.java

En esta clase se implementan las funcionalidades de la pantalla de registro, para ello se necesitan instanciar elementos de otras clases auxiliares como los que se muestran a continuación:

Se crea la base de datos interna donde se van a ir almacenando los datos de la sesión de usuario.

miBDHelper = new LogopediaSQLite(this);

También se crea un elemento de tipo spinner y un adapter para implementar el selector de enfermedades

```
spListaEnfermedades = (Spinner) findViewById(R.id.spinner1);
ArrayAdapter<CharSequence>adapter=ArrayAdapter.createFromReso
urce(this,R.array.tiposEnfermedades,android.R.layout.simple_spinner_
item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
spListaEnfermedades.setAdapter(adapter);
```

Con la función *OnClickListener(),* se espera a que el usuario pulse el botón para capturar los datos de los campos de escritura y el selector de enfermedades. Una vez el usuario ha pulsado, se usan los datos obtenidos para crear la base de datos y se saca en pantalla un mensaje tipo Toast, en el que indica que la base de datos se ha creado con éxito.

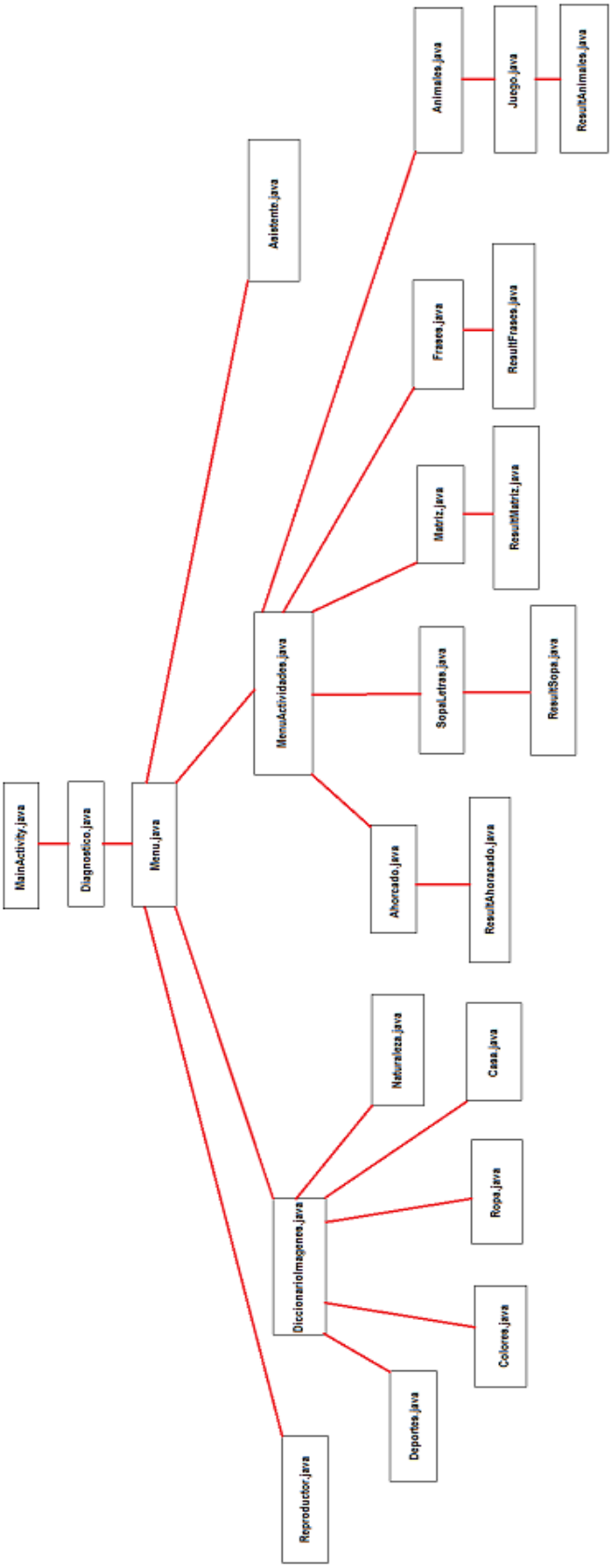

Figura 35. Diagrama de clases de la aplicación LOGOPEDIA

Una vez hecho esto se pasa el hilo del programa a la clase Diagnostico.java mediante un elemento Intent.

Diagnostico.java

Esta clase hace una consulta en la base de datos creada en MainActivity.java, para seleccionar la enfermedad que el paciente ha indicado, y muestra por pantalla en el campo destinado a tal efecto en el layout que instancia está clase (diagnostico.xml).

Una vez hecho esto implementa un elemento OnClickListener() sobre el botón comenzar y una vez que este es pulsado crea un Intent hacia la clase Menu.java para pasar a este el hilo del programa.

Menu.java

La clase menú.java, implementa una serie de eventos OnClickListener() para cada uno de los botones que se añaden en el layout menú.xml y según el botón pulsado lanza un evento tipo Intent a cada una de las siguientes clases:

- MenuActividades.java
- Asistente.java
- DiccionarioImagenes.java
- Reproductor.java

Para el botón **ver datos de sesión**, se generar un mensaje en el que se pide al usuario el correo electrónico al que debe enviar el correo. Después se comprueba que la dirección sea correcta y si es correcta se hace una consulta a la base de datos y se genera un email automático.

MenuActividades.java

La clase MenuActividades.java, como en el caso de la clase Menu.java, implementa una serie de eventos OnClickListener(), sobre los botones definidos en el layout menuactividades.xml y realiza un Intent para pasar el hilo del programa a las siguientes clases:

- Ahocado.java
- SopaLetras.java
- Frases.java
- Matriz.java
- Animales.java

Ahorcado.java

En la clase ahorcado.java se implementa el juego clásico del ahorcado mediante el reconocimiento de voz. Para poder implementar las funcionalidades que requiere este juego, se han tenido que importar los siguientes paquetes de Android:

```
import android.app.Activity;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
```

En esta clase se define array con palabras de uso cotidiano que serán las que el usuario tiene que mostrar. Luego se definirán los siguientes métodos que son los encargados de proporcionar la funcionalidad a esta actividad.

public void IniciarJuego()

Esta función inicializa el juego, teniendo en cuenta los huecos que tiene que mostrar al usuario, en función de la longitud de la palabra, e inicializa también el dibujo de la horca que determina el número de jugadas que tiene el usuario antes de perder.

public String filtroVoz(String s)

Esta función filtra el reconocimiento de voz obtenido y lo convierte en una letra.

public Drawable getDrawa(String s)

Esta función devuelve la localización de la imagen correspondiente a la letra a mostrar.

private void inicializarReconocimiento()

Función que usa el paquete *import android.speech.RecognizerIntent;* para hacer el reconocimiento de voz

protected void onActivityResult(int requestCode, int resultCode, Intent data);

Esta última función es en las que se realizan todas las comprobaciones y la física del juego.

Una vez se completa el juego se realiza un Intent a la clase ResultAhorcado.java pasándole los parámetro de cómo ha quedado el juego.

```
Intent i = new Intent(this, ResultadosAhorcado.class);
            i.putExtra("resultado", "ganador");
            i.putExtra("elegido", elegido);
            i.putExtra("nombre",nombre );
           i.putExtra("edad", edad);
```

ResultAhorado.java

En esta clase se reciben los parámetros de la clase Ahorcado.java y se muestra por pantalla en la etiqueta fijada en resultahorcado.xml, un texto indicando al usuario el resultado.

SopaLetras.java

Para la actividad sopa de letras se usan las mismas funciones que para la clase Ahorcado.java. Para ello modificamos los arrays por matrices y modificamos el método de obtener letras por cadenas de string.

ResultSopa.java

En esta clase se reciben los parámetros de la clase SopaLetras y se muestra por pantalla en la etiqueta fijada en resultsopaletras.xml un texto indicando al usuario el resultado.

Frases.java

En la clase frases.java importamos los mismos paquetes que en la clase Ahorcado.java. Existe un string de frases, en las que cada frase corresponde a una imagen donde le presentamos al usuario la frase desordenada. Luego se activa el reconocimiento de voz y se compara la respuesta del usuario con las respuestas almacenadas en el array de String. Después se evalúa si la respuesta es correcta, en función del resultado se actualiza la base de datos y se lanza un Intent hacia la clase resultFrases.java con los datos del Intent

ResultFrases.java

En esta clase se reciben los parámetros de la clase Frases.java y se muestra por pantalla en la etiqueta fijada en resultfrases.xml un texto indicando al usuario el resultado.

Animales.java

La clase Animales.java, como en el caso de la clase Menu.java, implementa una serie de eventos OnClickListener(), sobre los botones definidos en el layout animales.xml y realiza un Intent para pasar a la clase juego donde por parámetro le indicamos que nivel de dificultad a elegido el usuario entre los siguientes niveles de dificultad:

- Fácil
- Medio
- Difícil

Juego.java

La clase Juego crea una instancia del juego teniendo en cuenta el nivel seleccionado en el menú anterior, se apoya para ello en las dos clases auxiliares:

GameLoopThread.java

En esta clase se establecen un bucle en el que se van mostrando pantallazo de la clase GameView que se detalla a continuación.

GameView

En esta clase que extiende a la clase surfaceview de Android, es donde se definen las características de la pantalla, los personajes participantes en el juego(Sprites) y como estos se mueven.

Para conseguir esta sensación de movimiento, los personajes tienen una referencia con respecto al eje de coordenadas y las medidas de la pantalla, y su posición se va actualizando en la pantalla con cada interacción del bucle GameLoopThread

Asistente.java

Esta clase implementa las librerías:

```
import android.speech.RecognizerIntent;
import android.speech.tts.TextToSpeech;
import android.speech.tts.TextToSpeech.OnInitListener;
```

En ella se espera un evento OnClickListener en los botones definidos en el layout asistente.xml sobre los botones de reconocimiento o síntesis de texto a voz.

Una vez se ha pulsado el botón de reconocimiento se activa el reconocedor de voz y se muestra el texto entendido en una etiqueta de texto habilitada en el layout para tal efecto.

En el caso del TTS, el usuario debe escribir un texto en la etiqueta de introducción de texto habilitada a tal efecto, una vez se actúe sobre el botón asociado al TTS, se leerá ese campo de texto y se mandará al altavoz para ser reproducido.

Reproductor.java

En esta clase se ha utilizado el paquete android.media.MediaPlayer para la reproducción de archivos.

Se han guardado las dos sesiones de relajación en la carpeta res/raw habilitada para guardar los recursos que vayamos a tener que reproducir.

Con las siguientes funciones asociadas a eventos OnClickListener que actúan sobre los botones creados en el layout reproductor.xml

MediaPlayer.create(Reproductor.this, R.raw.ejercicio1);
MediaPlayer.create(Reproductor.this, R.raw.ejercicio2);

DiccionarioImagenes.java

La clase Diccionario.java como en el caso de la clase Menu.java, implementa una serie de eventos OnClickListener(), sobre los botones definidos en el layout diccionarioimagenes.xml y realiza un Intent para pasar el hilo del programa a las siguientes clases:

- Ropa.java
- Colores.java
- Deportes.java
- Naturaleza.java
- Casa.java

Clases Ropa.java,Colores.java,Deportes.java,Naturaleza.java y Casa.java

Las clases derivadas del diccionario de imágenes, están todas implementadas con la misma lógica. Implementan la librería TTS de Android y contienen una serie de imágenes almacenadas en la carpeta res habilitada a tal efecto.

Las imágenes se presentan con un slider que se presenta como un array de button y tienen otro array de String asociado al mismo. Un primer evento OnClickListener() sobre los elementos del array de botones fija la imagen principal en un objeto y un segundo evneto OnClickListener() activa el TTS y reproduce la representación fonética de la imagen que se presenta en el texto principal.

5.3 Layouts

Los layouts son archivos en formato xml destinados a fijar la apariencia y los elementos de los que van a disponer. Estos elementos generan una ID en la clase R.java ,la cual le permite interactuar con las funciones definidas en las diferentes clases y actuar sobre ellos.

Los archivos xml de la aplicación, han sido creados para cumplir con las especificaciones de diseños expuestos a lo largo de esta memoria.

Eclipse dispone de una herramienta para el diseño gráfico, en la cual se van añadiendo los elementos de manera gráfica, y el editor va añadiendo los comandos equivalentes al diseño gráfico como al script xml.

En la siguiente figura se muestra el editor gráfico para interface, que nos proporciona eclipse con su componente ADT

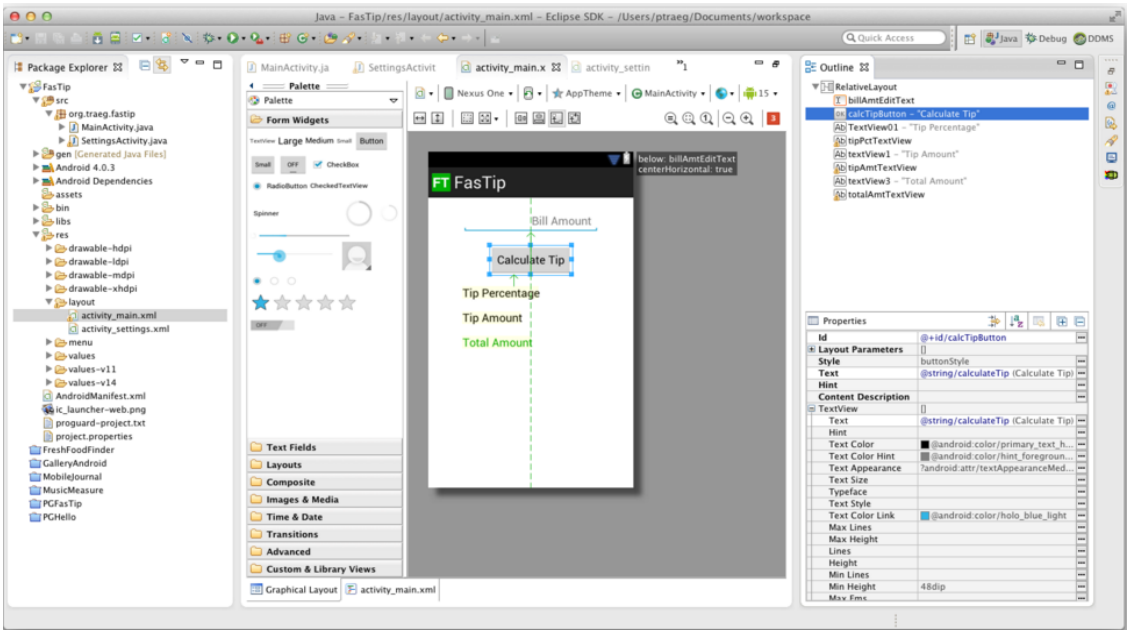

Figura 36. Editor gráfico para layouts de eclipse

Los elementos que se han incluido para la realización de la aplicación son los siguientes:

- **EditTex:** Elemento destinado a que el usuario pueda introducir texto.
- **TextView:** Elemento destinado a mostrar un texto en la interfaz.
- **ImageView:** Elemento destinado a mostrar imágenes en la interfaz.
- **Button:** Elemento que muestra un botón en la interfaz.
- **Spinner:** Elemento que proporciona un desplegable con distintas opciones a seleccionar.
- **ScrollView:** Elemento que engloba a otros elementos y les añade una barra deslizante**.**

Las pantallas generadas para la aplicación Logopedia han sido mostradas a lo largo de la memoria, en el capítulo de diseño de la aplicación.

Los layouts son instanciados en los métodos Oncreate() de cada clase de la siguiente forma:

```
protected void onCreate(Bundle savedInstanceState) {
                    super.onCreate(savedInstanceState);
                    setContentView(R.layout.ahorcado);
}
```

A continuación se van a nombrar todos los archivos XML desarrollados para esta aplicación indicando la clase desde la que son instanciadas.

mainactivity.xml: Archivo correspondiente a la pantalla inicial o de registro. Este layout se instancia desde la clase MainActivity.java en su método OnCreate();

diagnostico.xml: Layout correspondiente a la página de diagnóstico. Instanciada desde el método OnCreate() de la clase Diagnostico.java

menu.xml: Layout correspondiente a la página de diagnóstico. Instanciada desde el método OnCreate() de la clase Menu.java.

menuactividades.xml: Layout correspondiente a la página de diagnóstico. Instanciada desde el método OnCreate() de la clase MenuActividades.java.

Animales.xml: Layout correspondiente a la página del menú del juego de salvar animales. Instanciada desde el método OnCreate() de la clase Animales.java.

matriz.xml: Layout correspondiente a la página de la actividad de la matriz de letras. Instanciada desde el método OnCreate() de la clase Matriz.java.

frases.xml: Layout correspondiente a la página de la actividades de frases desordenada. Instanciada desde el método OnCreate() de la clase Frases.java.

ahorcado.xml: Layout correspondiente a la página de diagnóstico. Instanciada desde el método OnCreate() de la clase Ahorcado.java.

diccionarioimagenes.xml: Layout correspondiente a la página del menú de categorías del diccionario de imágenes. Instanciada desde el método OnCreate() de la clase DiccionarioImagenes.java.

color.xml: Layout correspondiente a la página de los colores del diccionario de imágenes. Instanciada desde el método OnCreate() de la clase Color.java.

ropa.xml: Layout correspondiente a la página de la ropa del diccionario de imágenes. Instanciada desde el método OnCreate() de la clase Ropa.java.

deportes.xml: Layout correspondiente a la página de los deportes del diccionario de imágenes. Instanciada desde el método OnCreate() de la clase Deportes.java.

naturaleza.xml: Layout correspondiente a la página de los colores del diccionario de imágenes. Instanciada desde el método OnCreate() de la clase Naturaleza.java.

reproductor.xml: Layout correspondiente a la página del reproductor de sesiones de relajación. Instanciada desde el método OnCreate() de la clase Reproducotor.java.

asistente.xml: Layout correspondiente a la página del asistente de conversación. Instanciada desde el método OnCreate() de la clase Asistente.

Capítulo 6
Evaluación de la aplicación

Este capítulo expone la evaluación a la que ha sido sometida la aplicación LOGOPEDIA en un entorno real. Para ello se distribuyó la aplicación en un colegio concertado de educación especial en San Javier (Murcia), el colegio CCEE AIDEMAR.

Este colegio proporcionó un feedback de los errores que presento la fase de pruebas de la aplicación y posteriormente la profesional de la Logopedia María Asunción Aguilera Musso, contestó a una encuesta telefónica acerca de los distintos aspectos de la aplicación que se habían tratado con los alumnos y personal.

6.1 Encuesta realizada

A continuación se detallan las preguntas y respuestas realizas con el fin de evaluar la aplicación.

Pregunta 1: ¿Considera que la aplicación LOGOPEDIA es sencilla de usar?

Respuesta: "Para los educadores nos ha resultado una herramienta útil y sencilla de usar, en cuanto al alumnado hemos tenido que guiarles en algunos puntos como el registro en la aplicación o para poder enviar los datos, pero se han adaptado bien al resto de actividades"

Pregunta 2: ¿Cree que la aplicación puede ser beneficiosa para las personas afectadas por enfermedades del habla?

Respuesta: "La aplicación ha conseguido captar la atención de los usuarios y les permite interactuar todo el tiempo que deseen. Es pronto para saber si obtienen una mejora, ya que las terapias de rehabilitación requieren de un proceso largo para dar resultados, pero el camino es interesante"

Pregunta 3: ¿Cuál es el principal problema que dirías que presenta la aplicación?

Respuesta: "En nuestro colegio no disponemos de WiFi en el aula, por lo tanto el reconocimiento de voz fallaba en alguno de los móviles que no disponían de internet, por lo que eso hacía que no todo el mundo pudiese usarla"

Pregunta 4:¿Cree que la aplicación cumple con los objetivos para los que fue diseñada?

Respuesta: "La aplicación va por el buen camino, aunque la pronunciación de las palabras es un poco artificial para ejercicios de pronunciación, pero las fases del tratamiento están bien definidas y con unos ligeros retoques podría usarse a nivel terapéutico. Lo que consideramos una herramienta muy potente es el asistente del habla

para gente con problemas graves del habla o traqueotomías, ya que es muy sencilla de usar y les permite expresarse con soltura, además de no necesitar internet **”**

Pregunta 5 ¿Qué propondría mejorar de la aplicación para desarrollos futuros?
Respuesta: “Trataría de que en el diccionario de imágenes, las palabras se expresasen con una mejor pronunciación en lugar de una voz artificial, además se podrían incluir más actividades ya que hemos vistos que estas captan la atención de los pacientes y fomentan su participación”

La encuesta realizada aclara muchos puntos que se tendrán en cuenta en el Capítulo 7 de conclusiones y desarrollos futuros de la aplicación.

Capítulo 7
Conclusiones y desarrollos futuros

En este último capítulo de la memoria, se van a detallar las conclusiones a las que se han llegado tras analizar los objetivos que se planteaban al principio de la memoria y que se han ido desarrollando a lo largo de la ejecución de este proyecto. También se propondrán las posibles líneas a seguir para seguir desarrollando y mejorando la aplicación Logopedia desarrollada a lo lardo del PFC

7.1 Conclusiones

A tenor de la aceptación de la aplicación Logopedia entre las personas que padecen enfermedades del habla y han interactuado con ella, se llega a la conclusión de que se ha conseguido el objetivo principal de este PFC que es el de hacer una aplicación de logopedia que interactuase por voz con el usuario y le ayudase a mejorar en su enfermedad del habla. Una vez evaluado el objetivo principal se va a concluir si se han cumplido los objetivos en los puntos propuestos en el objetivo

- A lo largo de esta memoria se ha hecho una comparativa del sistema operativo Android con los principales sistemas operativos, se ha descrito a su vez las funcionalidades que aporta y como está estructurado internamente. Se han proporcionado al lector de esta memoria, todas las herramientas y procedimientos para poder crear una aplicación en Android sin necesidad de tener conocimientos previos. Con toda esta información esta memoria sirve como elemento ilustrativo para la gente que quiere empezar a desarrollar en Android o desea conocer cómo funciona y cuáles son sus ventajas e inconvenientes.

- En el capítulo del estado del arte, se han presentado los sistemas de reconocimiento de voz y de síntesis de texto a voz. Se ha hecho una descripción didáctica de cuál es su funcionamiento y se ha puesto a disposición del lector la información del API de Android para que puedan incluir estos elementos en sus desarrollos en caso de considerarlo necesario. Posteriormente se ha ilustrado mediante el desarrollo como funciona este API y como invocar a los métodos propios de ella dentro de las clases de Java.

- Se ha hecho un estudio de las enfermedades del habla distinguiendo a que personas afecta y la rehabilitación que precisan. Se han asilado las posibles funcionalidades que podría tener una aplicación de Logopedia y se han puesto las herramientas para que los usuarios afectados por estas enfermedades se beneficien del uso de la aplicación LOGOPEDIA. Además se le ha aportado una herramienta para que puedan estar controlados a distancia por un profesional.

7.2 Desarrollo futuro

Para el desarrollo futuro de la aplicación es conveniente indicar que se debe mejorar la aplicación sin perder de vista que la interfaz y las funcionalidades que se añadan a la aplicación deben ser sencillas de utilizar, ya que está aplicación está dirigida en la mayoría de los casos a personas con afectaciones cerebrales. Por ello vamos a poner los puntos en los que creemos que se podría encaminar el desarrollo futuro de esta aplicación:

- **Introducción de nuevos juegos y ejercicios:** Los métodos de rehabilitación mejoran constantemente y aparecen nuevos ejercicios. Durante el desarrollo de este PFC se han propuesto una serie de ejercicios, pero podrían introducirse muchos más que enriquezcan la aplicación y le den más opciones al usuario.

- **Comunicación medico paciente:** En este proyecto se ha querido proporcionar una herramienta sencilla de usar, para que el paciente se comunique con el médico, pero cabría la posibilidad de establecer una conexión del médico con la pantalla de diagnóstico del paciente, para que fuese este quien le recomiende los ejercicios a realizar dentro de la aplicación.

- **Creación de TTS propio:** El TTS es una gran herramienta, ya que proporciona al paciente una amplio abanico de palabras, pero posee la desventaja de que en ocasiones no implementa un pronunciación muy natural debido a la forma que tiene de formar las palabras por partes. Para las parte de la aplicación que tienen un vocabulario predefinido como puede ser el diccionario de imágenes, se podría crear una base de dato propia de palabras, haciendo énfasis en la pronunciación.

- **Añadir enfermedades:** Por la distinta sintomatología de las enfermedades del habla, se ha decidido adaptar la aplicación LOGOPEDIA al tratamiento de la afasia y la dislexia, que son las enfermedades que tienen más elementos comunes en sus procesos de rehabilitación, extrapolables al uso del reconocedor y del TTS, pero existen más enfermedades a las cuales se le podrían adaptar ejercicios específicos para el tratamiento de sus enfermedades.

Glosario

- **Activity:** componente utilizado en las aplicaciones Android para representar las pantallas que componen la interfaz de usuario.

- **ADT:** siglas de Android Development Tools**,** es un *plug-in* de Eclipse que permite el desarrollo de aplicaciones Android.

- **API:** siglas de *Application Programming Interface*, es una colección de funciones o métodos recopilados en forma de biblioteca para crear una capa de abstracción sobre la que trabaja el desarrollador. Define la forma en que un módulo software interactúa con otro.

- **APK:** siglas de *Applicacion PacKage File*, es un paquete de aplicación para la plataforma Android.

- **Códec:** elemento software-hardware capaz de codificar y descifrar un archivo con un flujo de datos. Muy utilizado en videoconferencias y archivos multimedia.

- **Dalvik:** entorno de ejecución de Android, derivado de una versión reducida de la máquina virtual de Java.

- **Driver:** programa que controla la interacción entre el sistema operativo y un periférico.

- **GPL:** siglas de *General Public License*, es la licencia creada por la *Free Sofware Foundation* que define las bases de distribución, modificación y uso de software libre.

- **HTML:** del inglés *HyperText Markup Language*, estándar que define el lenguaje de marcado utilizado en el desarrollo de páginas web.

- **JDK:** siglas en inglés de *Java Development Kit*, es el conjunto de herramientas de desarrollo utilizadas para la creación de programas Java. Actualmente lo suministra Oracle.

- **Kernel:** núcleo de un sistema operativo, encargado de proporcionar acceso al hardware y gestionar recursos del sistema mediante una capa de abstracción.

- **Layouts:** son ficheros XML que definen la estructura visual de la interfaz de usuario en Android.

- **MySQL:** sistema de gestión de bases de datos relacionales multihilo, multiusuario y estable, suministrado bajo licencia GPL.

- **Plug-in:** componente software que se integra y amplía la funcionalidad de otro.

- **RAH:** siglas de Reconocedor Automático del Habla, módulo dentro de los sistemas de diálogo que captan y procesan la voz del usuario para obtener una cadena de texto.

- **Smartphone:** término inglés del que surge el concepto de teléfono inteligente.

Bibliografía

[1] Gantt . “Gantt Download Page”,
obs-edu.com [En línea] . Disponible en:
http://www.obs-edu.com/blog-project-management/diagramas-de-gantt/ganttproject/

[2] Presupuesto “Presupuesto Download Page”,
Portal.uc3m.es [En línea] . Disponible en:
http://portal.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20(3)_1.xlsx

[3] Penetración smartphones “Penetración Smartphones Download Page”,
Ditrendia.es [En línea] . Disponible en:
http://www.ditrendia.es/wp-content/uploads/2014/07/Ditrendia-Informe-Mobile-en-Espa%C3%B1a-y-en-el-Mundo.pdf

[4] Descargas App Store “Descargas App Store Download Page”,
Deloitte.com [En línea] . Disponible en:
http://www2.deloitte.com/content/dam/Deloitte/es/Documents/tecnologia-media-telecomunicaciones/Deloitte_ES_TMT_Consumo-movil-espana-2014-def.pdf

[5] Tipos de aplicaciones móviles existentes “aplicaciones moviles existentes Download Page”,
lancetalent.com [En línea] . Disponible en:
https://www.lancetalent.com/blog/tipos-de-aplicaciones-moviles-ventajas-inconvenientes/

[6]Evolución IOS “Evolución IOS Download Page”,
Enter.co [En línea] . Disponible en:
http://www.enter.co/especiales/vida-digital/del-1-al-8-la-evolucion-del-sistema-operativo-ios/

[7]Evolución Android “Evolución Android Download Page”,
tuexpertomovil.com [En línea] . Disponible en:
http://www.tuexpertomovil.com/2014/10/20/android-el-sistema-operativo-cuya-historia-se-resume-en-seis-anos/

bloglayer.com [En línea] . Disponible en:
http://blog.layer.com/how-we-leverage-ios-push-notifications/

[8] Funciones API “Funciones API Download Page”,
developer.android.com [En línea] . Disponible en:
http://developer.android.com/reference/android/speech/SpeechRecognizer.html

[9] API TTS “API TTS Download Page”,
developer.android.com [En línea] . Disponible en:
http://developer.android.com/reference/android/speech/tts/TextToSpeech.html

[11] ACTIVITY “ACTIVITY Download Page”,
desarrolloweb.com [En línea] . Disponible en:
http://www.desarrolloweb.com/articulos/android-que-es-una-activity-o-actividad.html

[11] VIEW “VIEW Download Page”,
developer.android.com [En línea] . Disponible en:
http://developer.android.com/reference/android/view/View.html