UNIVERSIDAD DE VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Diseño y desarrollo de una aplicación Android de seguimiento y rehabilitación cognitiva y de las actividades básicas de la vida diaria

Autor:
D. Abdelali Elbihari Elmersy

Tutor:
Dña. MÍRIAM ANTÓN RODRÍGUEZ

Valladolid, 14 de enero de 2020

TÍTULO:	**Diseño y desarrollo de una aplicación Android de seguimiento y rehabilitación cognitiva y de las actividades básicas de la vida diaria**
TITLE:	**Design and development of an Android application for monitoring and cognitive rehabilitation and basic activities of daily living**
AUTOR:	**D. Abdelali Elbihari Elmersy**
TUTOR:	**Dña. Míriam Antón Rodríguez**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL	
PRESIDENTE:	**Mª ÁNGELES PÉREZ JUÁREZ**
VOCAL:	**MÍRIAM ANTÓN RODRÍGUEZ**
SECRETARIO	**DAVID GONZÁLEZ ORTEGA**
SUPLENTE	**JAVIER AGUIAR PÉREZ**
SUPLENTE	**MARIO MARTÍNEZ ZARZUELA**

FECHA:	**14 de enero de 2020**
CALIFICACIÓN:	

Resumen y Palabras Clave

En la actualidad, las Tecnologías de la Información y las Comunicaciones (TIC) están presentes en una gran variedad de ámbitos como son la educación, la política, la seguridad o la sanidad. Es en este último donde la aplicación de las TIC ha supuesto un gran avance en la forma en la que el paciente y el profesional de la salud interactúan entre sí. Como resultado de este avance, surge lo que se conoce como *eHealth*, y dentro de este sector, se encuentra un campo más específico conocido como *mHealth*. Este campo representa la conjunción de internet y movilidad aplicados al ámbito de la salud, en este sentido, las *apps* para dispositivos móviles dirigidas al *mHealth* son el mejor representante de la aplicación de las TIC al ámbito de la salud.

A pesar de los avances tecnológicos y su aplicación en el ámbito de la medicina, todavía hay áreas dentro de esta última donde la aplicación de la TIC es casi inexistente, en concreto, la atención sociosanitaria dirigida a la tutela de personas mayores y su rehabilitación cognitiva. Para ello, se están desarrollando aplicaciones móviles para facilitar el trabajo de los profesionales de este sector, especialmente los voluntarios.

Es pues dentro de este contexto donde se desarrolla el presente TFG, con el desarrollo de una aplicación móvil para Android de seguimiento y para la realización de juegos de rehabilitación cognitiva y de las actividades básicas de la vida diaria. En este contexto, el papel del paciente está representado por el tutelado y en el papel del profesional de la salud se encuentra el voluntario cuyo cometido es el de realizar un seguimiento del tutelado y asistirlo en el uso de la aplicación para la realización de las actividades o juegos dirigidos a ayudar al tutelado con su rehabilitación cognitiva y las actividades básicas de vida diaria.

Además del desarrollo de la aplicación, se realizaron pruebas para comprobar el su correcto funcionamiento y se ha elaborado un manual de usuario para facilitar su uso parte del usuario. Y como último punto, se han analizado las posibles mejoras y posibilidades futuras a la hora de ampliar las funcionalidades de la aplicación para adaptarla a las futuras necesidades de los usuarios.

Palabras Clave

TIC, *eHealth*, Atención sociosanitaria, Apps, Rehabilitación cognitiva, Android, API, Base de Datos

Abstract and Keywords

Nowadays, Information and Communications Technology (ICT) is present in a wide variety of fields such as education, politics, security or health. It is in the latter where the application of ICT has been a great advance in the way in which the patient and the healthcare professional interact with each other. As a result of this progress, we have what is known as eHealth, and more specifically mHealth, which is a subfield within eHealth. This field represents the conjunction of internet and mobility within the scope of healthcare, in this sense, the mobile apps aimed for mHealth are the best portrayal of the application of ICT to the field of healthcare.

Despite the technological advances and their application in the field of medicine, there are still areas within the latter where the application of ICT is almost non-existent, particularly the socio-health care aimed for the caretaking of the elderly and their cognitive rehabilitation. For this purpose, mobile applications are being developed to facilitate the work of the professionals in this sector, especially the volunteers.

Therefore, It is within this context where the present final thesis is developed, with the development of a mobile application for Android aimed for monitoring and carrying out cognitive rehabilitation games and the basic activities of daily life. In this context, the role of the patient is represented by elderly person, whom the caretaking is aimed for, and in the role of the health professional, we have the volunteer, whose task is to do a follow up of the elderly person and assist him in the use of the app for carrying out the activities or games that are intended for helping with his cognitive rehabilitation and the basic activities of daily life.

In addition to the development of the app, some testing was carried out to assure its correct functioning as well as the creation of a guide in order to help the user. And as a last point, improvements and future possibilities has been analysed in terms of expanding the application's functionalities to adapt it to the future needs of the users.

Keywords

eHealth, Socio-health care, Cognitive rehabilitation, ICT, Apps, Android, API, Database

Agradecimientos

En este apartado, me gustaría agradecer a todas las personas que han hecho posible el desarrollo y de este Trabajo Fin de Grado.

En primer lugar, a mis padres, que han luchado para que pueda tener una educación de calidad.

En segundo lugar, a mis amigos, por su apoyo y su motivación para que nunca pierda la esperanza en el futuro a pesar de todas las dificultades que me he encontrado a lo largo de estos años y que han supuesto un retraso importante en mi carrera y en mis estudios.

En tercer lugar, a mi tutora Miriam, por todo su esfuerzo y, sobre todo, su paciencia para ayudarme con la realización de este proyecto.

Finalmente, agradecer a todos los compañeros/as, profesores/as y resto de personas involucradas en mi vida, de las cuales he aprendido valiosas lecciones que me han ayudado en mi vida.

Índice General

Índice de Figuras y Tablas

Figuras

Tablas

Capítulo 1 - Introducción

Capítulo 1 - Introducción

Este primer capítulo estará dedicado a dar una visión resumida del papel que juegan las aplicaciones de las TIC en el campo de la atención sociosanitaria. Dicha atención comprenderá dos partes fundamentales como son la tutela de personas mayores y su rehabilitación cognitiva. Además, se expondrán los objetivos, las fases y los medios para la realización del TFG. Finalmente se describirá la estructura básica de la memoria del TFG.

1.1 Atención sociosanitaria

La atención sociosanitaria comprende el conjunto de cuidados destinados a aquellos enfermos, generalmente crónicos, que por sus especiales características pueden beneficiarse de la actuación simultánea y sinérgica de los servicios sanitarios y sociales para aumentar su autonomía, paliar sus limitaciones o sufrimientos y facilitar su reinserción social (Vicente Fuentes, 2013).

Es un hecho, ampliamente documentado, que la estructuración tradicional de los servicios de salud y de los servicios sociales no se adapta bien a la naturaleza mixta de determinadas necesidades, y que esa inadaptación determina su incapacidad para responder a esas necesidades con la prestación de la atención más idónea, es decir, de una atención integral y personalizada, orientada a garantizar la buena articulación y la continuidad de los cuidados (Ararteko, 2015).

La constatación de esta realidad y el creciente aumento de las necesidades que requieren la intervención combinada de ambos sectores, llevan a considerar la necesidad de articular un espacio mixto, que pretende aplicar fórmulas de atención y servicios de forma combinada y complementaria. Cuando alcanzan su nivel máximo de coordinación, estas fórmulas pueden llegar a adquirir entidad propia. Dicha entidad se conoce como “atención sociosanitaria”, “espacio sociosanitario” o, simplemente, “coordinación sociosanitaria” (Vicente Fuentes, 2013).

El siguiente esquema trata de reflejar el conjunto de relaciones englobadas en este enfoque (Vicente Fuentes, 2013):

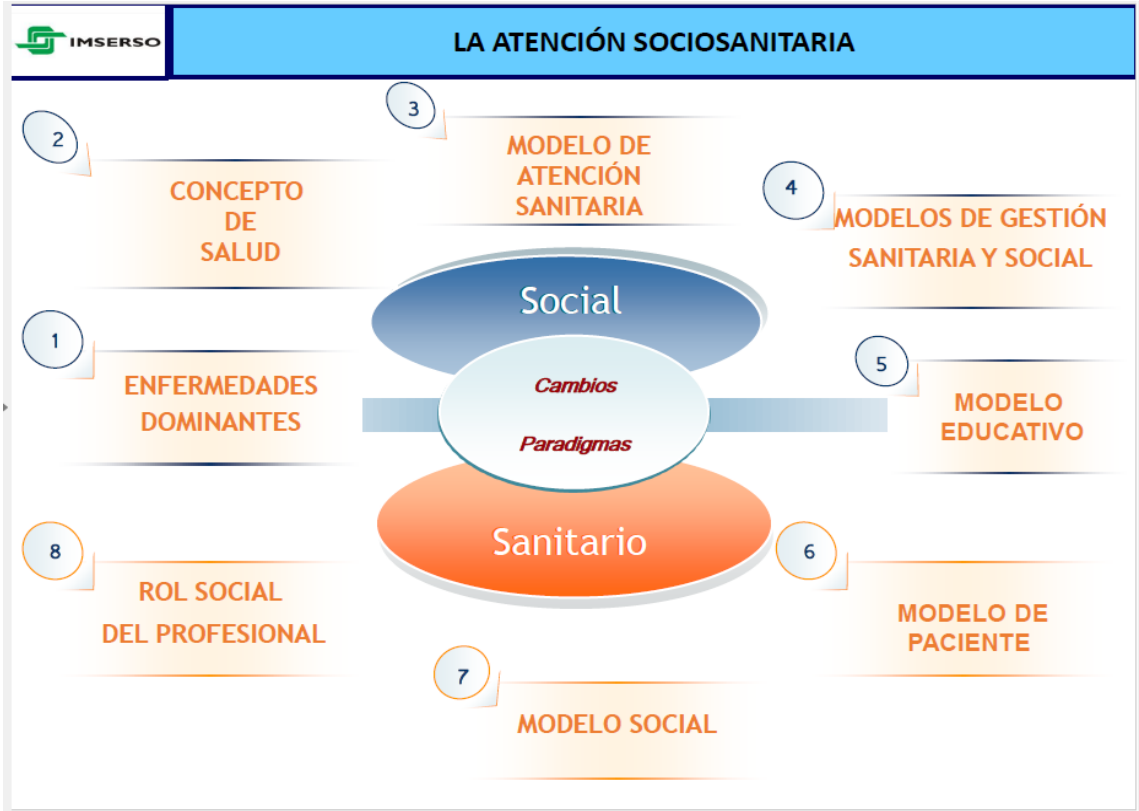

Figura 1 - Conjunto de las relaciones englobadas en la atención sociosanitaria (Vicente Fuentes, 2013)

A pesar de las diferentes denominaciones y a pesar también de la diversidad de modelos y de formas organizativas que se engloban bajo las mismas, existen algunos elementos que pueden considerarse definitorios de la atención sociosanitaria (Ararteko, 2015):

- Responde a situaciones de necesidad complejas que requieren de intervenciones de naturaleza mixta, sociales y sanitarias, de forma simultánea, o también secuencial, pero en todo caso complementaria y estrechamente articulada.
- Tiene un objetivo principal: garantizar la continuidad de los cuidados, evitando desajustes materiales, solapamientos y vacíos o déficit de cobertura, y desajustes temporales, desfases o tiempos de espera entre los diferentes servicios.
- Recurre, para su prestación, a una serie de estrategias e instrumentos de colaboración que se articulan en torno al principio de interdisciplinariedad.
- Tiene su sentido y fundamento en un enfoque de atención integral centrado en la persona y orientado a garantizar el máximo nivel de bienestar, calidad de vida y autonomía, otorgando prioridad y facilitando, siempre que sea posible, su permanencia en su entorno social habitual y la atención en su domicilio.

Algunos ejemplos de grupos de población que constituyen el objeto prioritario de la atención sociosanitaria serían los siguientes (Ararteko, 2015):

- **Personas mayores dependientes**

  Este grupo es el más representativo, primero, porque es el más numeroso, el envejecimiento determina, más que otras edades, la aparición de problemas o dificultades físicas, psicológicas y sociales cada vez más complejas. Además, cuando las personas mayores presentan un episodio agudo, es frecuente que éste, al margen de requerir un periodo más largo de recuperación que el que precisan personas más jóvenes, se presente en el contexto de otra enfermedad, de carácter crónico o de larga duración, que requiere un tratamiento largo, complejo e integral y, por lo tanto, precisa la coordinación de una serie de competencias y conocimientos aportados por diferentes especialidades médicas, pero también por otras disciplinas profesionales, como, en particular, el trabajo social.

- **Personas con condiciones crónicas**

  El concepto "condiciones crónicas" permite agrupar una serie de contingencias de larga duración que, siguiendo a la Organización Mundial de la Salud en su informe sobre "Cuidado innovador para las condiciones crónicas: agenda para el cambio", incluyen básicamente enfermedades no transmisibles (por ejemplo, cardiopatía, diabetes, cáncer), enfermedades transmisibles persistentes (por ejemplo, sida), trastornos mentales y deficiencias físicas y sensoriales. Independientemente de su etiología, son todas ellas condiciones crónicas y todas ellas requieren cambios, por tiempo indefinido, en el modo de vida de las personas afectadas y en sus pautas de cuidado de su propia salud. Todas estas condiciones crónicas también requieren, del sistema sanitario, pautas de atención que, sin perjuicio de las especificidades asociadas a la particular naturaleza de cada una de ellas, presentan, en su esencia, numerosas similitudes.

Son necesarios ciertos sistemas socio sanitarios en conjunción con la aplicación de las Tecnologías de la Información y la Comunicación (TIC) para ofrecer una buena atención socio sanitaria. Este TFG se centra en tres de dichos sistemas, los cuales de definen a continuación:

- ***eHealth***: el uso de las TIC para el apoyo al cuidado de la salud tanto a nivel local como a distancia (WHO, 2019).

- ***mHealth*****:** el uso de dispositivos móviles, como son los smartphones, aparatos para el seguimiento de pacientes y otros dispositivos inalámbricos, para la práctica médica y la sanidad pública (WHO, 2011).

- **Envejecimiento activo:** el proceso de optimización de las oportunidades de salud, participación y seguridad con el fin de mejorar la calidad de vida de las personas a medida que envejecen (WHO, 2015).

1.2 Introducción al *eHealth*

El impacto de las TIC es decisivo en el desarrollo de la medicina y clave en la gestión de pacientes crónicos, mejorando la calidad asistencial y la seguridad y cambiando el concepto de equidad y accesibilidad, al salvar las barreras tanto geográficas como políticas, económicas y administrativas, ayudando a mejorar la continuidad del proceso asistencial en base a la eficiencia, eficacia y efectividad, reduciendo costes, evitando duplicidad de pruebas, agilizando procesos, evitando el desarrollo de morbilidades y posibilitando el desarrollo de la medicina personalizada. La adopción de tecnologías en *cloud (en la nube)* dinamizará aún más la implantación de la telemedicina (*eHealth*) y de los Historiales Médicos Electrónicos (HME) en nuestro entorno asistencial (Cabo Salvador, 2015).

El *eHealth* es aplicado en un amplio rango de aspectos que afectan el cuidado de la salud, el **seguimiento** de los pacientes, la proporción considerable de ventajas en materia de información e incluso favorecer la obtención de diagnósticos alternativos a través de actividades como por ejemplo **juegos de rehabilitación cognitiva** como se verá más adelante.

El *eHealth* se está consolidando poco a poco pero no es un modo alternativo o adicional de atención sanitaria como consecuencia de la aplicación de las TIC, sino formas diferentes de prestar servicios ordinarios; en muchos casos, de forma más eficiente y efectiva, y en otros, de forma más equitativa, gracias al potencial de cambio que las mismas facilitan, para poder mejorar: la accesibilidad, rapidez en la atención, reducción de tiempos de respuesta, implantación de alertas, ahorro de costes, precocidad diagnóstica, mejora de la efectividad diagnóstica o terapéutica, mejora en la calidad del servicio, etc. (*Jadad y Lorca,* 2009).

El *eHealth* aplicado exclusivamente en el entorno de las **tecnologías móviles y dispositivos como Smartphones y Tablets** suele ser denominado ***mHealth*.** Dado que este proyecto se enfoca en el desarrollo de una aplicación para dispositivos móviles Android**,** el *mHealth* será el sistema socio sanitario dentro del cual se enmarca el TFG.

1.3 Tutela de personas mayores

El envejecimiento viene acompañado en ocasiones de un cierto declive cognitivo, un deterioro de la memoria, desorientación temporal y espacial, una merma de la capacidad de juicio, así como un desgaste de otras capacidades cognitivas que impiden la correcta toma de decisiones. Llegados a un punto límite y a efectos de salvaguardar sus propios intereses, resulta necesario que alguien del entorno de la persona mayor asuma estas responsabilidades por su propia persona. Sin embargo, hay casos de personas mayores que no tienen familiares ni otros apoyos sociales idóneos, y ahí es cuando entran en juego asociaciones que ayudan a las personas mayores y para ello han de ser sin ánimo de lucro y figurar entre sus fines la protección de las personas incapacitadas (Larescv, 2018). Una de estas asociaciones es Fundamay y es para la cual va dirigida la aplicación cuyo desarrollo se detalla en este TFG.

Fundamay es una entidad sin ánimo de lucro que se encarga del ejercicio de la tutela jurídica de personas mayores en la Comunidad de Castilla y León. Teniendo en cuenta que el ejercicio tutelar está pensado como un mecanismo de ayuda y de protección a la persona incapacitada, para poder dar un servicio adecuado y personalizado, es necesario que la entidad tutelar cuente con el personal necesario para poder llevar a cabo sus funciones con los criterios de calidad establecidos; siempre pensando en los beneficios y garantías de la persona tutelada. La incapacitación y tutela de las personas mayores, no significa que hayan de ser marginadas, sino que requieren una protección y garantía para realizar determinados actos de su vida. Esto significa “participar, gozar, tener protagonismo, seguir su historia, tener una función, unos objetivos y contribuir así, de forma activa, a la construcción de su cotidianeidad”. Todo ello se puede conseguir si existe una red de voluntariado que pueden prestar estos apoyos (Fundamay, 2019).

Para atender estas situaciones Fundamay cuenta con la figura del “voluntario Tutelar”, persona que, de forma comprometida, asume aquellos aspectos de la tutela que son difíciles de garantizar desde las fundaciones como la cercanía, el seguimiento y acompañamiento puntual de las actividades de su vida cotidiana. Este voluntario está en contacto con los profesionales de la fundación para un mejor conocimiento del tutelado, garantizando una atención e intervención más acorde no solo con sus necesidades, sino también con sus gustos y preferencias, buscando siempre el bienestar y la felicidad de la persona tutelada (Fundamay, 2019).

1.4 Rehabilitación cognitiva

La rehabilitación cognitiva es el conjunto de procedimientos y técnicas que tienen por objetivo alcanzar los máximos rendimientos intelectuales, la mejor adaptación familiar laboral y social en aquellos sujetos que sufren o sufrieron una injuria

cerebral. Se aplica en numerosas enfermedades, entre las que destacan el daño cerebral traumático, el ataque cerebrovascular y las demencias. Su fundamento neurobiológico es la plasticidad sináptica, fenómeno bien conocido y demostrado en experimentación animal, cultivos celulares, y en el ser humano. Básicamente se utilizan estrategias de restauración, compensación y sustitución, de las funciones cognitivas, ya sea en base a modificaciones ambientales, entrenamiento de habilidades compensatorias o reentrenamiento directo de las áreas comprometidas (Lorenzo Otero y Fontán Scheitler, 2001).

La rehabilitación cognitiva coge especial relevancia a la hora de intervenir en ciertos procesos cognitivos como son el **lenguaje**, la **memoria**, la **atención**, la capacidad de **cálculo**, la **orientación**, la **percepción** y el **razonamiento**. Procesos en los que se va a enfocar en esta App

Algunos de los objetivos de las actividades y juegos propuesto en la APP son: mejorar las distintas áreas de la vida diaria del tutelado, ayudar a maximizar el grado de autonomía de la personas mayores, fomentar el aprendizaje sin error, la visualización, etc. Todos estos objetivos tienen el fin de aumentar la calidad de vida tanto del paciente como de sus familiares y cuidadores.

1.5 Objetivos del TFG

El objetivo fundamental de este TFG es la elaboración y documentación de una aplicación de rehabilitación cognitiva y de las actividades básicas de la vida diaria, basado en juegos y seguimientos, que se sustente en tecnologías móviles, en este caso para dispositivos móviles Android, y que ayude a los usuarios (voluntarios) en su tarea de asistir a los pacientes a la hora recuperar y ejercitar sus capacidades y facultades cognitivas.

La aplicación en cuestión habrá de tener una serie de características que faciliten al usuario llevar a cabo las funciones que realiza de forma fácil y rápida. Algunas de estas características y objetivos que habrá de cumplir la aplicación son:

- Permitir a los voluntarios llevar a cabo el seguimiento puntual del estado de salud física y mental de los tutelados.
- Visualizar información sobre los tutelados, así como también acceder a los seguimientos previamente realizados por el usuario o por otros voluntarios.
- Ayudar a los tutelados en sus actividades de vida cotidiana proponiéndoles la realización de algunos de los juegos y ejercicios de la aplicación.

- Recoger información de los resultados de los juegos para cada tutelado en particular y guardarla en el servidor a disposición de los profesionales sanitarios para que pueda ser utilizada para ver el progreso de los tutelados y también para diseñar las actividades y juegos de manera más efectiva para cada tutelado.

1.6 Fases y medios para la realización del TFG

Fases

Para llevar a cabo los objetivos descritos en el apartado anterior, se ha dividido el TFG en las siguientes fases para su desarrollo:

- **Motivación e investigación del problema*:*** En esta fase se ha investigado el papel y la importancia de las aplicaciones móviles en el ámbito sanitario. Se ha tratado de analizar la viabilidad del uso de las aplicaciones móviles para dar respuesta al problema que trata de resolver este TFG.
- **Documentación y Análisis**: En esta fase se ha consultado la bibliografía y sobre todo los recursos digitales para adquirir los conocimientos necesarios para poder realizar el desarrollo de la aplicación y la escritura del TFG. Además, se diseñan y analizan todos los requisitos funcionales y no funcionales que debe implementar la aplicación a desarrollar.
- **Aprendizaje**: Debido a que no se disponía de conocimientos previos de desarrollo Android, más allá de conocimientos básicos de Java, se ha tenido que aprender los conocimientos técnicos necesarios para el desarrollo de aplicaciones para dispositivos Android, incluyendo la consolidación de Java como lenguaje de programación, el diseño de la arquitectura interna de la aplicación, la interfaz gráfica y la adquisición del conocimiento necesario para utilizar API externas como *OkHttp* para las comunicaciones entre el servidor Web y la aplicación.
- **Desarrollo**: Después de haber adquirido los conocimientos necesarios, se ha procedido a desarrollar la aplicación. Para ello, en un principio se utilizó el entorno de desarrollo Eclipse; sin embargo, más adelante se cambió al entorno de desarrollo Android Studio debido a que se verificó que era más adecuado para el desarrollo de la aplicación. También se desarrolló una API en lenguaje PHP para poder acceder y tratar los datos de la base de datos con la que trabaja la aplicación. Dicha base de datos, se guardó en un servidor MySQL local para facilitar y agilizar el desarrollo.
- **Pruebas y depuración**: En una primera fase de desarrollo, todas las pruebas se hicieron en un entorno virtual ofrecido por Android Studio y la comunicación con la base de datos se hacía en un servidor local. Más tarde cuando, el desarrollo

estuvo finalizado, se procedió a realizar pruebas en un entorno real que consta de varios dispositivos Android y un servidor Web alojado en un servicio de Hosting gratuito, emulando así un funcionamiento completamente real.

- **Conclusiones y Líneas Futuras**: en esta fase se han sacado las conclusiones sobre el trabajo realizado en las fases del desarrollo del presente TFG. Además, se han determinado las posibles mejoras y funciones que se pueden añadir a la aplicación para así sacarle todo el potencial a las ideas que han dado como resultado presente proyecto de desarrollo.

Medios para la realización del TFG

A continuación, se describen los recursos hardware y software utilizados principalmente para el desarrollo de la aplicación y también para la investigación y escritura del TFG.

- Ordenador sobremesa con procesador Intel CORE i7, 16 GB de memoria RAM y sistema operativo Windows 7 de 64 bits.
- Android Studio 1.5.1 (diciembre 2015).
- AppServ 2.5.10 32-bit que incluye: Servidor Apache 2.2.8, PHP 5.2.6, phpMyAdmin 2.10.3 y MySQL 5.0.51b.
- Dispositivos Android entre los que se incluyen: Samsung Galaxy Note 3, Nexus 5, Oneplus 5, Xiaomi Mi9T.
- Servicio de hosting gratuito *Hostinger* para el alojamiento de la base de datos y la API PHP, se ha utilizado este servicio para las pruebas en un entorno real.

1.7 Estructura de la Memoria del TFG

En lo referente a la estructura de la memoria de este TFG, se ha dividido en siete capítulos con el siguiente contenido:

Capítulo 1: "Introducción"

Este primer capítulo estará dedicado a dar una visión resumida del papel que juegan las aplicaciones de las TIC en el campo del *eHealth*, centrándonos especialmente en la tutela de personas mayores y su rehabilitación cognitiva. Además, se expondrán los objetivos, las fases y los medios para la realización del TFG. Finalmente se describirá la estructura básica de la memoria del TFG.

Capítulo 2: "Tecnologías móviles"

En este capítulo se describirán, en varios bloques, las tecnologías móviles más destacadas del mercado, sus características, así como los dispositivos móviles en las que son utilizadas. Todo ello permitirá una mayor comprensión de

las tecnologías móviles lo cual nos permitirá elegir la tecnología móvil adecuada para el desarrollo de la aplicación.

Capítulo 3: "Android"

En este capítulo se describe en profundidad, de manera técnica, la arquitectura y los componentes que integran la plataforma Android. Además del desarrollo para Android, las versiones de Android lanzadas hasta la fecha y los entornos de desarrollo para Android más destacados.

Capítulo 4: "Aplicación socio-sanitaria. Descripción Funcional"

En este capítulo se pretende describir las funcionalidades de la aplicación, así como los tipos de usuarios de la aplicación y sus roles. Todo ello descrito desde un punto de vista funcional.

Capítulo 5: "Aplicación socio-sanitaria. Descripción Técnica"

En este capítulo, se pretende realizar una descripción técnica que explique y muestre los distintos fundamentos técnicos utilizados para desarrollar la aplicación y la comunicación de ésta última con el servidor Web de la fundación. Todo ello explicado desde un punto de vista técnico.

Capítulo 6: "Aplicación socio sanitaria. Manual de usuario"

En este capítulo se pretende elaborar un manual de usuario en el cual se detallan los pasos a seguir por parte del usuario para poder hacer uso de las funcionalidades de la aplicación.

Capítulo 7: "Conclusiones y Perspectivas Futuras"

En este capítulo se recogen las conclusiones sobre el trabajo desarrollado y se determinan las posibles mejoras y funciones que se pueden añadir a la aplicación en futuras versiones de la misma.

Bibliografía

En este apartado se indica las fuentes bibliográficas y recursos digitales consultados para la realización de este TFG.

Capítulo 2 - Tecnologías Móviles

Capítulo 2 – Tecnologías Móviles

En este capítulo se describirán, en varios bloques, las tecnologías móviles más destacadas del mercado, sus características, así como los dispositivos móviles en las que son utilizadas. Todo ello permitirá una mayor comprensión de las tecnologías móviles lo cual nos permitirá elegir la tecnología móvil adecuada para el desarrollo de la aplicación.

2.1. Dispositivos Móviles

Una gran cantidad de dispositivos electrónicos se clasifican actualmente como dispositivos móviles, desde teléfonos hasta tablets, pasando por dispositivos como lectores de RFID (identificación por radio frecuencia). Con tanta tecnología clasificada como móvil, puede resultar complicado determinar cuáles son las características de los dispositivos móviles (Morillo Pozo, 2012).

En la mayoría de los casos, un dispositivo móvil puede definirse con cuatro características que lo diferencian de otros dispositivos que, aunque pudieran parecer similares, carecen de algunas de las características de los verdaderos dispositivos móviles. Estas cuatro características son (Morillo Pozo, 2012):

- **Movilidad**: se entiende por movilidad la cualidad de un dispositivo para ser transportado o movido con frecuencia y facilidad. Por tanto, el concepto de movilidad es una característica básica. Los dispositivos móviles son aquellos que son lo suficientemente pequeños como para ser transportados y utilizados durante su transporte.
- **Tamaño reducido:** se entiende por tamaño reducido la cualidad de un dispositivo móvil de ser fácilmente usado con una o dos manos sin necesidad de ninguna ayuda o soporte externo. El tamaño reducido también permite transportar el dispositivo cómodamente por parte de una persona.
- **Comunicación inalámbrica:** por comunicación inalámbrica se entiende la capacidad que tiene un dispositivo de enviar o recibir datos sin la necesidad de un enlace cableado.
- **Interacción con las personas:** se entiende por interacción el proceso de uso que establece un usuario con un dispositivo. Entre otros factores, en el diseño de la interacción intervienen disciplinas como la usabilidad y la ergonomía.

En la actualidad, los dispositivos móviles disponen de unas prestaciones de hardware y software muy potentes, lo cual hace posible el desarrollo de aplicaciones

con funcionalidades tan avanzadas y tan complejas que era imposible de imaginar en la época de los teléfonos comunes.

Algunas de las características de los dispositivos móviles que los hacen ideales para el desarrollo de aplicaciones son las que siguen (Morillo Pozo, 2012):

- Son aparatos tamaño reducido.
- La mayoría de estos aparatos se pueden transportar en el bolsillo del propietario o en un pequeño bolso.
- Tienen unas capacidades de procesamiento muy altas.
- Tienen conexión permanente o intermitente a Internet.
- Tienen grandes capacidades de memoria (RAM, tarjetas MicroSD, flash, etc.).
- Normalmente se asocian al uso individual de una persona, tanto en posesión como en operación, la cual puede adaptarlos a su gusto.
- Tienen una alta capacidad de interacción mediante la pantalla o el teclado.
- Pantallas de alta resolución y soporte de gráficos 3D y pantallas de nueva generación que ayudan a crean experiencias únicas para el usuario.
- Disponibilidad de una gran variedad de sensores tales como sensores de proximidad, de luz, de temperatura, giroscopio y acelerómetro...
- Pueden llevar un sistema operativo (ej.: Android, iOS, Windows, …).

2.1.1. Smartphones

Un Smartphone es un dispositivo móvil con funcionalidades y capacidades de conectividad y procesamiento mucho más cercanas a las de un ordenador que a las de un teléfono móvil convencional. Los smartphones combinan los conceptos de teléfono móvil y ordenadores portátiles en un único dispositivo. Los smartphones permiten el acceso a la información (por ejemplo, correos electrónicos) e instalar programas, además de usar un teléfono móvil en un único dispositivo. Por ejemplo, un smartphone podría considerarse como un teléfono móvil con funciones de PDA (asistente personal digital) integradas en el dispositivo o viceversa (Morillo Pozo, 2012).

La creación del primer smartphone data de principios de los años 90. IBM diseñó el primer teléfono inteligente llamado Simon. El concepto fue generado desde la década de los años 70 pero el teléfono inteligente fue presentado hasta 1992, en Las Vegas en la antigua feria de la industria informática conocida como COMDEX. Además de realizar y recibir llamadas, Simon también podía enviar correos electrónicos, copias de páginas web, permitía recibir y enviar faxes, crear notas a la mano, tenía una pantalla táctil para marcar los números, también tenía la posibilidad de poseer una aplicación para un calendario, una libreta de direcciones, calculadora, y bloc de notas. IBM también demostró que el teléfono

podía mostrar mapas, generar informes de stocks, y buscar noticias y otras aplicaciones de terceros con ciertas modificaciones. (Certideal, 2018).

Los smartphones, en esencia, han reprogramado cómo interactuamos y experimentamos el mundo. Pero no aparecieron de la noche a la mañana. Hubo muchas tecnologías que progresaron, compitieron, evolucionaron y convergieron, para conocer lo que hoy puede ser un iPhone X o un Samsung Galaxy S10 (Certideal, 2018).

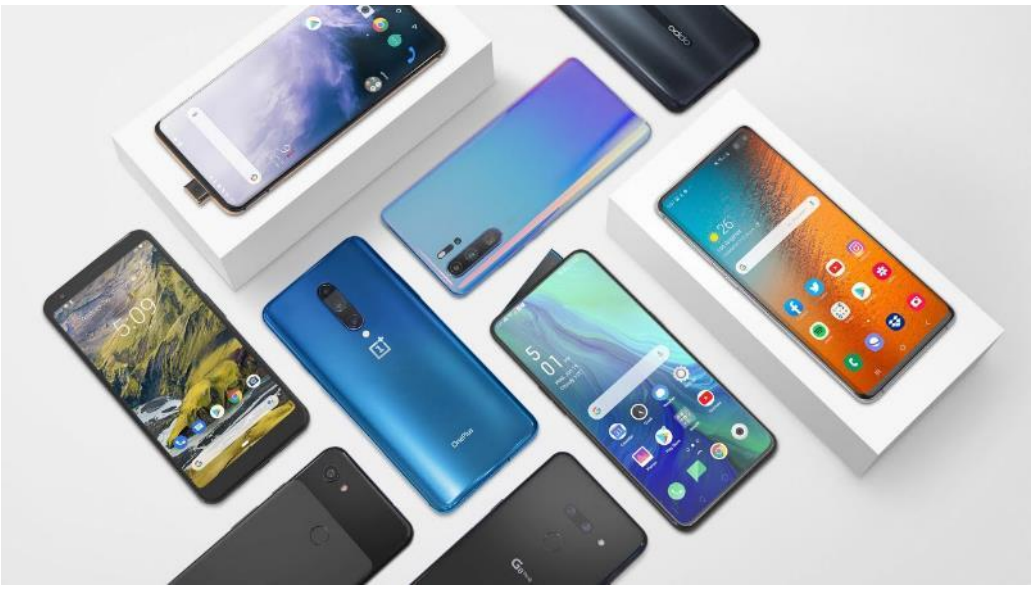

Figura 2 - Algunos de los Smartphones de gama alta de 2019

Estos smartphones se encuentran prácticamente al alcance de todo el mundo, y además, integran sistemas operativos como iOS o Android, que, gracias a su interfaz de usuario que recuerda las disponibles en los ordenadores comunes, permiten que sus múltiples funciones puedan ser utilizadas intuitivamente por cualquiera.

2.1.2. Tablets

Se podría describir una Tablet como un dispositivo que tiene unas prestaciones similares a las de un ordenador portátil pero que se presenta en una sola pieza, sin teclado físico, con un diseño plano, fino y compacto el cual contiene todos los componentes esenciales para su funcionamiento de forma autónoma, todo ello comprimido en una sola pieza aparente que está compuesta por pantalla táctil, CPU, puertos y conectores, unidades de almacenamiento, etc. (Tablet Area, 2010).

Los dispositivos Tablet revolucionan el concepto de movilidad por ser fácilmente portables y permitir estar conectados a Internet de forma permanente y prácticamente en cualquier lugar además de permitirnos ejecutar un sin fin de aplicaciones tanto locales como remotas (Tablet Area, 2010).

Algunas de las características y ventajas de los Tablets (El Androide Libre, 2013):

- **Portabilidad:** no hay duda de que una de las ventajas que tienen las Tablets frente a un ordenador es su portabilidad; su reducido peso y tamaño hacen que este dispositivo sea muy fácil de manejar.
- **Autonomía:** la batería dura, en general, bastante más que en un portátil debido sobre todo al tipo de CPU (normalmente de arquitectura ARM) que llevan las Tablets, suelen ser enérgicamente más eficientes, frente a los que llevan los ordenadores portátiles. Una Tablet puede durar cómodamente todo un día de uso, cosa que con un portátil es difícil de conseguir.
- **Comodidad de lectura:** otro punto a favor es que se pueden almacenar libros electrónicos y se pueden leer en cualquier parte. Esta ventaja gana mucho si se habla de Tablets de 7”, ya que tienen el tamaño ideal para ser sujetadas con una mano.
- **Estación multimedia:** la capacidad multimedia de las tablets es una de las características más destacadas, se pueden ver películas, escuchar música, ver fotos, YouTube, realizar videoconferencias… Es decir, un gran número de posibilidades al alcance de nuestras manos y con la máxima comodidad posible.
- **Variedad:** hay Tablets de varios tipos y de varias gamas en función de nuestras necesidades. Tamaños, capacidades, rendimiento, precios, diseño. El rango de mercado que ofrecen es muy amplio y ese es un punto a su favor.

Figura 3 - Algunas de las Tablets más destacadas

La versatilidad que ofrecen estos dispositivos es lo que los diferencia de los ordenadores, ya que se pueden mover en muchos campos y situaciones en las cuales un PC no es capaz de ofrecer las mismas comodidades.

2.1.3. Wearables

Wearable hace referencia al conjunto de aparatos y dispositivos electrónicos que se incorporan en alguna parte de nuestro cuerpo interactuando continuamente con el usuario y con otros dispositivos con la finalidad de realizar alguna función específica, gafas, relojes inteligentes o smartwatchs, zapatillas de deportes con GPS incorporado y pulseras que monitorizan nuestro estado de salud son ejemplos entre otros de este tipo tecnología que se encuentra cada vez más presente en nuestras vidas (Dispositivos Wereables, 2014).

Los dispositivos Wearables son una nueva generación de dispositivos que pretenden aprovechar las capacidades de la tecnología y llevarlas un paso más allá. Se encuentran presentes en un amplio abanico de sectores que satisfacen nuestras necesidades con la finalidad de aumentar nuestra calidad de vida mejorando por ejemplo la salud de los pacientes, la seguridad de las personas que se exponen a ciertos riesgos en su trabajo diario o el entrenamiento de los deportistas (Dispositivos Wereables, 2014).

Algunos ejemplos de este tipo de dispositivos:

Figura 4 - Dispositivos Wearables

Gadgets de muñeca

Estos complementos digitales almacenan toda la información sobre el estilo de vida del usuario como los kilómetros que camina durante el día, el ritmo cardíaco, los ciclos de sueño, etc.... También permiten interactuar con otros dispositivos electrónicos gracias a la tecnología inalámbrica de estos wearables pudiendo abrir las puertas de una casa, encender el motor de un coche, pagar las compras sin

necesidad de sacar la tarjeta de crédito, cargar y descargar todo tipo de archivos como documentos, imágenes, vídeo... todo ello con un solo gesto de muñeca (Dispositivos Wereables, 2014).

Otros usos de los wearables

La seguridad de los trabajadores se incrementará gracias a la tecnología wearable, a modo de ejemplo actualmente existen cascos de bomberos que monitorizan los niveles de oxígeno y la temperatura que soporta el bombero durante los trabajos extinción de los incendios, además de llevar incorporados un localizador GPS el cual permite conocer en cualquier momento el punto exacto donde se encuentra (Quees, 2015).

El sector textil también ha sucumbido a esta nueva forma de tecnología wearable, prendas infantiles que indican y mandan una señal a al smartphone cuando un bebe tiene fiebre, ropa de deporte que ayuda a realizar los movimientos correctos mientras se aprende a jugar al golf, sudaderas con leds que iluminan en las calles oscuras mientras se hace running a la vez que contabiliza los kilómetros recorridos, el ritmo cardíaco, las calorías quemadas, etc. (Quees, 2015).

Este tipo de gadgets representan, al menos de momento, un nicho de mercado en expansión. Todos dependen de la conexión con un smartphone para funcionar. Si se popularizan, tal vez, en los próximos años, éste ni siquiera tenga que salir del bolsillo para ser útil (Quees, 2015).

2.1.4. Smart TV

La palabra Smart TV deriva de la de teléfono inteligente o Smartphone. Un Smart TV por lo tanto es un televisor inteligente. Inteligente porque son televisores en los que aparte de servir para ver la televisión, con ellos podemos navegar por internet, grabar y reproducir películas, utilizar nuestro correo electrónico, descargar videos de YouTube, etc., así como conectar dispositivos externos a través de sus puertos. Incluso podemos instalar software en nuestro propio televisor y usarlo como si fuera un ordenador (ABC Color, 2013).

Los Smart TV no son precisamente dispositivos móviles, sin embargo, entran dentro de este marco ya que suelen llevar instalados sistemas operativos, tales como Android TV, que han sido desarrollados para dispositivos móviles. Ya existen App Stores (tiendas online con juegos y aplicaciones) para descargarse e instalar apps en nuestro propio Smart TV. Estas tiendas suelen ser propias de

cada marca de televisor, como ocurre con los Smartphone, aunque en general si los televisores incorporan el sistema Android, pueden compartir la tienda de Apps con los Smartphones y Tablets. Todos estos televisores tienen una interface fácil e intuitiva de manejar, que hacen que sean muy fácil su uso y con su propio sistema operativo. Además, las pantallas de estos televisores son de última generación LCD o LED (ABC Color, 2013).

Figura 5 - Interfaz típica de los Smart TV

Algunas de las funciones que se pueden realizar con un Smart TV (ABC Color, 2013):

- Navegar por internet.
- Correo electrónico, redes sociales, comunicaciones en tiempo real, etc.
- Ver y grabar películas, series, documentales, etc.
- Instalar software.
- El cine en casa.
- Entradas para dispositivos de almacenamiento externos.
- Equipo de música.
- Compartir contenidos multimedia.
- Descargar videos, etc.
- Ver Canales de Televisión por internet.

2.2. Las Aplicaciones Móviles

2.2.1. Introducción

Las aplicaciones, o simplemente *apps*, están presentes en los dispositivos móviles desde hace varios años; desde las primeras versiones de sistemas operativos como los de Nokia o BlackBerry, ya entonces se podían encontrar aplicaciones incluidas en dichos sistemas. Los dispositivos móviles de aquella época, contaban con una potencia muy limitada y con pantallas reducidas y la mayoría no táctiles, o si lo eran, solamente eran de tipo resistente y no capacitivo, por tanto, no ofrecían la misma comodidad a la hora de utilizar las aplicaciones en ese tipo de pantallas comparados a los Smartphones más actuales (Cuello y Vittone, 2013).

Una aplicación móvil no deja de ser un software, muy similares a los programas que podemos encontrar en los ordenadores de escritorio. Actualmente encontramos aplicaciones de todo tipo, que cubre casi cualquier necesidad, pero en los primeros teléfonos su uso era más específico y simple, pues estaban enfocadas en mejorar la productividad personal: se trataba de apps simples tales como alarmas, calendarios, calculadoras y clientes de correo(Cuello y Vittone, 2013).

No es hasta la salida del iPhone al mercado cuando comenzaron a surgir nuevos modelos de negocio que hicieron de las aplicaciones algo rentable, tanto para desarrolladores como para los mercados de aplicaciones, como App Store, Google Play Store. A partir de ese momento, se empezaron a lanzar dispositivos móviles cada vez más potente y variados que ofrecerían unas características y unas prestaciones ideales para el desarrollo de las aplicaciones móviles. Al mismo tiempo, también mejoraron las herramientas de las que disponían diseñadores y desarrolladores para la creación y publicación de apps (Cuello y Vittone, 2013).

2.2.2. Posibilidades de desarrollos de aplicaciones móviles

Una de las primeras cuestiones a las que nos enfrentamos al desarrollar Apps, es qué tecnologías y lenguaje de programación usar. Actualmente, podemos encontrar gran variedad de aplicaciones móviles en el mercado, de muchos tipos y en muchos formatos. Las aplicaciones móviles se pueden clasificar según el entorno de su desarrollo y ejecución. Podemos distinguir dos categorías generales de aplicaciones móviles: **Aplicaciones Web** y **Aplicaciones Nativas.**

Aplicaciones *web*

Una aplicación web se basa en HTML, JavaScript o CSS. Puesto que se carga en el servidor web y se ejecuta en el navegador, no requiere ninguna instalación. El espectro de aplicaciones web es muy amplio, yendo desde pequeñas herramientas hasta software de gráficos o juegos de navegador, pasando por las adaptaciones de conocidos programas, como servicios de mensajería instantánea o paquetes de Office (Iconos, 2019).

Figura 6 - Financial Times web app para iPhone

- Se puede acceder a ellas desde cualquier dispositivo móvil con navegador e internet y funcionan en todos ellos.
- No se pueden instalar por el usuario y por tanto no se pueden encontrar en las tiendas de aplicaciones, lo cual supone una desventaja respecto a las apps nativas. Aunque como punto positivos, no es necesario que el usuario las actualice para disponer de la última versión.
- Se pueden integrar fácilmente con los servicios habituales del sitio web, de modo que las actualizaciones y los contenidos por suscripción se vuelven más accesibles y sencillos.

Las páginas Web que se adaptan especialmente a un dispositivo móvil se llaman «web responsivas» y son ejemplo del diseño líquido, ya que se puede pensar en ellas como un contenido que toma la forma del contenedor, mostrando la información según sea necesario. Así, columnas enteras, bloques de texto y gráficos de una web, pueden acomodarse en el espacio de una manera diferente, o incluso desaparecer, de acuerdo a si se entra desde un teléfono, una tablet o un ordenador (Cuello y Vittone, 2013).

Aplicaciones nativas

- **Nativas**: La aplicación nativa está desarrollada y optimizada específicamente para el sistema operativo determinado y la plataforma de desarrollo del fabricante (Android, iOS, etc.). Este tipo de aplicaciones se adapta completamente a las funcionalidades y características del dispositivo obteniendo así una mejor experiencia de uso. Algunos ejemplos de aplicación nativa, serían WhatsApp o Facebook (Raona, 2017).
- **Nativas Hibridas:** Este tipo de aplicación aprovecha al máximo la versatilidad de un desarrollo web y tiene la capacidad de adaptación al dispositivo como una app nativa. Permite utilizar los estándares de desarrollo web (HTML5) y aprovechar las funcionalidades del dispositivo tales como la cámara, el GPS o los contactos. Además, comporta un menor coste que una aplicación nativa y una mejor experiencia de uso que una aplicación web (Raona, 2017).
- **Nativas Generadas:** Son aplicaciones desarrolladas usando herramientas como Xamarin, React Native y la más reciente Flutter. El desarrollo se realiza usando técnicas y lenguajes específicos de la herramienta y luego se genera la App en el lenguaje de la plataforma destino para ser compilada con las herramientas nativas. Son lo que se denomina falsas nativas, ya que sus características se acercan bastante a aquellas de las app nativas. Por esta razón se denominan Generadas (InnovaAge, 2018).

Comparativa de las posibilidades de desarrollo

Dado que el presente TFG se desarrolla bajo la premisa de la creación de una aplicación específicamente para Android, podemos descartar las aplicaciones web como opción de desarrollo. Ahora queda elegir una las opciones nativas mencionadas anteriormente para el desarrollo de la aplicación. Para ello, se ha hecho uso de un análisis de las tres opciones y así poder realizar una comparación que nos permite decantarnos por una de ellas.

- **Tiempos y Costes de desarrollo (InnovaAge, 2018)**

  El hecho de tener que desarrollar una versión de la aplicación para cada plataforma hace que las Apps nativas queden en la peor posición en este punto, triplicando el costo de desarrollo. Por otro lado la simplicidad del HTML + JavaScript, sumado a la capa de abstracción que ofrecen los plugins de Phonegap, haciendo incluso más simple el acceso a funciones nativas como la geolocalización, pushnotifications, acelerómetro, etc.,

que el desarrollo nativo, sumado a los recursos disponibles en la web para esta tecnología, todo mediante un API cien por ciento neutral respecto al sistema operativo, hace que desarrollar una App con esta tecnología tenga un coste más reducido y de menor duración. En el caso de herramientas como Xamarin o Flutter, se requiere un conocimiento por parte del desarrollador de la API de cada plataforma, así como la consideración de estas excepciones durante el desarrollo.

- **Curva de aprendizaje**

  Basándonos en la descripción y las características de cada uno de los tipos de aplicaciones, es evidente que las apps nativas presentan una curva de aprendizaje más lenta frente a las web. Pues, se requiere aprender sobre las distintas plataformas, lenguajes y APIs, etc. La simplicidad de lenguajes como HTML, CSS y JavaScript, sumado a la facilidad que ofrecen los *frameworks* como Phonegap a la hora de exportar aplicaciones web a diferentes plataformas, hace que la mejor alternativa en este caso sean las apps nativas.

- **Calidad de la App**

  Experiencia de usuario (InnovaAge, 2018):

  La experiencia de usuario depende de factores de diseño, usabilidad, interacción, accesibilidad y calidad visual así como de factores como las emociones, construcción y trasmisión de marca, confiabilidad, etc. En este caso, nos limitaremos únicamente al diseño y aspecto visual específico de la plataforma. Con el desarrollo nativo, los controles y transiciones de la plataforma están disponibles en la herramienta de desarrollo por lo que su uso es natural e inmediato. Además de otras funcionalidades, como por ejemplo, la disponibilidad de librerías (MPAndroidChart, AndroidPlot, etc.) para la inclusión de gráficas en la app desarrollada. En la soluciones híbridas con HTML5, con excepción de los controles que nativamente ofrece el WebView, el resto deben ser simulados usando HTML5 y CSS3. Las Apps generadas, usan los controles nativos, aunque el aspecto visual general de la App no siempre es óptimo, sobre todo cuando se sale de lo previsto, ya que el diseño se realiza con las reglas de la herramienta que luego deben ser interpretadas y convertidas a nativo.

  En conclusión, las apps nativas son las que mejor experiencia de usuario ofrece con respecto a las demás alternativas.

Rendimiento:

El rendimiento final de la App dependerá de varios factores como son: la arquitectura general de la solución, su ingeniería, la inteligencia y resolución de sus componentes por parte del servidor, el uso inteligente del almacenamiento local, cache, la capacidad de los servicios web, etc. (InnovaAge, 2018).

De esta forma, cualquiera de las alternativas analizadas se despliega de forma aceptable, pues en general todas cumplen, en cierta manera con los factores necesarios para ofrecer un rendimiento aceptable de cara al usuario final, haciendo que en algunos casos, la diferencia en rendimiento de aplicaciones sencillas sea poco notable. Sin embargo, si se trata de desarrollar una app compleja y potente o un juego con gran carga gráfica, lógicamente la alternativa más adecuada sería las nativas puesto que son las que mayor rendimiento ofrecen al permitir el acceso directo y completo a las capacidades del dispositivo.

CONCLUSIÓN

Tal y como se ha visto en la comparativa anterior y el resumen de la figura 7, no existe una opción que destaque muy por encima de las otras.

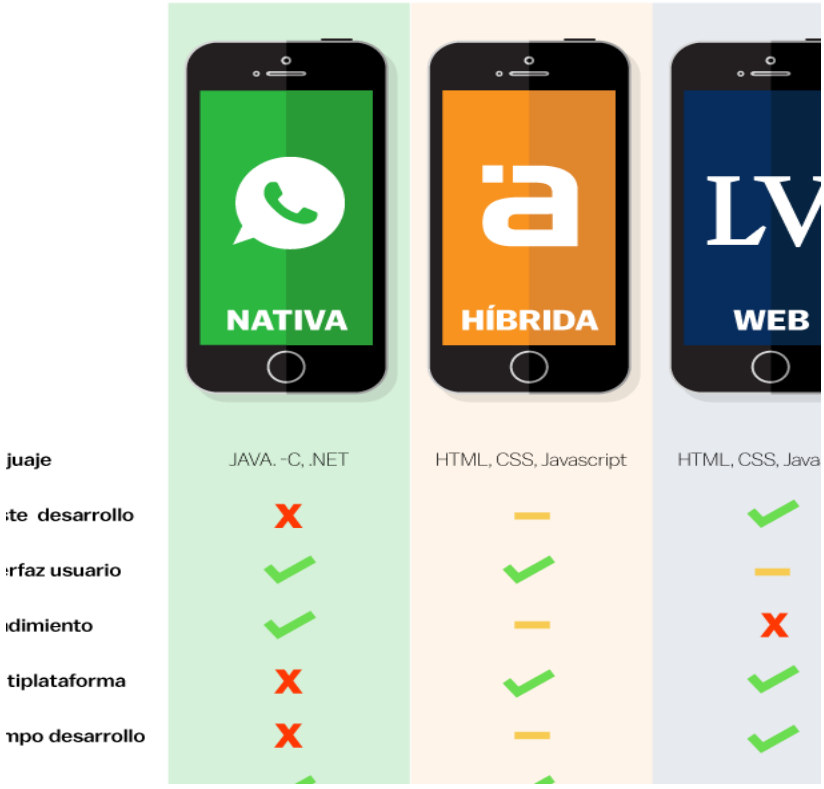

Figura 7 - Comparativa de tipos de apps (Raona, 2017)

Sin embargo, debido al tipo de aplicación que se desarrolla en este TFG y a los objetivos que se pretende que tenga la aplicación, entre los cuales se encuentran el ofrecer el mejor rendimiento y la mejor experiencia de usuario posibles, es lógico que la opción más destacable es el **desarrollo nativo**. Otro de los aspectos a destacar y que ha pesado en la decisión de elegir el desarrollo nativo es la disponibilidad de librerías como **MPAndroidChart** para la inclusión de gráficas en la app desarrollada. Este último aspecto también está disponible en la generadas pero debido a que el desarrollo de gráficas en esta alternativa aún es muy reciente y no ofrece muchas posibilidades para la inclusión de gráficas.

2.2.3. Mercado de las Apps

En la actualidad, según el informe de Statista (Statista, 2019), hay más de 5 millones de aplicaciones disponibles divididos entre las principales tiendas de aplicaciones como Google Play, Apple Store, Windows, Amazon o BlackBerry. Y no sólo es cuestión de numero de apps, sino que también entran en juego los ingresos que generan dichas apps. Solamente iOS de Apple supone dos tercios de los ingresos de estas apps para smartphone y tablet. Pese a que Android supera en número de aplicaciones a su antagónico iOS, los usuarios de éste destacan por ser quienes más se gastan a la hora de adquirir este producto. Además, todo apunta a que iOS mantendrá su liderazgo en este sentido.

Las aplicaciones Móviles se suelen relacionar principalmente con los juegos online o la mensajería. Sin embargo, las nuevas tendencias marcan un creciente interés por tecnologías y usos muy diferentes, como: Aplicaciones Corporativas para Marcas, **Salud**, Deporte, Educación, Transporte o de Servicios, entre otros. Pero definitivamente el uso de las apps va en aumento y su crecimiento es verdaderamente significativo así como lo muestra el Informe Ditrendia: Mobile en España y el Mundo 2018. Ya que se cumplen 10 años desde que llegó al mercado la primera aplicación móvil y desde entonces las hemos ido utilizando cada vez más para nuestras tareas diarias hasta suponer en 2017 más del 80% del tiempo que dedicamos al uso del móvil en el mundo. En 2017 se descargaron 178,1 miles de millones de aplicaciones móviles y se espera que en 2022 la cifra ascienda a 258,2 miles de millones de descargas. De hecho un smartphone tiene de media 80 aplicaciones instaladas, de las cuales solo se usan mensualmente la mitad (Ditrendia, 2018).

Figura 8 - Evolución de las descargas de aplicaciones a nivel mundial (miles de millones) (Ditrendia, 2018)

Existen infinitos tipos de aplicaciones disponibles pero finalmente los usuarios dedican la mayor parte de su tiempo (dos tercios) a cuatro categorías. Las redes sociales son las que mayor promedio de participación tienen en todos los mercados, siendo Facebook quien lidera el ranking de aplicaciones más utilizadas en el mundo seguido de WhatsApp (Ditrendia, 2018).

Ranking mundial de aplicaciones móviles por usuarios activos mensualmente

	NOMBRE APP	DESARROLLADOR / COMPAÑÍA
01	FACEBOOK	FACEBOOK
02	WHATSAPP MESSENGER	FACEBOOK
03	WECHAT	TENCENT
04	FACEBOOK MESSENGER	FACEBOOK
05	QQ	TENCENT
06	INSTAGRAM	FACEBOOK
07	TAOBAO	AUBABA GROUP
08	ALIPAY	ANT FINANCIAL SERVICES GROUP
09	WIFI MASTER KEY	SHANGAI LANTERN NETWORK
10	TENCENT VÍDEO	TENCEN

Figura 9 - Ranking mundial de aplicaciones por usuarios activos mensualmente (Ditrendia, 2018)

2.3 Sistemas Operativos Móviles

Un Sistema Operativo es un Programa (software) que se inicia al encender el ordenador o el teléfono móvil y se encarga de gestionar todos los recursos del sistema informático, tanto de hardware (partes físicas, disco duro, almacenamiento, pantalla, teclado, etc.) como el software (programas e instrucciones) permitiendo así la comunicación entre el usuario y el ordenador. En definitiva, controlan el ordenador, el teléfono móvil o la tablet y nos permite comunicarnos con ellos de forma sencilla. Los sistemas operativos móviles para los Smartphone son bastantes más simples que los de los PC y están más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos (Areatecnologia, 2015).

2.3.1 Android

Android es un sistema operativo y una plataforma software, basado en Linux para teléfonos móviles. Además, también usan este sistema operativo, Tablets y Netbooks. Android permite programar en un entorno de trabajo de Java, aplicaciones sobre una máquina virtual Dalvik (en versiones de Android inferiores 5.0) y ART a partir de la versión 5.0 de Android. Además, lo que lo diferencia de otros sistemas operativos, es que cualquier persona que sepa programar puede crear nuevas aplicaciones, widgets, o incluso, modificar el propio sistema operativo, dado que Android es de código libre, por lo que sabiendo programar en lenguaje Java, es fácil comenzar a programar en esta plataforma (Sanz, Saucedo y Torralbo, 2012).

Historia de Android

Los dispositivos móviles constituyen cada vez más una realidad que ofrece al usuario, en un mismo y reducido aparato, funciones de comunicación y procesamiento de datos que van mucho más allá de las simples llamadas telefónicas o la ejecución de aplicaciones básicas. El gigante de Internet Google presentó un sistema operativo para este tipo de dispositivos, buscando así ofrecer un sistema operativo que permita a los fabricantes de dispositivos móviles poder competir con Apple y su sistema iOS ya que en ese momento, las únicas alternativas de las que disponían eran los ya desaparecidos Symbian y Windows Mobile, que no eran los suficientemente potentes como para competir con las características que ofrecía iOS a nivel de experiencia de usuario y rendimiento. Android es un sistema operativo del móvil que utiliza una versión modificada del kernel de Linux. Fue desarrollado por Android Inc., empresa que en 2005 fue comprada por Google, aunque no fue hasta 2008 cuando se popularizó, gracias a la unión al proyecto de Open Handset Alliance, un consorcio formado por 48 empresas de desarrollo hardware, software y telecomunicaciones, que decidieron promocionar el software libre. Pero ha sido Google quien ha publicado la mayor parte del código fuente

del sistema operativo, gracias al software Apache, que es una fundación que da soporte a proyectos software de código abierto. Dado que Android está basado en el núcleo de Linux, tiene acceso a sus recursos, pudiendo gestionarlo, gracias a que se encuentra en una capa por encima del Kernel, accediendo así a recursos como los controladores de pantalla, cámara, memoria flash… (Sanz et al., 2012).

Las capas que conforman la arquitectura básica de Android son: la que forma parte del propio Kernel de Linux, donde Android puede acceder a diferentes controladores, las librerías creadas para el desarrollo de aplicaciones Android, la siguiente capa que organiza los diferentes administradores de recursos, y, por último, la capa de las aplicaciones a las que tiene acceso. Esta arquitectura de Android se explicará más en profundidad en el **capítulo 3**.

2.3.2 iOS

iOS, es un sistema operativo propiedad de Apple orientado a sus dispositivos móviles táctiles como el iPhone, el iPod touch y el iPad. La idea de sus creadores fue la de proveer a sus dispositivos móviles con un sistema operativo adaptado y optimizado, que fuera capaz de funcionar sin la necesidad de un teclado físico y que permitiera, no solamente realizar llamadas telefónicas como lo hacían los teléfonos móviles de ese entonces, que aún no eran smartphones, sino también entre otras cosas, navegar por internet mediante Safari y almacenar una gran cantidad de canciones al mejor estilo iPod. (Swiftlatino, 2017).

Historia

El lanzamiento del iPhone OS tuvo lugar el 29 de junio de 2007. La primera versión de iOS no contaba con la mayoría de las funcionalidades que hoy vemos como indispensables, como la multitarea, el App Store, el kit de desarrollo SDK, etcétera. Su interfaz era sencilla y fluida, contaba con una cantidad pequeña de programas desarrollados por Apple o de terceros en conjunto con la empresa. El diseño se orientaba al skeumorfismo, esto implica que los íconos eran dibujados de manera tal de parecerse lo máximo posible al objeto que representa. La novedad que presentaba iOS fue que en lugar de usar botones para interactuar con la interfaz y las aplicaciones, se usarían gestos, como deslices y toques. En la actualidad, este sistema operativo ha avanzado mucho y junto a Android son los dos sistemas más utilizados en smartphones a nivel global (Swiftlatino, 2017).

Arquitectura y características de iOS

El kernel del sistema operativo, al igual que macOS, su versión para portátiles, está basado en UNIX. Si bien UNIX es más conocido por ser el kernel de las distribuciones de Linux, eso no implica que cualquier sistema que lo utilice deba ser de código abierto. De hecho, tanto iOS como los demás sistemas usados en los dispositivos de Apple son propietarios de la empresa y solo se usan en el mercado en los dispositivos para los cuales fueron creados. Esto supone una ventaja para la compañía en términos de velocidad y sinergia entre software y hardware, ya que el código existente se encuentra optimizado para el hardware en el que corre y no existe código extra. Esta es la diferencia más importante con Android ya que la visión de Google es diferente. Google busca crear un sistema operativo que pueda usarse en la mayor cantidad posible de dispositivos distintos a lo largo del mundo, con una variedad prácticamente infinita de especificaciones de hardware, lo que hace que se deba incluir en el código del sistema muchas líneas extra para abarcar todas las posibilidades existentes (Swiftlatino, 2017).

La arquitectura de iOS está constituida por una serie de capas o niveles de abstracción, las cuales sirven como intermediario entre el hardware del dispositivo y las aplicaciones instaladas en él. En general, iOS se compone de cuatro capas (Intellipaat, 2019):

- **Core OS Layer**: engloba las funcionalidades de bajo nivel sobre las que están construidas la mayoría de *frameworks* del sistema: ficheros del sistema, manejo de memoria, seguridad, drivers del dispositivo, etc.
- **Core Services Layer: c**ontiene los servicios fundamentales del sistema que usan todas las aplicaciones.
- **Media Layer:** provee los servicios de gráficos y multimedia a la capa superior.
- **Cocoa Touch Layer**: es la capa más importante y provee de un conjunto de Frameworks para desarrollar aplicaciones como son EventKit framework, GameKit framework, MapKit Framework y UIKit Framework entre otros

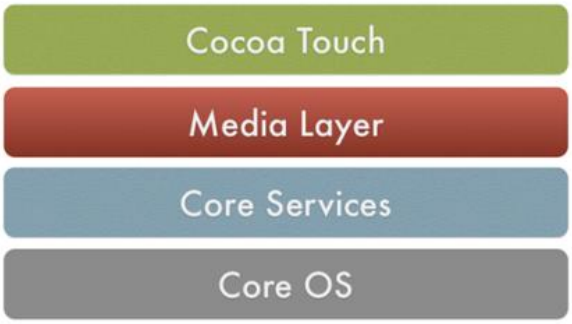

Figura 10 - Capas de la arquitectura de iOS (Intellipaat, 2019)

Algunas de las funcionalidades más destacables de iOS son (Swiftlatino, 2017):

- **Desbloqueo de pantalla**: En las versiones iniciales se contaba con un deslizador que desbloqueaba la pantalla y pedía la contraseña en caso de haberse configurado así. Esto supuso una novedad respecto a los teléfonos aquel entonces que tenían teclado físico y se desbloqueaban usando alguna

combinación de teclas. En el año 2013, en la versión 7 de iOS, se introduce el touch ID como método de desbloqueo de pantalla mediante huellas dactilares y más tarde, en la versión 11 de iOS, se introduce el Face ID, el cual desbloquea el dispositivo mediante el reconocimiento facial del usuario.

- **Diseño minimalista:** iOS muestra un listado de apps que el usuario puede seleccionar, y que además se pueden agrupar en carpetas. En la parte inferior se encuentra el Dock que corresponde a una selección de 4 aplicaciones como máximo y que aparecen fijos en ese lugar, independientemente de que el usuario se deslice hacia los listados laterales de apps. En los primeros años de vida del sistema se utilizaba un diseño orientado al skeumorfismo, así por ejemplo, la aplicación para grabar voces diseñada por Apple tenía una apariencia exacta a un micrófono. En los años siguientes, se abandonó ese estilo pasando a uno más minimalista y sencillo, lo que hace que iOS represente de una mejor manera el ADN de Apple.
- **App Store:** desde la versión 2 del sistema operativo iOS, Apple permite a los desarrolladores de software a crear aplicaciones que aprovechen las funcionalidades del sistema y sus recursos y venderlas o distribuirlas a través de esta tienda de apps. Los usuarios, por su parte, son libres de descargar aquellas aplicaciones que más consideren convenientes o bien usar las ya provistas por el sistema.
- **Siri**: a partir de la versión 5 de iOS, se introduce el asistente virtual llamado Siri. Su objetivo fue el de proveer el sistema con una funcionalidad en la que se utiliza el procesamiento de lenguaje natural para poder interactuar con el usuario y así ofrecer una experiencia interactiva parecida a la humana.

iPadOS

iPadOS sistema operativo con nuevo nombre para reflejar la experiencia característica del iPad. iPadOS parte de la tecnología de iOS y añade nuevas funciones y prestaciones intuitivas que aprovechan la gran pantalla y la versatilidad del iPad. iPadOS presenta nuevas formas de usar apps en varias ventanas, una pantalla de inicio rediseñada para mostrar más información y novedades para usar el Apple Pencil de manera natural. iPadOS también incorpora las fantásticas novedades de iOS 13 para que el iPad sea la opción perfecta para consumidores y profesionales (Apple, 2019).

Figura 11 - Pantalla de inicio de iPadOS

Según Craig Federighi, vicepresidente sénior de Ingeniería del Software de Apple:

> *"El iPad ha transformado nuestra forma de trabajar y de expresar nuestra creatividad, y ahora con iPadOS hemos ido un paso más allá con interesantes funciones que aprovechan la gran pantalla y la versatilidad del dispositivo"*
>
> *"iPadOS estrena prestaciones fantásticas como una nueva pantalla de inicio con widgets, prestaciones multitarea más potentes y herramientas con las que usar el Apple Pencil resulta aún más natural."*

Algunas de las características más importantes de iPadOS son (Apple, 2019)**:**

- **Nueva pantalla de inicio:** la pantalla de inicio estrena un formato que muestra más apps por página. La vista Hoy puede añadirse a la pantalla de inicio para acceder rápidamente a los widgets que muestran información al vuelo con noticias, meteorología, calendario, eventos, consejos y mucho más.
- **Más posibilidades con Split View y Slide Over:** la actualización de Split View permite trabajar de forma simultánea con varios archivos y documentos de una misma app, y Slide Over permite ver y cambiar rápidamente entre apps.
- **Más formas de usar el Apple Pencil:** con iPadOS, el Apple Pencil se integra todavía más en el iPad. Los usuarios pueden anotar y enviar páginas web, documentos o emails en el iPad con solo deslizar el Apple Pencil desde la esquina de la pantalla
- **Más potencia para la app Archivos:** la app Archivos es un lugar centralizado para abrir y gestionar rápidamente los documentos, y la experiencia mejora aún más con iPadOS y la posibilidad de compartir carpetas en iCloud Drive.

- **Navegar con Safari como en un ordenador:** con iPadOS, las páginas web se ven como en un ordenador, con la versión de escritorio ajustada a la escala de la pantalla del iPad y optimizada para la interacción táctil.
- **Edición de texto mejorada:** la edición de texto en el iPad da un gran salto con iPadOS. Todo es más ágil y sencillo: señalar incluso con más precisión y rapidez, seleccionar texto deslizando o cortar, copiar, pegar y deshacer con los nuevos gestos.
- **Prestaciones adicionales de iPadOS:** se añade el modo oscuro para hacer que la interfaz de usuario de las aplicaciones sea más cómoda a la vista del usuario cuando haya poca luz en el entorno del usuario. También se añaden nuevas fuentes personalizadas, teclado flotante y algunas mejoras de rendimiento.

2.3.3 Windows

Windows 10 Mobile

Windows 10 Mobile es un sistema operativo móvil desarrollado por Microsoft y diseñado para teléfonos inteligentes y tabletas. Sin embargo, el primer sistema operativo móvil de Microsoft fue Windows Mobile (Pocket PC en sus comienzos) allá por el año 2000 y presentó varias versiones hasta la 6.5 liberada en el 2009. En el año 2010 Windows Phone sustituyó a Windows Mobile, con una serie de mejoras entre las que destacaba una interfaz totalmente renovada. La última actualización de Windows Phone fue la 8.1 en el 2014, dejando paso a Windows 10 Mobile, que salió al mercado en febrero de 2015 y estaba más enfocada a todo tipo de plataformas (smartphones, tablets y pc). Su objetivo es principalmente llevar la integración y unificación con su homólogo de PC Windows 10, y proporcionar una plataforma para los teléfonos inteligentes, y pequeñas tabletas (Babel, 2016).

Debido a la baja demanda de dispositivos móviles que utilizan Windows 10 Mobile y la poca implicación de los desarrolladores de apps en cuanto al desarrollo nativo específicamente para este sistema, Microsoft ha decidido finalizar el soporte para este sistema operativo. Windows 10 Mobile dejaría de recibir actualizaciones el 10 de diciembre de 2019 y desde la propia Microsoft recomiendan se recomienda a los clientes que pasen a un dispositivo Android o iOS, para los que Microsoft lleva tiempo desarrollando aplicaciones nativas, donde destacan sus aplicaciones ofimáticas (Microsoft, 2019a).

Plataforma universal de Windows (UWP)

La finalización del soporte y desarrollo a lo largo de los años de Windows phone y más recientemente de Windows 10 Mobile, ha dado paso a una estrategia por parte de Microsoft más unificadora en forma de una plataforma llamada UWP (Universal Windows Platform) y su propósito es el de permitir ejecutar la misma aplicación en cualquier dispositivo con Windows 10, desde teléfonos hasta equipos de escritorio. Estas aplicaciones se desarrollan con Visual Studio y las herramientas de desarrollo de aplicaciones Windows universales (Microsoft, 2019b).

Figura 12 - Aplicaciones universales de UWP para todo tipo de dispositivos (Microsoft, 2019b)

La idea básica es que se ejecute una aplicación en un teléfono con Windows 10, un equipo de escritorio con Windows 10 o una consola Xbox a través del mismo paquete de aplicaciones. Con la introducción del núcleo unificado de Windows 10, un mismo paquete de la aplicación puede ejecutarse en todas las plataformas. Varias plataformas tienen SDK de extensión que se puede agregar a la aplicación para aprovechar los características específicas de cada plataforma. Windows se ha refactorizado para que tenga un núcleo común para todas las plataformas Windows 10. Hay un código fuente común, un kernel de Windows común, una pila común de E/S de archivos y un modelo de aplicación común. Para la interfaz de usuario, solo hay un marco de interfaz de usuario XAML y un marco de interfaz de usuario HTML. Esto facilita al desarrollador poder crear una aplicación que pueda ejecutarse en distintos dispositivos con Windows 10 (Microsoft, 2019b).

Características principales de UWP (Microsoft, 2019c).

- Segura: Las aplicaciones para UWP declaran cuáles son los recursos y datos de dispositivos a los que acceden. El usuario debe autorizar ese acceso.
- Se puede usar una API común en todos los dispositivos que ejecutan Windows 10.
- Se pueden usar funcionalidades específicas del dispositivo y adaptar la interfaz de usuario para los tamaños de pantalla de otro dispositivo, distintas resoluciones.

- Una store de aplicaciones unificada hace que la aplicación esté disponible en dispositivos con Windows 10 como PC, tabletas, Xbox, HoloLens, Surface Hub y dispositivos de Internet de las cosas (IoT).
- Las apps pueden instalarse y desinstalarse sin riesgo para el dispositivo ni degradación en su rendimiento
- Atractiva: usa iconos dinámicos, notificaciones push y actividades del usuario que interactúan con la línea de tiempo de Windows y la función Continuar donde lo dejé de Cortana para atraer a los usuarios.
- Programable en C#, C++, Visual Basic y JavaScript. Para la interfaz de usuario, usa XAML, HTML o DirectX.

2.4 Comparación de los Sistemas Operativos Móviles

En este apartado se va a comparar las características de las dos principales plataformas móviles disponibles en la actualidad además del ya abandonado Windows Mobile. Dada la gran cantidad de datos que se indican, se ha hecho uso de una tabla (Tabla 1) para representar la información. De esta forma resulta más sencillo comparar las distintas plataformas.

	Android	IOS	Windows (UWP)
Año de lanzamiento	*2008*	*2008*	*2015*
Versión actual	*Android 9.0*	*IOS 12*	*Windows 10*
Lenguajes de programación	*C, C++, JAVA*	*C, C++, Objective-C*	*C, C++*
Licencia de software	*Software libre y abierto*	*Propietaria*	*Propietaria*
Soporte Flash	*Sí*	*No*	*Sí*
HTML5	*Sí*	*Sí*	*Sí*
Tienda de aplicaciones	*Google Play*	*App Store*	*Microsoft Store*
Coste de publicación de APPS	*25$ pago único*	*$99 al año*	*$99 al año*
Plataforma de desarrollo	*Windows, Mac, Linux*	*Mac*	*Windows*
Interfaz personalizare	*Sí*	*No*	*Sí*

Actualizaciones automáticas del S.O.	*Depende del fabricante*	*Sí*	*Sí*
Soporte memoria externa	*Sí*	*No*	*Sí*
Fabricante único	*No*	*Sí*	*No*
Variedad de dispositivos	*Muy alta*	*Muy baja*	*Baja*
Aplicaciones nativas	*Sí*	*Sí*	*Sí*

Tabla 1 - Comparativa de las principales plataformas móviles

Uno de los puntos de comparación más interesante y que indica la popularidad y la extensión de cada una de las plataformas es la cuota de mercado. Tal y como se puede ver en la Figura 13 (Richter, 2019), Android es el Sistema operativos móvil más utilizada seguido por iOS y Windows.

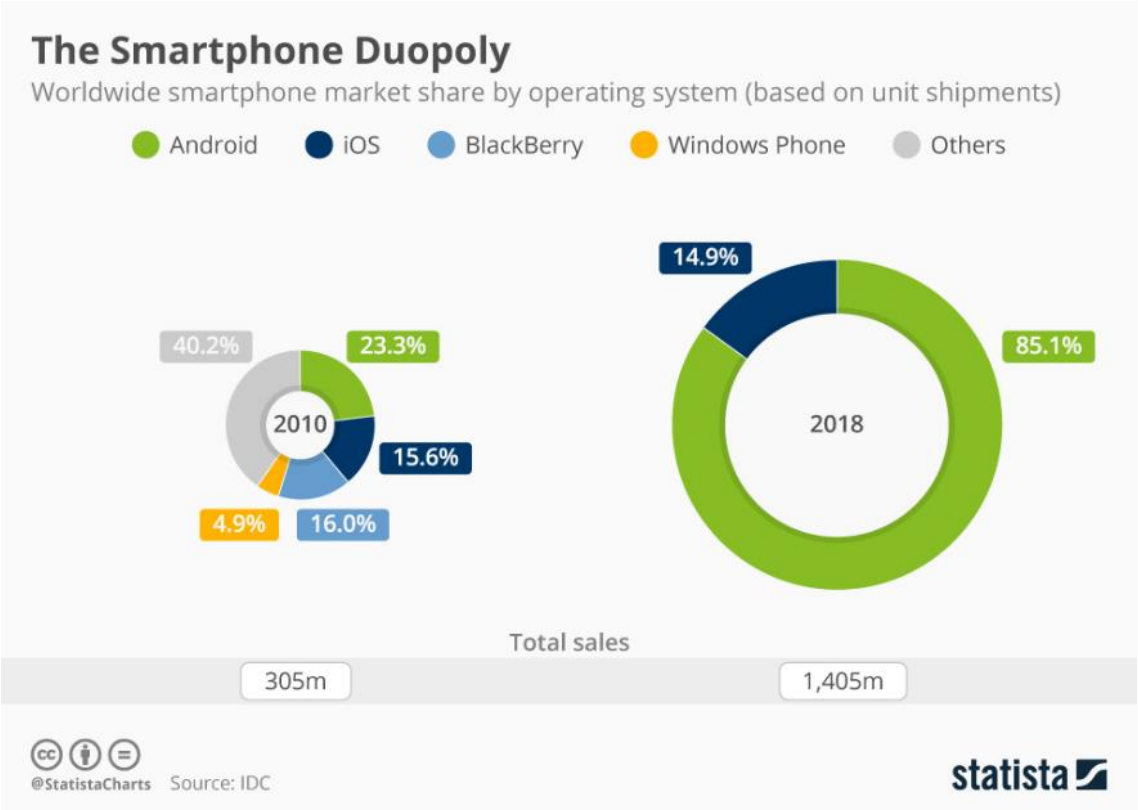

Figura 13- Cuota de mercado de las plataformas móviles (Richter, 2019)

Además, existen otros aspectos que diferencian una plataforma de la otra, a continuación, veremos las ventajas y desventajas de estos tres sistemas:

Android

Ventajas

Algunas de las ventajas de Android son (Evelyn Porras, 2012a):

- Soporta los teléfonos más avanzados: dada la posibilidad de que Android pueda instalarse prácticamente en todo tipo de dispositivos, sean móviles, portátiles e incluso microondas, hace que Android siempre esté presente en los terminales más potentes del mercado siendo una apuesta importante por fabricantes y operadoras por la posibilidad de que independientemente del dispositivo, Android podrá adaptarse a la perfección a todo tipo de necesidades.
- Código abierto: el hecho de que Android esté liberado con licencia Apache y código abierto lo convierte en un sistema operativo totalmente libre para que un desarrollador pueda modificar su código y mejorarlo. A través de esas mejoras puede publicar el nuevo código y con el ayudar a mejorar el sistema operativo para futuras versiones sin depender de fabricantes u operadoras. Al ser código abierto garantiza que, en caso de haber un bug o error, sea detectado y reparado con mayor presteza al no existir ninguna traba legal para indagar en su interior ni depender de nadie para pedir autorización a su cambio.
- Libertad: Android da completa libertad al propietario de un dispositivo de instalar lo que quiera, no limitando la libertad del usuario ni imponiendo software propietario para poder instalar música, archivos, documentos directamente desde el cable USB como si fuera un disco externo La misma libertad tienen los desarrolladores o empresas pudiendo realizar aplicaciones o complementos como Flash, Opera o cualquier otro software sin tener que pedir permiso a nadie para ofrecerlo a los usuarios que libremente podrán instalarlo.
- Sin fronteras: El desarrollo de Android no está determinado por operadoras, fabricantes o proveedores. Android es libertad en todos los aspectos permitiendo que todos puedan disfrutarlo. La libertad del código de Android ha hecho que en poco tiempo se implante en multitud de dispositivos electrónicos, desde móviles hasta ordenadores portátiles, netbooks, microondas, lavadoras, marcos digitales, navegadores GPS, relojes e incluso en navegadores de abordo de coches. Esto convierte a Android en un sistema operativo multifunción y completamente escalable que garantizará su crecimiento y expansión, así como ayudará a fabricantes a tener un motor inteligente para sus fabricaciones.

- Comunidad: Android cuenta con la comunidad más grande mundial de desarrolladores y el mayor movimiento de estos con multitud de eventos, concursos, competiciones y reuniones, así como múltiples vías de comunicación como foros y chats oficiales para fomentar la participación y la colaboración para encontrar mejoras e ideas para futuras versiones. Por otro lado, las modificaciones o mejoras no dependen de un limitado equipo de desarrolladores de una empresa sino contarán con el apoyo, respaldo y participación de todos los desarrolladores del mundo.

- Coste y gustos: Debido a que Android se puede instalar en diferentes dispositivos, los usuarios pueden disfrutar de una amplia gama de dispositivos de diferentes precios y gamas para determinados poderes adquisitivos sin tener que forzar o limitar un sistema operativo o terminal a determinadas clases dando la opción de que toda persona pueda adquirir el terminal que más le guste, así como de diferentes características o costes o incluso fabricantes.

- Vistas personalizadas: Al ser abierto y libre, Android es completamente personalizable tanto por usuarios instalando fondos de pantalla, animaciones, widgets, skins o temas como para fabricantes con la posibilidad de crear sus propias capas permitiendo poder personalizar los teléfonos de la mejor manera posible y dando a elegir al usuario la interfaz más adecuada para su gusto evitando imponer un determinado estilo o interfaz.

- Multitarea: Android tiene capacidad de hacer múltiples tareas a la vez con su sistema de multitarea inteligente es capaz de gestionar varias aplicaciones abiertas a la vez dejando en suspensión aquellas que no se utilicen y cerrarlas en caso de resultar ya inútiles para evitar un consumo de memoria.

Desventajas

A pesar de presentar muchas ventajas también tiene una serie de desventajas como (Evelyn Porras, 2012a):

- El hecho de tener varias aplicaciones abiertas con el sistema multitarea hace que el consumo de la batería aumente. Además, el hecho de que Android este desarrollado de manera que se necesita usar la máquina virtual de java como intermediaria entre las apps y el hardware, hace que el rendimiento y el consumo de batería sea mayor comparado con iOS y Windows.

- Poco intuitivo: Para la mayoría el sistema operativo es muy complicado. Por ejemplo, se vuelve complicado configurar el teléfono, esto te puede llevar mucho tiempo, y esto es generado por la interfaz de Android.

- La necesidad tediosa de instalar aplicaciones externas para solucionar problemas de uso normal: efectivamente el equipo funciona con solo encenderlo, pero sí de facilitar el uso se trata, se hace necesario instalar aplicaciones parte de las que vengan predefinidas.
- No es un sistema operativo completo, Android es poco intuitivo al principio y que hay que configurar una buena cantidad de aplicaciones para que funcione correctamente.
- Fragmentación y Actualizaciones: Android ha sido criticado múltiples veces a causa de la inmensa variedad de terminales que soportan el sistema operativo. Este hecho no es en sí mismo negativo, pero la diferencia de hardware existente entre sus terminales y la influencia de las operadoras móviles y de los fabricantes en el ritmo de llegada de las actualizaciones de Android a los terminales, hacen que exista una diversidad enorme de versiones del sistema operativo utilizados por los clientes. En la Figura 14 se pueden apreciar un gráfico con las distintas versiones de *Android* que han accedido a *Google Play* en los 7 días previos al final del mes de mayo del 2019 (Android Developers, 2019a).

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

Figura 14 - Fragmentación en Android (Android Developers, 2019a).

iOS

***Ventajas*:**

El sistema operativo iOS tiene muchas ventajas, algunas de las más importantes son (Evelyn Porras, 2012b):

- Presenta interfaz gráfica es muy agradable y muy intuitiva. Con buen diseño, funcionalidad, facilidad de uso y una variedad de aplicaciones y juegos.
- Tiene perfecta integración con servicios en la nube y equipos PC como Mac, por lo que el correo, redes sociales, fotos, imágenes, videos y demás esta sincronizado.
- Los dispositivos presentan gran capacidad de almacenamiento interno.
- En reciente versiones del sistema, se ha incluido la capacidad de multitarea, para tener varias aplicaciones en uso sin necesidad de cerrarlas y ejecutarlas cada vez que se necesiten.
- Las notificaciones son un gran avance, que permite tener saber que pasa en las aplicaciones que se encuentran en segundo plano, como las redes sociales, mails o cualquier otro cambio en las notificaciones.
- Facilidad de uso de la cámara y edición de fotografías, tiene la velocidad de captura de fotos superiores a la competencia. Una vez sacada la fotografía puede editarse con la aplicación incluida en el dispositivo.
- Integración con redes sociales, como Twitter que posibilita iniciar sesión una sola vez y luego se puede hacer tweets desde cada aplicación compatible con un solo toque, es muy sencillo, además se puede compartir fotos y videos con solo compartir el contenido.
- Integración con iMessage, una app de Apple que es un servicio de mensajería que sirve para comunicarse entre dispositivos de Apple usando WI-FI o conexión de datos. Permite el envío mensajes normales hasta de multimedia. Esta aplicación de Apple funciona muy rápido e integra todo en un mismo lugar, siendo cómoda de usar.
- La seguridad, este sistema está diseñado en gran parte para proteger la información personal del usuario.

Desventajas

A pesar de tener un sinfín de grandiosas características el sistema operativo iOS tiene varias desventajas (Evelyn Porras, 2012b):

- Al ser propietario de Apple, solo puede instalarse en dispositivos de la misma empresa que sean compatibles con tecnología táctil.
- Las aplicaciones que se instalan muchas no son gratis y el hardware que soporte el sistema operativo tiene un elevado precio.
- No cuenta con servicios de radio FM, por lo que solo puede reproducir audio almacenado en el dispositivo.
- El diseño del iOS es cerrado, es quiere decir menos posibilidades de cambiar la forma de funcionar del teléfono y hay un control rígido de las aplicaciones publicadas para iOS.

Windows UWP

***Ventajas*:**

Entre las ventajas más destacadas encontramos (Medium, 2018):

- Un store de apps para todas las plataformas, se puede comprar una aplicación en un dispositivo móvil por ejemplo y se puede usar en cualquier otro tipo de dispositivo con Windows s10, ya sea un PC, una tablet, etc.
- Sincronización entre todos los dispositivos que soporte UWP, lo que aumenta la productividad de manera significativa.
- La UWP usa varios lenguajes de programación muy populares como C++, C#, JavaScript, etc. lo cual hace que esta plataforma sea atractiva para los desarrolladores y eso se traduce en una mayor cantidad de apps para los usuarios.
- Windows,, al igual que iOS, está desarrollado de manera que no es necesaria una máquina virtual intermedia entre las app y el hardware, y por tanto significa que el rendimiento es muy alto y el consumo de energía esta muy reducido ya que se ejecuta código nativo accediendo directamente a las prestaciones del hardware del dispositivo.
- UWP proporciona un control total sobre el ciclo de vida de las app, los administradores podrán ofrecer un soporte más rápido y dinámico a los usuarios.
- El usuario tendrá la misma experiencia de uso ya que la app es la misma en todos los dispositivos.

- Gran variedad de dispositivos disponible con Windows, permitiendo al usuario poder elegir el que mejor se adecue a la tarea que desea realizar.

Desventajas

En cuanto a las desventajas encontramos algunas:

- Las apps UWP están limitadas a los dispositivos Windows 10, no funcionarían en versiones antiguas de Windows que no soportan la plataforma universal.
- El hecho de tener que lidiar con varios SDK extensos dependiendo del dispositivo, puede suponer una dificultad para los desarrolladores al no poder concentrarse en un único tipo de dispositivo y así aprovechar todo el potencial de dicho dispositivo. Esto se traduciría en menos apps para los usuarios.
- Ya no se fabrican smartphones con Windows 10 y considerando que el smartphones es uno de los dispositivos más utilizado en el mundo, pues esto supone un gran problema a la hora de elegir plataforma para desarrollar una app para Windows hoy en día. Esto supone la desventaja más grande frente a Android e iOS.

2.6 Elección del sistema operativo

Tal y como se indica en el título del presente TFG, el objetivo es el desarrollo de una aplicación Android. Por tanto, no se ha tenido que elegir la plataforma en el cual desarrollar la app ya que era requisito del TFG que sea Android. Sin embargo, a continuación, se enumeran las razones por las cuales se elegiría Android en el caso de que hubiese libertad para elegir el sistema operativo para desarrollar la APP.

- En el caso de Windows, simplemente no existen smartphones con este sistema, por lo que el desarrollo para este sistema quedaría desacertado a la hora de desarrollar la app de este TFG.
- El desarrollo de apps para iOS requiere un sistema Mac, este sentido, Android no requiere de un sistema específico.
- Desarrollar para iOS requiere el pago de una cuota de 99 $ al año para poder registrarse y publicar en la App Store. Android requiere una única cuota, bastante más baja comparada con iOS.
- Los dispositivos reales para el testeo de la app habrían de ser un iPhone o un iPad. Dichos dispositivos tienen un coste excesivamente elevado en comparación con los dispositivos Android.

Capítulo 3 - Android

Capítulo 3 - Android

3.1 Introducción

La telefonía móvil está cambiando la sociedad actual de una forma tan significativa como lo ha hecho Internet. Esta revolución no ha hecho más que empezar, los nuevos terminales ofrecen unas capacidades similares a las de un ordenador personal, lo que permite que puedan ser utilizados para leer nuestro correo o navegar por Internet. Pero a diferencia de un ordenador, un teléfono móvil siempre está en el bolsillo del usuario. Esto permite un nuevo abanico de aplicaciones mucho más cercanas al usuario. De hecho, muchos autores coinciden en que el nuevo ordenador personal del siglo veintiuno será un terminal móvil (Tomás Gironés, 2013).

Lo que se pretende en este tercer capítulo es hacer un estudio sobre la arquitectura y los componentes que integran la plataforma Android.

3.2 Arquitectura

Antes de entrar en el tema de desarrollo de aplicaciones, conviene comprender la arquitectura interna del sistema operativo Android. Siendo dicha arquitectura de naturaleza compleja, va a requerir de una explicación en profundidad para comprender bien la estructura interna de Android. A pesar de ello, se va a tratar de destacar los puntos más importantes para explicar la arquitectura de Android.

Android, como conjunto, está diseñado en forma de capas que se comunican entre ellas para gestionar los recursos del dispositivo en el que está instalado el sistema operativo. Android hace uso del núcleo de Linux para tener acceso a sus recursos y poder gestionarlos. Esto es posible gracias a que se encuentra en una capa por encima de dicho Núcleo, de esta forma es sencillo para Android poder gestionar recursos como los controladores de pantalla, cámara, memoria NAND…

En la siguiente figura se muestran las capas que conforman el sistema operativo Android (Android Developers, 2019g):

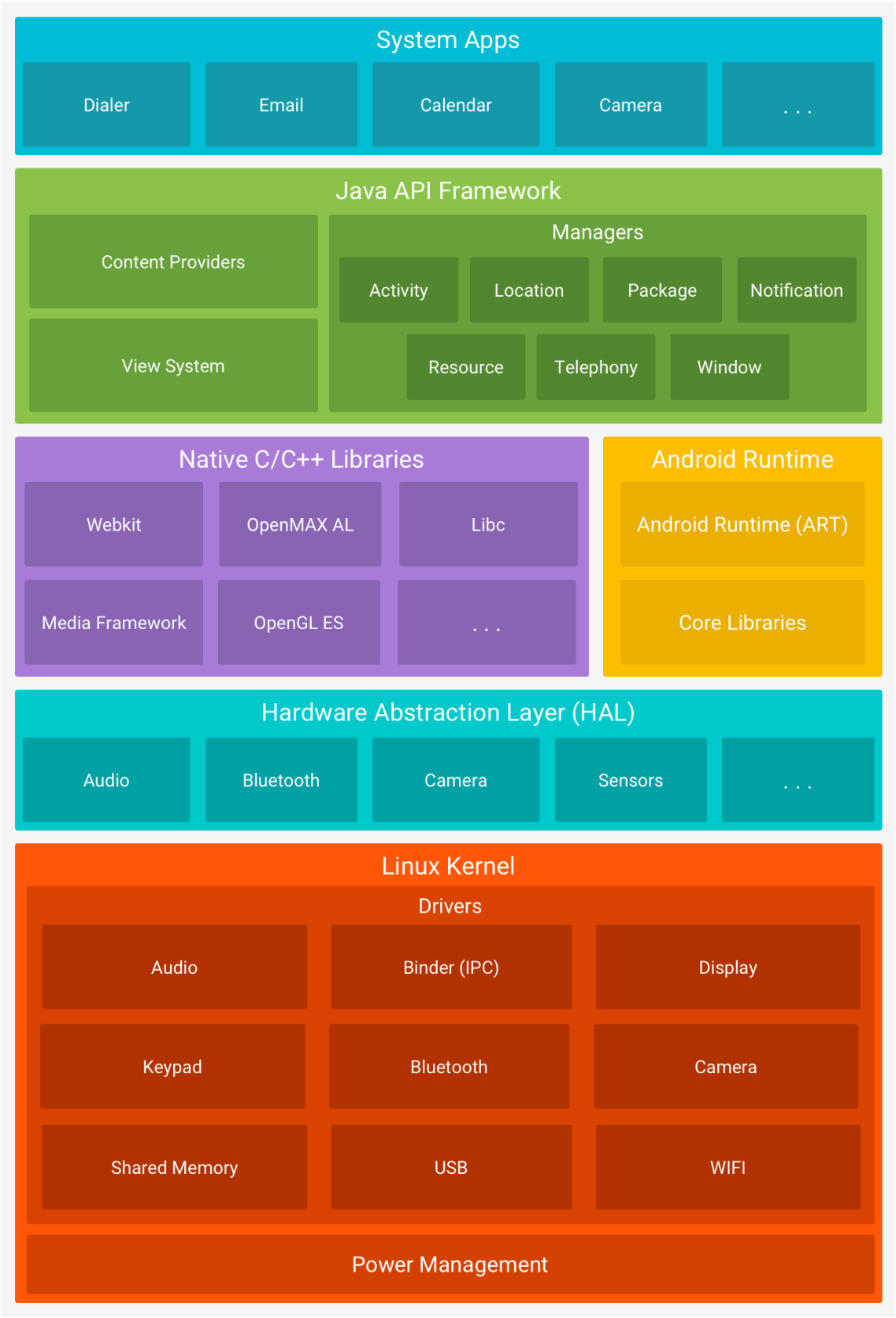

Figura 15 - Capas del sistema operativo Android (Android Developers, 2019g)

A continuación, se procede a detallar cada una de las capas que forman parte de Android.

3.2.1 Kernel de Linux

La base de la plataforma Android es el kernel de Linux. Por ejemplo, el tiempo de ejecución de Android (ART) se basa en el kernel de Linux para funcionalidades subyacentes, como la generación de subprocesos y la administración de memoria de bajo nivel. El uso del kernel de Linux permite que Android aproveche funciones de seguridad claves y, al mismo tiempo, permite a los fabricantes de dispositivos desarrollar controladores de hardware para un kernel conocido (Android Developers, 2019g).

La primera versión de Android está basada en la versión 2.6 del kernel de Linux y se va actualizando, normalmente con cada nueva versión de Android, a este proceso se le llama *kernel upstreaming*. Normalmente, Android no utiliza la última versión del kernel Linux, y la explicación radica en que fabricante compila y modifica su propia versión del kernel debido a la gran variedad de hardware que utiliza cada fabricante, este tipo de prácticas, tal y como se ha mencionado en el capítulo anterior, no hacen más que contribuir a uno de los mayores puntos negativos de Android, la fragmentación.

El kernel actúa de middleware, es decir, una capa de abstracción (en inglés, Hardware Abstraction Layer o HAL) entre el hardware y el resto de las capas de la arquitectura. Al controlar el hardware, el kernel debe incluir los controladores (en inglés, drivers) correspondientes a cada componente del sistema (pantalla, bluetooth, wifi, NFC…). El kernel también se encarga de gestionar los diferentes recursos del teléfono (frecuencias de los núcleos de la CPU, GPU, gestión de energía, memoria, etc.) y del sistema operativo en sí: actividades, procesos, elementos de comunicación, etc.

3.2.2 Librerías

La siguiente capa corresponde a las librerías nativas de Android. Esta capa gestiona los diferentes tipos de datos que se intercambia y se procesan en el sistema. Están escritas en el lenguaje de programación C o C++ y son específicas para cada tipo de hardware. Como se puede deducir, estas librerías proporcionan a Android la mayor parte de sus capacidades más características. Dada la importancia de esta capa, se podría decir que es la que, junto al kernel de Linux, forman el núcleo grueso de Android.

Algunas de las librerías nativas más importantes son (Universidad Carlos III, 2015):

- Librería ***libc*** y ***libc++***: Incluye todas las cabeceras y funciones según el estándar del lenguaje C para *libc* y C++ para *libc++*. Todas las demás librerías se definen en este lenguaje.
- Librería ***Surface Manager***: Es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.
- ***OpenGL/SL y SGL***: Representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.
- Librería ***Media Libraries***: Proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.)
- Librería ***FreeType***: Permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.
- Librería ***SSL***: Posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.
- Librería ***SQLite***: Creación y gestión de bases de datos relacionales.
- Librería ***WebKit***: Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

3.2.3 Entorno de ejecución

El entorno de ejecución está en el mismo nivel que las librerías de Android. Está formado por las Core Libraries, que son librerías con una gran variedad de clases Java y la máquina virtual Dalvik para las versiones de Android inferiores a la 5.0 o su sustituta, ART, en el caso de las versiones de Android 5.0 o superiores.

Máquina virtual Dalvik

Es una máquina virtual desarrollada por Google para Android. Aunque es muy parecida a la máquina virtual de Java, la máquina virtual Dalvik se diferencia en que ocupa menos espacio, es capaz de ejecutar conjuntos de instrucciones de 16 bits en vez de los 8 bits de la máquina virtual de Java y la tabla de constantes (constant pool) se ha modificado para utilizar solo índices de 32 bits para simplificar. La máquina virtual Dalvik se caracteriza por realizar compilaciones “justo a tiempo” (JIT, Just-In-Time). Es decir,

cada vez que se ejecuta una aplicación, el compilador de la máquina virtual Dalvik traduce el bytecode de dicha aplicación con la herramienta “dexopt” en tiempo de ejecución a código máquina, y lo ejecuta. La máquina virtual Dalvik abre el fichero APK y descomprime el fichero DEX del interior. Luego creará en caché o actualizará el fichero ODEX si los permisos lo permiten, y luego lo ejecutará (Domínguez Chávez, 2016).

En la figura 16 se puede observar un diagrama del funcionamiento de la máquina virtual Dalvik:

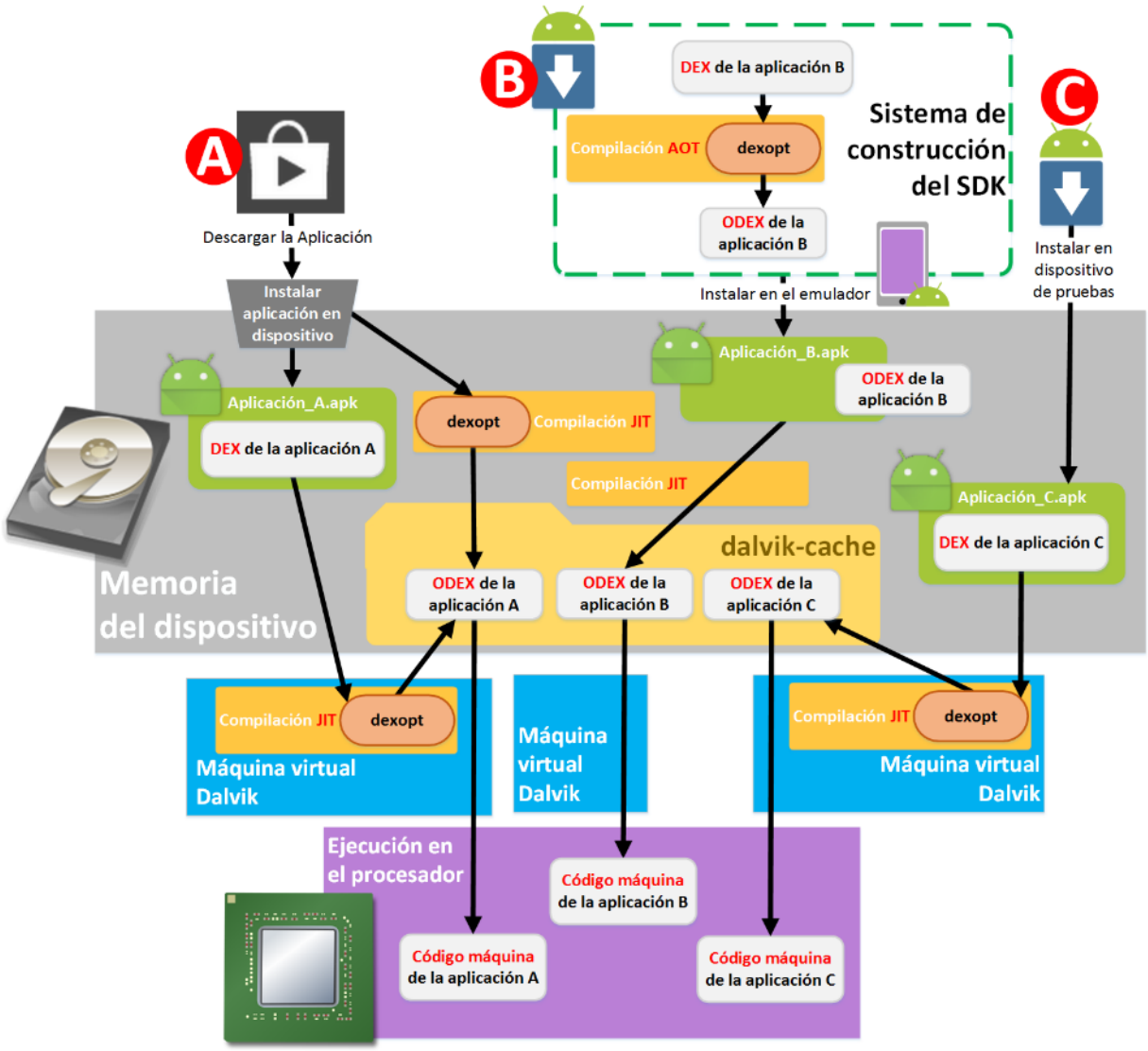

Figura 16 - Diagrama del funcionamiento de la máquina virtual Dalvik (Domínguez Chávez, 2016)

Entorno de ejecución ART

ART (Android Runtime, Entorno de ejecución Android), realiza una compilación AOT en el momento de la instalación para obtener el ejecutable ELF (sustituye al fichero ODEX de Dalvik). La compilación AOT consiste en compilar el bytecode en código máquina antes de que se ejecute el programa. En Android se compila el bytecode DEX en el momento en el que se instala la aplicación en el dispositivo, convirtiéndose en el fichero ejecutable ELF (Domínguez Chávez, 2016).

De esta manera se simplifica sustancialmente la compilación y ejecución de las aplicaciones. En la figura 15 (Domínguez Chávez, 2016). se puede observar un diagrama de las fases de ART:

Figura 17 - Diagrama de las fases de ART (Domínguez Chávez, 2016)

Algunas de las mejoras de ART respecto a la máquina virtual Dalvik (Domínguez Chávez, 2016):

- Mejora el rendimiento del recolector de basura
- Mejora la depuración de aplicaciones
- Consume menos energía
- Mejora el rendimiento general del sistema
- No hay caché de código en tiempo de ejecución
- Al utilizar solo memoria RAM se puede paginar (Dalvik utilizaba memoria Caché que no era paginable)
- Pre-inicializa el conjunto de clases en tiempo de compilación lo que mejora el rendimiento de la memoria RAM
- Compilar con la herramienta "dex2oat" desde un fichero DEX a un ELF consume casi tres veces más de tiempo que, compilar con la herramienta "dexopt" de DEX a ODEX. Pese a esto, el resultado es una compilación completa en los ficheros ELF, al contrario que en los ODEX que solo se compilaba una parte
- Otra gran ventaja es que ART mantiene la retrocompatibilidad con Dalvik al utilizar sus mismos bytecode ".dex". Pero ART no utiliza ".odex", sino que los sustituye por los ejecutables "ELF".

3.2.4 Framework de aplicaciones

Esta capa representa al conjunto de herramientas de desarrollo de cualquier aplicación, es decir, todas las clases, servicios y APIs que utilizan las aplicaciones para realizar sus funciones. Estos componentes, escritos en Java, acceden a los recursos de las capas inferiores mediante la máquina virtual Dalvick o mediante ART en las versiones de Android 5.0 o superiores.

Algunos de los componentes más importantes de esta capa son (Universidad Carlos III, 2015):

- *Activity Manager*: Conjunto de APIs que gestiona el ciclo de vida de las aplicaciones en Android.
- *Window Manager*: Gestiona las ventanas de las aplicaciones y utiliza la librería *Surface Manager*.
- *Telephony Manager*: Incluye todas las API vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).
- *Content Provider:* Permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc. será accesible para otras aplicaciones.

- *View System*: Proporciona un gran número de elementos para poder construir interfaces de usuario (*GUI*), como listas, mosaicos, botones, *check-boxes*, tamaño de ventanas, control de las interfaces mediante teclado, etc. Incluye también algunas vistas estándar para las funcionalidades más frecuentes.
- *Location Manager*: Posibilita a las aplicaciones la obtención de información de localización y posicionamiento.
- *Notification Manager*: Mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución: una llamada entrante, un mensaje recibido, conexión Wi-Fi disponible, ubicación en un punto determinado, etc. Si llevan asociada alguna acción, en Android denominada Intent, (por ejemplo, atender una llamada recibida) ésta se activa mediante un simple clic.
- *Sensor Manager*: Nos permite manipular los elementos de hardware del teléfono como el acelerómetro, giroscopio, sensor de luminosidad, sensor de campo magnético, brújula, sensor de presión, sensor de proximidad, sensor de temperatura, etc.

- *XMPP Service*: Colección de API para utilizar este protocolo de intercambio de mensajes basado en XML.

- *Package Manager*: Esta biblioteca permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes. Con paquete nos referimos a la forma en que se distribuyen las aplicaciones Android, estos contienen el archivo .apk, que a su vez incluyen los archivos .dex con todos los recursos y archivos adicionales que necesite la aplicación, para facilitar su descarga e instalación.

- *Camera*: Con esta librería podemos hacer uso de la(s) cámara(s) del dispositivo para tomar fotografías o para grabar vídeo.

- Multimedia: Permiten reproducir y visualizar audio, vídeo e imágenes en el dispositivo.

3.2.5 Aplicaciones

Esta última capa contiene las aplicaciones, tanto las incluidas por defecto de Android como aquellas que el usuario instala posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API y librerías de las capas anteriores.

Las aplicaciones tienen un tipo de archivo específico para Android, el .APK (.APK (Android Applicacion Package). Este archivo es el resultado del proceso de construcción de una aplicación Android y dentro encontraremos los siguientes componentes (Android Developers, 2019c):

- Archivo *Android Manifest*: Este archivo es la definición de todas las características principales que tiene una aplicación al ejecutarse en un dispositivo móvil. Con características, se refiere a los bloques (*Activities*) que posee la aplicación, los permisos, su versión, las versiones previas soportadas, las dimensiones de la pantalla, etc.
- Archivo *classes.dex*: Este será el fichero compilado preparado para ejecutarse en la Máquina Virtual Dalvik o en ART.
- La carpeta *Resources*: Aquí encontramos todos los archivos externos usados para construir el proyecto, como por ejemplo iconos, audio, archivos de texto plano, los archivos .xml de diseño, etc.
- Librerías nativas: El archivo .APK también contiene aquellas librerías externas de las cuales depende la aplicación.
- Carpeta *META-INF*: En ella se guardan archivos que corresponden a las Firmas Digitales de la aplicación. Con esta especificación se puede indicar quién es el creador y dueño de la aplicación, además se debe indicar el ID de desarrollador para ser reconocido y autenticado en procesos de comercialización.

3.2.6 Firma y publicación de Apps

El sistema Android requiere que todas las aplicaciones instaladas estén firmadas digitalmente con un certificado cuya clave privada se lleva a cabo por el desarrollador de la aplicación. Este certificado no se utiliza para controlar las aplicaciones que el usuario puede instalar ni tiene que ser firmado por una autoridad de certificación: es perfectamente admisible, y típico, para aplicaciones de Android utilizar certificados auto firmados.

Los puntos importantes en la firma de aplicaciones son (Android Developers, 2019d):

- Todas las aplicaciones deben ser firmadas. El sistema no instala las aplicaciones que no estén firmadas.

- Se puede utilizar certificados auto firmados para firmar las aplicaciones. No se necesita ninguna autoridad de certificación.
- Antes de publicar aplicación para su uso por parte de los usuarios finales, se debe firmar con una clave privada adecuada. No se puede publicar una aplicación firmada con la clave de depuración, *debug.keystore*, que se genera por las herramientas del SDK.
- El sistema comprueba la fecha de expiración del certificado sólo en el momento de la instalación. Si un certificado de una aplicación expira después de que la aplicación esté instalada, la aplicación seguirá funcionando normalmente.
- Se utilizan las herramientas estándar - *Keytool* y *Jarsigner* - para generar las claves y firmar los archivos .7APK.
- Una vez firmada la aplicación, es recomendable utilizar la herramienta *zipalign* para optimizar el paquete .APK final.

Una vez completado el proceso de firma de las aplicaciones, el siguiente paso es la publicación. Publicar una aplicación implica realizar las pruebas, empaquetar la aplicación correctamente y ponerla a disposición de los usuarios finales de dispositivos Android. Algunas de las consideraciones a tener en cuenta son (Android Developers, 2019d):

- Probar la aplicación a fondo en un dispositivo real
- Añadir una licencia de usuario final en la aplicación
- La posibilidad de añadir soporte de licencias
- Especificar un icono y nombre en el manifiesto de la aplicación
- Desactivar el registro y depuración y limpiar datos y archivos
- Antes de hacer la última compilación de la aplicación:
    - Indicar la versión de la aplicación
    - Obtener una clave criptográfica adecuada
    - Obtener la clave para el API de Google Maps, si la aplicación utiliza el componente *MapView*.
- Compilar la aplicación
- Firmar la aplicación

Una vez considerados los aspectos anteriores, el siguiente paso es el de la publicación de la aplicación en **Google Play**. Para ello, primero hay que registrarse en el servicio, en la dirección *https://play.google.com/apps/publish/signup* utilizando una cuenta de Google y de acuerdo con los términos del servicio. El registro tiene un coste de 25 *$* para todo el mundo. Una vez registrado, se puede subir la aplicación en el servidor, actualizar tantas veces como se desee, y posteriormente publicarla aplicación cuando esté lista. Después de este paso, los usuarios pueden ver la aplicación, descargarla, comentarla y puntuarla.

3.2.7 La gestión multitarea

Android es el primer sistema operativo para móviles en implementar multitarea real, similar a Windows en PC. Otros sistemas como iOS o Windows Phone implementaban un tipo de multitarea incompleto ya que cuando se cambiaba de aplicación, la aplicación anterior se congelaba hasta que se volviese a ella. En cambio, en Android, si una aplicación pasa a segundo plano, ésta sigue ejecutándose y realizando sus funciones a costa de consumir recursos del sistema. Por una parte, esto facilita y mejora la experiencia del usuario, pero, por otra parte, tiene un impacto negativo en el rendimiento general del dispositivo. Sin embargo, este problema ya no es tan significativo en los dispositivos actuales, pues los hay que cuentan con procesadores de hasta 10 núcleos e incluso algunos cuentan con hasta 12 GB de RAM.

3.3 Las versiones de Android

Uno de los aspectos más importantes a tener en cuenta a la hora de desarrollar en Android es la versión mínima a la cual va dirigida la aplicación. En las diferentes versiones de Android hay clases y métodos que están disponibles a partir de una cierta versión, si las vamos a utilizar hemos de conocer la versión mínima necesaria.

Cada vez que se ha lanzado una nueva plataforma, siempre ha sido compatible con las versiones anteriores. Es decir, solo se añaden nuevas funcionalidades y en el caso de modificar alguna funcionalidad no se elimina, se etiquetan como obsoletas, pero se pueden continuar utilizando, la mayoría de entornos de desarrollo se encargan de avisarnos si alguna funcionalidad está obsoleta.

A continuación, se describen las versiones de Android lanzadas hasta la fecha con una breve descripción de las novedades introducidas. Las plataformas se identifican de tres formas: Nombre comercial, versión y nivel de API. Para los nombres comerciales se han elegido postres en orden alfabético Cupcake (v1.5), Donut (v1.6), Éclair (v2.0), Froyo (v2.2), Gingerbread (v2.3), etc., aunque las dos primeras versiones, que hubieran correspondido a las letras A y B, no recibieron nombre (Android, 2019).

Las primeras versiones

Android 1.0 Nivel de API 1 (septiembre 2008)

Esta es la primera versión de Android. Nunca se utilizó comercialmente, por lo que no se debería desarrollar para esta plataforma.

Android 1.1 Nivel de API 2 (febrero 2009)

No se añadieron apenas funcionalidades simplemente se arreglaron algunos errores de la versión anterior. Si se quiere desarrollar una aplicación compatible con todos los dispositivos Android, esta es la versión a escoger. No obstante, apenas existen usuarios con esta versión.

Cupcake

Android 1.5 Nivel de API 3 (abril 2009)

Es la primera versión con usuarios, aunque en la actualidad apenas quedan.

Como novedades de esta versión tenemos (Universidad Politécnica de Valencia, 2018a):

- Una de las más importantes es que el sistema pasa a estar basado en un núcleo Linux 2.6.27
- La posibilidad de teclado con predicción de texto en pantalla gracias al aumento de tamaño de ésta, los terminales ya no tienen que tener un teclado físico,
- La capacidad de grabación avanzada de audio y vídeo.
- También aparecen los *widgets* de escritorio y *live folders*.
- Incorpora soporte para *bluetooth* estéreo, por lo que permite conectarse automáticamente a auriculares *bluetooth*.
- Las transiciones entre ventanas se realizan mediante animaciones.

Donut

Android 1.6 Nivel de API 4 (septiembre 2009)

La primera revisión de Android 1.6 incluye mejoras para usuarios y desarrolladores y, además, cambios en el framework de APIs de Android.

Entre los principales cambios están (Universidad Politécnica de Valencia, 2018a):

- Permite capacidades de búsqueda avanzada en todo el dispositivo.
- También se incorpora la API *gestures* y la síntesis de texto a voz.
- Se facilita que una aplicación pueda trabajar con diferentes densidades de pantalla.
- Soporte para resolución de pantallas WVGA.
- Aparece un nuevo atributo XML, *onClick*, que puede especificarse en una vista.
- Soporte para CDMA/EVDO, 802.1x y VPNs.

- Acceso a las funciones de cámara, videocámara y galería desde la misma aplicación.

Éclair

Android 2.0 Nivel de API 5 (octubre 2009)

Esta versión de API apenas cuenta con usuarios, dado que la mayoría de fabricantes pasaron directamente de la versión 1.6 a la 2.1 (Universidad Politécnica de Valencia, 2018a).

Como novedades cabría destacar:

- Incorpora una API para manejar el *bluetooth* 2.1.
- Nueva funcionalidad que permite sincronizar adaptadores para conectarlo a cualquier dispositivo.
- Ofrece un servicio centralizado de manejo de cuentas.
- Soporte de sincronización email con múltiples cuentas.
- Mejora la gestión de contactos y ofrece más ajustes en la cámara.
- Se ha optimizado la velocidad de *hardware*.
- Se aumenta el número de tamaños de ventana y resoluciones soportadas.
- Nueva interfaz del navegador y soporte para HTML5.
- Mejoras en el calendario y soporte para Microsoft Exchange.
- La clase *MotionEvent* ahora soporta eventos en pantallas multitáctil.

Android 2.1 Nivel de API 7 (enero 2010)

Se considera una actualización menor, por lo que conservó el nombre de Éclair. En esta versión destaca (Universidad Politécnica de Valencia, 2018a):

- El reconocimiento de voz, que permite introducir un campo de texto dictando sin necesidad de utilizar el teclado.
- También permite desarrollar fondos de pantalla animados.
- Se puede obtener información sobre la señal de la red actual que posea el dispositivo.
- En el paquete WebKit se incluyen nuevos métodos para manipular bases de datos almacenadas en Internet.

Froyo

Android 2.2 Nivel de API 8 (mayo 2010)

Esta es una de las versiones que todavía se utiliza en la actualidad. Proporciona nuevas características a usuarios y desarrolladores, cambios en la API y corrección de pequeños errores o bugs.

Como características más destacadas se puede indicar (Universidad Politécnica de Valencia, 2018a):

- La mejora de velocidad de ejecución de las aplicaciones (ejecución del código de la CPU de 2 a 5 veces más rápido que en la versión 2.1 de acuerdo a varios test de rendimiento o *benchmarks*). Esto se consigue con la introducción de un nuevo compilador JIT de la máquina virtual Dalvik.
- Se añaden varias mejoras relacionadas con el navegador Web, como el soporte de Adobe Flash 10.1 y la incorporación del motor Javascript V8 utilizado en Chrome.
- En el desarrollo de aplicaciones permite las siguientes novedades: se puede preguntar al usuario si desea instalar una aplicación en un medio de almacenamiento externo (como una tarjeta SD), como alternativa a la instalación en la memoria interna del dispositivo. Las aplicaciones se actualizan de forma automática cuando aparece una nueva versión. Proporciona un servicio para la copia de seguridad de datos que se puede realizar desde la propia aplicación para garantizar al usuario el mantenimiento de sus datos. Por último, se facilita que las aplicaciones interaccionen con el reconocimiento de voz y que terceras partes proporcionen nuevos motores de reconocimiento.
- Se mejora la conectividad dando la posibilidad al teléfono para dar acceso a Internet a otros dispositivos (tethering), tanto por USB como por Wi-Fi. También se añade el soporte a Wi-Fi IEEE 802.11n y notificaciones *push*.
- Se añaden varias mejoras en diferentes componentes: En el API gráfica OpenGL ES se pasa a soportar la versión 2.0.
- Para finalizar, permite definir modos de interfaz de usuario (“automóvil” y “noche”) para que las aplicaciones se configuren según el modo seleccionado por el usuario.

Gingerbread

Android 2.3 Nivel de API 9 (diciembre 2010)

Debido al éxito de Android en las nuevas Tablets de aquella época, Android pasa a dar soporte mayores tamaños de pantalla y resoluciones (WXGA y superiores) e incorpora un nuevo interfaz de usuario con un diseño actualizado (Universidad Politécnica de Valencia, 2018a).

Dentro de las mejoras y novedades destacan:

- Se mejora la funcionalidad de "cortar, copiar y pegar" y un teclado en pantalla con capacidad multitáctil.
- Se incluye soporte nativo para varias cámaras, pensado en la cámara delantera usada en videoconferencia. La incorporación de esta segunda cámara ha propiciado la inclusión de reconocimiento facial para identificar el usuario del terminal.
- La máquina virtual de Dalvik introduce un nuevo recolector de basura (*Garbage collector*) que minimiza las pausas de la aplicación, ayudando a garantizar una mejor animación y el aumento de la capacidad de respuesta en juegos y aplicaciones similares. Se trata de corregir así una de las lacras de este sistema operativo móvil, que en versiones previas no ha sido capaz de cerrar bien las aplicaciones en desuso que queda en segundo plano.
- Se dispone de mayor apoyo para el desarrollo de código nativo (NDK).
- Se mejora la gestión de energía y control de aplicaciones.
- Se cambia el sistema de ficheros, que pasa de YAFFS a ext4.
- Soporte nativo para telefonía a través de Internet (VoIP/SIP).
- Soporte para reproducción de vídeo WebM/VP8 y codificación de audio AAC.
- Soporte para la tecnología NFC.
- Se añaden facilidades en el audio, gráficos y entradas para los desarrolladores de juegos.
- El soporte nativo para más sensores (como giroscopios y barómetros).
- Un gestor de descargas para las descargas largas.

Honeycomb

Android 3.0 Nivel de API 11 (febrero 2011)

Para mejorar la experiencia de Android en las nuevas Tablets se lanza la versión 3.0 optimizada para dispositivos con pantallas grandes. En esta versión la interfaz de usuario ha sido completamente rediseñada con paradigmas nuevos para la interacción y navegación (Universidad Politécnica de Valencia, 2018a).

Entre las novedades introducidas destacan:

- Los *fragments*, con los que podemos diseñar diferentes elementos del interfaz de usuario posibilitando así el diseño adaptativo.
- Se introduce la barra de acciones (*Action bar*), donde las aplicaciones pueden mostrar un menú siempre visible y las teclas físicas son reemplazadas por teclas en pantalla.
- La nueva interfaz se pone a disposición de todas las aplicaciones, incluso las construidas para versiones anteriores de la plataforma. Esto se consigue gracias a la introducción de librerías de compatibilidad (Support libraries) que pueden ser utilizadas en versiones anteriores a la 3.0.
- Se mejora los gráficos 2D/3D gracias al renderizador OpenGL acelerado por hardware. Aparece el nuevo motor de gráficos *Renderscript*, que saca mayor rendimiento al hardware e incorpora su propia API.
- Se incorpora un nuevo motor de animaciones mucho más flexible, conocido como animación de propiedades.
- Primera versión de la plataforma que soporta procesadores multinúcleo. Lo que lleva a que la máquina virtual Dalvik sea optimizada para permitir multiprocesado (Multithreading), lo que permite una ejecución más rápida de las aplicaciones, incluso aquellas que son de hilo único.
- Se incorporan varias mejoras multimedia, como listas de reproducción M3U a través de HTTP Live Sreaming, soporte a la protección de derechos musicales (DRM) y soporte para la transferencia de archivos multimedia a través de USB con los protocolos MTP y PTP.
- En esta versión se añaden nuevas alternativas de conectividad, como las nuevas APIS de Bluetooth A2DP y HSP con streaming de audio. También, se permite conectar teclados completos por USB o Bluetooth.
- Se mejora el uso de los dispositivos en un entorno empresarial. Entre las novedades introducidas destacan las nuevas políticas administrativas con encriptación del almacenamiento, caducidad de contraseña y mejoras para administrar los dispositivos de empresa de forma eficaz.

A pesar de la nueva interfaz gráfica optimizada para tabletas, Android 3.0 es compatible con las aplicaciones creadas para versiones anteriores a pesar de que éstas no aprovechen del todo las bondades de Android Honeycomb.

Android 3.1 Nivel de API 12 (mayo 2011)

Como novedades en esta versión destaca la posibilidad de manejar dispositivos conectados por USB (tanto host como dispositivo) y la inclusión de protocolos de transferencia de fotos y vídeo (PTP/MTP) y de tiempo real (RTP) (Universidad Politécnica de Valencia, 2018a).

Android 3.2 Nivel de API 13 (julio 2011)

Se incluyen optimizaciones para distintos tipos de Tablets, el Zoom se vuelve compatible para aplicaciones de tamaño fijo y se añade la sincronización multimedia desde SD (Universidad Politécnica de Valencia, 2018a).

Ice Cream Sandwich

Android 4.0 Nivel de API 14 (octubre 2011)

La característica más importante es que se unifican las dos versiones anteriores (2.x para teléfonos y 3.x para tablets) en una sola compatible con cualquier tipo de dispositivo (Universidad Politécnica de Valencia, 2018a).

Entre las características más interesantes destaca:

- Se introduce una nueva interfaz de usuario totalmente renovada. Por ejemplo, se reemplazan los botones físicos por botones en pantalla (característica heredada de las versiones 3.x).
- Nueva API de reconocimiento facial que permite entre otras muchas cosas, el desbloqueo del teléfono al reconocer la cara a su propietario.
- También se mejora en el reconocimiento de voz. Por ejemplo, se puede empezar a hablar en cuanto pulsamos el botón.
- Aparece un nuevo gestor de tráfico de datos de Internet, donde podremos ver el consumo de forma gráfica y donde podemos definir los límites a ese consumo para evitar cargos inesperados con la operadora.
- Incorpora herramientas para la edición de imágenes en tiempo real, con herramientas para distorsionar, manipular e interactuar con la imagen al momento de ser capturada.
- Se mejora el API para comunicaciones por NFC y la integración con redes sociales.

En diciembre del 2011 aparece una actualización de mantenimiento (versión 4.0.2) que no aumenta el nivel de API.

Android 4.0.3 Nivel de API 15 (diciembre 2011)

Se introducen ligeras mejoras en algunas APIs incluyendo el de redes sociales, calendario, revisor ortográfico, texto a voz y bases de datos entre otros. En marzo de 2012 aparece la actualización 4.0.4 que corrige errores y bugs e introduce mejoras de seguridad (Universidad Politécnica de Valencia, 2018a).

Jelly Bean

Android 4.1 Nivel de API 16 (julio 2012)

En esta versión se hace hincapié en mejorar un punto débil de Android: la fluidez del interfaz de usuario. Con este propósito surge el llamado *Project Butter* gracias al cual se incorporan varias técnicas para mejorar la fluidez, como: sincronismo vertical, triple búfer y el aumento de la velocidad del procesador al tocar la pantalla (Universidad Politécnica de Valencia, 2018a).

Otras de las novedades y mejoras a destacar son:

- Se mejoran las notificaciones con un sistema de información expandible personalizada.
- Los Widgets de escritorio pueden ajustar su tamaño y hacerse sitio de forma automática al situarlos en el escritorio.
- El dictado por voz puede realizarse sin conexión a Internet (aunque en ese momento sólo en inglés).
- Se introducen varias mejoras en Google Search. Se potencia la búsqueda por voz con resultados en forma de ficha. La función Google Now permite utilizar información de posición, agenda y hora en las búsquedas.
- Se incorporan nuevo soporte para usuarios internacionales: como texto bidireccional y teclados instalables.
- Para mejorar la seguridad, las aplicaciones son cifradas.
- Y también se permiten las actualizaciones parciales de aplicaciones.

Android 4.2 Nivel de API 17 (noviembre 2012)

Una de las novedades más importantes es la posibilidad de crear varias cuentas de usuario en el mismo dispositivo. Aunque, esta característica solo estaría disponible en tablets. Cada cuenta tendría sus propias aplicaciones y configuración (Universidad Politécnica de Valencia, 2018a).

Los Widgets de escritorio pueden aparecer en la pantalla de bloqueo. Se incorpora un nuevo teclado predictivo deslizante al estilo del teclado Swype. Posibilidad de conectar dispositivo y televisores mediante wifi (Miracast). Mejoras menores en las notificaciones. Nueva aplicación de cámara que incorpora la funcionalidad Photo Sphere para hacer fotos panorámicas inmersivas (en 360º) (Universidad Politécnica de Valencia, 2018a).

Android 4.3 Nivel de API 18 (julio 2013)

Esta versión introduce mejoras en múltiples áreas. Entre ellas los perfiles restringidos (disponible sólo en tabletas) que permiten controlar los derechos de los usuarios para ejecutar aplicaciones específicas y para tener acceso a datos específicos. Igualmente, los programadores pueden definir restricciones en las apps, que los

propietarios puedan activar si quieren. Se da soporte para Bluetooth Low Energy (BLE) que permite a los dispositivos Android comunicarse con los periféricos con bajo consumo de energía. Se agregan nuevas características para la codificación, transmisión y multiplexación de datos multimedia. Se da soporte para OpenGL ES 3.0. Se mejora la seguridad para gestionar y ocultar las claves privadas y credenciales (Universidad Politécnica de Valencia, 2018a).

KitKat

Android 4.4 Nivel de API 19 (octubre 2013)

El principal objetivo de la versión 4.4 es hacer que Android esté disponible en una gama aún más amplia de dispositivos, incluyendo aquellos con tamaños de memoria RAM de solo 512 MB. Para ello, todos los componentes principales de Android han sido recortados para reducir sus requerimientos de memoria, y se ha creado una nueva API que permite adaptar el comportamiento de la aplicación en dispositivos con poca memoria. Algunas de las mejoras y nuevas características son (Universidad Politécnica de Valencia, 2018a):

- Más visibles son algunas nuevas características de la interfaz de usuario. El modo de inmersión en pantalla completa oculta todas las interfaces del sistema (barras de navegación y de estado) de tal manera que una aplicación puede aprovechar el tamaño de la pantalla completa. WebViews (componentes de la interfaz de usuario para mostrar las páginas Web) se basa ahora en el software de Crome de Google y por lo tanto puede mostrar contenido basado en HTM5 (Universidad Politécnica de Valencia, 2018a).
- Se mejora la conectividad con soporte de NFC para emular tarjetas de pago tipo HCE, varios protocolos sobre Bluetooth y soporte para mandos infrarrojos. También se mejoran los sensores para disminuir su consumo y se incorpora un sensor contador de pasos.
- Se facilita el acceso de las aplicaciones a la nube con un nuevo marco de almacenamiento. Este marco incorpora un tipo específico de *content provider* conocido como *document provider*, nuevas intenciones para abrir y crear documentos y una ventana de dialogo que permite al usuario seleccionar ficheros. Se incorpora un administrador de impresión para enviar documentos a través de WiFi a una impresora. También se añade un content provider para gestionar los SMS.

Desde una perspectiva técnica, hay que destacar la introducción la nueva máquina virtual ART, que consigue tiempos de ejecución muy superiores a la máquina Dalvik. Sin embargo, todavía está en una etapa experimental. Por defecto se utiliza la máquina virtual de Dalvik, permitiendo a los programadores activar opcionalmente ART para

verificar que sus aplicaciones funcionan correctamente (Universidad Politécnica de Valencia, 2018a).

Lollipop

Android 5.0 Nivel de API 21 (noviembre 2014)

Android 5.0 es la versión de Google lanzada en junio de 2014, se conoció con el nombre Android L, entre las mejoras y novedades más destacables encontramos (Universidad Politécnica de Valencia, 2018a):

- La novedad más importante de Lollipop es la extensión de Android a nuevas plataformas, incluyendo Google Wear, Google TV y Google Card. Hay un cambio significativo en la arquitectura, al utilizar la máquina virtual ART en lugar de Dalvik. Esta novedad ya había sido incorporada en la versión anterior a modo de prueba. ART mejora de forma considerable el tiempo de ejecución del código escrito en Java. Además, se soporta dispositivos de 64 bits en procesadores ARM, x86, y MIPS. Muchas aplicaciones del sistema (Chrome, Gmail…) se han incorporado en código nativo para una ejecución más rápida.
- Desde el punto de vista del consumo de batería, hay que resaltar que en Lollipop el modo de ahorro de batería se activa por defecto. Este modo desconecta algunos componentes en caso de que la batería esté baja. Se incorpora una nueva API (*android.app.job.JobScheduler*) que nos permite que ciertos trabajos se realicen solo cuando se cumplan determinadas condiciones (por ejemplo, con el dispositivo cargando). También se incluyen completas estadísticas para analizar el consumo que nuestras aplicaciones hacen de la batería.
- En el campo Gráfico Android Lollipop incorpora soporte nativo para OpenGL ES 3.1. Además, esta versión permite añadir a nuestras aplicaciones un paquete de extensión con funcionalidades gráficas avanzadas (*fragment shader, tessellation, geometry shaders, ASTC…).*
- Otro aspecto innovador de la nueva versión lo encontramos en el diseño de la interfaz de usuario. Se han cambiado los iconos, incluyendo los de la parte inferior (Retroceder, Inicio y Aplicaciones), que ahora son un triángulo, un círculo y un cuadrado.

- El nuevo enfoque se centra en Material Design, que consiste en una guía completa para el diseño visual, el movimiento y las interacciones a través de plataformas y dispositivos. Google pretende aplicar esta iniciativa a todas las

plataformas, incluyendo wearables y Google TV. La nueva versión también incluye varias mejoras para controlar las notificaciones. Ahora son más parecidas a las tarjetas de Google Now y pueden verse en la pantalla de bloqueo.

- Se incorporan nuevos sensores como el de pulso cardiaco, el de inclinación (para reconocer el tipo de actividad del usuario), y sensores de interacción compuestos para detectar ciertos gestos.

Marshmallow

Android 6.0 Nivel de API 22 (octubre 2015)

Android 6.0 Marshmallow fue lanzado en setiembre de 2015 y se considera una actualización muy transitoria de Android, como lo fue Android 4.2 JellyBean. Muchos equipos fabricados para tener Lollipop de serie, recibieron actualización a Marshmallow.

Una de las novedades más interesantes es el administrador de permisos. Los usuarios podrán conceder o retirar ciertos permisos a cada aplicación. Con esto el sistema da mucha más protección a la privacidad de los usuarios. Ahora, el sistema realiza una copia de seguridad automática de todos los datos de las aplicaciones. Esto resulta muy útil al cambiar de dispositivo o tras restaurar valores de fábrica (Universidad Politécnica de Valencia, 2018a).

Entre sus características más destacables, tenemos:

- Batería mucho más inteligente y eficiente.
- Google Now disponible para anticipase a lo que se necesita.
- Ahora permite definir qué compartir y cuándo.
- Seguridad simplificada mediante huella dactilar.
- Aplicaciones solicitan permisos que sean realmente necesarios.
- Mejora en el rendimiento de las aplicaciones, permitiendo reducir el consumo de memoria para volver ágil la gestión multitarea.
- Mejora de funciones de redacción y del procesamiento de textos.

Nougat

Android 7.0 Nivel de API 24 (junio 2016)

Android Nougat es la versión numerada como 7.0 e incluye a su sucesor 7.1. Representa una evolución natural de Material Design, nacido con Android Lollipop, aunque pocos equipos pudieron actualizar Android Nougat desde esa versión.

Ahora los usuarios pueden abrir varias aplicaciones al mismo tiempo en la pantalla. Se puede configurar la aplicación para que se visualice con unas dimensiones mínimas o inhabilitar la visualización de ventanas múltiples. (Universidad Politécnica de Valencia, 2018a)

Android Nougat incorpora la plataforma de realidad virtual Daydream. Se trata de una propuesta de Google que complementa la iniciativa *Cardboard*. Incluye especificaciones software y hardware que nos permitirán diferenciar a los dispositivos compatibles. Los principales fabricantes de móviles se han unido a esta iniciativa. (Universidad Politécnica de Valencia, 2018a).

Novedades y mejoras:

- Emojis actualizados.
- Posibilidad de usar 2 o más idiomas al mismo tiempo.
- Ejecución de aplicaciones multiventana.
- Gráficos 3D de alto rendimiento con la API de Vulkan.
- Modo de realidad virtual VR.
- Modo *descanso* para el ahorro en el uso de batería.
- Configuración más personalizada y notificaciones inteligentes.
- Poderosas capas de encriptación y seguridad para la protección de datos.

En cuanto a Android 7.1, la principal novedad son los accesos directos a aplicaciones. Desde el icono de la aplicación, con una pulsación prolongada, aparecen varias opciones que podremos seleccionar. Por ejemplo, podremos iniciar una navegación privada con Chrome de forma directa. Los accesos directos que quieras incorporar a tu aplicación, podrás configurarlo por medio de *intents*, que han de especificarse en un fichero de configuración. También se incorporan otras novedades

como la posibilidad de insertar imágenes desde el teclado, de la misma forma que ahora insertamos emoticonos. (Universidad Politécnica de Valencia, 2018a).

Oreo

Android 8.0 Nivel de API 26 (agosto 2017)

Oreo es el nombre de la versión 8.0 de Android que anunció la firma Google el día 21 de marzo de 2017. Su nombre fue revelado el día 21 de agosto de 2017. Se lanzó por primera vez como una versión previa para desarrolladores en fase alfa el 21 de marzo de 2017, para los móviles de Google (Nexus y Pixel). La segunda versión para desarrolladores se lanzó el 17 de mayo de 2017 y se considera en fase beta. La tercera versión para desarrolladores fue lanzada el 8 de junio de 2017 y contiene las API definitivas.

Destacan las siguientes mejoras (Universidad Politécnica de Valencia, 2018a):

- En seguridad, se introduce Google Play Protect, que escanea regularmente las aplicaciones en busca de *malware*. La opción "Orígenes desconocidos" desaparece. Ahora podemos indicar que aplicaciones pueden instalar apks y cuáles no. Desde la opción "Acceso especial de aplicaciones" podemos configurar que aplicaciones pueden realizar ciertas acciones.
- Pensando en los países emergentes, se lanza Android Go, una distribución adaptada para dispositivos de gama baja (1 GB de RAM o menos). Se preinstalan apps ligeras y en Google Play Store se destacan aplicaciones ligeras adecuadas para estos dispositivos. Estas aplicaciones han de cubrir 3 requisitos, trabajar sin red, pesar menos de 10 MB y un buen rendimiento de batería.
- Con el fin de reducir la fragmentación de Android, aparece el proyecto *Treble* que facilitará las actualizaciones a los fabricantes. Se reestructura la arquitectura de Android para definir una interfaz clara entre capa del Núcleo Linux (con sus drivers) de las capas del Framework. Esto permite actualizar Android sin tener que tocar la capa del Núcleo Linux.
- Las notificaciones presentan varias mejoras: Podemos añadir color de fondo. Se ordenan por importancia. Las aplicaciones pueden crear canales de notificaciones y el usuario decidir cuales quiere recibir. Podemos posponer una notificación o verlas pulsando sobre el icono de la aplicación.

- Los iconos tendrán que estar diseñados en dos capas: el icono y el fondo del icono. Esto permite adaptarse al dispositivo. Además, el usuario podrá escoger entre iconos circulares, cuadrados o de esquinas redondeadas.
- Ahora podemos reproducir un vídeo en una ventana flotante mientras utilizamos otras aplicaciones. Al selecciona un texto se nos sugieren acciones cuando se trata de un número de teléfono o una dirección. Auto completar de Google, que antes estaba disponible en Chrome para guardar contraseñas, ahora se puede usar en cualquier aplicación Android.

Pie

Android 9.0 Nivel de API 28 (agosto 2018)

Android Pie es la novena versión de Android. Primero fue anunciado por Google el 7 de marzo de 2018, y la primera vista previa de desarrollador fue lanzada el mismo día. La segunda vista previa, considerada de calidad beta, se lanzó el 8 de mayo de 2018. La tercera vista previa, llamada Beta 2, fue lanzada el 6 de junio de 2018.5 La cuarta vista previa, llamada Beta 3, fue lanzada el 2 de julio de 2018. La versión definitiva se lanzó el 6 de agosto de 2018.

Destacan las siguientes mejoras (Universidad Politécnica de Valencia, 2018a):

- Una de las novedades más interesantes es el nuevo API WiFi RTT introducido en IEEE 802.11mc. Permite estimar la distancia entre nuestro dispositivo y los puntos de acceso cercanos, lo que permite sistemas de posicionamiento en interiores con una precisión de 1 a 2 metros.
- Otro importante cambio es la navegación por gestos. Se reemplazan los tres botones en pantalla (triangulo, círculo y cuadrado) por solo dos (retroceder e inicio).
- El botón de inicio admite diferentes gestos que nos permite ir al asistente de Google, cambiar entre apps recientes o abrir el menú de apps.
- Una innovación interesante es el uso de Inteligencia Artificial, para mejorar diferentes aspectos. La idea consiste en aprender nuestros hábitos a la hora de usar las aplicaciones. Con esta información se puede quitar preferencia sobre el uso de la CPU a las apps menos utilizadas, consiguiendo una reducción de hasta un 30 %. Este menor uso de la CPU prolongará la vida de la batería. Usando técnicas similares se pretende aprender cuando el usuario va a arrancar una aplicación o una acción de esta. De esta forma el sistema puede cargar en memoria la aplicación antes incluso que el usuario decida utilizarla.

- Se introducen algunas mejoras que fomentan un uso responsable y saludable del móvil. Por ejemplo, desde el Dashboard podemos consultar el uso que hacemos cada día, en cada aplicación. Podemos establecer alarmas de uso excesivo muy interesantes para el control parental. En esta línea, se introducen nuevos modos de relajación y no molestar para favorecer la desconexión digital.

La fragmentación en Android

Tal y como se ha mencionado en el capítulo anterior, uno de los principales problemas de Android es la fragmentación que existe debido a sus múltiples versiones. Nos podemos encontrar con capas de personalización que van, desde las que son prácticamente un calco de Android puro o las que son pesadas y no siempre muy eficientes, como suelen ser las capas orientales que acostumbran incorporar iconos y estilos muy cargados.

La existencia de capas tan distintas hace que el software no sea igual en todos los teléfonos y la gran cantidad de posibilidades que nos encontramos en el hardware de los terminales Android es muy significativa, y todo sumado provoca que cada fabricante necesite mucho tiempo para poder amoldar la versión pura que Google le presta a cada compañía. Otro problema que tiene Android es que la mayor parte de fabricantes solo actualiza cada año los teléfonos con una edad inferior a dos años, lo cual provoca que no se reciban más de una o dos actualizaciones, en el mejor de los casos (Android Ayuda, 2017).

Project Treble

Project Treble es el plan de Google para acabar con la fragmentación de Android. El principal problema de las actualizaciones consiste en que las compañías siempre debían adaptarse a lo que Google estaba ofreciendo. Esto significaba que, a cada nueva versión, había que adaptar las particularidades de cada capa de personalización, algo que se refinaba del todo una vez llegaba a los usuarios. A partir de Android Oreo, Google ha decidido implementar Project Treble, un nuevo proceso que modulariza las actualizaciones de Android y promete simplificar su implantación. También debería abrir el abanico de dispositivos que reciban actualizaciones de forma regular (Android Ayuda, 2017).

Los fabricantes de chips como procesadores y sensores debían adaptar primero sus componentes electrónicos y sus controladores de código cerrado para que otras

empresas como Sony, o Samsung pudieran actualizar. Lo que Google propone es que ese código de esos fabricantes, como por ejemplo Qualcomm, empiece a ser un código aparte. Además, ese código debe seguir siempre unos estándares para comunicarse con Android (Android Ayuda, 2017).

Esto habilita que los chips y las capas personalizadas de Android trabajen siempre de la misma manera. El resto de Android se implementará sobre esos cimientos, acelerando el proceso. Sony, Samsung, Nokia… las compañías podrían trabajar por su cuenta en el nuevo sistema sin tener problemas porque el hardware no funcione (Android Ayuda, 2017).

En el siguiente esquema se puede observar el funcionamiento del proceso:

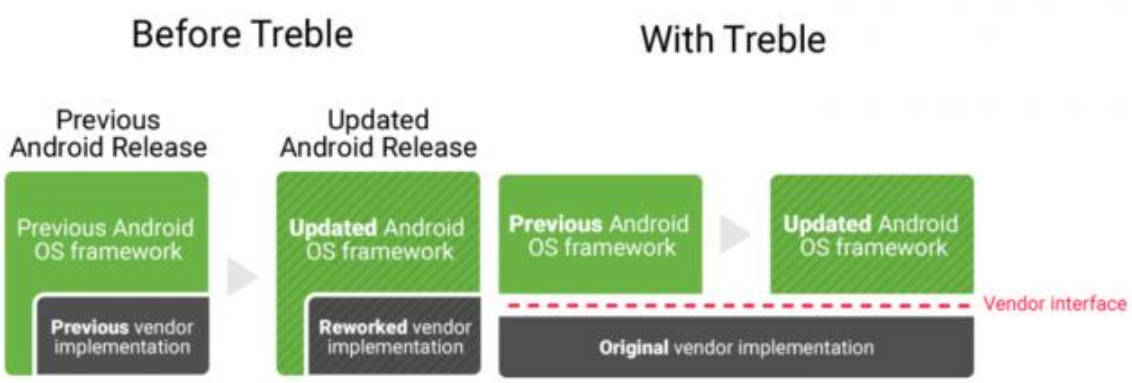

Figura 18 - Esquema del funcionamiento de Treble (Android Ayuda, 2017).

Así, Android se separa en varias partes y se liman las asperezas para facilitar la llegada de nuevas versiones. Este es un movimiento inteligente que permite atacar la raíz de uno de los mayores problemas del sistema (Android Ayuda, 2017).

3.4 Desarrollo en Android

3.4.1 Componentes de una aplicación

Las aplicaciones en Android pueden descomponerse en una serie de elementos clave que resultan imprescindibles para el desarrollo de aplicaciones en Android. Una aplicación en Android debe declarar todas sus actividades, los puntos de entrada, la comunicación, las capas, los permisos, y las intenciones a través de AndroidManifest.xml. Es muy importante tener en consideración cómo estos

componentes impactan en el tiempo de vida del proceso asociado con una aplicación, porque si no son empleados de manera apropiada, el sistema detendrá el proceso de la aplicación aun cuando se esté haciendo algo importante. (Universidad Carlos III, 2015)..

A continuación, se va a realizar una descripción de algunos de estos elementos.

Activity

Un componente Activity refleja una determinada actividad llevada a cabo por una aplicación, y que lleva asociada típicamente una ventana o interfaz de usuario; es importante señalar que no contempla únicamente el aspecto gráfico, sino que éste forma parte del componente Activity a través de vistas representadas por clases como View y sus derivadas. Este componente se implementa mediante la clase de mismo nombre Activity (Universidad Carlos III, 2015).

Intent

Un Intent consiste básicamente en la voluntad de realizar alguna acción, generalmente asociada a unos datos. Lanzando un Intent, una aplicación puede delegar el trabajo en otra, de forma que el sistema se encarga de buscar qué aplicación de entre las instaladas, es la que puede llevar a cabo la acción solicitada. Por ejemplo, abrir una URL en algún navegador web, o escribir un correo electrónico desde algún cliente de correo. Los Intents están incluidos en el AndroidManifest porque describen dónde y cuándo puede comenzar una actividad. Cuando una actividad crea un Intent, éste puede tener descriptores de lo que se quiere hacer. Una vez se está ejecutando la aplicación, Android compara esta información del Intent con los Intents de cada aplicación, eligiendo el más adecuado para realizar la operación especificada por el llamante (Universidad Carlos III, 2015).

View

Las vistas son los elementos que componen la interfaz de usuario de una aplicación: por ejemplo, un botón o una entrada de texto. Todas las vistas van a ser objetos descendientes de la clase View, y por tanto, pueden ser definidas utilizando código Java. Sin embargo, lo habitual será definir las vistas utilizando un fichero XML y dejar que el sistema cree los objetos por nosotros a partir de este fichero. Esta forma de trabajar es muy similar a la definición de una página web utilizando código HTML (Universidad Politécnica de Valencia, 2018b).

Layout

Un *layout* es un conjunto de vistas agrupadas de una determinada forma. Vamos a disponer de diferentes tipos de *layouts* para organizar las vistas de forma lineal, en

cuadrícula o indicando la posición absoluta de cada vista. Los *layouts* también son objetos descendientes de la clase View. Igual que las vistas, los *layouts* pueden ser definidos en código, aunque la forma habitual de definirlos es utilizando código XML (Universidad Politécnica de Valencia, 2018b).

Fragment

Un fragment está formado por la unión de varias vistas para crear un bloque funcional de la interfaz de usuario. Una vez creados los fragments, podemos combinar uno o varios fragments dentro de una actividad, según el tamaño de pantalla disponible. El uso de fragments puede ser algo complejo, sobre todo si no se maneja bien conceptos como actividad, vista y *layout* (Universidad Politécnica de Valencia, 2018b).

Este componente ha sido de gran utilidad a la hora de desarrollar la interfaz y los menús de la app desarrollada gracias a la flexibilidad que ofrece. El hecho de poder combinar varios *fragments* en una sola actividad, supone un ahorro de memoria y de tiempo de ejecución significativo, además, ofrece cierta libertad estética que no se podría conseguir con el uso exclusivo de *activities* para cada sección, menú o sub menú de la interfaz.

Broadcast Intent Receiver

Un componente Broadcast Intent Receiver se utiliza para lanzar alguna ejecución dentro de la aplicación actual cuando un determinado evento se produzca (generalmente, abrir un componente Activity*)*. Por ejemplo, una llamada entrante o un SMS recibido. Este componente no tiene interfaz de usuario asociada, pero puede utilizar el API *Notification Manager* para avisar al usuario del evento producido a través de la barra de estado del dispositivo móvil. Este componente se implementa a través de una clase de nombre BroadcastReceiver. Para que Broadcast Intent Receiver funcione, no es necesario que la aplicación en cuestión sea la aplicación activa en el momento de producirse el evento. El sistema lanzará la aplicación si es necesario cuando el evento monitorizado tenga lugar (Universidad Carlos III, 2015).

Service

Un componente Service representa una aplicación ejecutada sin interfaz de usuario, y que generalmente tiene lugar en segundo plano mientras otras aplicaciones (éstas con interfaz) son las que están activas en la pantalla del dispositivo. Un ejemplo típico de este componente es un reproductor de música. La interfaz del reproductor muestra al usuario las distintas canciones disponibles, así como los típicos botones de reproducción, pausa, volumen, etc. En el momento en el que el usuario reproduce una canción, ésta se escucha mientras se siguen visualizando todas las acciones anteriores, e incluso puede ejecutar una aplicación

distinta sin que la música deje de sonar. La interfaz de usuario del reproductor sería un componente Activity, pero la música en reproducción sería un componente Service, porque se ejecuta en *background*. Este elemento está implementado por la clase de mismo nombre Service (Universidad Carlos III, 2015).

Content Provider

Con el componente Content Provider, cualquier aplicación en Android puede almacenar datos en un fichero, en una base de datos SQLite o en cualquier otro formato que considere. Además, estos datos pueden ser compartidos entre distintas aplicaciones. Una clase que implemente el componente Content Provider contendrá una serie de métodos que permiten almacenar, recuperar, actualizar y compartir los datos de una aplicación (Universidad Carlos III, 2015).

3.4.2 Ciclo de vida de las aplicaciones Android

En Android, cada aplicación se ejecuta en su propio proceso. El sistema se encarga de lanzar y parar todos los procesos, gestionar su ejecución y decidir qué hacer en función de los recursos disponibles y de las órdenes dadas por el usuario. Esta gestión hecha por Android es transparente al usuario, ya que, este último, de lo único que es consciente es de pasar de una aplicación a otra y que puede volver a cualquiera de ellas en el momento que lo desee. En Android es posible ejecutar tantos procesos como permitan los recursos del sistema. Cada proceso está formado por una o varias Activities independientes de esa aplicación. Cuando se realiza un cambio de una aplicación a otra o simplemente se ejecuta una nueva aplicación, el sistema duerme el proceso y realiza una copia de su estado para poder recuperarlo más tarde. El proceso y la actividad siguen existiendo en el sistema, pero están dormidos y su estado ha sido guardado. Es entonces cuando crea, o reanuda en caso de que ya exista previamente, el proceso para la aplicación que debe ser lanzada, siempre y cuando haya suficientes recursos para llevar a cabo dicha tarea. Cada uno de los componentes básicos de Android tiene un ciclo de vida bien definido; esto implica que el desarrollador puede controlar en cada momento en qué estado se encuentra dicho componente, pudiendo así programar las acciones que mejor convengan (Universidad Carlos III, 2015).

Una actividad normalmente puede estar en uno de los siguientes tres estados (Android Developers, 2019e):

- Reanudada (Resumed): la actividad se encuentra en el primer plano de la pantalla y tiene la atención del usuario (a veces, este estado también se denomina "en ejecución").

- Pausada (Paused): otra actividad se encuentra en el primer plano y tiene la atención del usuario, pero esta todavía está visible. Es decir, otra actividad está visible por encima de esta y esa actividad es parcialmente transparente o no cubre toda la pantalla. Una actividad pausada está completamente "viva" (el objeto Activity se conserva en la memoria, mantiene toda la información de estado y miembro y continúa anexado al administrador de ventanas), pero el sistema puede eliminarla en situaciones en que la memoria sea extremadamente baja.

- Detenida (Stopped): la actividad está completamente opacada por otra actividad (ahora la actividad se encuentra en "segundo plano"). Una actividad detenida también permanece "viva" (el objeto *Activity* se conserva en memoria, mantiene toda la información de estado y miembro, pero no está anexado al administrador de ventanas). Sin embargo, ya no está visible para el usuario y el sistema puede eliminarla cuando necesite memoria en alguna otra parte.

Si se pausa o se detiene una actividad, el sistema puede excluirla de la memoria al solicitarle que se cierre (llamando a su método *finish*()), o simplemente eliminando su proceso. Cuando se vuelve a abrir la actividad (después de haber sido eliminada o destruida), es necesario crearla nuevamente.

El ciclo de vida completo de una actividad está definido por una serie de métodos llamados *callback*. A continuación, se describen los posibles ciclos de una actividad y los métodos *callback* que intervienen en cada ciclo (Android Developers, 2019e):

- El ciclo de vida completo de una actividad transcurre entre la llamada a *onCreate()* y la llamada a *onDestroy().* Tu actividad debe configurar el estado "global" (como la definición del diseño) en *onCreate(),* y liberar todos los recursos restantes en *onDestroy().* Por ejemplo, si hay un subproceso de descarga de datos de la red en ejecución en segundo plano en tu actividad, esta podría crear ese subproceso en *onCreate()* y luego detenerlo en *onDestroy().*

- El ciclo de vida visible de una actividad transcurre entre la llamada a *onStart()* y la llamada a *onStop().* Durante ese tiempo, el usuario puede ver la actividad en pantalla e interactuar con ella. Por ejemplo, se llama a *onStop()* cuando se inicia una nueva actividad y esta ya no está visible. Entre estos dos métodos, puedes conservar los recursos necesarios para mostrarle la actividad al usuario. Por ejemplo, puedes registrar un BroadcastReceiver en *onStart()* para controlar los cambios que afecten tu IU y anular su registro en *onStop()* cuando el usuario ya no pueda ver lo que muestras. El sistema podría llamar a *onStart()* y *onStop()* muchas veces durante el ciclo de vida completo de la actividad, ya que la actividad pasa de ser visible y a estar oculta para el usuario.

- El ciclo de vida en primer plano de una actividad transcurre entre la llamada a *onResume()* y la llamada a *onPause().* Durante ese tiempo, la actividad se encuentra al frente de todas las demás actividades en la pantalla y tiene el foco en la interacción del usuario. Con frecuencia, una actividad puede entrar y salir de primer plano, por ejemplo, se llama a *onPause()* cuando el dispositivo entre en suspensión o cuando aparece un diálogo. Dado que este estado puede cambiar con frecuencia, el código en estos métodos debe ser bastante liviano para evitar las transiciones lentas que hacen que el usuario deba esperar.

Dichos ciclos de vida y métodos callback se pueden observar en el siguiente esquema (Android Developers, 2019e).

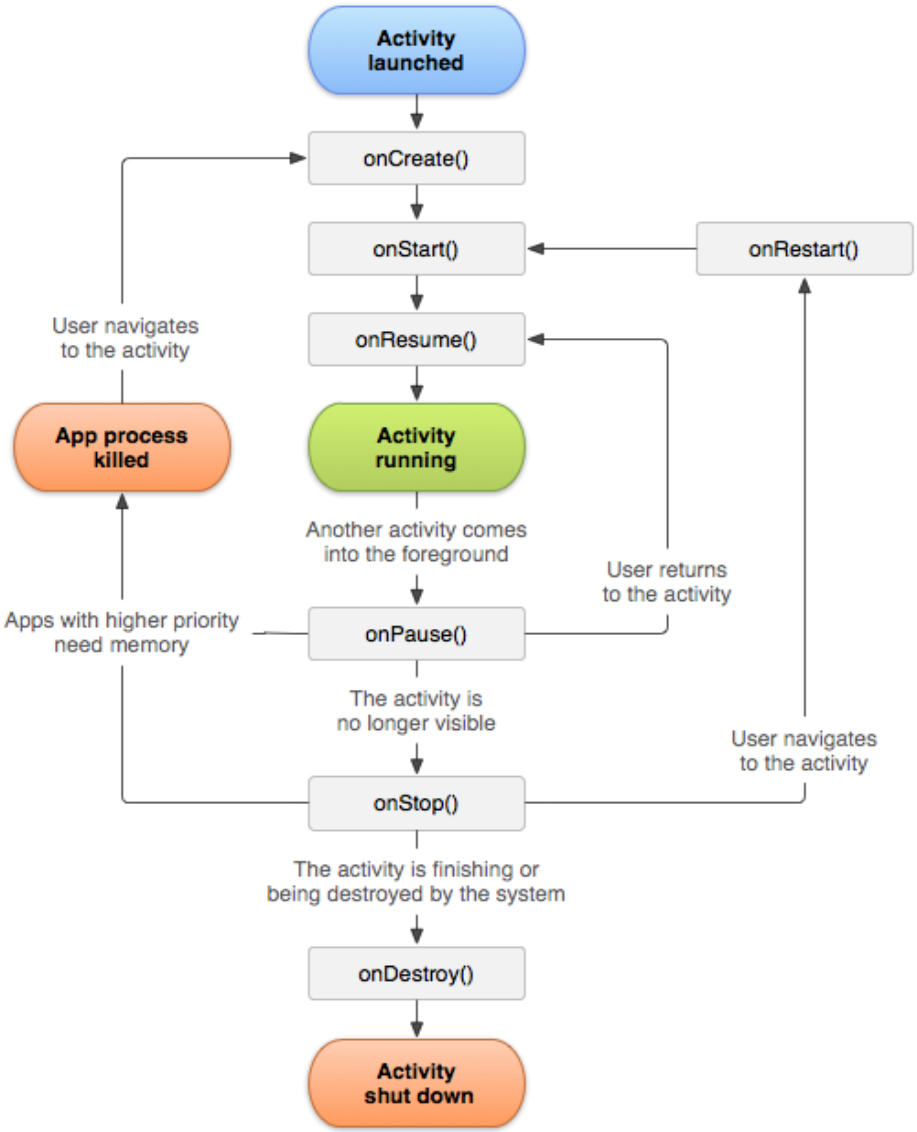

Figura 19 - Ciclo de vida de una Activity (Android Developers, 2019e)

Los mismos métodos callback del ciclo de vida se indican en la siguiente tabla, que describe cada uno de los métodos callback en más detalle y ubica a cada uno dentro del ciclo de vida total de la actividad, incluido si el sistema puede eliminar la actividad después de que se completa el método callback (Android Developers, 2019e).

Método	Descripción	¿Se puede eliminar después?	A continuación
onCreate()	Recibe una llamada cuando se crea la actividad por primera vez. Aquí es donde debes realizar todas las configuraciones estáticas normales: crear vistas, enlazar datos con listas, etc. A este método se le pasa un objeto agrupado que contiene el estado anterior de la actividad, si se hubiera capturado ese estado. Siempre seguido por onStart().	No	onStart()
onRestart()	Recibe una llamada después de que se detiene la actividad, justo antes de que vuelva a iniciarse. Siempre seguido por onStart().	No	onStart()
onStart()	Recibe una llamada justo antes de que la actividad se vuelva visible para el usuario. Seguido por onResume() si la actividad pasa a primer plano, o por onStop() si se oculta.	No	onResume() o onStop()
onResume()	Recibe una llamada justo antes de que la actividad comience a interactuar con el usuario. En este momento la actividad se encuentra en la parte superior de la pila de actividades, y recibe las entradas del usuario. Siempre seguido por onPause().	No	onPause()
onPause()	Recibe una llamada cuando el sistema está a punto de reanudar otra actividad. Este método generalmente se usa para confirmar los cambios sin guardar como datos persistentes, para detener animaciones y otras tareas que podrían estar consumiendo CPU, etc. Lo que sea que haga, debe hacerlo muy rápido porque la siguiente actividad no se reanudará hasta que el método regrese.	Sí	onResume() o onStop()

	Seguido por onResume() si la actividad vuelve al primer plano, o por onStop() si se vuelve invisible para el usuario.		
onStop()	Recibe una llamada cuando la actividad ya no es visible para el usuario. Esto puede ocurrir porque se la destruyó o porque se reanudó otra actividad (ya sea una actividad existente o una nueva) y la está cubriendo. Seguido por onRestart() si la actividad vuelve a interactuar con el usuario, o por onDestroy() si la actividad desaparece.	Sí	onRestart() o onDestroy()
onDestroy()	Recibe una llamada antes de que se destruya la actividad. Esta es la última llamada que recibirá la actividad. Se puede llamar a este método porque la actividad está finalizando (porque se llamó a finish() para esa actividad), o porque el sistema destruye temporalmente esa instancia de la actividad para ahorrar espacio. Se puede diferenciar estos dos escenarios con el método isFinishing().	Sí	*nada*

Tabla 2 - Resumen de los métodos de callback del ciclo de vida de una actividad

3.4.3 Elección del entorno de desarrollo

Existe una gran variedad de entornos de desarrollo en el mundo del desarrollo de software, sin embargo, cuando se trata de Android, hay dos grandes entornos que dominan este campo, se trata de Android Studio y Eclipse. A continuación, se realiza un análisis a estos dos entornos y en función de ellos se va a decidir cuál de los dos nos conviene más a la hora de desarrollar la APP.

Antes de seguir, convendría aclarar el concepto de **IDE**, que no es más que un programa informático compuesto por un conjunto de herramientas de programación. Es un entorno de programación empaquetado como un programa o aplicación, que nos provee de un marco de trabajo agradable para la mayoría de los lenguajes de programación (Academiaandroid, 2014).

Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Fue presentado por Google el 16 de mayo del 2013 en el congreso de desarrolladores Google I/O, con el objetivo de crear un entorno dedicado en exclusiva a la programación de aplicaciones para dispositivos Android, proporcionando a Google un mayor control sobre el proceso de producción. Se trata pues de una alternativa real a Eclipse, el IDE recomendado por Google hasta la fecha, pero que presentaba problemas debido a su lentitud en el desarrollo de versiones que solucionaran las carencias actuales ya que Eclipse es una plataforma de desarrollo, diseñada para ser extendida a través de plugins (Academiaandroid, 2014).

Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan la productividad durante la compilación de apps para Android, como las siguientes (Android Developers, 2019f):

- Un sistema de compilación basado en Gradle flexible.
- Un emulador rápido con varias funciones.
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.
- Instant Run para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK.
- Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código.
- Gran cantidad de herramientas y frameworks de prueba.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK.
- Soporte incorporado para Google Cloud Platform, lo que facilita la integración de Google Cloud Messaging y App Engine.

Estructura de un proyecto en Android Studio

La estructura de un proyecto en Android studio presenta un aspecto similar al que se muestra en la siguiente figura (Android Developers, 2019i):

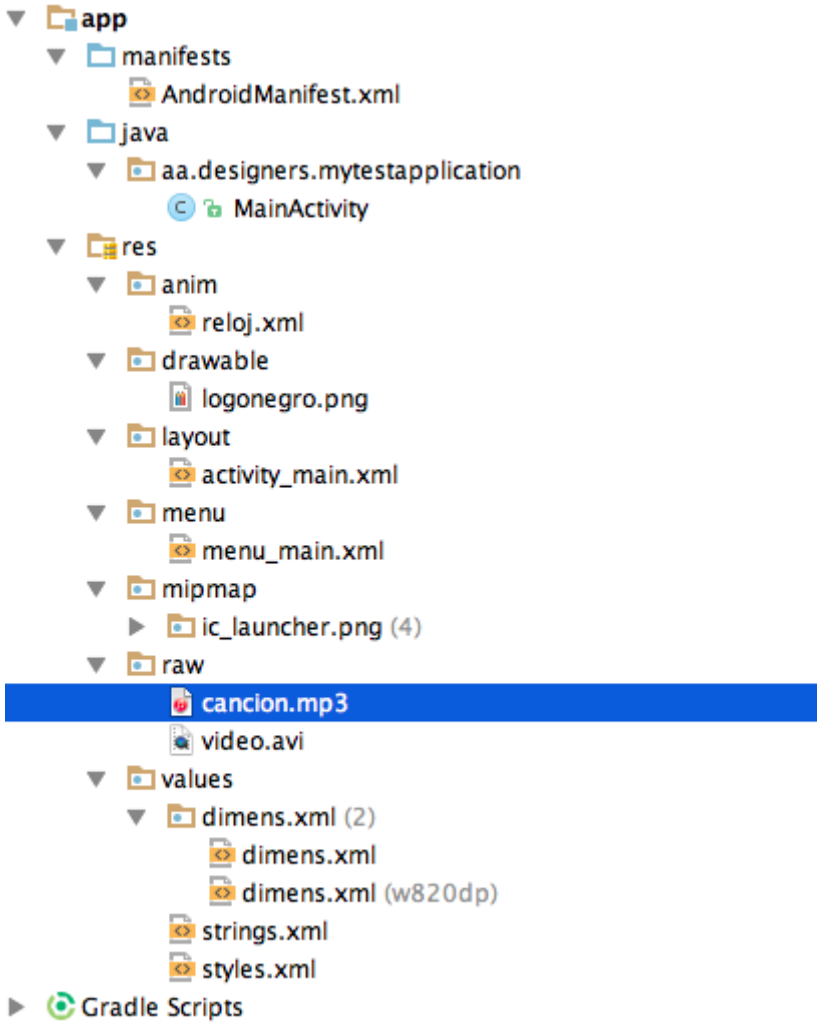

Figura 20 - Estructura de un proyecto en Android Studio (Android Developers, 2019i)

En la figura anterior, se encuentran una serie de componentes que forman parte de un proyecto de Android Studio que se explican a continuación (Sgoliver, 2015a):

Carpeta /app/src/main/java

Esta carpeta contendrá todo el código fuente de la aplicación, clases auxiliares, etc. Inicialmente, Android Studio creará el código básico de la pantalla (*actividad* o *activity*) principal de la aplicación, MainActivity, y siempre bajo la estructura del paquete java definido durante la creación del proyecto.

Carpeta /app/src/main/res/

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, *layouts*, cadenas de texto, etc.

Los diferentes tipos de recursos se pueden distribuir entre las siguientes subcarpetas:

Carpeta	Descripción
/res/drawable/	Contiene las imágenes y otros elementos gráficos usados por la aplicación. Para poder definir diferentes recursos dependiendo de la resolución y densidad de la pantalla del dispositivo, se suele dividir en varias subcarpetas: • /drawable (recursos independientes de la densidad) • /drawable-ldpi (densidad baja) • /drawable-mdpi (densidad media) • /drawable-hdpi (densidad alta) • /drawable-xhdpi (densidad muy alta) • /drawable-xxhdpi (densidad muy muy alta)
/res/mipmap/	Contiene los iconos de lanzamiento de la aplicación (el icono que aparecerá en el menú de aplicaciones del dispositivo) para las distintas densidades de pantalla existentes. Al igual que en el caso de las carpetas /drawable, se dividirá en varias subcarpetas dependiendo de la densidad de pantalla: • /mipmap-mdpi • /mipmap-hdpi • /mipmap-xhdpi • …
/res/*layout*/	Contiene los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica. Para definir distintos *layouts* dependiendo de la orientación del dispositivo se puede dividir también en subcarpetas: • /*layout* (vertical) • /*layout*-land (horizontal)
/res/anim/ **/res/animator/**	Contienen la definición de las animaciones utilizadas por la aplicación.
/res/color/	Contiene ficheros XML de definición de listas de colores según estado.

/res/menu/	Contiene la definición XML de los menús de la aplicación.
/res/xml/	Contiene otros ficheros XML de datos utilizados por la aplicación.
/res/raw/	Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.
/res/values/	Contiene otros ficheros XML de recursos de la aplicación, como por ejemplo cadenas de texto (*strings.xml*), estilos (*styles.xml*), colores (*colors.xml*), arrays de valores (*arrays.xml*), tamaños (*dimens.xml*), etc.

Tabla 3 - Subcarpetas de los diferentes tipos de recursos de un proyecto Android

Fichero /app/src/main/AndroidManifest.xml

Contiene la definición en XML de muchos de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, icono, …), sus componentes (pantallas, servicios, …), o los permisos necesarios para su ejecución.

Fichero /app/build.gradle

Contiene información necesaria para la compilación del proyecto, por ejemplo, la versión del SDK de Android utilizada para compilar, la mínima versión de Android que soportará la aplicación, referencias a las librerías externas utilizadas, etc.

En un proyecto pueden existir varios ficheros *build.gradle*, para definir determinados parámetros a distintos niveles. Podemos ver que existe un fichero *build.gradle* a nivel de proyecto, y otro a nivel de módulo dentro de la carpeta /app. El primero de ellos definirá parámetros globales a todos los módulos del proyecto, y el segundo sólo tendrá efecto para cada módulo en particular.

Carpeta /app/libs

Puede contener las librerías java externas (ficheros *.jar*) que utilice nuestra aplicación. Normalmente no incluiremos directamente aquí ninguna librería, sino que haremos referencia a ellas en el fichero *build.gradle* descrito en el punto anterior, de forma que entren en el proceso de compilación de nuestra aplicación.

Carpeta /app/build/

Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que compilamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente java dirigidos, entre otras muchas cosas, al control de los recursos de la aplicación. Dado que estos ficheros se generan automáticamente tras cada compilación del proyecto es importante que no se modifiquen manualmente bajo ninguna circunstancia.

Eclipse

Eclipse es un entorno de desarrollo, de código abierto y gratuito, cuyo diseño sigue un patrón de actualización basado en plugins. Su objetivo es convertirse en una plataforma de integración de herramientas de desarrollo. Es un IDE que podríamos denominar genérico, ya que no fue concebido para ser utilizado con un solo lenguaje de programación, sino que es compatible con una gran variedad de lenguajes (Academiaandroid, 2014).

Podríamos destacar de Eclipse los siguientes aspectos o características (Academiaandroid, 2014):

- Gestión de Proyectos: el desarrollo sobre Eclipse se basa en proyectos, que son un conjunto de recursos relacionados entre sí, como pueden ser el código fuente, documentación, ficheros, etc.
- Depurador de Código: Eclipse incluye un potente depurador de código, fácil e intuitivo, que nos proporciona de forma gráfica una opción de mejorar nuestros proyectos. Dispone de una perspectiva dedicada a la depuración donde podremos realizar y supervisar dicha tarea.
- Perspectivas, Editores y Vistas: en Eclipse el concepto de trabajo se basa en las perspectivas, que son una preconfiguración de ventanas y editores que nos permiten trabajar en un determinado entorno de trabajo de forma óptima.
- Colección de Plugins: están disponibles una gran cantidad de plugins, tanto desarrollados por Eclipse como de terceros. Los hay de pago y gratuitos con diversas licencias. Actualmente el número de ellos es muy

alto, rondando los 1.280 plugins que pueden aumentar las funcionalidades del IDE

Estructura de un proyecto Android en Eclipse

En la figura 21, se puede observar un ejemplo de la estructura de un proyecto Android en Eclipse, y cuyos componentes se explican a continuación (Sgoliver, 2015b):

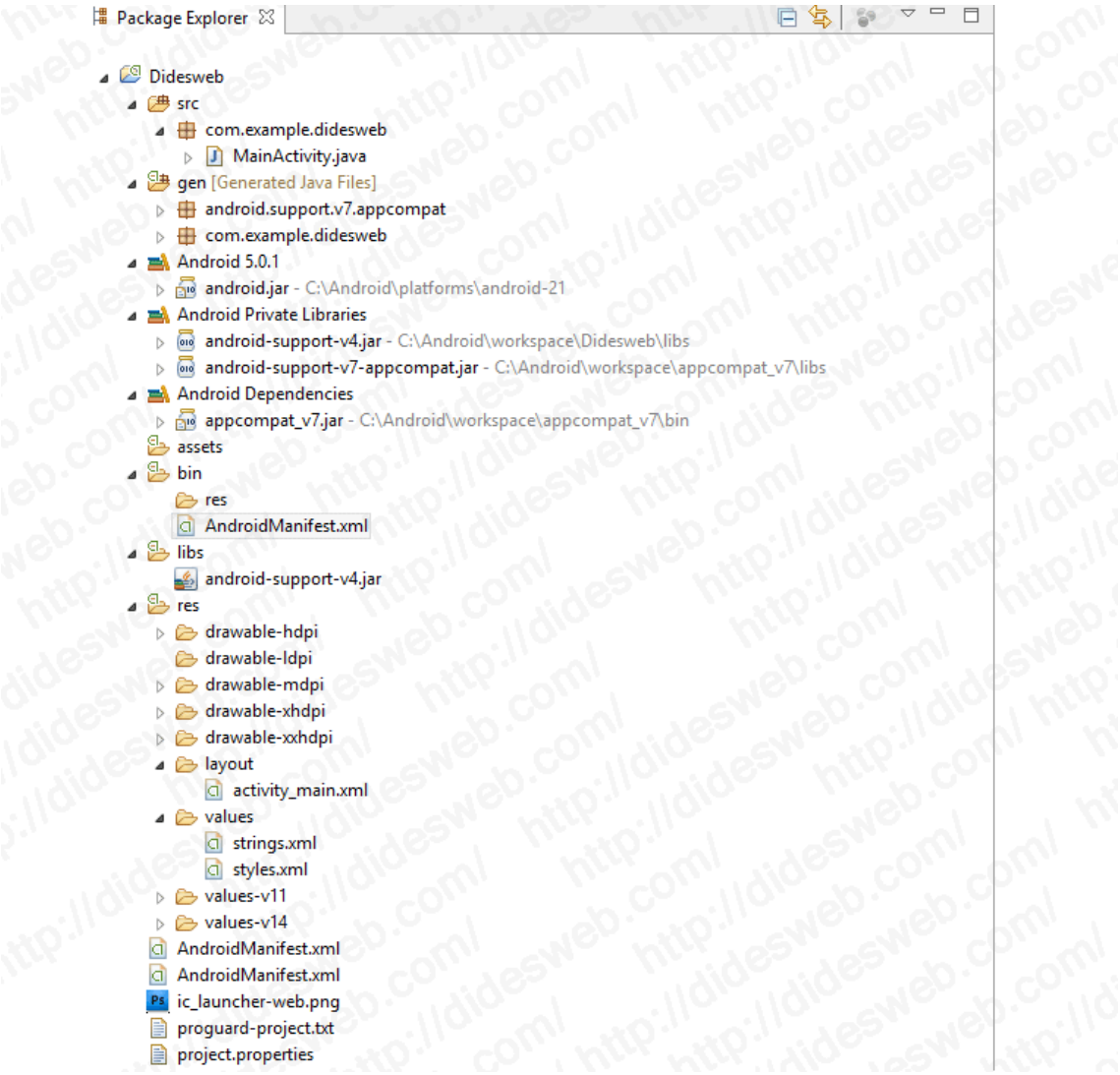

Figura 21 - Estructura de un proyecto Android en Eclipse (Sgoliver, 2015b)

Carpeta /src/

Esta carpeta contendrá todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc. Inicialmente, Eclipse creará por nosotros el

código básico de la pantalla (Activity) principal de la aplicación y siempre bajo la estructura del paquete java definido.

Carpeta /res/

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc. Los diferentes tipos de recursos se distribuyen de la misma forma que en el caso de Android studio, que se podrá ver en la Tabla 3 vista anteriormente.

Carpeta /gen/

Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que generamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente java dirigidos al control de los recursos de la aplicación.

Carpeta /assets/

Contiene todos los demás ficheros auxiliares necesarios para la aplicación (y que se incluirán en su propio paquete), como por ejemplo ficheros de configuración, de datos, etc. La diferencia entre los recursos incluidos en la carpeta /res/raw/ y los incluidos en la carpeta /assets/ es que para los primeros se generará un ID en la clase R y se deberá acceder a ellos con los diferentes métodos de acceso a recursos. Para los segundos sin embargo no se generarán ID y se podrá acceder a ellos por su ruta como a cualquier otro fichero del sistema. Usaremos uno u otro según las necesidades de nuestra aplicación.

Carpeta /bin/

Esta es otra de esas carpetas que en principio no tendremos por qué tocar. Contiene los elementos compilados de la aplicación y otros ficheros auxiliares. Cabe destacar el fichero con extensión ".apk", que es el ejecutable de la aplicación que se instalará en el dispositivo.

Carpeta /libs/

Contendrá las librerías auxiliares, normalmente en formato ".jar" que utilicemos en nuestra aplicación Android.

Fichero AndroidManifest.xml

Contiene la definición en XML de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, versión, icono, …), sus componentes (pantallas, mensajes, …), las librerías auxiliares utilizadas, o los permisos necesarios para su ejecución.

Comparativa entre Android Studio y ADT Eclipse

Para una mayor comprensión de las diferencias y novedades que presenta Android Studio con respecto al IDE Eclipse, y más concretamente con el ADT para Android, se propone la siguiente tabla comparativa entre ambas opciones (Academiaandroid, 2015):

Características	Android Studio	ADT
Sistema de construcción	Gradle	ANT
Construcción y gestión de proyectos basado en Maven (herramienta de software para la gestión y construcción de proyectos Java, similar a Apache ANT, pero su modelo es más simple ya que está basado en XML)	Si	No (es necesario instalar un plugin auxiliar)
Construir variantes y generación de múltiples APK (muy útil para Android Wear)	Si	No
Refactorización y completado avanzado de código Android	Si	No
Diseño del editor gráfico	Si	Si
Firma APK y gestión de almacén de claves	Si	Si
Soporte para NDK (Native Development Kit: herramientas para implementar código nativo escrito en C y C++)	Próximas versiones	Si
Soporte para Google Cloud Platform	Si	No
Vista en tiempo real de renderizado de *layouts*	Si	No
Nuevos módulos en proyecto	Si	No
Editor de navegación	Si	No
Generador de assets	Si	No
Datos de ejemplo en diseño de *layout* (sin renderizar en tiempo de ejecución)	Si	No
Visualización de recursos desde editor de código	Si (a la izquierda de la línea de	No

	asignación del recurso)	

Tabla 4 - Comparativa de las características de Android Studio frente a ADT Eclipse

Basándonos en la tabla anterior, se podría anticipar que Android estudio nos ofrecería más posibilidades y una mayor flexibilidad y comodidad a la hora de desarrollar nuestra APP, aunque sería necesario tener en cuenta otros aspectos que puedan facilitarnos la tarea de decidir cuál de los dos IDE utilizar para nuestra APP.

Ventajas de Android Studio frente a Eclipse (Academiaandroid, 2015):

- Android Studio se convirtió en el entorno recomendado para el desarrollo de aplicaciones en Android, al tratarse de un IDE oficial de Google en colaboración con JetBrains (compañía de desarrollo software especializada en diseño de IDEs).
- Android Studio permite la creación de nuevos módulos dentro de un mismo proyecto, sin necesidad de estar cambiando de espacio de trabajo para el manejo de proyectos, algo habitual en Eclipse.
- Con la simple descarga de Android Studio se disponen de todas las herramientas necesarias para el desarrollo de aplicaciones para la plataforma Android.
- Su nueva forma de construir los paquetes .apk, mediante el uso de Gradle, proporciona una serie de ventajas más acorde a un proyecto Java:
    - Facilita la distribución de código, y por lo tanto el trabajo en equipo.
    - Reutilización de código y recursos.
    - Permite compilar desde línea de comandos, para aquellas situaciones en las que no esté disponible un entorno de desarrollo.
    - Mayor facilidad para la creación de diferentes versiones de la misma aplicación, que proporciona numerosas ventajas como puede ser la creación de una versión de pago y otra gratuita, o por ejemplo diferentes dispositivos o almacén de datos.

Desventajas de Android Studio frente a Eclipse (Academiaandroid, 2015):

- Al estar en una fase de desarrollo constante, siempre es susceptible de introducirse más cambios que puedan provocar inestabilidad entre proyectos de diferentes versiones.
- Curva de aprendizaje más lenta para nuevos desarrolladores de Android.
- El sistema de construcción de proyectos Gradle puede resultar complicado inicialmente.
- En comparativa con Eclipse, menor número de plugins.

Considerando las ventajas y desventajas de ambos IDEs, se ha decantado finalmente por **Android Studio** como entorno de desarrollo para la APP.

Capítulo 4 – Aplicación socio sanitaria. Descripción Funcional

Capítulo 4 – Aplicación socio sanitaria. Descripción Funcional

En este capítulo se pretende describir las funcionalidades de la aplicación, así como los tipos de usuarios de la aplicación y sus roles. Todo ello descrito desde un punto de vista funcional.

4.1 Introducción

La aplicación desarrollada se puede dividir en dos grandes bloques: **Seguimientos** y **Juegos**. Cada uno de estos bloques va dirigido a uno de los tipos de usuario de la aplicación. El bloque **Seguimientos** va dirigido al tipo de usuario llamado **Voluntarios** y el bloque Juegos va dirigido al tipo de usuario llamado **Tutelados** (a través de los Voluntarios, que son los que inician sesión para poder usar la app)**.** En la Figura 22 queda reflejada la división descrita anteriormente.

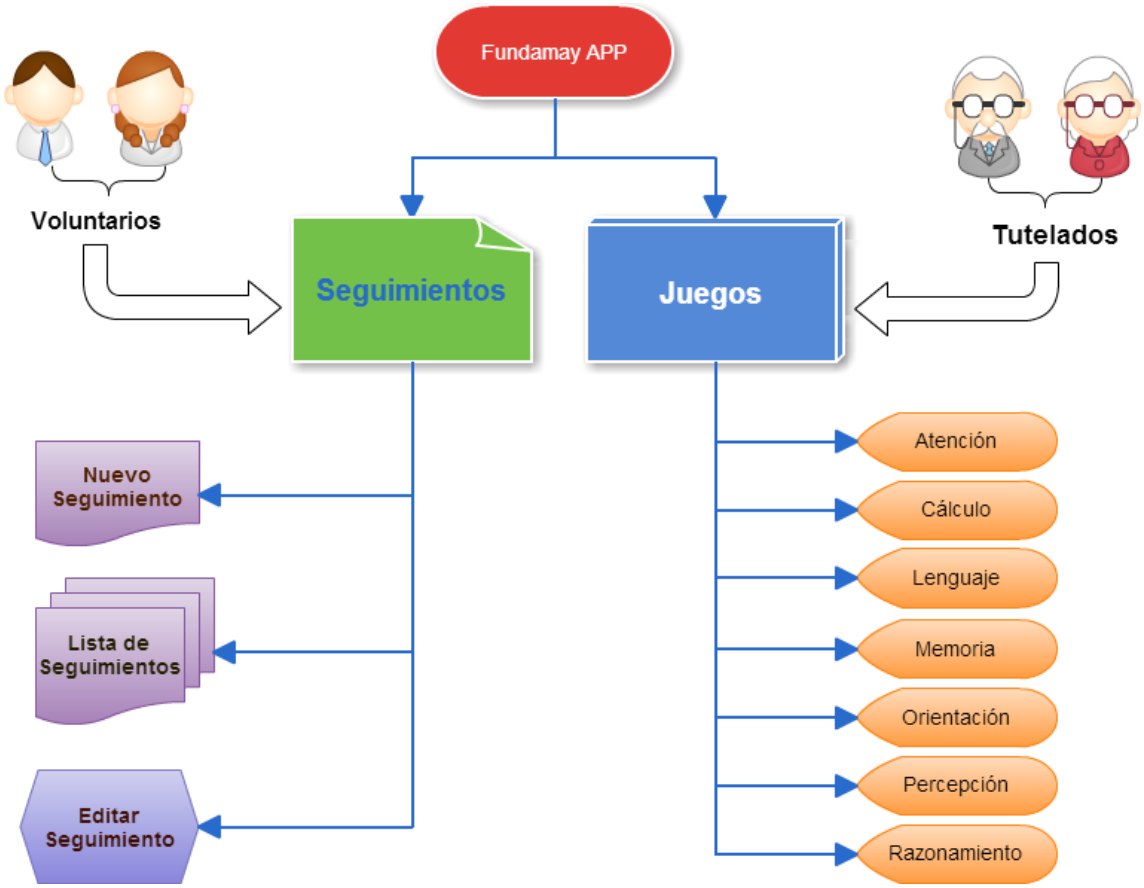

Figura 22 - Esquema de los Bloques de los que se compone la APP

A continuación, se procede a describir a los tipos de usuarios de la aplicación, cada uno de los bloques a los que acceden, los sub bloques de cada bloque y sus componentes.

4.2 Voluntarios

Tal y como ya se ha dicho en la introducción del presente TFG, los voluntarios, son personas que, de forma comprometida, asumen aquellos aspectos de la tutela que son difíciles de garantizar desde las fundaciones como la cercanía, **el seguimiento y acompañamiento puntual de las actividades de su vida cotidiana**. Y como tales, son uno de los dos tipos de usuarios de la aplicación desarrollada y tendrán dos grandes funciones que realizar con la ayuda de la aplicación desarrollada. Por una parte, realizarán el **seguimiento de las personas tuteladas** y por otra, ayudarán a los tutelados en sus actividades de vida cotidiana proponiéndoles la **realización de algunos de los juegos y ejercicios de la aplicación**. En la Figura 23 se pueden ver las funciones de los voluntarios.

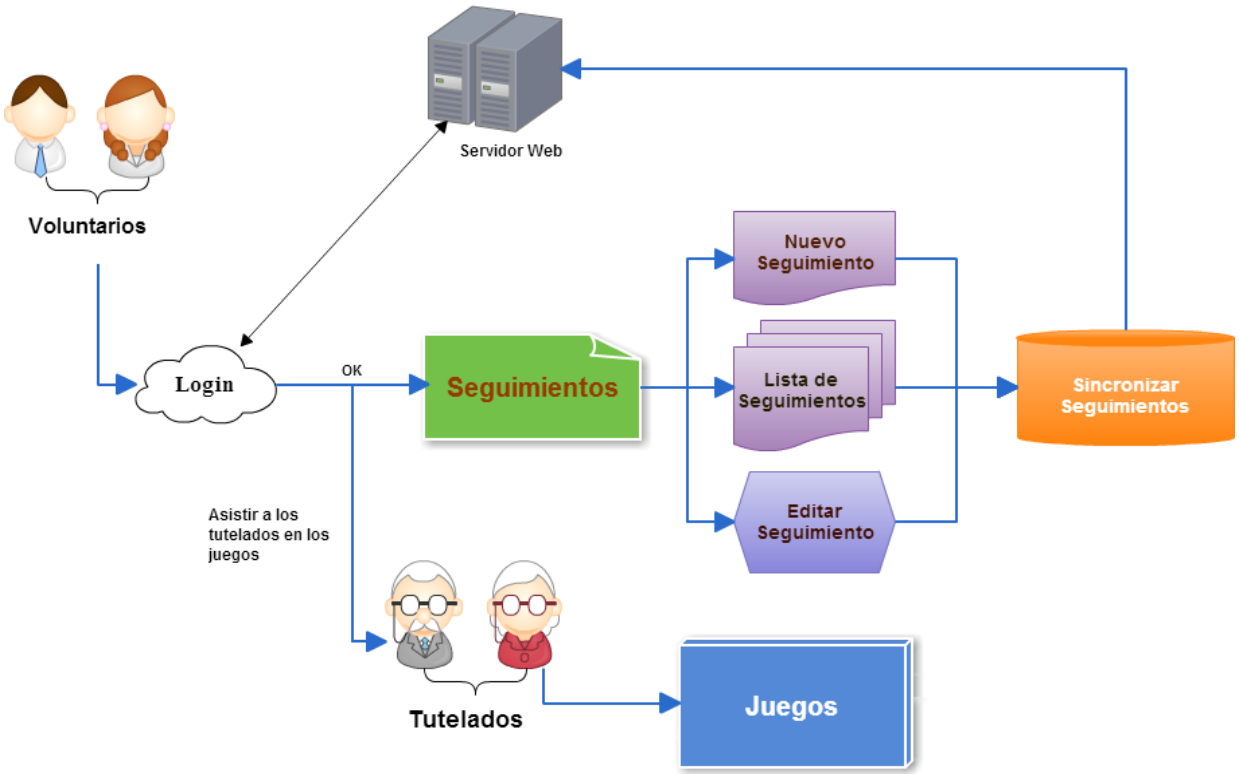

Figura 23 - *Esquema de las funciones del Voluntario*

Los voluntarios serán los usuarios principales de la aplicación, pues serán ellos los poseedores de los dispositivos móviles proporcionado por la fundación. Debido a que los tutelados son personas mayores con dificultades para manejar este tipo de dispositivos y, por tanto, esta aplicación, el papel de los voluntarios, además de realizar los seguimientos, será el de asistir a los tutelados, que tenga asignados, a la hora de

realizar los ejercicios de los juegos de la aplicación. Es decir, el voluntario, después de haber iniciado la aplicación y habiéndose identificado, ejecutará el juego correspondiente y le irá indicando al tutelado los pasos necesarios para realizar el ejercicio propuesto, ya sea leyendo los textos que el tutelado pueda tener dificultades al leer o pulsando los botones y demás controles que puedan presentar alguna dificultad.

4.3 Tutelados

Como ya hemos visto en el punto anterior, los voluntarios son los usuarios principales de la aplicación ya que recae sobre ellos una responsabilidad mayor. Sin embargo, los usuarios más importantes son los Tutelados, pues se espera que la aplicación les ayude lo máximo posible en su rehabilitación cognitiva. Para ello, se han desarrollado una serie de juegos pertenecientes a distintas categorías, y cada una de estas categorías pretende estimular la mente del tutelado para ejercitar un tipo de habilidades específicas, tales como son la memoria, el cálculo, la percepción…

El tutelado sólo podrá acceder a los juegos y solamente acompañado por voluntario que le asistirá y le ayudará en su uso. La razón de ello es que los tutelados son personas mayores que tienen ciertas dificultades, ya sean cognitivas o físicas, para realizar sus actividades de vida diaria. Lo cual, obliga a que estén siempre acompañados por los voluntarios a la hora de realizar los ejercicios propuestos en los juegos.

En la Figura 24 se puede ver un esquema de las funciones de los tutelados en la aplicación.

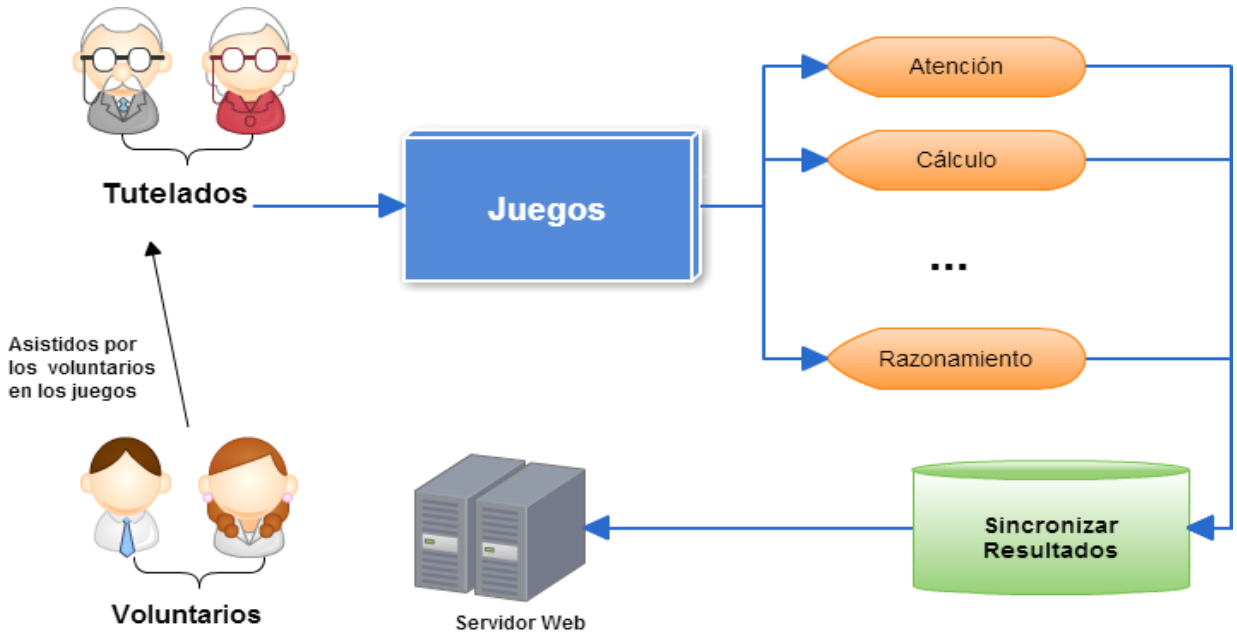

Figura 24 - Esquema de las funciones del Tutelado

4.4 Análisis de requisitos

Una vez vistos los usuarios que harán uso de la aplicación y las funcionalidades a las que podrán acceder. Se procede a enumerar los requisitos, funcionales y no funcionales, necesarios que se han tenido en cuenta para el desarrollo.

4.4.1 Requisitos funcionales

Los requisitos funcionales se definen para cada uno de los módulos descritos en el capítulo de Introducción, con el objetivo de descomponer el desarrollo del software en partes más pequeñas que faciliten su construcción.

Como complemento a la descripción textual de los requisitos funcionales del sistema, se incluyen diagramas UML de cada una de las secciones de la app.

Los requisitos funcionales de la aplicación son:

- **Seguimientos**:
    - **Seleccionar Tutelado**: permitir al voluntario seleccionar al tutelado para quien se quiera hacer seguimiento.
    - **Nuevo Seguimiento**: permitir al voluntario crear un nuevo seguimiento de un tutelado.
    - **Editar Seguimiento:** permitir al voluntario editar un seguimiento previamente guardado en la BDD.
    - **Ver Lista Seguimientos:** permitir al voluntario ver la lista de seguimientos del tutelado.
    - **Ver Seguimiento:** permitir al voluntario ver un seguimiento de la lista sin la posibilidad de alterarlo.
    - **Guardar Seguimiento:** la aplicación debe guardar el seguimiento en el servidor después de que éste haya sido creado o modificado y descargar los seguimientos cuando sean consultados por el usuario.

- **Juegos**:
    - **Seleccionar Tutelado**: permitir al voluntario seleccionar al tutelado para quien se quiera ejecutar un juego.
    - **Ejecutar Juego**: permitir al voluntario iniciar un nuevo juego para un tutelado.
    - **Jugar Partida**: permitir al tutelado jugar una partida de un juego elegido por el voluntario.

- **Jugar de nuevo**: permitir al voluntario iniciar una nueva partida del mismo juego.
- **Ver Resultados Partida:** permitir al voluntario corregir y/o ver los resultados de las partidas.

- **Usuarios**:
  - **Login**: permitir al usuario iniciar sesión con sus datos.
  - **Cerrar sesión**: permitir al usuario cerrar la sesión cuando quiera.
  - **Autentificación**: verificar si los datos de inicio de sesión son correctos y en caso de ser incorrectos, avisar al usuario mediante un mensaje de error.

4.4.2 Requisitos no funcionales

Los requisitos no funcionales para la aplicación, es decir, los que no especifican el comportamiento del sistema, son:

- **Rendimiento**: la aplicación debe desempeñar su función de una manera fluida. Se debe buscar la experiencia de uso más agradable para el usuario.
- **Accesibilidad**: la aplicación debe ser legible y tiene que seguir los patrones de accesibilidad de Google.
- **Usabilidad**: cualquier voluntario debe ser capaz de utilizar la aplicación y acceder a toda la funcionalidad sin ningún tipo de restricción.
- **Estabilidad**: la aplicación debe ser capaz de manejar los errores ocurridos durante la ejecución de la misma y avisando a este de la naturalidad del error.
- **Interfaz**: clara y concisa. No debe dar lugar a la confusión del usuario y debe seguir los estándares de diseño de interfaces de Material Design de Google.
- **Integración**: la aplicación debe de integrarse con todo el sistema operativo de Android. Hacer uso de las aplicaciones nativas si se necesita y mantener un diseño acorde al sistema.
- **Optimización**: el consumo de batería y de datos debe ser adecuado, y nunca dejar procesos sueltos que consuman memoria y batería. El tiempo de ejecución debe ser mínimo, para mejorar los tiempos de respuesta y la experiencia de uso del usuario.

4.5 Casos de uso

Una vez se han establecido los requisitos específicos del proyecto, el primer paso es definir con claridad los casos de uso.

Los casos de uso se utilizan para especificar el comportamiento deseado de un sistema o subsistema. Su especificación describe el conjunto de secuencias de acciones que se lleva a cabo en el sistema para producir un resultado para un actor. También, capturan el comportamiento deseado del sistema, sin especificar cómo se lleva a cabo dicho comportamiento.

4.5.1 Actores

Un actor comunica con el sistema o producto y es externo al sistema en sí. Por tanto, los actores podrían ser tanto personas, que interactúan con la aplicación, como otros sistemas o aplicaciones que de alguna forma comunican con ella.

En nuestro caso se definen dos tipos de actores:

- **Voluntario:** será la persona que ejecuta la aplicación en un dispositivo móvil, realizará los seguimientos y ayudará a un tutelado en el uso de los juegos.
- **Tutelado:** será la persona de edad avanzada que hará uso de los juegos y ejercicios con la ayuda del voluntario.

En los siguientes apartados, al igual que con los requisitos funcionales, los casos de uso también se han agrupado en función de la sección que contendría la funcionalidad de dicho caso de uso. Por cada módulo se detalla:

- Diagrama UML de los casos de uso con los actores y los tipos de relaciones entre los casos de uso y éstos.
- Descripción textual de cada caso de uso

4.5.2 Casos de uso de la sección de conexión

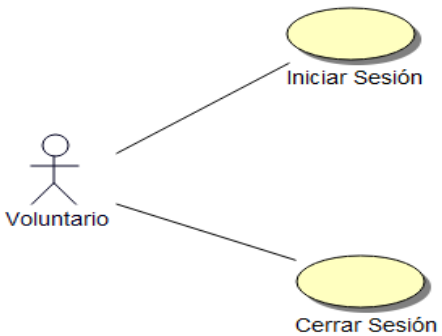

Figura 25 - Diagrama UML de la sección de conexión

Caso de uso 1	Iniciar Sesión
Resumen de la funcionalidad	Permite al usuario iniciar sesión y acceder a las funcionalidades de la aplicación.
Casos de uso relacionados	
Actores	Voluntario
Precondición	Los datos de autentificación del voluntario han de estar previamente registrados en la base de datos del servidor web.
Postcondición	Se mostrará la pantalla principal de la aplicación
Escenario de éxito	1. La aplicación muestra la pantalla de Iniciar sesión 2. El usuario introduce el correo electrónico y la contraseña y pulsa “Iniciar Sesión” 3. La aplicación accede al servidor, comprueba que los datos son correctos y se muestra la pantalla principal.
Extensiones	2a. El usuario introduce un correo electrónico no válido o no cumplimenta los campos 2a1. La aplicación muestra un mensaje de error al usuario 3a. La aplicación no puede conectar con el servidor y comprobar la identidad del usuario 3a1. La aplicación muestra un mensaje de error al usuario

Tabla 5 - Caso de uso Iniciar Sesión

Caso de uso 2	Cerrar Sesión
Resumen de la funcionalidad	Permite al usuario cerrar sesión y volver a la pantalla de Iniciar sesión
Casos de uso relacionados	Iniciar sesión
Actores	Voluntario
Precondición	El usuario debe haber accedido correctamente a la aplicación con sus datos de acceso.
Postcondición	Se cerrará la sesión y el usuario deberá introducir de nuevo sus datos de acceso para volver a acceder a la aplicación.

Escenario de éxito	1. El usuario pulsa “Salir” 2. Se cerrará la sesión y se vuelve a la pantalla de Iniciar sesión

Tabla 6 - Caso de uso Cerrar Sesión

4.5.3 Casos de uso de la sección de Seguimiento

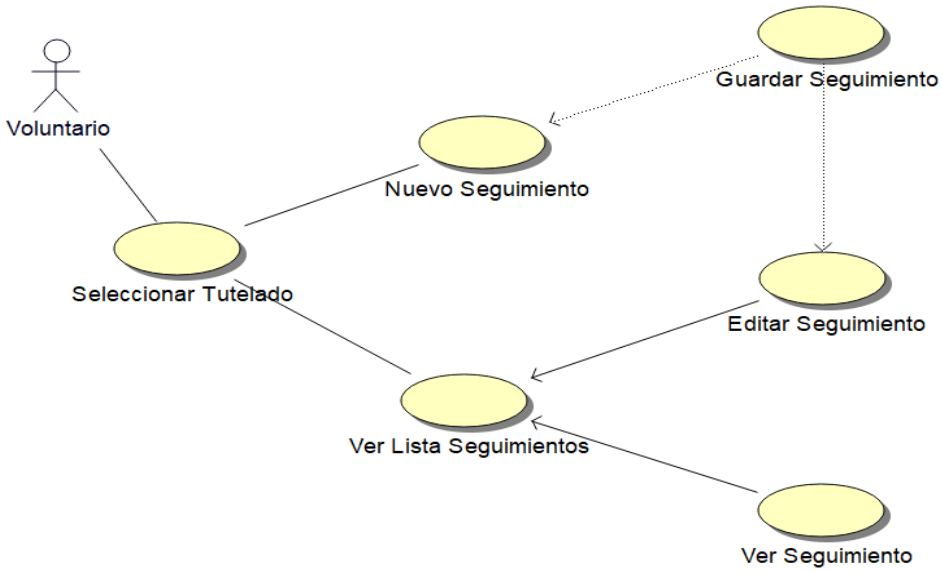

Figura 26 - Diagrama UML de la sección de Seguimiento

Caso de uso 1	**Seleccionar Tutelado**
Resumen de la funcionalidad	Permite al voluntario seleccionar un Tutelado a quien hacer el seguimiento
Casos de uso relacionados	
Actores	Voluntario
Precondición	Debe haberse descargado previamente la lista de tutelados y sus datos
Postcondición	
Escenario de éxito	1. Se carga la lista de tutelados 2. El voluntario selecciona un tutelado de la lista 3. Se muestran las opciones disponibles para hacer el seguimiento.

Tabla 7 - Caso de uso Seleccionar Tutelado (Sección de Seguimiento)

Caso de uso 2	Nuevo Seguimiento
Resumen de la funcionalidad	Permite al voluntario realizar un nuevo seguimiento del tutelado seleccionado.
Casos de uso relacionados	Seleccionar Tutelado, Guardar Seguimiento
Actores	Voluntario
Precondición	Se debe haber seleccionado el tutelado en el caso de uso “Seleccionar Tutelado”.
Postcondición	Una vez finalizado el seguimiento, se deberá seleccionar la opción guardar seguimiento
Escenario de éxito	1. El voluntario selecciona la opción “Nuevo Seguimiento” 2. Se muestra la pantalla de seguimiento 3. Se establece la fecha del seguimiento y se añade el texto correspondiente al seguimiento. 4. Se selecciona el botón de Guardar Seguimiento

Tabla 8 - Caso de uso Nuevo Seguimiento

Caso de uso 3	Ver Lista Seguimientos
Resumen de la funcionalidad	Permite al voluntario ver la lista de todos los seguimientos realizados del tutelado seleccionado.
Casos de uso relacionados	Seleccionar Tutelado
Actores	Voluntario
Precondición	Se debe haber seleccionado el tutelado en el caso de uso “Seleccionar Tutelado”.
Postcondición	
Escenario de éxito	1. El voluntario selecciona la opción “Ver Lista Seguimientos” 2. Se carga la lista de seguimientos desde el servidor 3. Se muestra en pantalla la lista de seguimientos del tutelado seleccionado.

Tabla 9 - Caso de uso Ver Lista Seguimientos

Caso de uso 4	Editar Seguimiento
Resumen de la funcionalidad	Permite al voluntario editar un seguimiento previamente realizado del tutelado seleccionado.
Casos de uso relacionados	Ver lista de Seguimientos, Guardar Seguimiento
Actores	Voluntario
Precondición	Se debe haber seleccionado el seguimiento a editar en el caso de uso “Ver lista de Seguimientos”.
Postcondición	Una vez finalizada la edición del seguimiento, se deberá seleccionar la opción guardar seguimiento
Escenario de éxito	1. El voluntario selecciona la opción “Editar Seguimiento” 2. Se carga el seguimiento a editar y se muestra la pantalla de editar seguimiento 3. Se establece la fecha de edición del seguimiento y se modifica el texto correspondiente al seguimiento. 4. Se selecciona el botón de Guardar Seguimiento

Tabla 10 - Caso de uso Editar Seguimiento

Caso de uso 5	Ver Seguimiento
Resumen de la funcionalidad	Permite al voluntario consultar un seguimiento previamente realizado del tutelado seleccionado.
Casos de uso relacionados	Ver lista de Seguimientos
Actores	Voluntario
Precondición	Se debe haber seleccionado el seguimiento a consultar en el caso de uso “Ver lista de Seguimientos”.
Postcondición	
Escenario de éxito	1. El voluntario selecciona la opción “Ver Seguimiento” 2. Se carga el seguimiento a consultar y se muestra la pantalla de ver seguimiento

Tabla 11 - Caso de uso Ver Seguimiento

Caso de uso 6	Guardar Seguimiento
Resumen de la funcionalidad	Permite al voluntario guardar un seguimiento del tutelado seleccionado.
Casos de uso relacionados	Editar Seguimiento, Nuevo Seguimiento
Actores	Voluntario
Precondición	
Postcondición	
Escenario de éxito **Extensiones**	1. El voluntario selecciona la opción “Guardar Seguimiento” 2. La aplicación se conecta al servidor para guardar los cambios realizados. 3. Se guarda el nuevo Seguimiento realizado. 3a. Se guardan los cambios realizados al seguimiento previamente realizado.

Tabla 12 - Caso de uso Guardar Seguimiento

4.5.4 Casos de uso de la sección de Juegos

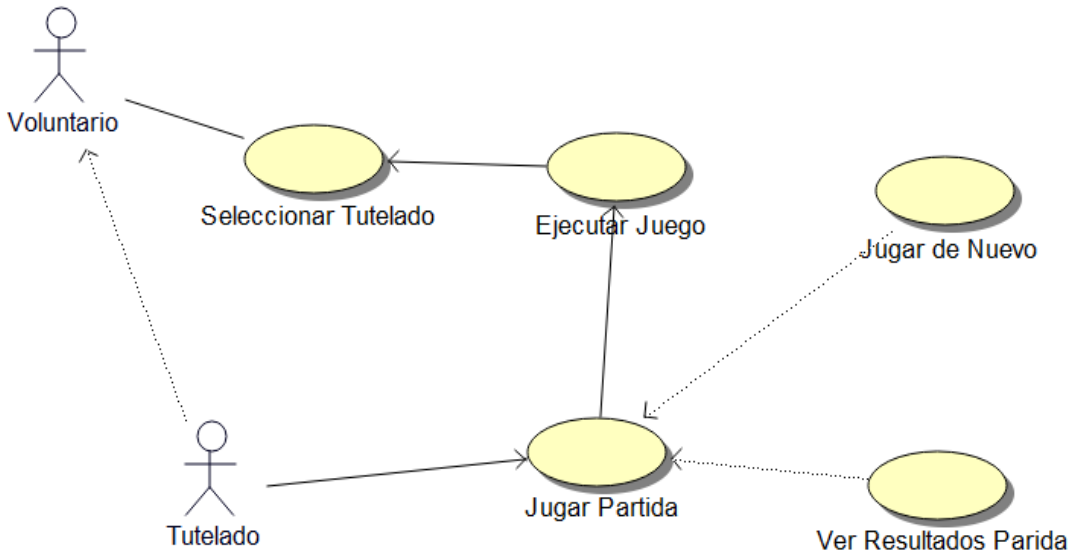

Figura 27 - Diagrama UML de la sección de Juegos

Caso de uso 1	Seleccionar Tutelado
Resumen de la funcionalidad	Permite al voluntario seleccionar un Tutelado para quien se desea ejecutar un juego.
Casos de uso relacionados	
Actores	Voluntario
Precondición	Debe haberse descargado previamente la lista de tutelados y sus datos
Postcondición	

Escenario de éxito	1. Se carga la lista de tutelados 2. El voluntario selecciona un tutelado de la lista 3. Se muestran los juegos disponibles para el tutelado.

Tabla 13 - Caso de uso Seleccionar Tutelado (Sección de Juegos)

Caso de uso 2	**Ejecutar juego**
Resumen de la funcionalidad	Permite al voluntario elegir un juego de la lista de juegos y ejecutarlo.
Casos de uso relacionados	Seleccionar tutelado
Actores	Voluntario
Precondición	
Postcondición	Se carga la pantalla del Juego seleccionado
Escenario de éxito	1. Se carga la lista de juegos. 2. El voluntario selecciona un juego de la lista. 3. Se carga la pantalla del Juego seleccionado.

Tabla 14 - Caso de uso Ejecutar Juego

Caso de uso 3	**Jugar Partida**
Resumen de la funcionalidad	Permite al voluntario elegir los parámetros de la partida e iniciarla para que el tutelado pueda jugar.
Casos de uso relacionados	Jugar de Nuevo, Ver Resultados Partida
Actores	Voluntario, Tutelado
Precondición	
Postcondición	Se muestran las opciones de Jugar de Nuevo y Ver Resultados Partida.
Escenario de éxito	1. El voluntario configura la partida y la inicia para que el tutelado juegue. 2. El tutelado juega la partida con la ayuda del voluntario si es necesario.

	3. Se muestran las opciones de Jugar de Nuevo y Ver Resultados Partida.

Tabla 15 - Caso de uso Jugar Partida

Caso de uso 4	**Jugar de nuevo**
Resumen de la funcionalidad	Permite al voluntario volver a ejecutar una nueva partida con los mismos parámetros.
Casos de uso relacionados	Jugar Partida
Actores	Voluntario
Precondición	
Postcondición	Se ejecuta una nueva partida.
Escenario de éxito	1. El voluntario elige la opción de Jugar de Nuevo. 2. Se carga una nueva partida con los mismos parámetros.

Tabla 16 - Caso de uso Jugar de Nuevo

Caso de uso 5	**Ver Resultados Partida**
Resumen de la funcionalidad	Permite al voluntario ver los resultados de la partida jugada.
Casos de uso relacionados	Jugar Partida
Actores	Voluntario
Precondición	
Postcondición	Se sincronizan los resultados de la partida con el servidor.
Escenario de éxito	1. El voluntario elige la opción de Ver resultados Partida. 2. Se muestran los resultados de la partida. 3. Se sincronizan los resultados con el servidor.

Tabla 17- Caso de uso Ver Resultados Partida

Capítulo 5 – Aplicación socio sanitaria. Descripción Técnica

Capítulo 5 – Aplicación socio sanitaria. Descripción Técnica

En este capítulo, después de haber realizado una descripción funcional y conceptual de la aplicación y sus componentes en el capítulo anterior, se pretende realizar una descripción técnica que explique e ilustre los distintos fundamentos técnicos utilizados para desarrollar la aplicación y la comunicación de ésta última con el servidor Web de la fundación. Todo ello explicado desde un punto de vista técnico.

Sin embargo, antes de entrar en los detalles técnicos de la aplicación como tal, primero se hará una pequeña introducción para explicar la jerarquía de vistas de Android y cómo se asocia cada uno de estos elementos, por regla general llamados *layouts*, a una Activity (o Fragment) de la aplicación. También se estudiarán los widgets, subclases predefinidas por el sistema Android de la clase View que implementan elementos que pueden ser incrustados en una jerarquía de vistas cualquiera, las librerías de soporte utilizadas en la interfaz *Material Design* o en la comunicación con el servidor y algún otro tipo de clase que también han sido utilizadas en la aplicación. Tras la introducción teórica, se prestará atención a los distintos *layouts*, clases View personalizadas y librerías que utiliza la aplicación.

5.1 Introducción

Una aplicación, vista desde el punto de vista del usuario, no es más que un conjunto de pantallas con controles y elemento gráficos que permiten al usuario interactuar con la aplicación para acceder las funciones y características de dicha aplicación. Sin embargo, a nivel de sistema operativo, en este caso Android, estas pantallas reciben el nombre de actividades (*Activities*) y son objetos que extienden a la clase *Activity*. Cada una de estas actividades se encarga de los elementos que contiene, y además existe la posibilidad de invocar actividades entre aplicaciones distintas.

Los elementos mencionados anteriormente y que se visualizan en una actividad extienden a la clase *View*. Y aquellos componentes que permiten agrupar a otros extienden a *ViewGroup*. Así los Widgets, que no son otra cosa que objetos de la interfaz de usuario completamente desarrollados, como *Button* o *Checkbox,* son componentes que extienden a *View* y un *LinearLayout* o *TableLayout* son componentes que extienden a *ViewGroup*. En la figura 28 se muestra un esquema de la jerarquía de vistas de Android.

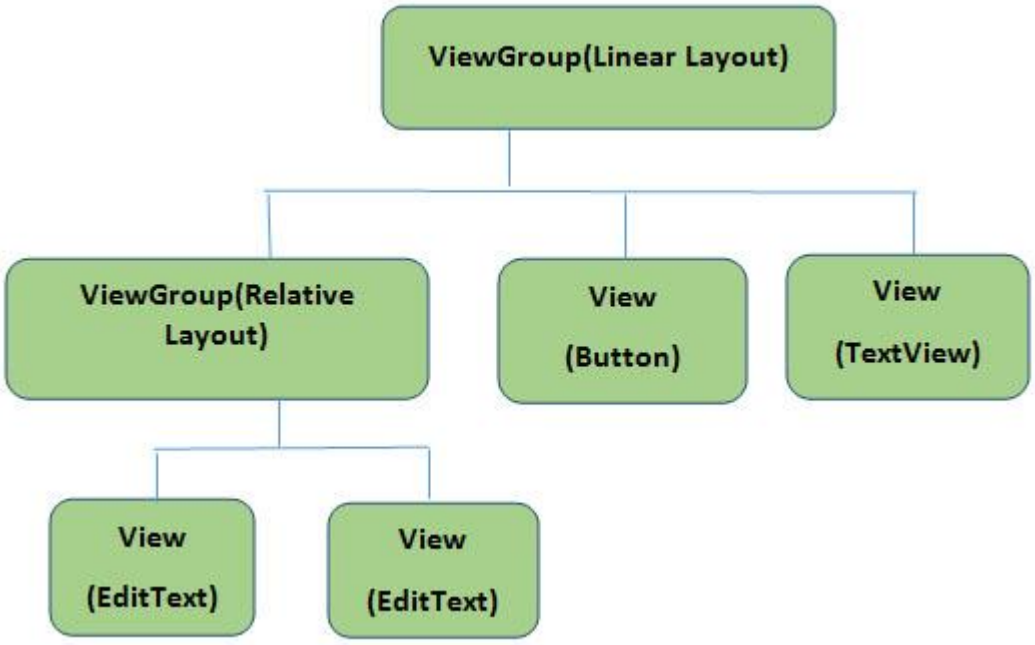

Figura 28 - Esquema de la jerarquía de vistas de Android

Un objeto View, 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*. Además, este objeto es capaz de gestionar sus propias medidas, y prácticamente cualquier interacción por parte del usuario en el área en el que está definido. La clase *view* es útil como clase base para los widgets, que son unas subclases ya implementadas que dibujan los elementos en la pantalla. Los widgets contienen sus propias medidas, pero se pueden usar para construir la interfaz más rápidamente. La lista de widgets que se pueden utilizar incluye: Text, EditText, InputMethod, MovementMethod, Button, RadioButton, CheckBox, y ScrollView (Universidad Carlos III, 2015).

Tal y como se puede ver en la Figura 28, todos los *layouts* y objetos han de ser herederos de *ViewGroup*. Los elementos más destacados se muestran a continuación en la siguiente tabla (Android Developers, 2019g):

Nombre	Descripción
FrameLayout	Éste es el más simple de todos los *layouts* de Android. Un *FrameLayout* coloca todos sus controles hijos alineados con su esquina superior izquierda, de forma que cada control quedará oculto por el control siguiente (a menos que éste último tenga transparencia).
GridView	Muestra una rejilla de *m* columnas y *n* filas que soporta *scroll*.

LinearLayout	Este *layout* apila uno tras otro todos sus elementos hijos en sentido horizontal o vertical según se establezca su propiedad *android:orientation*. los elementos contenidos en un *LinearLayout* pueden establecer sus propiedades *android:layout_width y android:layout_height* para determinar sus dimensiones dentro del *layout*.
ListView	Muestra una única lista en forma de columna con *scroll*.
RecyclerView	Una vista *RecyclerView* visualiza una lista o cuadrícula deslizable verticalmente de varios elementos, donde cada elemento puede definirse como un *Layout*
CardView	Un *CardView* no es más que una extensión de *FrameLayout* con esquinas redondeadas y una sombra inferior), dentro de un *CardView* podemos añadir todos los controles que necesitemos.
RelativeLayout	Este *layout* permite especificar la posición de cada elemento de forma relativa a su elemento padre o a cualquier otro elemento incluido en el propio *layout* .
Spinner	Muestra un único ítem a la vez de una lista de hijos.
SurfaceView	Proporciona acceso directo a una superficie de dibujo dedicada. Está pensada para aplicaciones que dibujan píxeles más que para las que funcionan con el conjunto predefinido de widgets de Android.
TableLayout	Un *TableLayout* permite distribuir sus elementos hijos de forma tabular, definiendo las filas y columnas necesarias, y la posición de cada componente dentro de la tabla.

Tabla 18- Layouts herederos de ViewGroup

5.2 Interfaz de Usuario de la Aplicación

5.2.1 Introducción

En la aplicación desarrollada se ha procurado que la interfaz sea lo más flexible y cuidada posible, además de que presente un aspecto atractivo para el usuario. Para ello se han seguido las pautas indicadas por la filosofía y concepto ***Material Design*** propuesta por Google en el Google I/O 2014.

Material Design es una guía integral para el diseño visual, de movimientos y de interacción en distintas plataformas y dispositivos. Para usar *Material Design* en aplicaciones de Android, se han de seguir las pautas descritas en la especificación de *Material Design* y usar los nuevos componentes y funcionalidades disponibles en Android 5.0 (nivel de API 21) y versiones posteriores. En Android, se proporcionan los siguientes elementos que permitirán crear aplicaciones en *Material Design*: un tema nuevo; nuevos widgets para vistas complejas y nuevas API (interfaces de programación de aplicaciones) para sombras y animaciones personalizadas (Android Developers, 2019h).

Una vez conocido lo que *es Material Design*, queda bastante claro su utilidad y su importancia en el futuro del desarrollo Android. Por ello, se ha decidido utilizar *Material Design* en la aplicación desarrollada. A pesar de ser un diseño introducido a partir de la versión 5.0 de Android, ello no impide su uso en versiones anteriores, aunque con ciertas limitaciones dependiendo de la versión de Android. Para que su uso sea posible en versiones inferiores a la 5.0, es necesaria principalmente la librería de soporte llamada *AppCompat v21***.**

Esta librería AppCompat (también conocida como ActionBarCompat) se implementó en la API de la ActionBar de Android 4.0 para dispositivos que ejecutaban Gingerbread, proporcionando una API común para su implementación. *AppCompat v21* es la versión que se ha actualizado con Android 5.0. En esta versión, todos aquellos elementos necesarios para poder utilizar *Material Design* en dispositivos con Android con versiones inferiores a la 5.0 (Chris Banes, 2014).

Recursos gráficos y de texto

Para poder llevar a cabo el desarrollo de la APP y dada la naturaleza de esta, se ha hecho uso de una serie de recursos gráficos y de texto que son todos de libre uso y sin protección de derechos de autor. La app contiene una cantidad significativa de imágenes y textos que son necesarios para el diseño e implementación de la interfaz de la aplicación, sobre todo en la sección de juegos.

Además, para implementar los juegos, se ha utilizado como base los juegos del ***Cuaderno de ejercicios de estimulación cognitiva (CPDC, 2008)*** proporcionado por la fundación **Fundamay**.

Estructura de archivos java del proyecto

Los detalles y contenido de todas las clases, métodos y *layouts* desarrollados, que forman parte de la *App,* nombrados en este capítulo se pueden consultar en el **código fuente de la aplicación desarrollada, cuya estructura de archivos *.java*** se puede ver en la Figura 29.

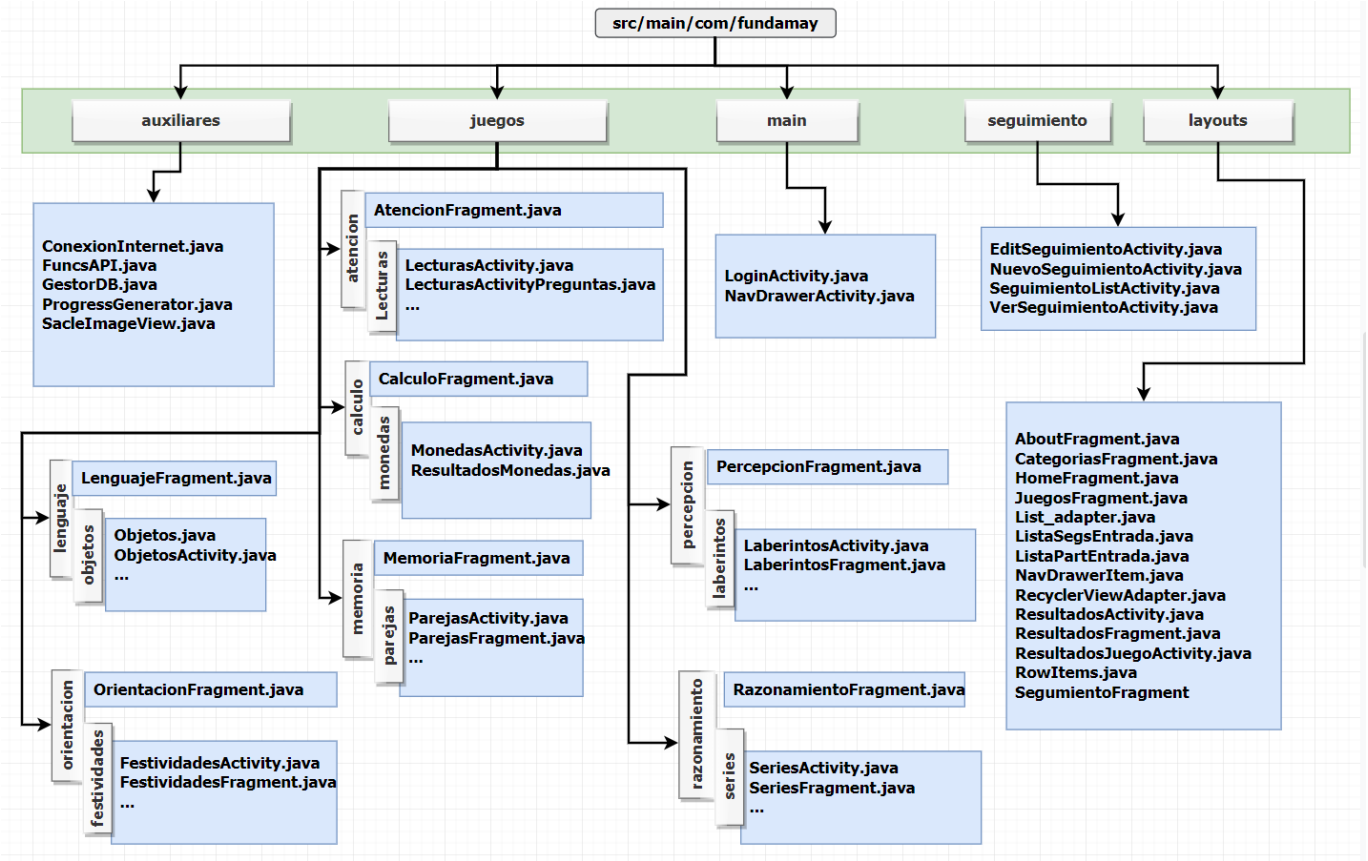

Figura 29 - Estructura de archivos java del proyecto

5.2.2 Pantalla de Login

Esta pantalla es donde puede autentificarse el voluntario que va a utilizar la aplicación. Se compone de dos campos *EditText* donde se puede introducir el email y la contraseña y también dispone de un botón animado que se pulsa para iniciar la sesión. En la siguiente figura se muestran los componentes más relevantes de la interfaz y su descripción.

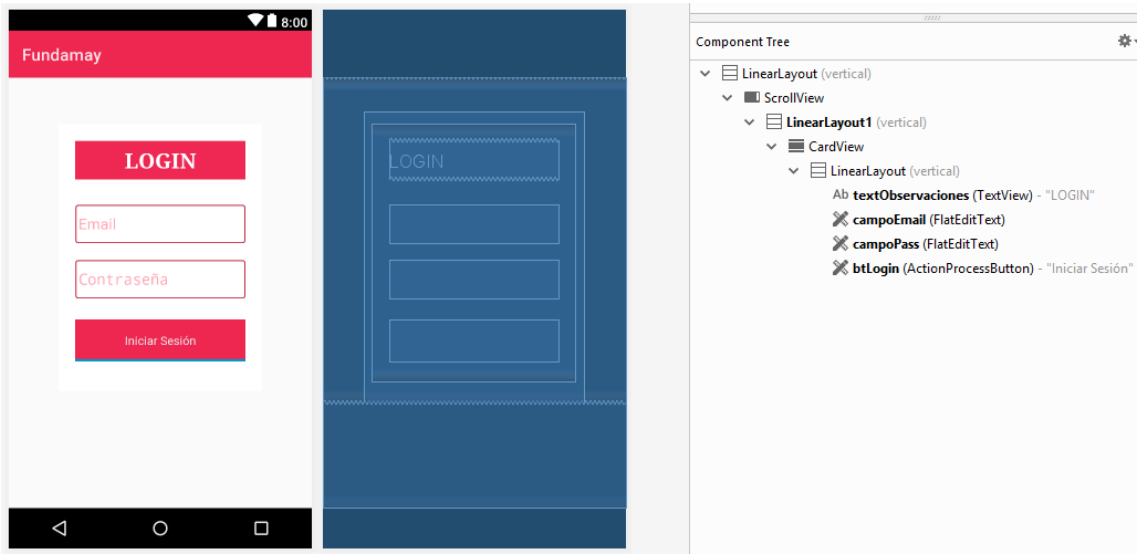

Figura 30 – Pantalla de Login: Interfaz y árbol de componentes

Este *layout*, *activity_login.xml*, que se puede observar en la Figura 30, está asociado a la activity *LoginActivity,* que extiende, como todas las activities de la aplicación, de la clase *AppCompatActivity* (introducida en *material design*). Esta activity se encarga de recoger los datos de Login introducidos por el usuario y conecta con el servidor web con el método *comprobarLogin()* para autentificar al voluntario. En este método se hace uso de la clase *Asynclogin* que extiende a la clase *AsyncTask*, con el objetivo de evitar que la interfaz se congele mientras espera la respuesta del servidor.
El envío de los datos de *Login* y la autenticación se realizan desde el método *loginVoluntario(email, password),* que está definido en la clase *FuncsAPI*, al que se le pasan como argumento los datos necesarios para poder realizar la autentificación con el servidor Web.
La clase *FuncsAPI* no es más que una clase java en la que están definidos varios métodos auxiliares a los cuales se accede desde varias otras clases de la aplicación.
Una vez hecha la autentificación de manera exitosa, se guardan los datos del voluntario en una base de datos local en el dispositivo mediante el método *comprobarLoginLocal*() para que el voluntario pueda entrar sin conexión a internet. Y posteriormente se lanza la actividad de la pantalla principal llamada *NavDrawerActivity* que se explica en el siguiente punto.

5.2.3 Pantalla principal de la aplicación

Esta interfaz no dispone de un *layout* definido en un *xml* en la carpeta *res/layout*, sino que es una librería externa llamada ***AdvancedMaterialDrawer*** con licencia *Apache*

License 2.0. Por consiguiente, los detalles sobre el *layout* y el código java se pueden encontrar en el repositorio *github* del creador de dicha librería (Schäfers, 2015). Por comodidad se ha importado la librería de manera local en la carpeta *Lib_AdvancedMaterialDrawer* de este mismo proyecto. Esta librería permite tener un menú lateral deslizable, tal y como se puede ver en la figura 31 en el cual se pueden ubicar los accesos a las distintas secciones de la aplicación. Tal y como está desarrollada la librería, las secciones son *fragments* que están asociados a la *activity* principal llamada *NavDrawerActivity*.

Secciones del menú lateral:

- **Fundamay**: es un *fragment* que actúa a modo de pantalla inicial y que contiene un sencillo banner con el logo y el nombre de Fundamay y un campo de texto que muestra el nombre del voluntario que está utilizando la aplicación. Se ha hecho uso de *Cardviews* para un mejor aspecto.
- **Seguimiento**: es otro *fragment* desde el cual se accede a las funciones que permiten realizar el seguimiento de los tutelados.
- **Juegos**: es otro *fragment* desde el cual se accede a los juegos incluidos en la aplicación para los tutelados.
- **Resultados**: es otro *fragment* desde el cual se accede a los resultados de las partidas jugadas por los tutelados.
- **Acerca de**: es un *fragment* con información sobre la aplicación.
- **Cerrar Sesión**: es un *button* para cerrar la sesión actual y volver a la pantalla de *Login*.

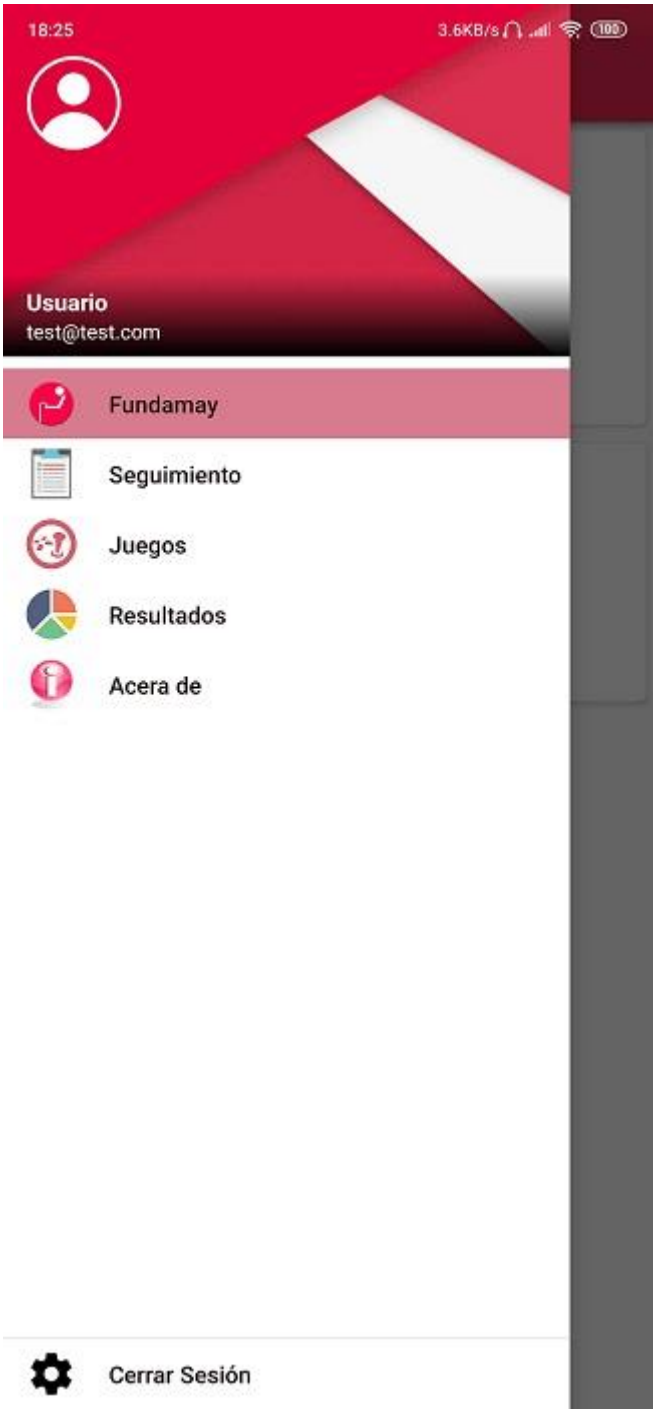

Figura 31 - Pantalla principal: Interfaz y menú lateral

5.2.4 Seguimientos

La sección Seguimientos es uno de los dos bloques principales de la aplicación. A continuación, se procede a detallar la interfaz de usuario relacionada con este bloque y las clases y actividades involucradas en su funcionamiento.

Al acceder a esta sección desde el menú lateral, se muestra un *fragment* (*fragment_seguimiento.xml*) con un *Listview* en el que se cargará la lista los tutelados asignados al voluntario que está utilizando la aplicación. Al seleccionar un tutelado de la lista, se muestran las opciones de **Nuevo Seguimiento** y **Ver Seguimientos,** tal y como se puede observar en la figura 73 del **capítulo 6**.

5.2.4.1 Nuevo Seguimiento

Esta opción permite al voluntario crear un nuevo seguimiento del tutelado, para ello, el voluntario introduce los datos en los distintos *widgets* del *layout* (*activity_nuevo_seguimiento.xml*, Figura 32) de esta activity.

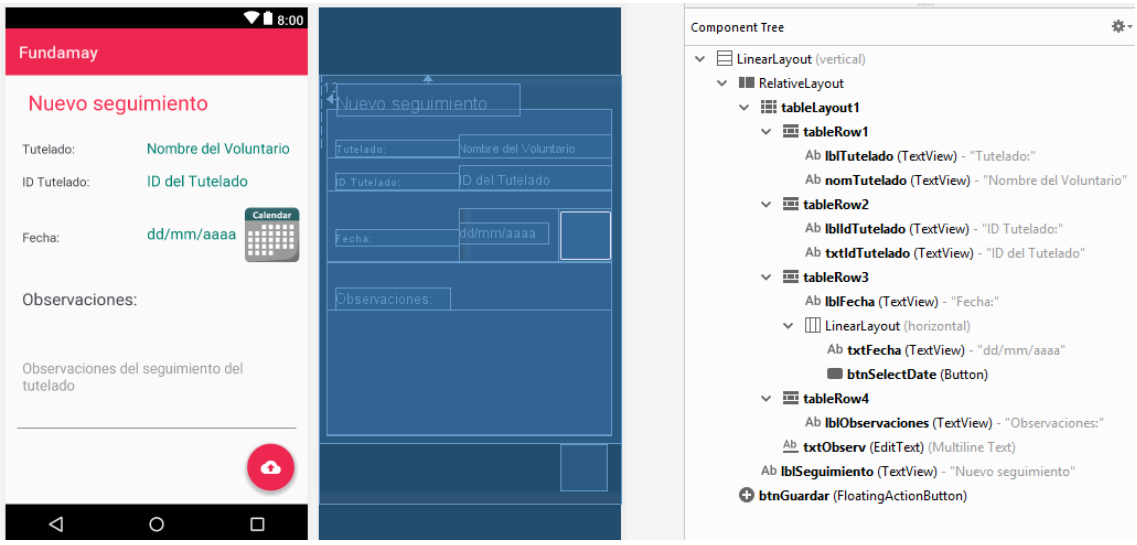

Figura 32 - Pantalla de Nuevo Seguimiento: Interfaz y árbol de componentes

La *activity NuevoSeguimientoActivity* se encarga de recoger los datos introducidos por el voluntario y enviarlos al servidor Web. Para ello, se hace uso de la clase *NuevoSeguimiento* que extiende a la clase *AsyncTask* para que la transferencia de los datos se realice de manera asíncrona y así evitar congelaciones de la interfaz. En esta clase, se llama al método *nuevoSeguimiento(idTutelado,idUsuario, fecha,tipo,observaciones)*, que está definido en la clase *FuncsAPI*, al que se le pasan como argumento los datos necesarios para poder guardar el seguimiento en la BD del servidor Web.

5.2.4.2 Ver Lista Seguimientos

Esta opción corresponde a la *activity SeguimientoListActivity* y permite al voluntario ver la lista de seguimientos del tutelado seleccionado que se mostrará en el *layout activity_seguimiento_list.xml de la Figura 33*.

La activity *SeguimientoListActivity,* se encarga de descargar la lista de seguimientos desde la BDD del servidor. Para ello, se hace uso de la clase *getSegs*, que extiende a la clase *AsyncTask*,en la cual se invoca al método *getSegs(idTutelado,idUsuario)*, que está definido en la clase *FuncsAPI*, al que se le pasan como argumento los datos necesarios para poder descargar la lista de seguimientos.

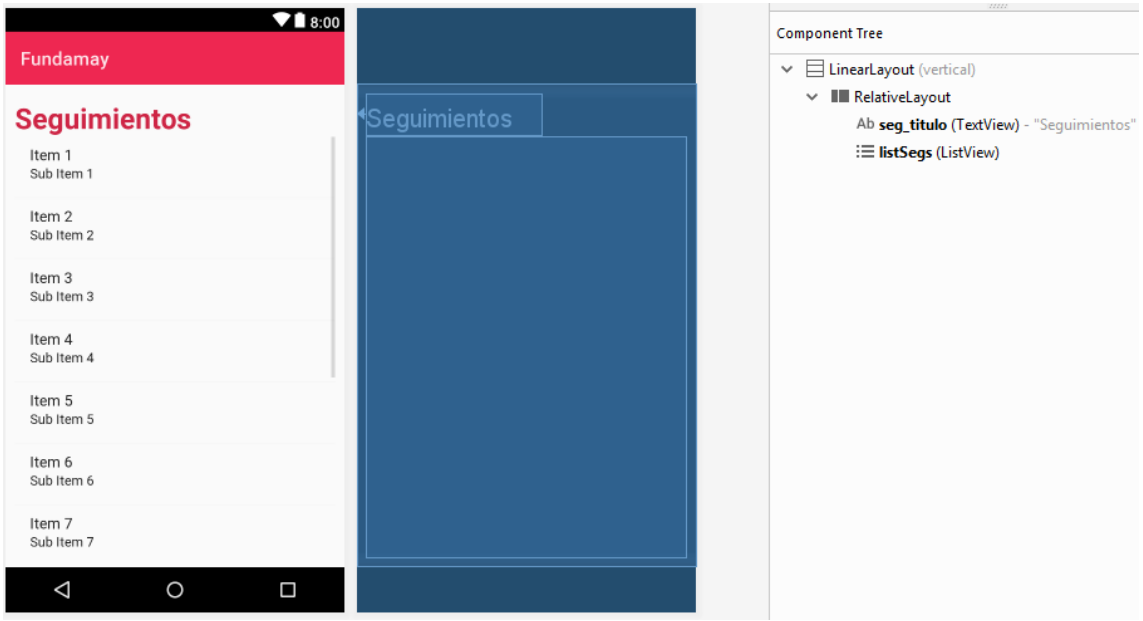

Figura 33 - Pantalla de Ver Lista Seguimientos: Interfaz y árbol de componentes

Al seleccionar un seguimiento de la lista, se muestran las opciones de **Editar Seguimiento** y **Ver Seguimiento** que se detallan a continuación.

5.2.4.3 Editar Seguimiento

Esta opción permite al voluntario editar un seguimiento previamente guardado en la BDD. Su funcionamiento es igual a la función de Nuevo Seguimiento con la diferencia de que en este caso se cargan los datos descargados de la BDD en el *layout* (que es el mismo que el de la función Nuevo Seguimiento, Figura 32) para poder ser editados

La *Activity* asociada a este *layout* es *EditSeguimientoActivity,* que, una vez modificados los datos y habiéndose pulsado el botón para guardar el seguimiento, se hace uso de la clase *EditSeguimiento*, que extiende a la clase AsyncTask, en la cual se invoca al método *editSeguimiento(idSeg, idTutelado, idUsuario ,fecha, tipo, observaciones)*, que está

definido en la clase *FuncsAPI*, al que se le pasan como argumento los datos necesarios para poder guardar el seguimiento modificado en la BD del servidor Web.

5.2.4.4 Ver Seguimiento

Esta opción corresponde a la *activity VerSeguimientoActivity* y permite al voluntario ver un seguimiento de la lista sin la posibilidad de alterarlo. Los datos del seguimiento se mostrarán en el *layout activity_ver_seguimiento.xml*, Figura 34.

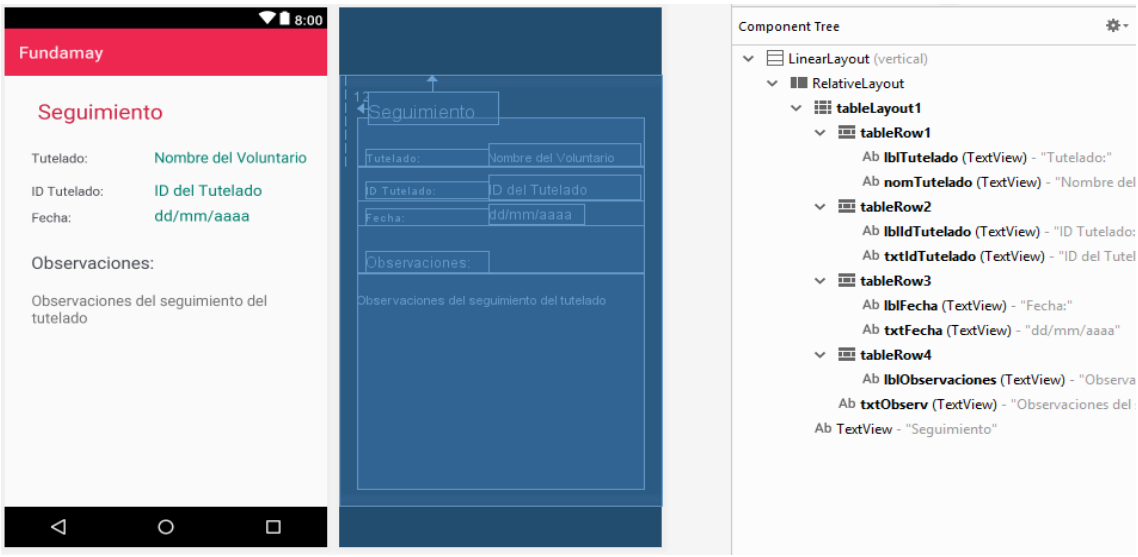

Figura 34 - Pantalla de Ver Seguimiento: Interfaz y árbol de componentes

La a*ctivity* asociada a este *layout* es *VerSeguimientoActivity, a la cual se le pasan los datos del seguimiento desde la activity SeguimientoListActivity y los muestra en el layout de la interfaz.*

5.2.5 Juegos

La sección Juegos es el otro de los bloques principales de la aplicación. A continuación, se procede a detallar la interfaz de usuario relacionada con este bloque y las clases y actividades involucradas en su funcionamiento.

Al acceder a esta sección desde el menú lateral, se muestra un *fragment* (fragment_juegos.xml, Figura 35) con un *Listview* en el que se cargará la lista los tutelados asignados al voluntario que está utilizando la aplicación.

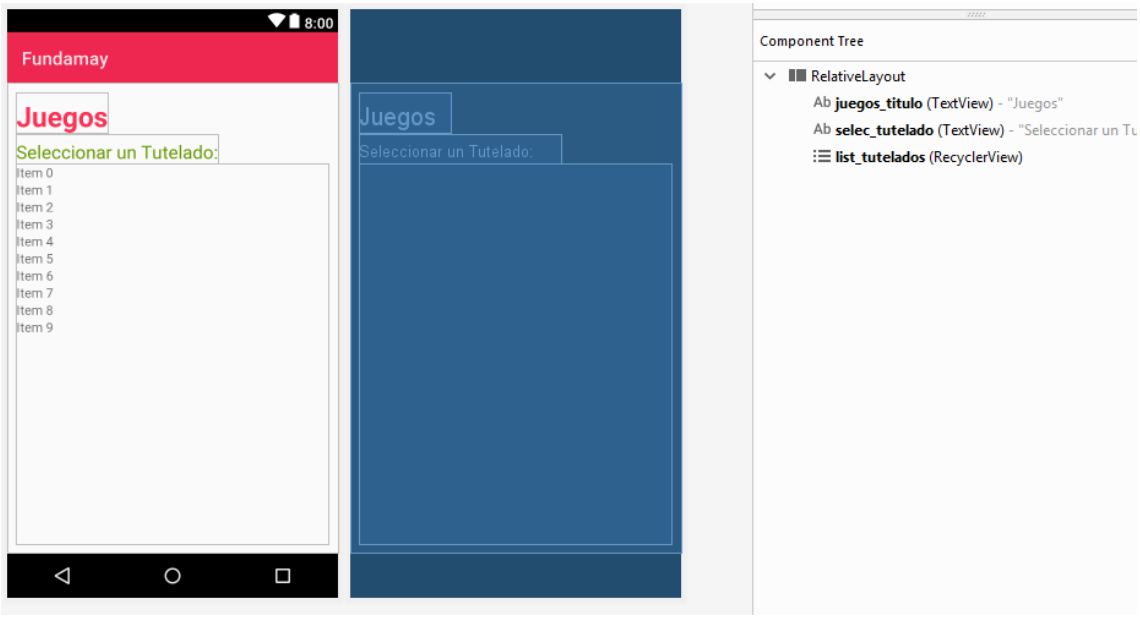

Figura 35 - Pantalla de Juegos: Interfaz y árbol de componentes

Al seleccionar un tutelado de la lista, se carga la lista de categorías de Juegos, cada categoría contiene distintos juegos que se detallan a continuación.

5.2.5.1 Atención

Esta categoría está creada en el *fragment AtencionFragment* a partir del cual se puede acceder al juego contenido en esta categoría. La interfaz correspondiente a esta categoría corresponde al *layout fragment_juegos_cat_atencion.xml* que se muestra en la Figura 36.

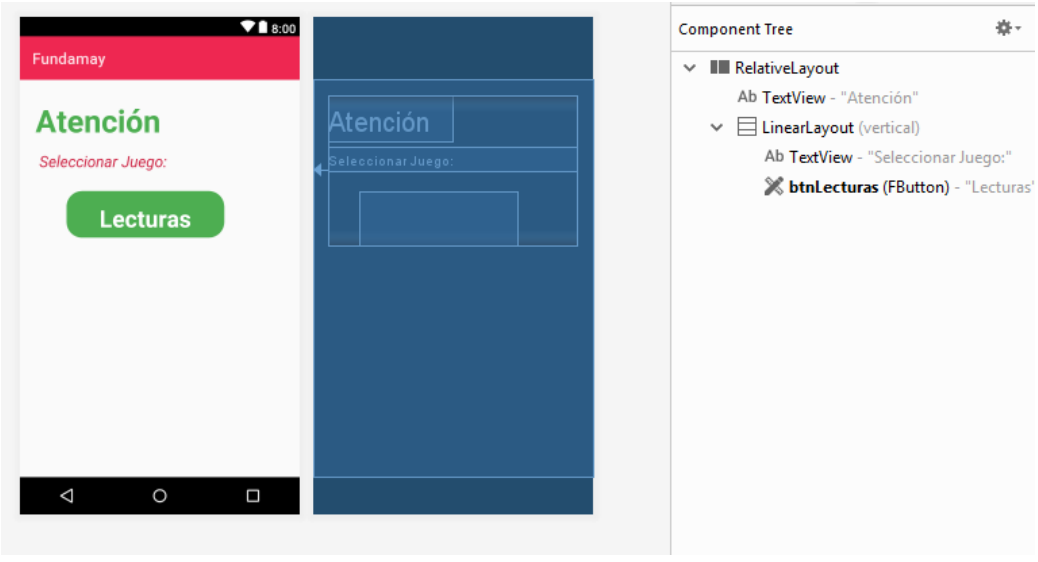

Figura 36 - Pantalla de la categoría Atención: Interfaz y árbol de componentes

- Juego: ***LECTURAS***

Este juego está formado por el *fragment LecturasFragment*, cuya interfaz corresponde al *layout fragment_juegos_cat_atencion_lecturas.xml,* Figura 37.

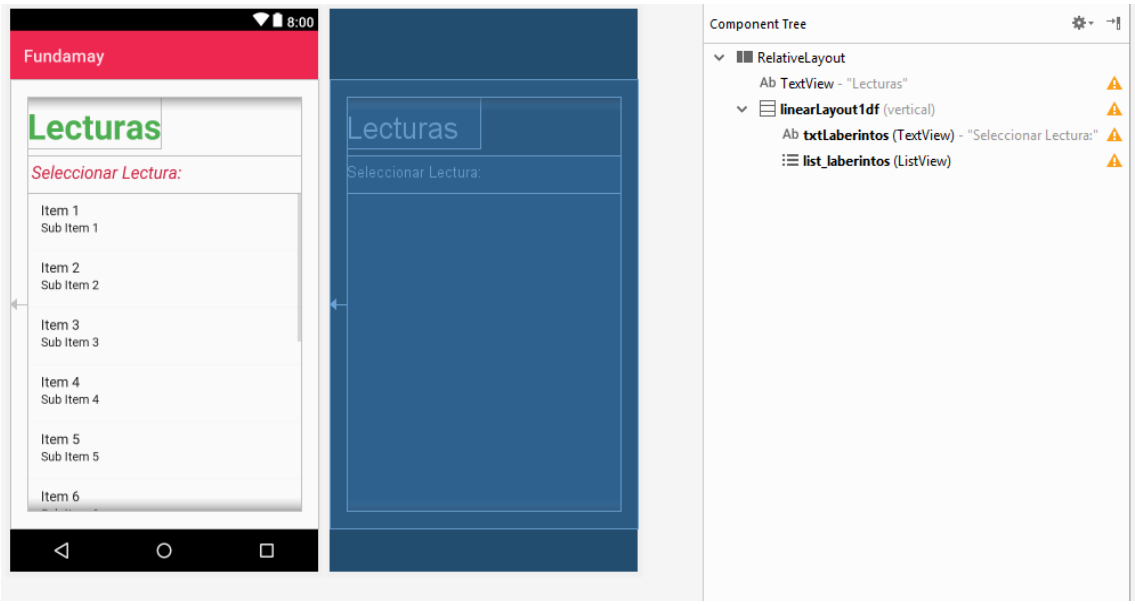

Figura 37 - Pantalla del juego Lecturas: Interfaz 1 y árbol de componentes

Y por las *activities LecturasActivity* y *LecturasActivityPreguntas,* cuyas interfaces están definidas en *activity_lecturas.xml* y *activity_lecturas_preguntas.xml*, Figuras 36 y 37.

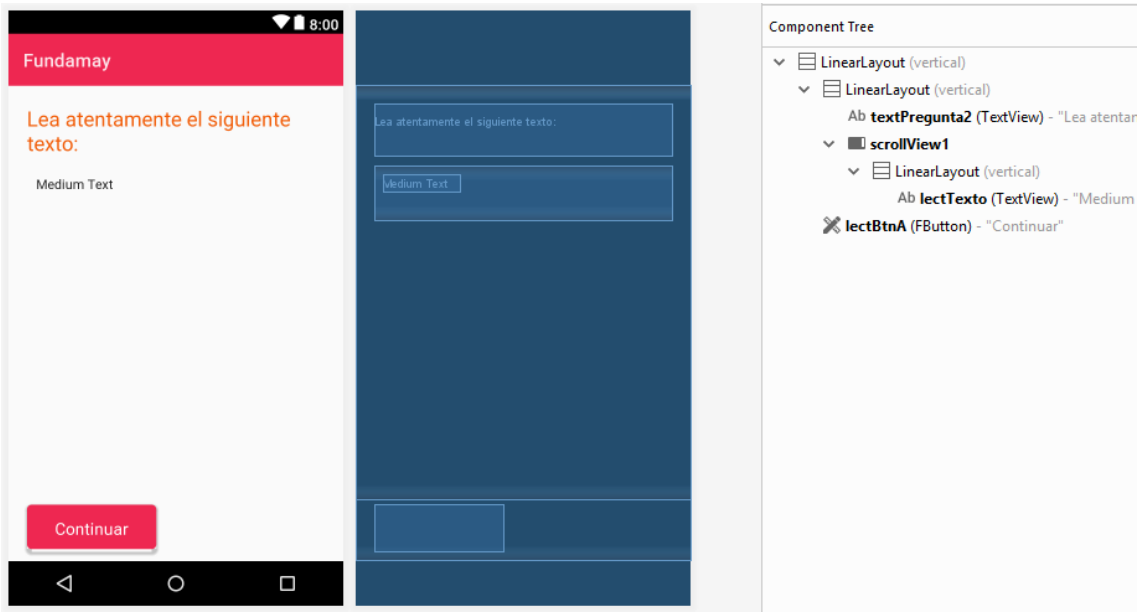

Figura 38 - Pantalla del juego Lecturas: Interfaz 2 y árbol de componentes

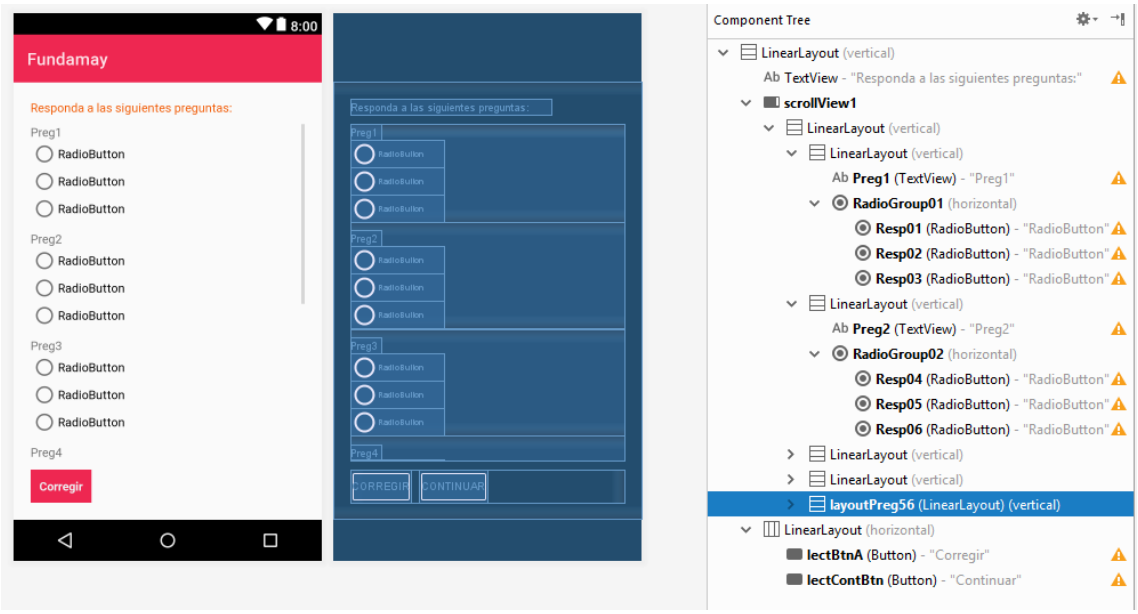

Figura 39 - Pantalla del juego Lecturas: Interfaz 3 y árbol de componentes

5.2.5.2 Cálculo

Esta categoría está creada en el *fragment CalculoFragment* a partir del cual se puede acceder al juego contenido en esta categoría. La interfaz correspondiente a esta categoría corresponde al *layout fragment_juegos_cat_calculo.xml* que se muestra en la Figura 40.

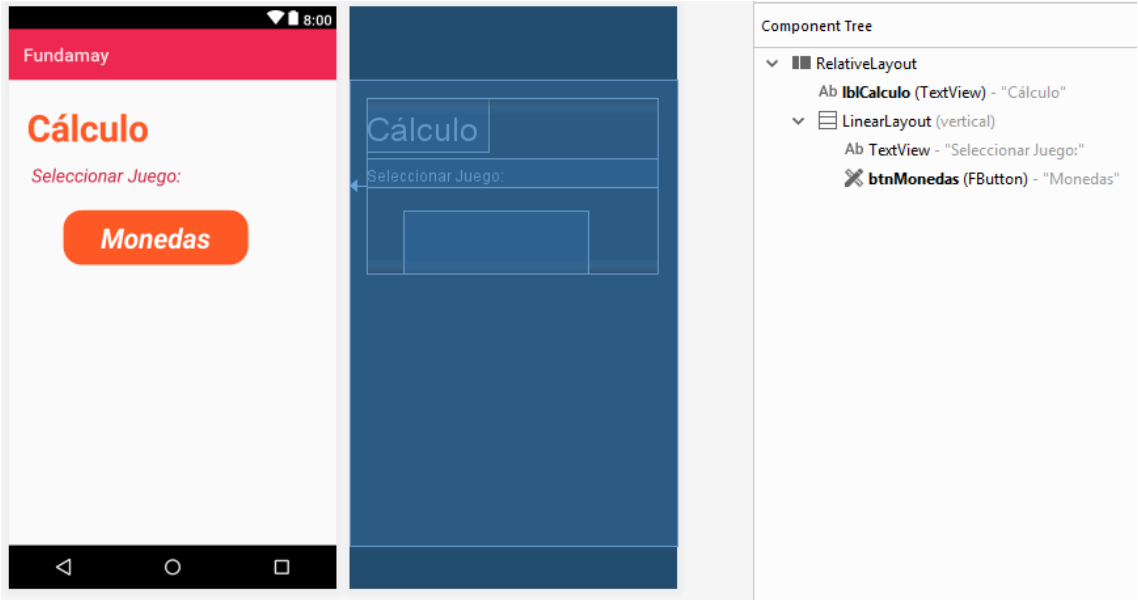

Figura 40 - Pantalla de la categoría Cálculo: Interfaz y árbol de componentes

❖ Juego: ***MONEDAS***

Este juego está formado por la *activity MonedasActivity* cuya interfaz está definida en *activity_monedas.xml* Figura 41.

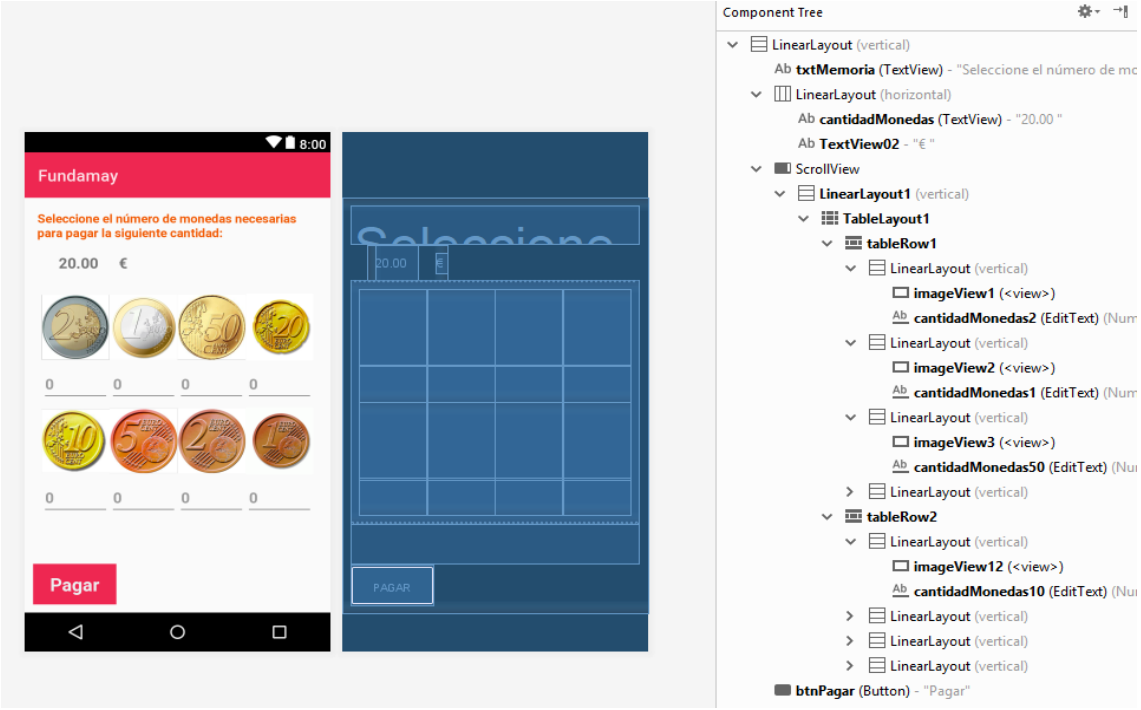

Figura 41 - Pantalla del juego Monedas: Interfaz y árbol de componentes

5.2.5.3 Lenguaje

Esta categoría está creada en el *fragment LenguajeFragment,* a partir del cual, se puede acceder al juego contenido en esta categoría. La interfaz correspondiente a esta categoría corresponde al *layout fragment_juegos_cat_lenguaje.xml* que se muestra en la Figura 42.

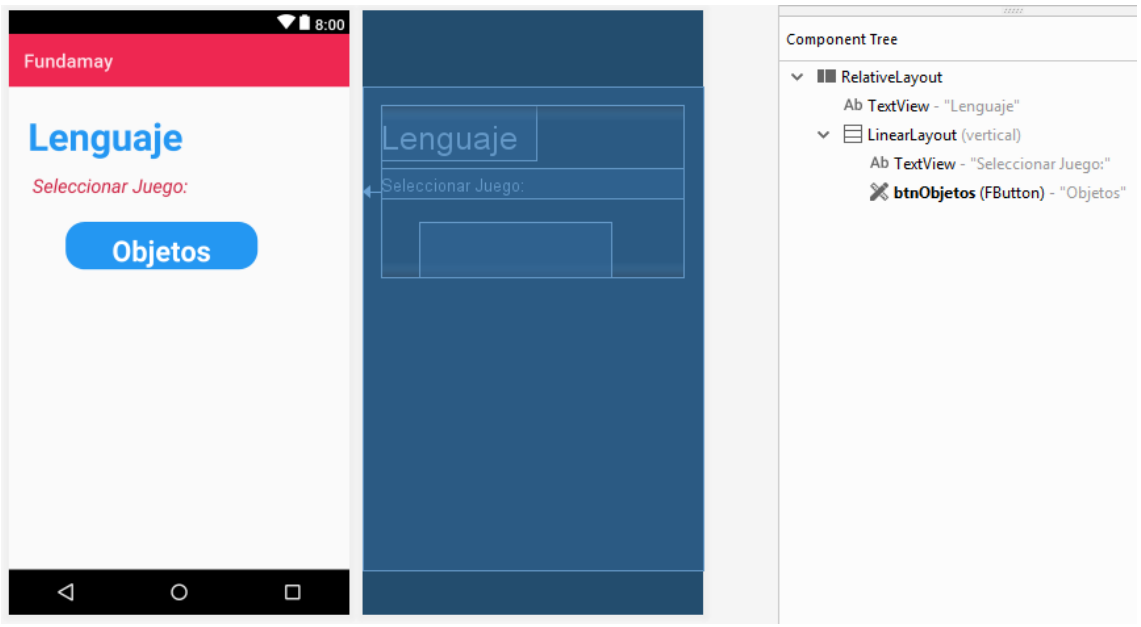

Figura 42 - Pantalla de la categoría Lenguaje: Interfaz y árbol de componentes

❖ Juego: ***OBJETOS***

Este juego está formado por la *activity ObjetosActivity* cuya interfaz está definida en *activity_objetos.xml* Figura 43.

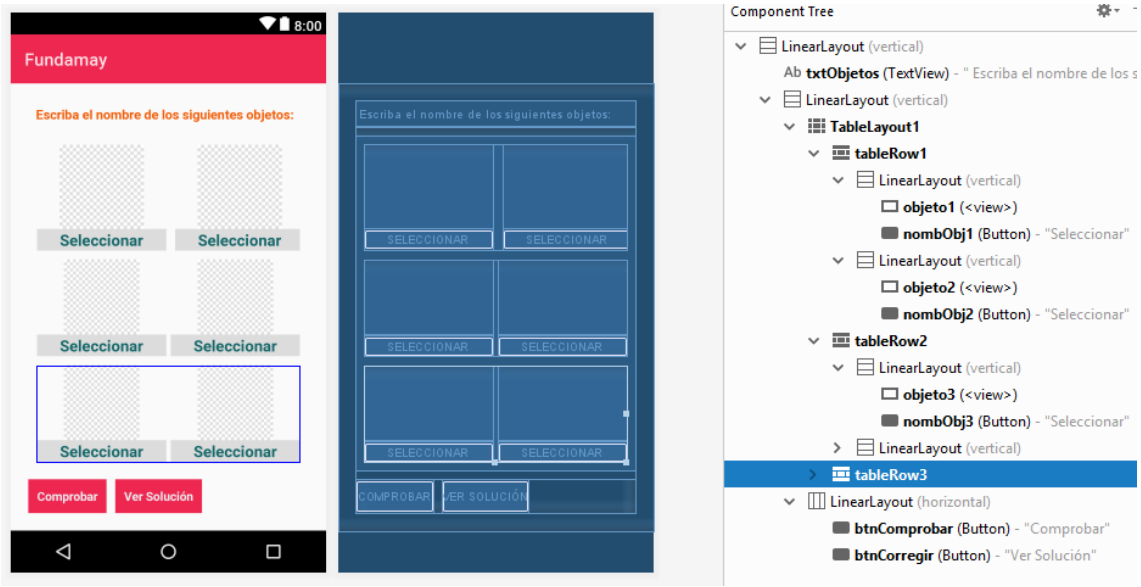

Figura 43- Pantalla del juego Objetos: Interfaz y árbol de componentes

5.2.5.4 Memoria

Esta categoría está creada en el *fragment MemoriaFragment,* a partir del cual, se puede acceder al juego contenido en esta categoría. La interfaz correspondiente a esta categoría corresponde al *layout fragment_juegos_cat_memoria.xml* que se muestra en la Figura 44.

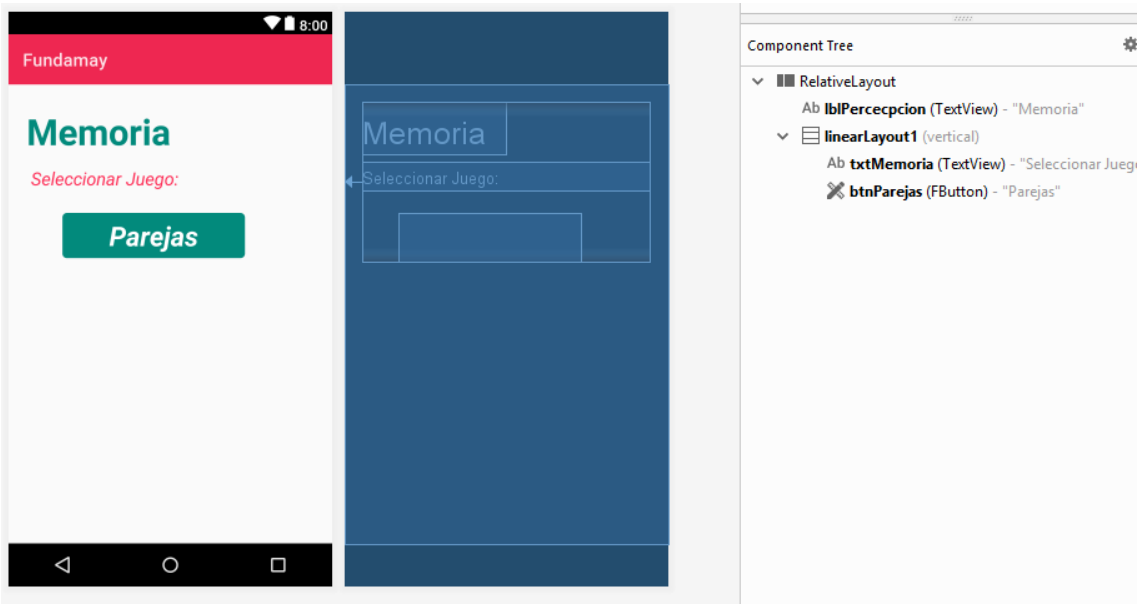

Figura 44 - Pantalla de la categoría Memoria: Interfaz y árbol de componentes

❖ Juego: ***PAREJAS***

Este juego está formado por el *fragment ParejasFragment,* cuya interfaz corresponde al *layout fragment_juegos_cat_memoria_parejas.xml,* Figura 45.

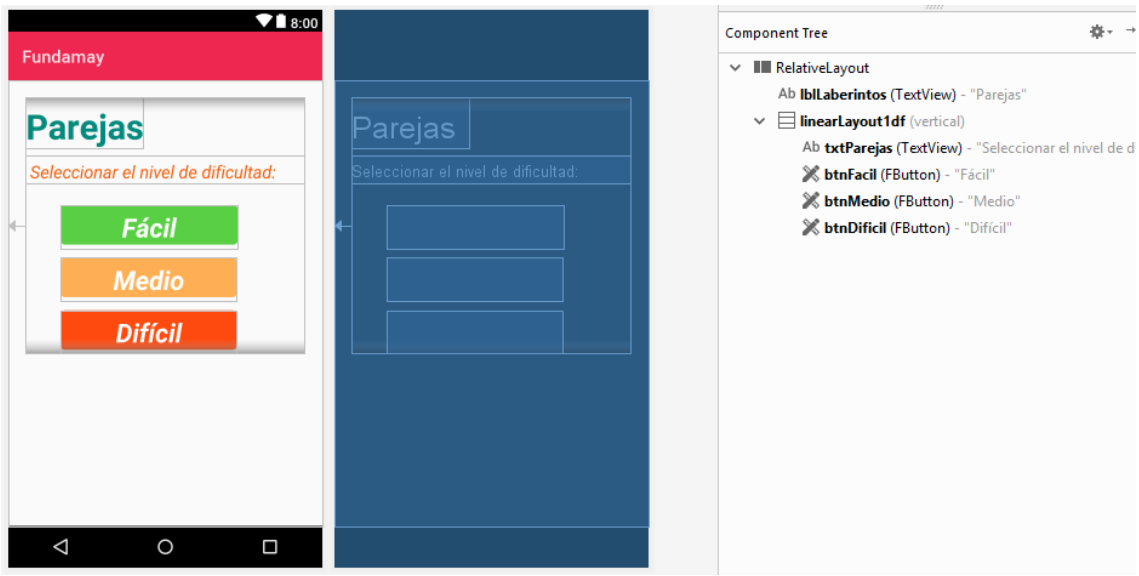

Figura 45 - Pantalla del juego Parejas: Interfaz 1 y árbol de componentes

Y por la *activity ParejasActivity* cuya interfaz está definida en *activity_parejas.xml*, Figura 46.

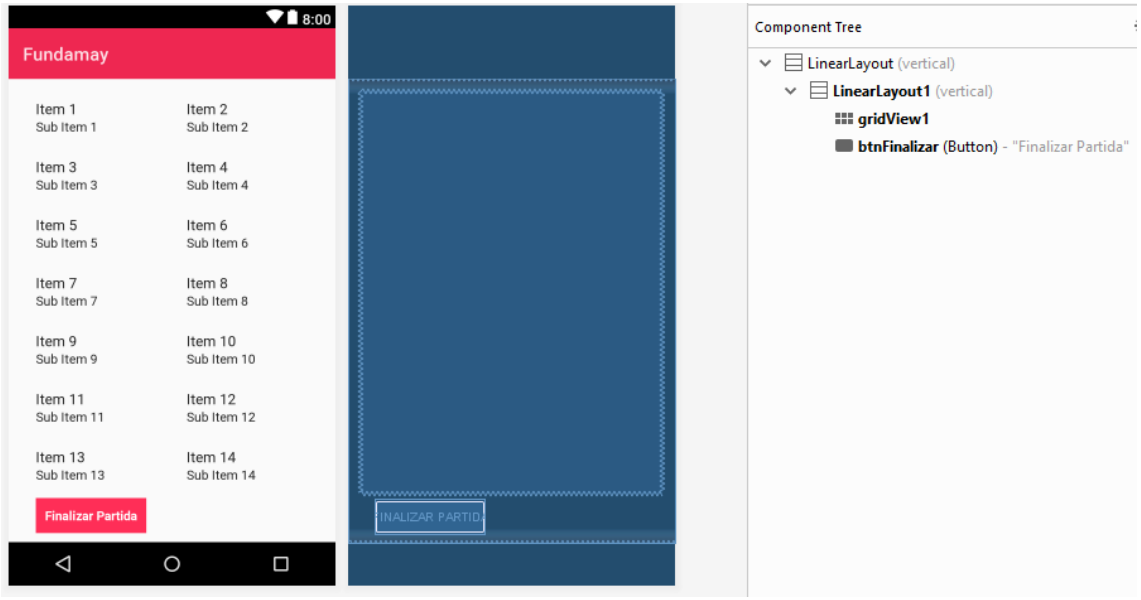

Figura 46 - Pantalla del juego Parejas: Interfaz 2 y árbol de componentes

5.2.5.5 Orientación

Esta categoría está creada en el *fragment OrientacionFragment* a partir del cual se puede acceder al juego contenido en esta categoría. La interfaz correspondiente a esta categoría corresponde al *layout fragment_juegos_cat_orientacion.xml* que se muestra en la Figura 47.

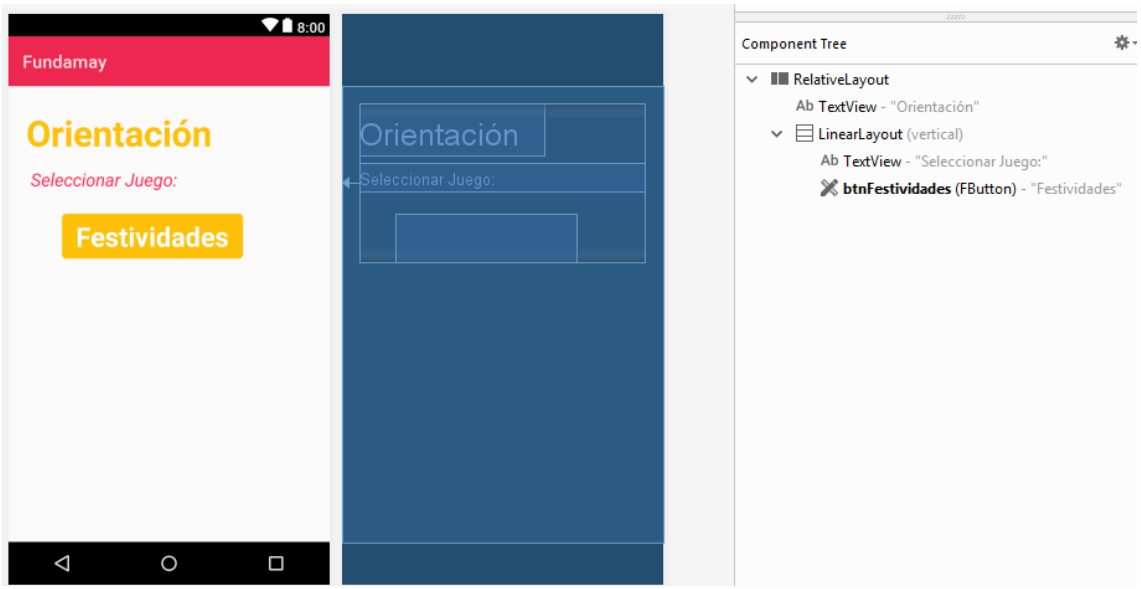

Figura 47 - Pantalla de la categoría Orientación: Interfaz y árbol de componentes

❖ Juego: ***FESTIVIDADES***

Este juego está formado por el *fragment FestividadesFragment*, cuya interfaz corresponde al *layout fragment_juegos_cat_orientacion_festividades.xml,* Figura 48.

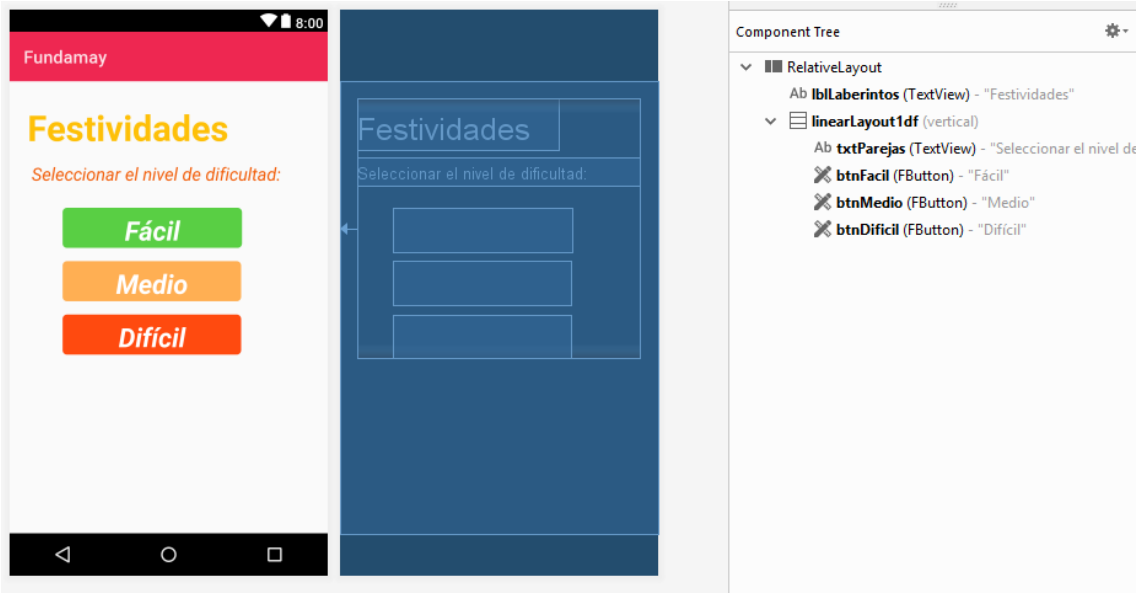

Figura 48 - Pantalla del juego Festividades: Interfaz 1 y árbol de componentes

Y por la *activity FestividadesActivity* cuya interfaz está definida en *activity_festividades.xml*, Figura 49.

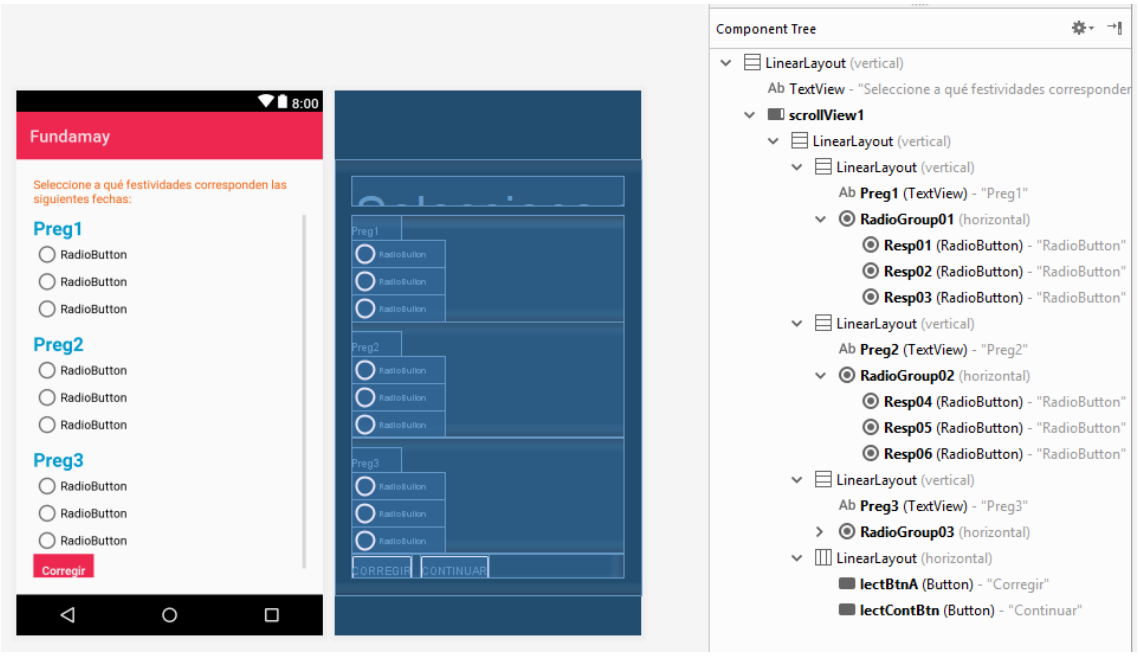

Figura 49 - Pantalla del juego Festividades: Interfaz 2 y árbol de componentes

5.2.5.6 Percepción

Esta categoría está creada en el *fragment PercepcionFragment* a partir del cual se puede acceder al juego contenido en esta categoría. La interfaz correspondiente a esta categoría corresponde al *layout fragment_juegos_cat_percepcion.xml* que se muestra en la Figura 50.

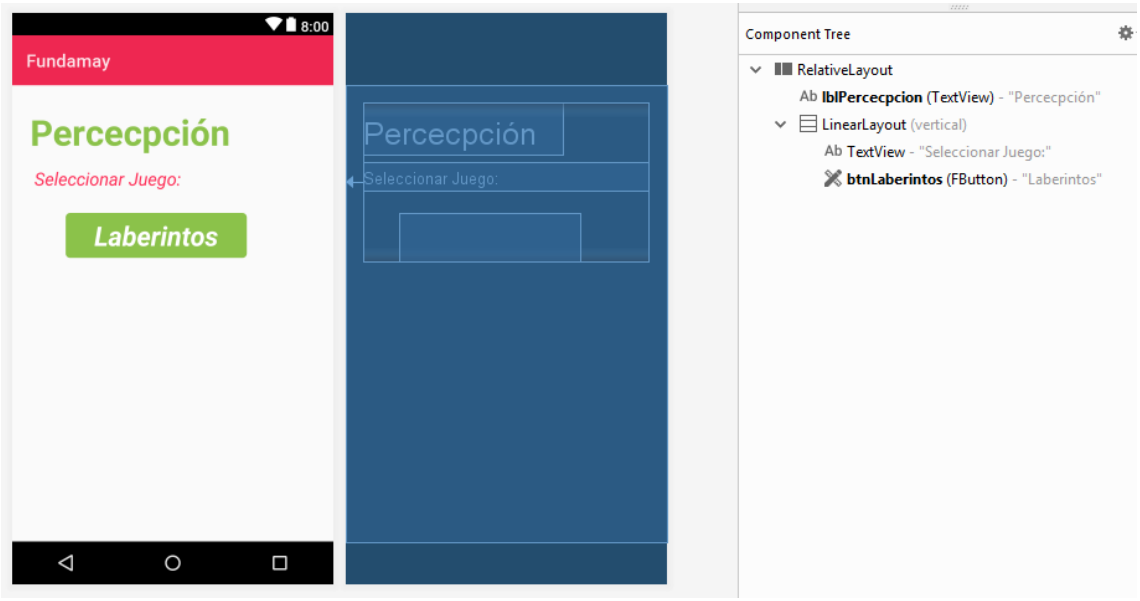

Figura 50 - Pantalla de la categoría Percepción: Interfaz y árbol de componentes

❖ Juego: ***LABERINTOS***

Este juego está formado por el *fragment LaberintosFragment*, cuya interfaz corresponde al *layout fragment_juegos_cat_percepcion_laberintos.xml,* Figura 51.

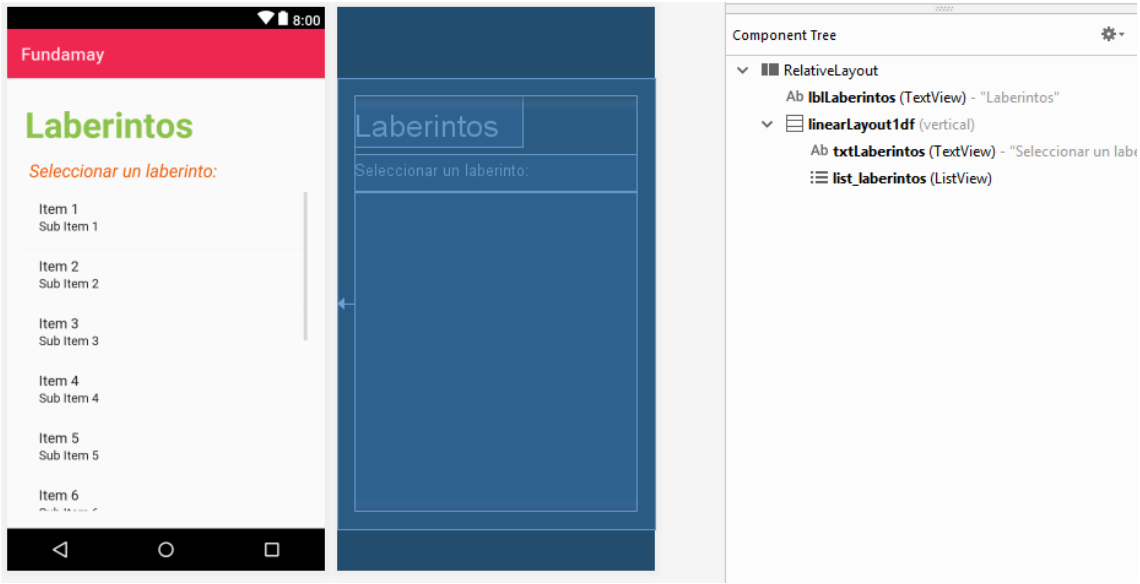

Figura 51 - Pantalla del juego Laberintos: Interfaz 1 y árbol de componentes

Y por la *activity LaberintosActivity* cuya interfaz está definida en *activity_laberintos.xml,* Figura 52.

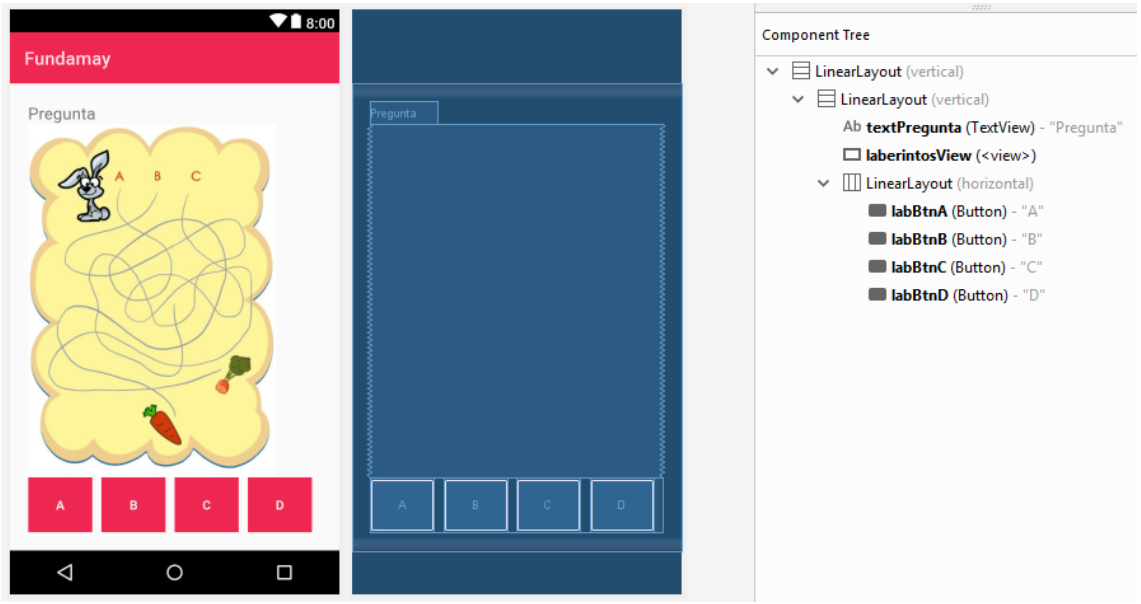

Figura 52 - Pantalla del juego Laberintos: Interfaz 2 y árbol de componentes

5.2.5.7 Razonamiento

Esta categoría está creada en el *fragment RazonamientoFragment* a partir del cual se puede acceder al juego contenido en esta categoría. La interfaz correspondiente a esta categoría corresponde al *layout fragment_juegos_cat_razonamiento.xml* que se muestra en la Figura 53.

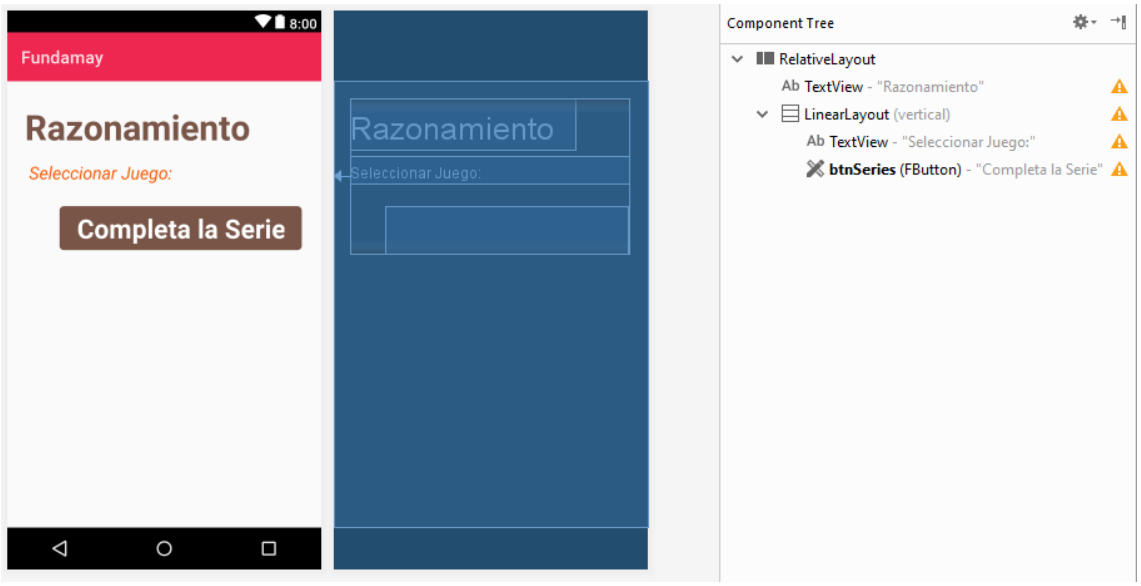

Figura 53 - Pantalla de la categoría Lenguaje: Interfaz y árbol de componentes

❖ Juego: ***COMPLETA LA SERIE***

Este juego está formado por el *fragment SeriesFragment*, cuya interfaz corresponde al *layout fragment_juegos_cat_razonamiento_series.xml,* Figura 54.

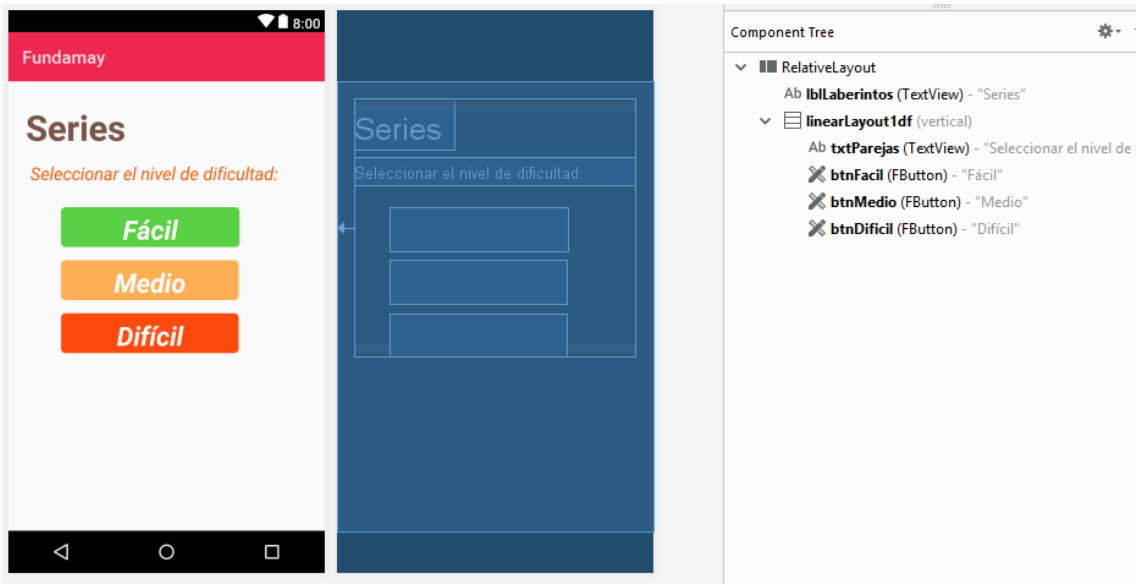

Figura 54 - Pantalla del juego Completa la Serie: Interfaz 1 y árbol de componentes

Y por la *activity SeriesActivity* cuya interfaz está definida *en activity_series.xml*, Figura 55.

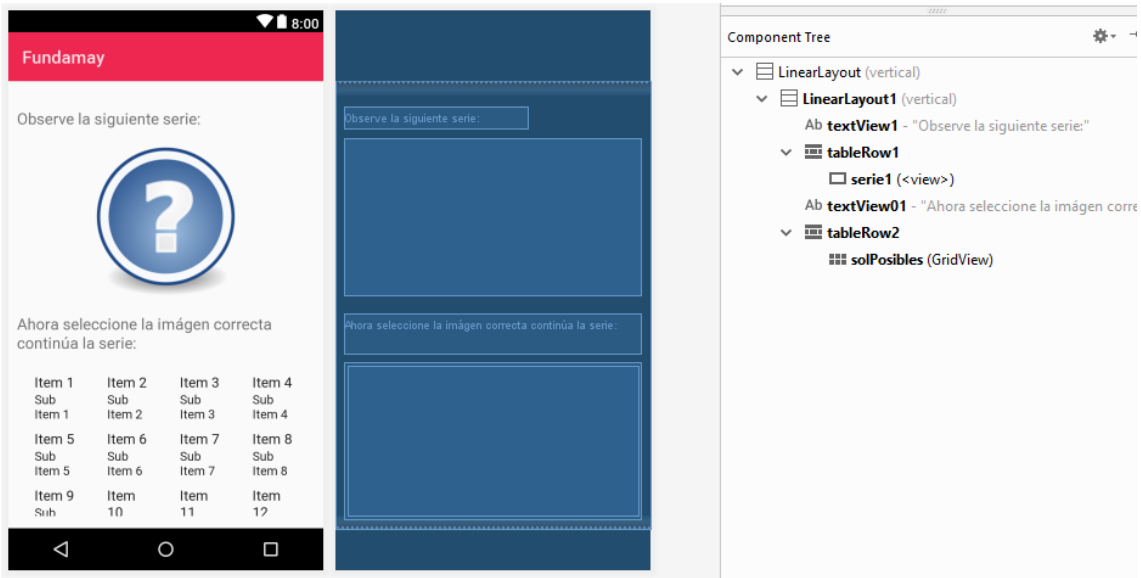

Figura 55 - Pantalla del juego Completa la Serie: Interfaz 2 y árbol de componentes

5.2.6 Resultados

La sección resultados pretende ofrecer un resumen de los resultados de las partidas jugadas por el tutelado, ofreciendo una visión general de las estadísticas de cada uno de los juegos. Para ello se ha hecho uso de graficas con la ayuda de la librería para graficas en Android llamada **MPAndroidChart** creada por Philipp Jahoda (Jahoda, 2014).

Al acceder a esta sección desde el menú lateral, se muestra un *fragment* (*fragment_resultados.xml*) con un *Listview* en el que se cargará la lista los tutelados asignados al voluntario que está utilizando la aplicación. Al seleccionar un tutelado de la lista, se accede a la *activity ResultadosActivity* cuyo *layout activity_resultados_general.xml* se puede ver en la Figura 56 en la que se hace uso de un *layout pieChart* de la librería MPAndroidChart en el que se dibujarán los datos de partidas jugadas por el titulado seleccionado en forma de grafica circular o *pie chart tal*. Los datos de las partidas se descargan del servidor mediante la clase *getPartidas* que se ejecuta en segundo plano ya que extiende a la clase *AsyncTask* de Android.

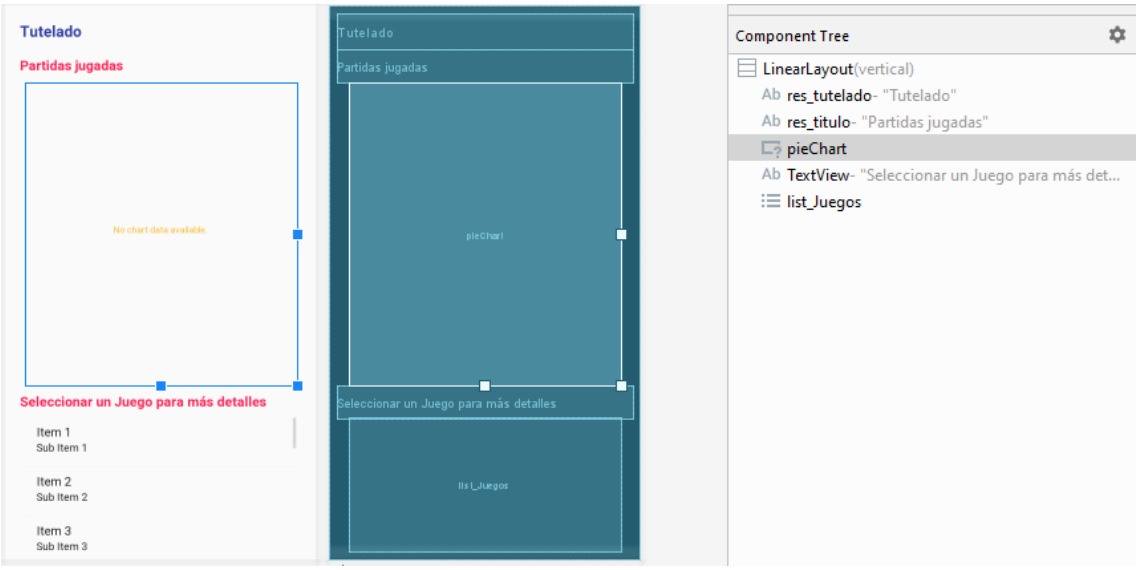

Figura 56 - Pantalla de Resultados: Interfaz y árbol de componentes

También se muestra un *ListView* donde se muestra la lista de juegos disponible y al seleccionar uno de estos juegos, se accede a otra activity donde se muestran más detalles de cada juego individualmente. Dicha *activity* se llama *ResultadosJuegoActivity* cuyo *layout* está definido en el archivo *activity_resultados_juego.xml*. Este *layout* se compone de un *pieChart* para dibujar la gráfica con el porcentaje de fallos y aciertos del juego seleccionado y de varios *TextView* para mostrar otras estadísticas del juego.

En la Figura 57 se puede ver el resultado producido por las clases y layouts descritos en este apartado:

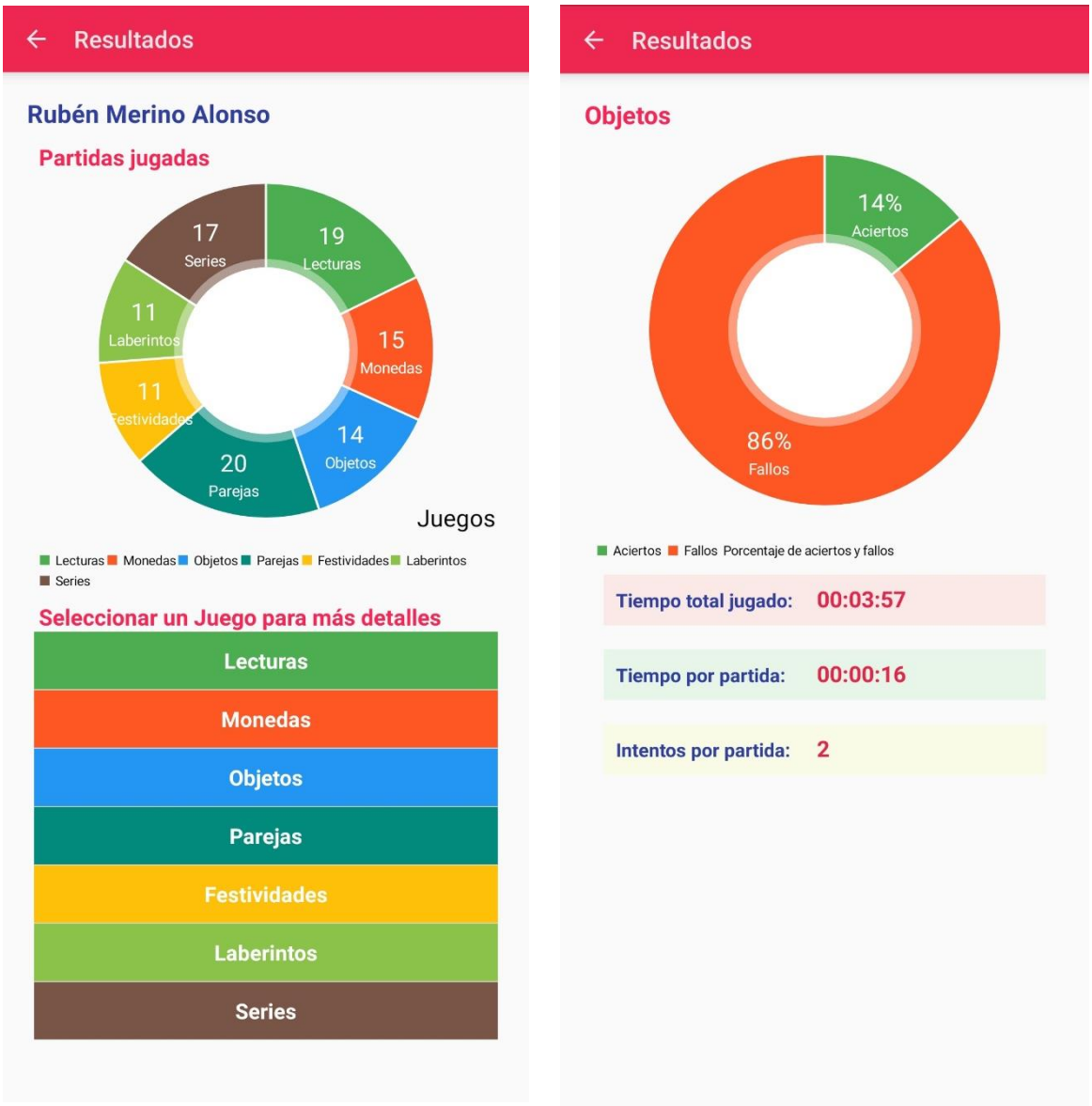

Figura 57 - Aspecto final de la sección Resultados

5.3 Comunicación entre la aplicación y el servidor

5.3.1 Introducción

Uno de los componentes fundamentales de la aplicación es la comunicación y sincronización con la base de datos alojada en el servidor, una comunicación que se lleva a cabo utilizando como interfaz intermedia una **API,** basada en el protocolo **REST** y desarrollada en el lenguaje **PHP,** gracias a la cual es posible el acceso de lectura y escritura en la **BDD** (Base de Datos). Se ha utilizado el protocolo **JSON** para la gestión y tratamiento de los datos intercambiados.

En este punto se procede a explicar detalladamente la base de datos, la API y las tecnologías y protocolos utilizados para llevar a cabo la comunicación completa entre la *app* y el servidor.

5.3.2 BDD

La base de datos con la que interactúa la app se compone de 6 tablas tal y como se puede observar el la Figura 58 de abajo.

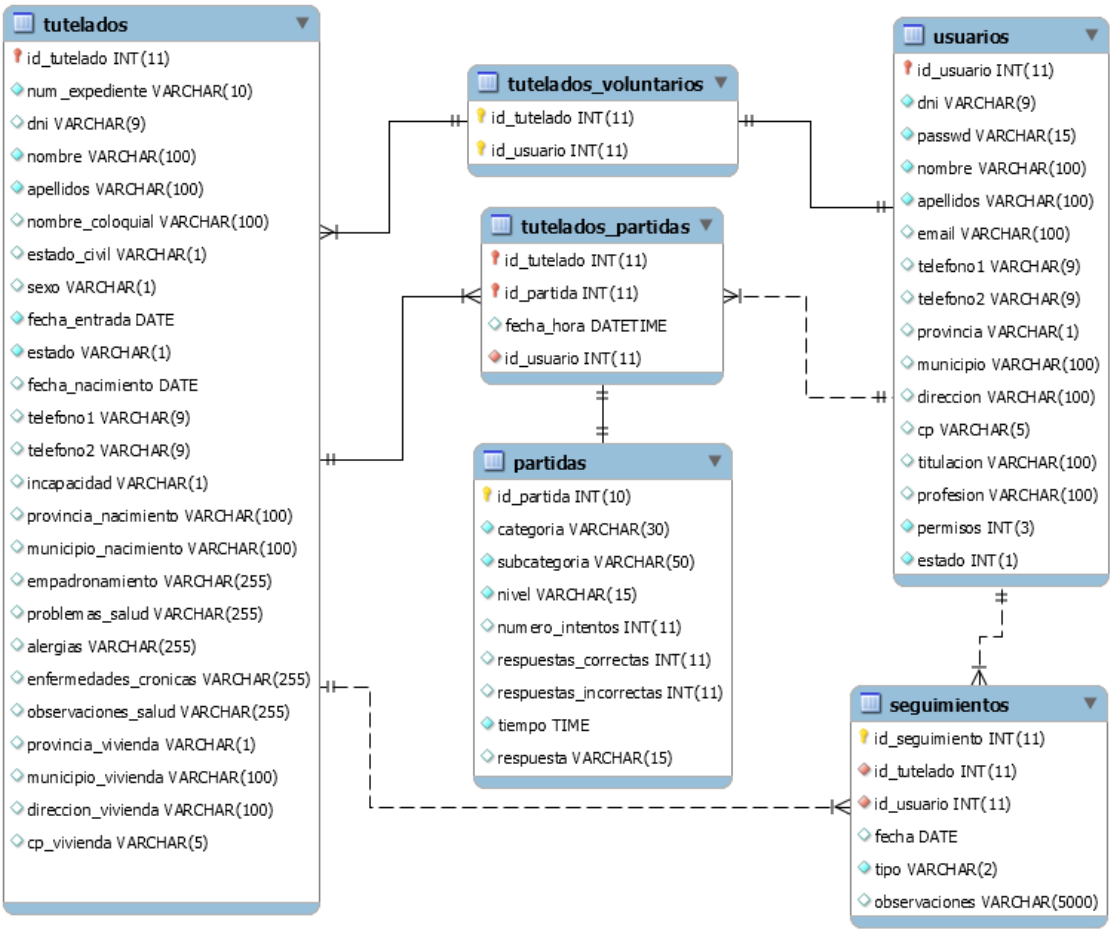

Figura 58 - Esquema de la base de datos de la APP

Las tablas son la siguientes:

- ***usuarios***: en esta tabla se guarda la información de los voluntarios, es decir los principales usuarios de la app.
- ***tutelados***: en esta tabla se guarda la información de los tutelados asignados a cada voluntario.
- ***seguimientos***: en esta tabla se guarda la información de los seguimientos hechos a los tutelados y llevados a cabo por los voluntarios.
- ***partidas***: en esta tabla se guardan los resultados de las partidas jugadas por los tutelados.
- ***tutelados_voluntarios***: en esta tabla se guarda el identificador de un tutelado y el identificador del voluntario al que está asignado. Es decir, se guarda la relación entre la tabla de ***tutelados*** y la de ***usuarios***.
- ***tutelados_partidas***: en esta tabla se guarda el identificador de una partida y la del tutelado al que corresponde. Es decir, se guarda la relación entre la tabla de ***tutelados*** y la de ***partidas***.

5.3.3 API

La **API** está alojada en el servidor web y, para su uso en la app, se ha de especificar la dirección URL del servidor. A continuación, se presenta un ejemplo que corresponde al servidor local del PC donde se hizo el desarrollo.

```
// URL de la API, Especificar la IP o la URL del servidor
private static final String API_URL = "http://192.168.0.151/mob_API/";
```

La API consta de varios componentes:

- *Index.php*: Es el fichero *php* principal que va a gestionar las peticiones de la APP a la API.

```php
<?php
header("Content-Type: text/html;charset=utf-8");

/**
 * Gestión de las peticiones a la API
 * GET y POST
 * Las peticiones se identifican mediante una etiqueta (tag)
 * Respuesta en formato JSON
 **/

  /**
 * POST
 */
if (isset($_POST['tag']) && $_POST['tag'] != '') {
    // get tag
    $tag = $_POST['tag'];

    // se incluyen la funciones de la BD
    require_once 'include/DB_Functions.php';
    $db = new DB_Functions();

    // respuesta
    $resp = array("tag" => $tag, "success" => 0, "error" => 0);
       // tipo de etiqueta login
    if ($tag == 'login') {
        $email = $_POST['email'];
        $password = $_POST['password'];

        // comprobar datos voluntario
        $voluntario = $db->getVoluntario($email, $password);
        if ($voluntario != false) {
            // voluntario encontrado
            // success = 1
            $resp["success"] = 1;
            $resp["voluntario"]["id"] = $voluntario["id_usuario"];
            $resp["voluntario"]["nombre"] = $voluntario["nombre"];
            $resp["voluntario"]["apellidos"] = $voluntario["apellidos"];
            $resp["voluntario"]["email"] = $voluntario["email"];
            $resp["voluntario"]["password"] = $voluntario["passwd"];
            //$resp["tutelados"]=$db->getTutelados($voluntario["id_voluntario"]);
            echo json_encode($resp);
        } else {
            // voluntario no encontrado
            // error = 1
            $resp["error"] = 1;
            $resp["error_msg"] = "Email o contraseña incorrectos!";
            echo json_encode($resp);
        }
    }
```

Figura 59 - Fichero Index.php de la API

- *config.php*: aquí se guardan los datos de acceso a la base de datos.
- *DB_Connect.php*: se definen las funciones *php* para conectar a la base de datos.
- *DB_Functions.php*: las funciones necesarias para hacer consultas y modificaciones en la base de datos.

```php
<?php
class DB_Functions {
    private $db;
    // constructor
    function __construct() {
        global $con;
        require_once 'DB_Connect.php';
        // Conexión con la BD
        $this->db = new DB_Connect();
        $con = $this->db->connect();
    }
    // destructor
    function __destruct() { }

    /**
     * Nuevo Seguimiento
     */
    public function addSeguimiento($idTutelado, $idUsuario, $fecha, $tipo, $observaciones) {
        global $con;
        $result = mysqli_query($this->$con,"INSERT INTO seguimientos(id_seguimiento, id_tutelado,
        id_usuario, fecha, tipo, observaciones) VALUES(NULL,'$idTutelado', '$idUsuario', '$fecha',
        '$tipo','$observaciones')");
        // Comprobación
        if ($result) {
            return true;
        } else {
            return false;
        }
    }

    /**
     * Editar Seguimiento
     */
    public function editSeguimiento($idSeg, $idTutelado, $idUsuario, $fecha, $tipo, $observaciones) {
        global $con;
        $result = mysqli_query($con,"UPDATE seguimientos
                                     SET id_seguimiento='$idSeg', id_tutelado='$idTutelado', id_usuario='$idUsuario',
                                     fecha='$fecha', tipo='$tipo', observaciones='$observaciones'
                                     WHERE id_seguimiento='$idSeg' ");
        // Comprobación
        if ($result) {
```

Figura 60 - Fichero DB_Functions.php de la API

5.3.4 PHP

PHP (acrónimo recursivo de *PHP: Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML (The PHP Group, 2001).

Ejemplo sencillo de uso del lenguaje PHP:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Ejemplo</title>
  </head>
  <body>
    <?php
      echo "¡Hola, soy un script de PHP!";
    ?>
  </body>
</html>
```

En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), las páginas de PHP contienen HTML con código incrustado que hace "algo" (en este caso, mostrar "¡Hola, soy un script de PHP!). El código de PHP está encerrado entre

las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP" (The PHP Group, 2019).

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga. Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales (The PHP Group, 2019).

Ejemplo de uso de PHP en la API

```php
/**
 * Get voluntario, email y password
 */
public function getVoluntario($email, $password) {
    global $con;
    $result = mysqli_query($con,"SELECT * FROM usuarios WHERE email = '$email'") or die(mysqli_error($con));
    // comprobar resultado
    $num_rows = mysqli_num_rows($result);
    if ($num_rows > 0) {
        $result = mysqli_fetch_array($result);
        //print_r($result);
        $password2 = $result['passwd'];
        // comprobar password
        if ($password2 == $password) {
            // autenticación correcta
            return $result;
        }
    } else {
        // voluntario no no encontrado
        return false;
    }
}
```

Figura 61 - Ejemplo de uso de PHP en la API

5.3.5 REST

La comunicación entre la *app* y la *API* se basa en el protocolo REST (Representational State Transfer), que es una interfaz para conectar varios sistemas basados en el protocolo HTTP y nos sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON (Openwebinars, 2018).

En el caso de la app desarrollada, se ha decido utilizar JSON como formato para el intercambio de datos entre la app y el servidor ya que es más ligero y legible en comparación al formato XML y además es el formato más usado en la actualidad.

Como punto de partida es posible enviar a un servicio un mensaje en formato JSON, el servicio lo recibirá y devolverá una respuesta tal y como se puede ver en la figura 62. Sin embargo, este intercambio se caracteriza por no tener ningún tipo de organización y es lo que habitualmente se denomina el **nivel 0** en la arquitectura REST.

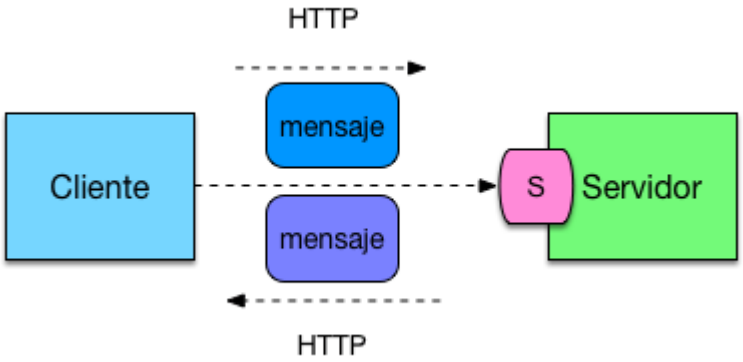

Figura 62 - Intercambio de mensajes en REST

Para solventar los problemas del nivel 0 serán necesarios nuevos niveles que se describen a continuación tal (Arquitecturajava, 2016):

Nivel 1 Recursos

En el **nivel 1**, en vez de tener servicios con métodos diversos, declaramos Recursos. Este estilo permite un primer nivel de organización permitiendo acceder a cada uno de los recursos de forma independiente, favoreciendo la reutilización y aumentando la flexibilidad.

Nivel 2 HTTP Verbs

Hasta este momento para realizar las peticiones se usa GET o POST indistintamente. En el nivel 2 las operaciones pasan a ser categorizadas de forma más estricta. Dependiendo de cada tipo de operación se utilizará un método diferente de envío.

- **GET:** Se usará para solicitar consultar a los recursos.
- **POST:** Se usará para insertar nuevos recursos.
- **PUT:** Se usará para actualizar recursos.
- **DELETE:** Se usará para borrar recursos.

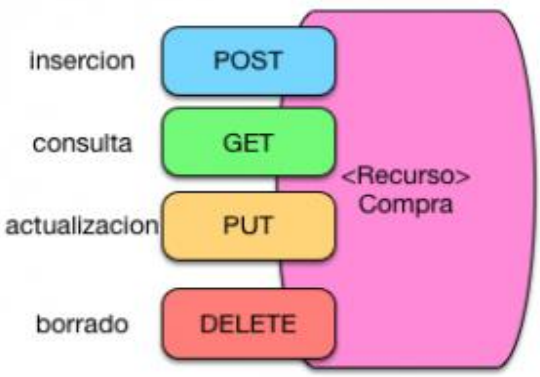

Figura 63 - HTTP Verbs en REST

HATEOAS (Nivel 3)

Todavía nos queda un nivel más que es el que se denomina HATEOAS (Hypertext As The Engine Of Application State). Para comprender el papel de este nivel, se plantea un ejemplo en el que se pretende acceder a un recurso Alumno vía REST. Si tenemos una Arquitectura a nivel 2, primero accederemos a ese recurso utilizando GET**.** En segundo lugar, deberemos acceder al recurso de Cursos para añadir al alumno al curso (línea roja, Figura 64) .

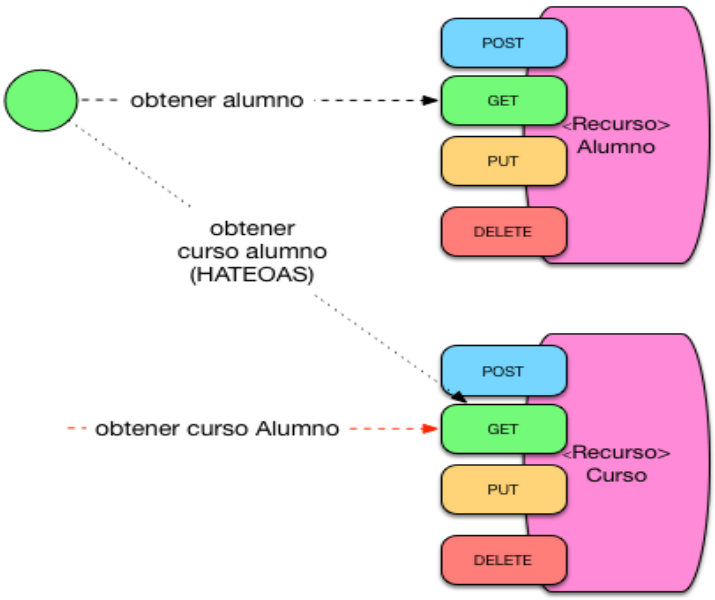

Figura 64 - HATEOAS (Nivel 3) en REST

Esto implica que el cliente que accede a los servicios REST asume un acoplamiento muy alto, debe conocer la *url* del Alumno y la del Curso. Sin embargo, si el recurso del Alumno contiene un link al recurso de Curso, esto no hará falta. Podríamos tener una estructura JSON como la siguiente:

{nombre:pedro, apellidos:"gomez", **cursos:"http://miurl/cursos"**}

Podremos acceder directamente al recurso de Curso utilizando las propiedades del Alumno, esto es **HATEOAS**. De esta forma se aumenta la flexibilidad y se reduce el acoplamiento. Construir arquitecturas sobre estilo REST no es sencillo y hay que ir paso a paso.

Algunas de las ventajas de REST son (Openwebinars, 2018):

- Nos **permite separar el cliente del servidor**, lo cual, mejora la portabilidad de la interfaz a otro tipo de plataformas, mejora la escalabilidad de los proyectos y además ayuda a que los componentes de un proyecto puedan avanzar de forma independiente.

- Es totalmente independiente de la plataforma, así que **podemos hacer uso de REST tanto en Windows, Linux, Mac** o el sistema operativo que nosotros queramos.
- Podemos hacer nuestra **API pública**, lo cual daría visibilidad al proyecto al estar al alcance de cualquier persona.

5.3.6 Okhttp

OkHttp es un cliente HTTP y HTTP/2 con licencia Apache para aplicaciones Android y Java. Es sencillo de usar y con unas funcionalidades muy interesantes (OkHttp, 2019)**:**

- Soporte HTTP/2 y SPDY
- Soporte realizar peticiones compartiendo un socket
- Soporte GZIP
- Cacheo peticiones
- Reintento si servicio tiene varias IPs
- Soporta Android y Java

A continuación, veremos algunos ejemplos de uso de *OkHttp*:

Extraer cadena de una URL

```
OkHttpClient client = new OkHttpClient();

String run(String url) throws IOException {
  Request request = new Request.Builder()
      .url(url)
      .build();

  Response response = client.newCall(request).execute();
  return response.body().string();
}
```

Figura 65 - Extraer cadena de una URL con OkHttp (OkHttp, 2019)

POST a un Servidor:

```
public static final MediaType JSON
    = MediaType.parse("application/json; charset=utf-8");

OkHttpClient client = new OkHttpClient();

String post(String url, String json) throws IOException {
  RequestBody body = RequestBody.create(JSON, json);
  Request request = new Request.Builder()
      .url(url)
      .post(body)
      .build();
  Response response = client.newCall(request).execute();
  return response.body().string();
}
```

Figura 66 - POST a un Servidor con OkHttp (OkHttp, 2019)

Se ha utilizado esta librería para gestionar las peticiones REST entre la API y la APP, para ello se han importado los componentes utilizados de la librería en el proyecto mediante su repositorio:

import com.squareup.okhttp.FormEncodingBuilder;

import com.squareup.okhttp.OkHttpClient;

import com.squareup.okhttp.Request;

import com.squareup.okhttp.RequestBody;

import com.squareup.okhttp.Response;

5.3.7 JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, *Standard ECMA-262 3rd Edition* Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos (Ecma, 2017).

JSON está constituido por dos estructuras: Una colección de pares de nombre/valor y Una lista ordenada de valores. Estas estructuras se podrían considerar universales ya que, virtualmente, todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras, que se presentan de estas formas (Ecma, 2017):

Un **objeto** es un conjunto desordenado de pares nombre/valor. Un objeto comienza con **{** (llave de apertura) y termine con **}** (llave de cierre). Cada nombre es seguido por **:** (dos puntos) y los pares nombre/valor están separados por "**,** (coma)".

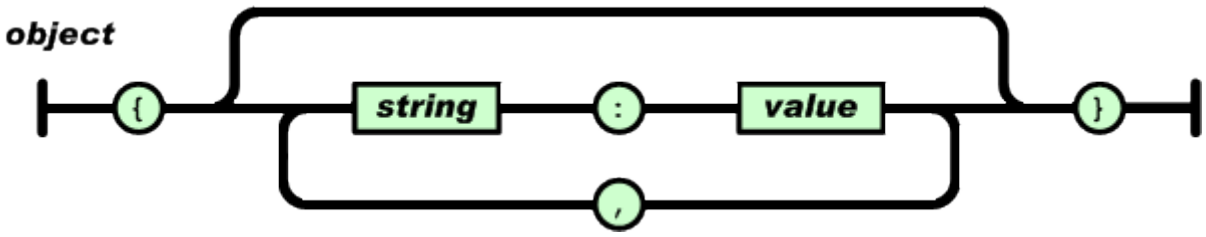

Figura 67 - Object en JSON (Ecma, 2017)

Un **array** es una colección de valores. Un array comienza con **[** (corchete izquierdo) y termina con **]** (corchete derecho). Los valores se separan por **,** (coma).

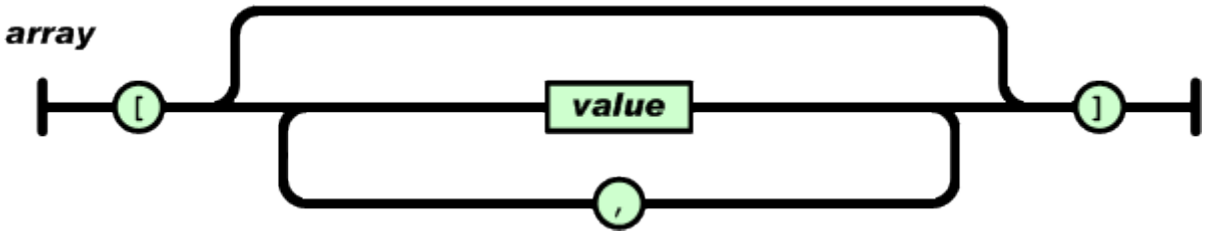

Figura 68 - Array en JSON (Ecma, 2017)

Un **valor** puede ser una *cadena de caracteres* con comillas dobles, o un *número*, o **true** o **false** o **null**, o un *objeto* o un *arreglo*. Estas estructuras pueden anidarse.

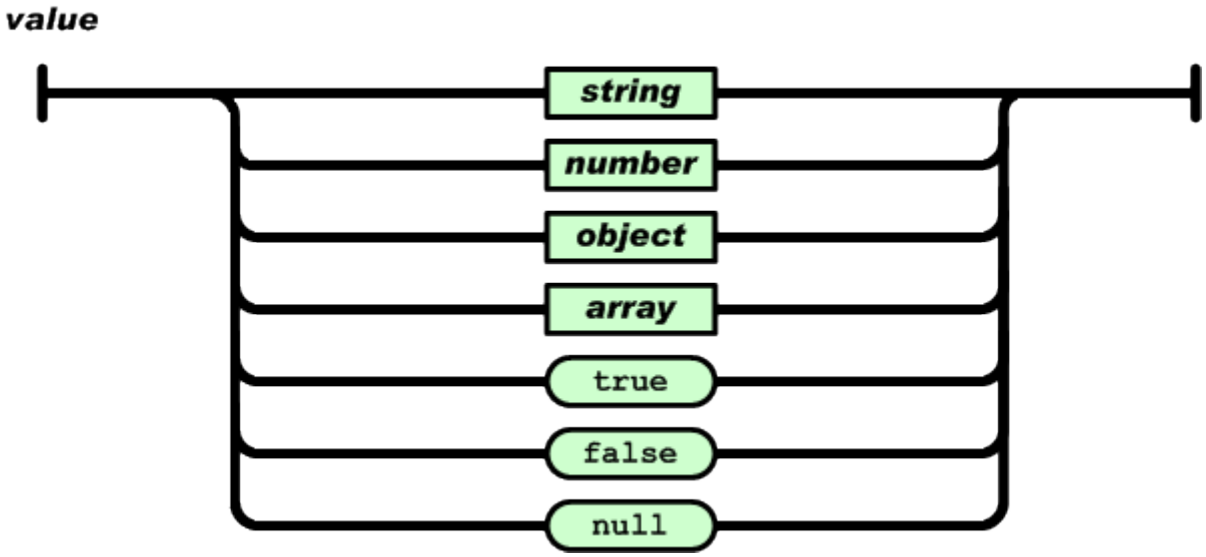

Figura 69 - Value en JSON (Ecma, 2017)

Una **cadena de caracteres** es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape. Un carácter está representado por una cadena de caracteres de un único carácter. Una *cadena de caracteres* es parecida a una cadena de caracteres C o Java.

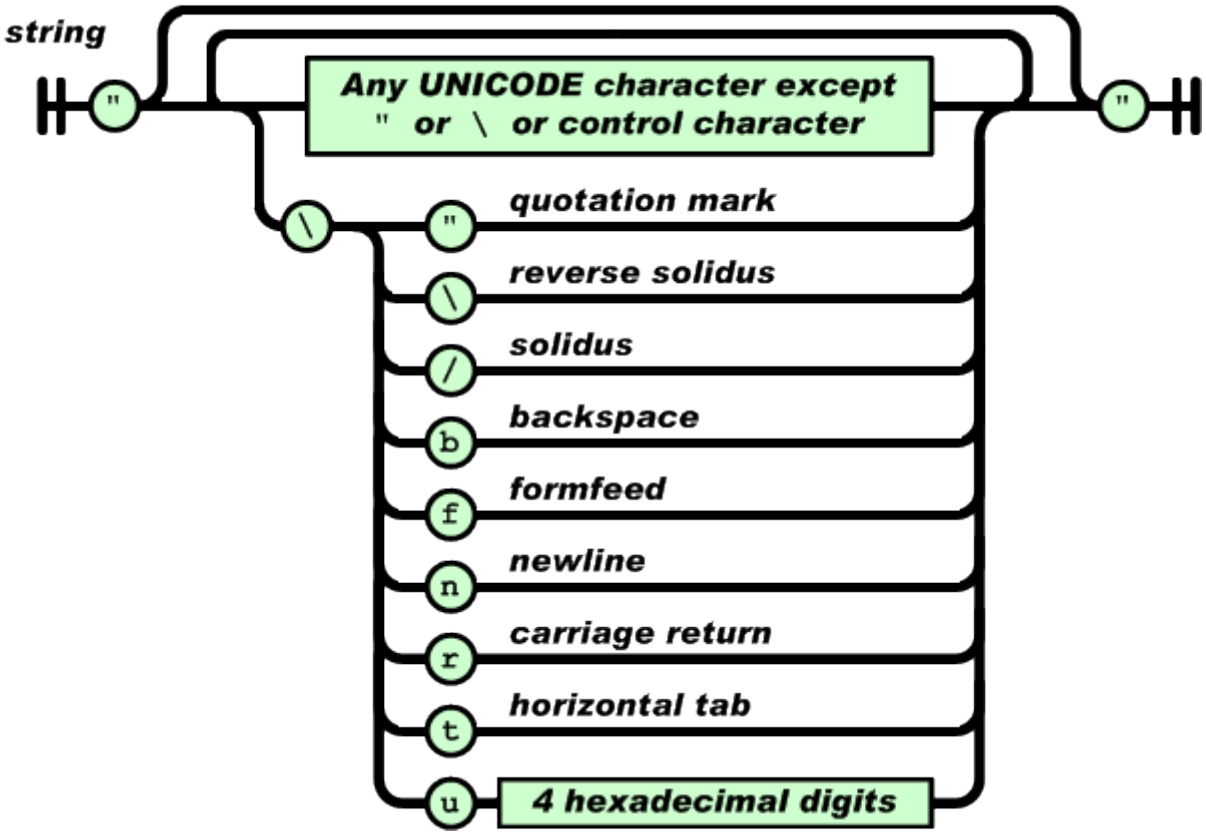

Figura 70 - String en JSON (Ecma, 2017)

Un **number** es similar a un número C o Java, excepto que no se usan los formatos octales y hexadecimales.

En nuestro caso, se ha empleado JSON para la gestión de los datos intercambiados entre la APP y el servidor. En la figura 71 se puede observar un ejemplo de uso de JSON *y* OkHttp en la app*:*

```
public JSONObject loginVoluntario(String email, String password)    {

    OkHttpClient client = new OkHttpClient();
    // Building Parameters
    String login_tag = "login";
    RequestBody formBody = new FormEncodingBuilder()
            .add( name: "tag", login_tag)
            .add( name: "email", email)
            .add( name: "password", password)
            .build();
    Request request = new Request.Builder()
            .url(API_URL)
            .post(formBody)
            .build();
    Response response;
    try {
        response = client.newCall(request).execute();
        String jsonData = response.body().string();
        ////Log.e("mobAPI", jsonData);
        if (response.isSuccessful()) {
            jObj = new JSONObject(jsonData);
            //Log.e("mobAPI", response.body().string());
        }           //Log.e("mobAPI", jsonData);
        if (response.isSuccessful()) {
            jObj = new JSONObject(jsonData);
            //Log.e("mobAPI", jsonData);
        }
        if (!response.isSuccessful()) {
            Log.e( tag: "mobAPI", jsonData);
        }
    } catch (IOException | JSONException e) {
        e.printStackTrace();
    }

    return jObj;
}
```

Figura 71 - Ejemplo de uso de JSON y OkHttp en la APP

Capítulo 6 – Aplicación socio sanitaria. Manual de Usuario

Capítulo 6 – Aplicación socio sanitaria. Manual de Usuario

En este capítulo se pretende elaborar un manual de usuario en el cual se detallan los pasos a seguir por parte del usuario para poder hacer uso de las funcionalidades de la aplicación.

6.1 Requisitos técnicos

A continuación, se exponen los requisitos que ha de cumplir el dispositivo en el cual se va a ejecutar la aplicación. Estos requisitos serán necesarios para que la aplicación funcione de manera correcta y sin errores relacionados con la capacidad de procesamiento requerida.

- Sistema operativo Android 4.0 y versiones superiores.
- Resolución de pantalla de 480p o superior
- Se recomienda al menos 1GB de RAM.
- Espacio mínimo para la instalación de la *app* 25 MB.
- Conexión a internet mediante Wifi o red de datos.

6.2 Voluntarios

Los voluntarios serán los usuarios principales de la aplicación, pues serán ellos los poseedores de los dispositivos móviles proporcionado por la fundación. Debido a que los tutelados son personas mayores con dificultades para manejar este tipo de dispositivos y por tanto esta aplicación, el papel de los voluntarios, además de realizar los seguimientos, será el de asistir a los tutelados a la hora de realizar los ejercicios de los juegos de la aplicación. Es decir, el voluntario, después de haber iniciado la aplicación y habiéndose identificado, ejecutará el juego correspondiente y le irá indicando al tutelado los pasos necesarios para realizar el ejercicio propuesto, ya sea leyendo los textos que el tutelado pueda tener dificultades al leer o pulsando los botones y demás controles que puedan presentar alguna dificultad.

6.2.1 Identificación del voluntario

Como usuario principal de la aplicación, el voluntario ha de identificarse (*Login*) antes de poder acceder a la aplicación. Para ello el voluntario, tal y como se muestra en la Figura 72, ha de introducir su email y su contraseña y pulsar el botón ***Iniciar Sesión***. La aplicación conectará con el servidor Web y comprobará si los datos son correctos o no. Una vez hecha la autentificación de manera correcta, se muestra la pantalla principal de la aplicación, desde la cual el voluntario podrá acceder a las distintas secciones de la aplicación.

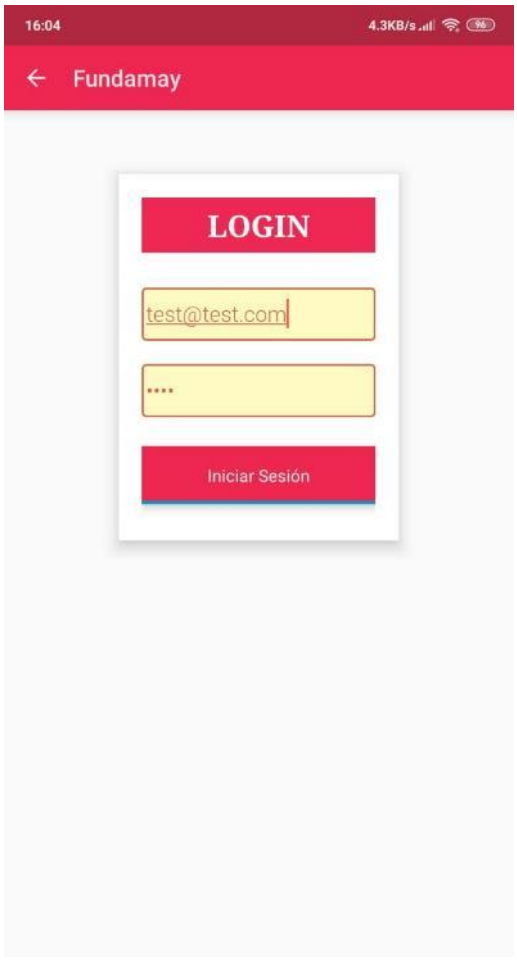

Figura 72 - Pantalla de autentificación del voluntario

La aplicación no permite registrar nuevos voluntarios en el sistema, el registro es realizador por el administrador de la fundación desde el sistema web. Por tanto, los voluntarios han de estar previamente registrados y dados de alta en el sistema web antes de poder acceder a la aplicación.

Otro de los aspectos a tener en cuenta es que la primera vez que se realiza el *Login*, como es lógico, es necesaria una conexión a Internet en el dispositivo utilizado. En posteriores autentificaciones ya no será necesaria una conexión a Internet, aunque si es recomendable tener una conexión ya que pudo haber cambios en la base de datos desde el ultimo acceso. Sin embargo, sí que será necesaria la conexión a internet para realizar las funciones del voluntario ya que la aplicación necesita descargar la lista de tutelados asignados a cada voluntario y también necesita descargar los seguimientos del servidor, así como también necesita una conexión para guardar los nuevos seguimientos que se realicen.

6.2.2 Seguimientos

Una vez autentificado el voluntario, podrá acceder a uno a uno de los dos grandes bloques de la aplicación, los seguimientos. En este bloque, el voluntario podrá realizar nuevos seguimientos, poder ver una lista de los seguimientos realizados hasta la fecha y también poder editar y actualizar los ya existentes. Antes de acceder a estas funciones, el voluntario ha de seleccionar un tutelado de una lista que corresponde a los tutelados asignados a él.

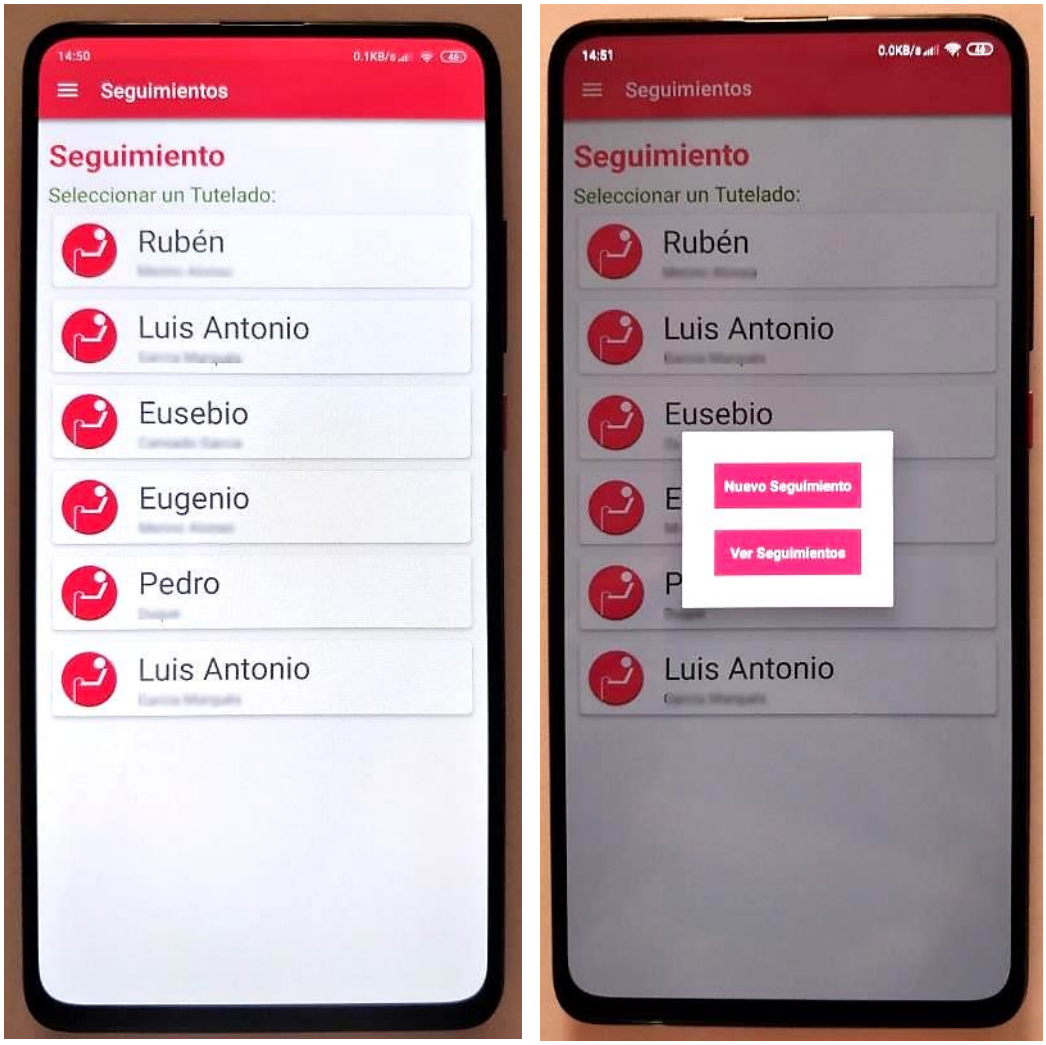

Figura 73 - Lista de tutelados

A continuación, se explican las tres funciones que el voluntario puede realizar en la sección de Seguimientos:

- **Nuevo Seguimiento:** El voluntario podrá realizar un nuevo seguimiento del tutelado seleccionado. Para ello ha de introducir la fecha del seguimiento y lo que es el seguimiento en sí, ha de introducirlo en el campo ***Observaciones.*** En la Figura 74 se puede ver el aspecto de esta sección.

  Una vez introducidos los datos y después de haber pulsado el botón para sincronizar, el seguimiento se envía al servidor Web donde se guardará en la base de datos para poder acceder a él desde otro dispositivo móvil o desde el propio sistema Web.

Figura 74 - Nuevo Seguimiento

- **Ver Lista de Seguimientos:** El voluntario podrá ver los seguimientos realizados hasta la fecha del tutelado seleccionado. Se muestra una lista de los seguimientos, tal y como se muestra en la Figura 75 y al pulsar sobre un seguimiento, se le ofrece al voluntario la opción de Ver Seguimiento y la de Editar seguimiento.

- **Ver Seguimiento:** El voluntario podrá leer el seguimiento seleccionado y no podrá realizar ninguna otra acción tal y como se muestra en la Figura 76.

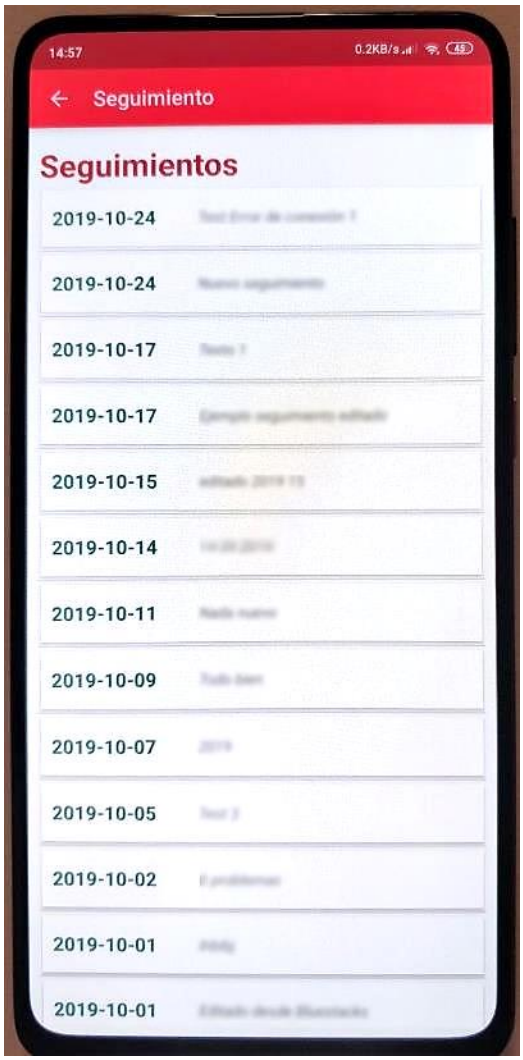

Figura 75 - Ver Lista de Seguimientos

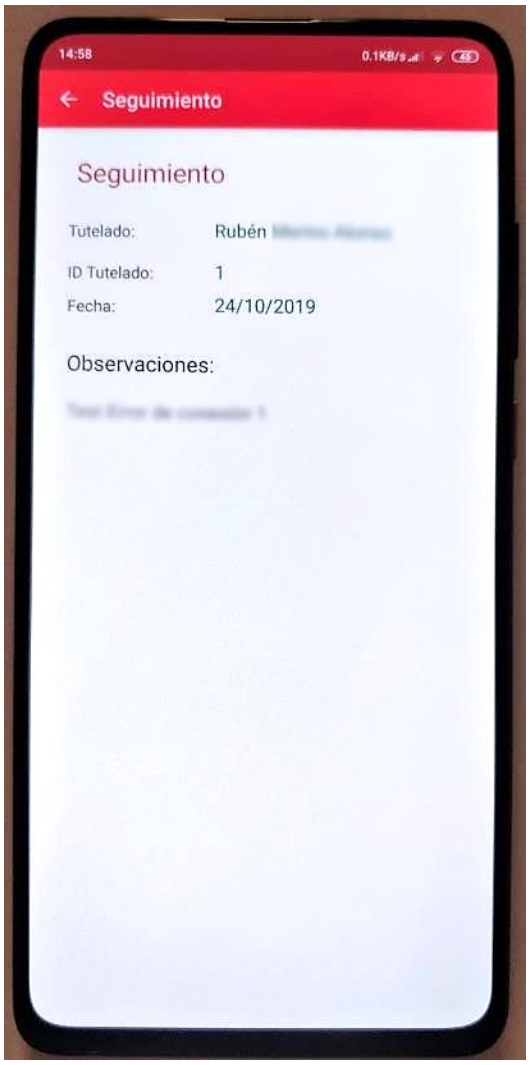

Figura 76 - Ver Seguimiento

- **Editar Seguimiento:** El voluntario podrá editar el seguimiento seleccionándolo en la lista de ***Seguimientos*** (Figura 77). Para ello ha de actualizar la fecha si es necesario e introducir las nuevas observaciones en el campo ***Observaciones.*** En la figura de la derecha se puede ver el aspecto de esta sección.

  Una vez actualizados los datos y después de haber pulsado el botón para sincronizar, el seguimiento se envía al servidor Web donde se guardará en la base de datos para poder acceder a él desde otro dispositivo móvil o desde el propio sistema Web.

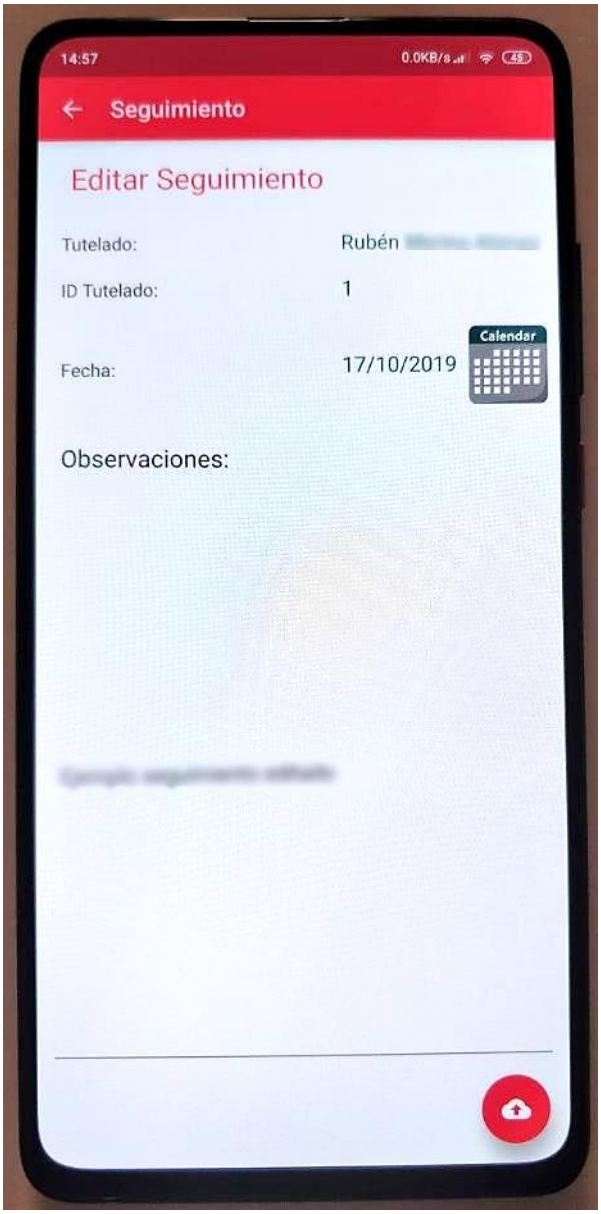

Figura 77 - Editar Seguimiento

6.2.3 Asistencia a los Tutelados

Otra de las funciones de los voluntarios será la de ayudar a los tutelados en sus actividades de vida diarias. Una de esas actividades es la de ejercitar sus mentes y realización de ejercicios y juegos que les ayuden en su rehabilitación cognitiva. Por tanto, la función de los voluntarios será la de proponerles los juegos adecuados para su rehabilitación.

A continuación, se destacan algunas de las tareas de los voluntarios a la hora de ofrecer asistencia al tutelado para la realización de los juegos:

- **Ejecutar un juego**: el voluntario seleccionará el juego adecuado y lo ejecutará. También deberá orientar el dispositivo en la orientación más adecuada para una mayor facilidad para el tutelado a la hora de realizar los ejercicios.
- **Leer el enunciado de un ejercicio**: el voluntario ayudará al tutelado en la lectura del enunciado del ejercicio en caso de que el tutelado tenga dificultades para leerlo.
- **Explicar el ejercicio y los pasos para su realización**: después de leer el enunciado y, siempre y cuando el tutelado no pueda entender en qué consiste el ejercicio por sí mismo, el voluntario ha de explicarle al tutelado en que consiste el juego o ejercicio e indicarle los pasos para su resolución, pero no ha de indicarle la solución del ejercicio.
- **Ayuda en los controles y mecanismos del juego**: otra de las funciones sería la de ayudar al tutelado en el manejo de los controles del juego, ya sea pulsando los botones y demás controles que puedan presentar alguna dificultad o seleccionando las figuras que le indique el tutelado.
- **Otras dificultades**: el voluntario ayudará al tutelado en cualquier dificultad que pueda presentarse a la hora de realizar los ejercicios, evitando por supuesto resolver los ejercicios en lugar del tutelado olvidando así el objetivo de los juegos, que no es más que el de ayudar a ejercitar las mentes de los tutelados.

6.3 Tutelados

Tal y como ya se ha explicado en el Capítulo 4, el tutelado sólo podrá acceder a los juegos y sólo en acompañamiento de los voluntarios que le asistirán y le ayudará en su uso. La razón de ello es que los tutelados son personas mayores que tienen ciertas dificultades, ya sean cognitivas o físicas, para realizar sus actividades de vida diaria. Lo cual obliga a que estén siempre acompañados por los voluntarios a la hora de realizar los ejercicios propuestos en los juegos. En la sección de juegos, la función de los voluntarios será la de proponerles los juegos adecuados para su rehabilitación.

6.3.1 Juegos

A continuación, se describen los juegos incluidos en la aplicación y una breve explicación de cómo resolverlos. Están organizados en distintas categorías y cada categoría corresponde a un tipo de habilidad que se pretende que el tutelado ejercite. En la versión actual de la aplicación solo se ha incluido un juego por categoría y la idea es que se amplíe el número de juegos en futuras versiones de la aplicación.

6.3.1.1 Atención

En esta categoría se pretende fortalecer la capacidad del tutelado para la asimilación de conceptos y su organización correcta. Para ello, los juegos de atención plantean distintos tipos de retos que han de ser solucionados por los tutelados de forma interactiva. Mediante la realización de los ejercicios propuestos a lo largo de estos juegos de atención, los tutelados mejoran su capacidad de concentración y desarrollan la capacidad de elaborar decisiones lógicas. Trabajar en el análisis de los retos y los problemas que plantean estos juegos de atención, y poder llegar a solucionarlos es un estímulo externo que beneficia su capacidad de aprendizaje y su memoria visual (CPDC, 2008).

❖ Juego: ***LECTURAS***

Este juego trata de leer un texto y responder a una serie de preguntas de tipo test relacionadas con el texto. Tal y como se muestra en la Figura 78, en el juego se presenta una lista de textos de dos niveles de dificultad, fácil y normal.

Una vez seleccionada una lectura de la lista, aparece una pantalla con el enunciado del ejercicio *"Lea atentamente el siguiente texto"* y se muestra el texto para poder ser leído y un botón para avanzar a la siguiente pantalla una vez leído el texto tal y como se muestra en la figura 79.

Figura 78 - Lista Lecturas

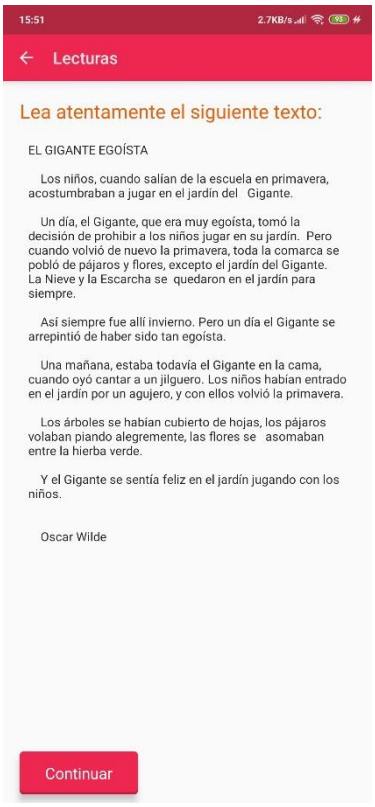

Figura 79 - Lecturas texto

En la siguiente pantalla se muestran las preguntas a contestar además de un botón para poder comprobar las respuestas elegidas y otros dos botones, que sólo aparecen después de haber pulsado el anterior, para poder finalizar el juego o volver a leer el texto y responder de nuevo. En la Figura 80 se muestra lo descrito anteriormente.

Al pulsar el botón para finalizar el juego, se muestra una pantalla con los resultados del juego, en los que se mide el tiempo empleado en la lectura del texto, el tiempo empleado en la resolución y el porcentaje de aciertos tal y como se muestra en la Figura 81. Estos resultados se sincronizan con el servidor.

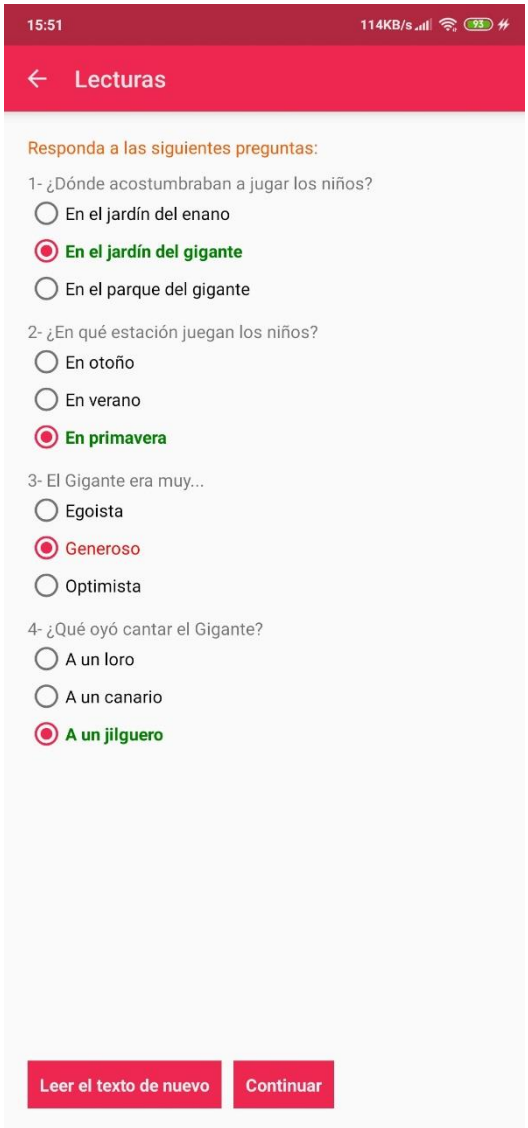

Figura 80 - Lecturas: Preguntas

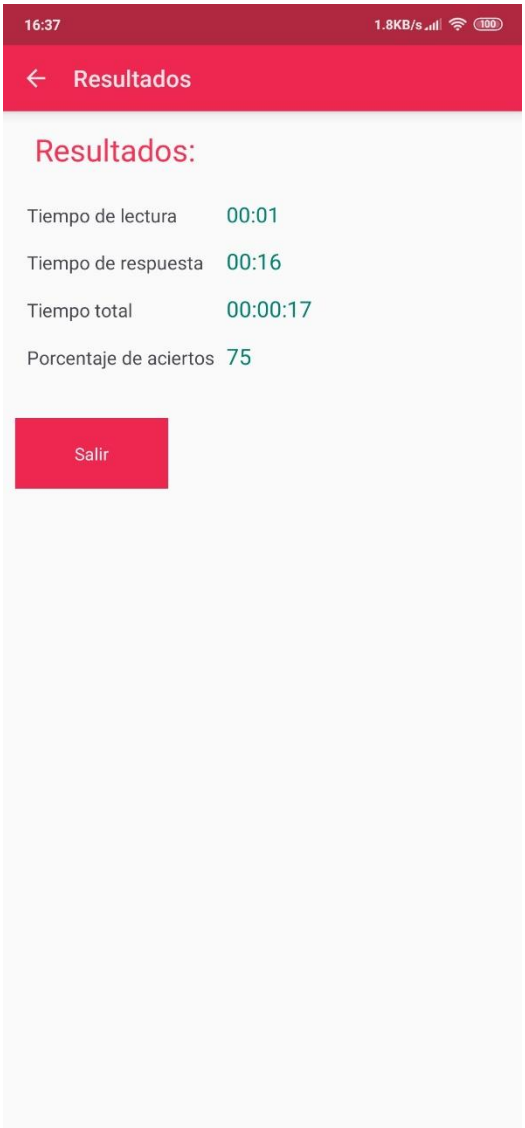

Figura 81 - Resultados Preguntas

6.3.1.2 Cálculo

En esta categoría se pretende que los tutelados ejerciten su capacidad de cálculo mental mediante los juegos incluidos. La habilidad para manipular números en una operación aritmética implica varios procesos:

- El conocimiento básico de las tablas numéricas.
- La comprensión de los conceptos de las operaciones aritméticas.
- La secuencia y el procedimiento necesario para realizar los cálculos.

Las habilidades numéricas son consideradas como una de las habilidades instrumentales más importantes en las sociedades de consumo (comprar, vender, utilización del dinero) (CPDC, 2008).

❖ Juego: ***MONEDAS***

Este juego pretende simular una situación de pagar una compra realizada en una tienda. Se trata de introducir el número correcto de monedas necesarias para poder pagar una cantidad determinada en euros. En el juego sólo se han incluido cantidades factibles de ser pagadas mediante todos los tipos de monedas de euro existentes (un céntimo, dos céntimos, … hasta dos euros). En la Figura 82 se muestra el aspecto de lo descrito anteriormente. Una vez introducidas las cantidades correspondientes, se pulsa el botón pagar y así finaliza el juego.

Figura 82 - Cálculo monedas

Al finalizar el juego, se muestra una pantalla con los resultados del juego, en los que se mide el tiempo empleado en la resolución del ejercicio y se indica si la respuesta es correcta o incorrecta tal y como se muestra en la Figura 83. Estos resultados se sincronizan con el servidor.

Figura 83 - Resultados Monedas

6.3.1.3 Lenguaje

En esta categoría se pretende que los tutelados ejerciten su capacidad de lenguaje mediante los juegos incluidos. Los objetivos que se pretende conseguir en esta categoría son (CPDC, 2008):

- Ejercitar lenguaje expresivo espontáneo y la fluidez verbal.
- Favorecer la capacidad de denominación.
- Ejercitar habilidades lectoescritura.
- Ejercitar lenguaje automático (pg.: mediante la asignación de nombres de objetos).
- Ejercitar otras funciones a través del lenguaje: abstracción, razonamiento, juicio crítico y memoria semántica.

❖ Juego: ***OBJETOS***

En este juego se muestra una serie de imágenes de objetos de diferente tipo y se le pide al tutelado que indique el nombre de cada uno de los objetos. Para ello, el tutelado ha de pulsar el botón *Seleccionar* que hay debajo de cada objeto y seleccionar el nombre del objeto de una lista de nombres tal y como se muestra en la figura 84.

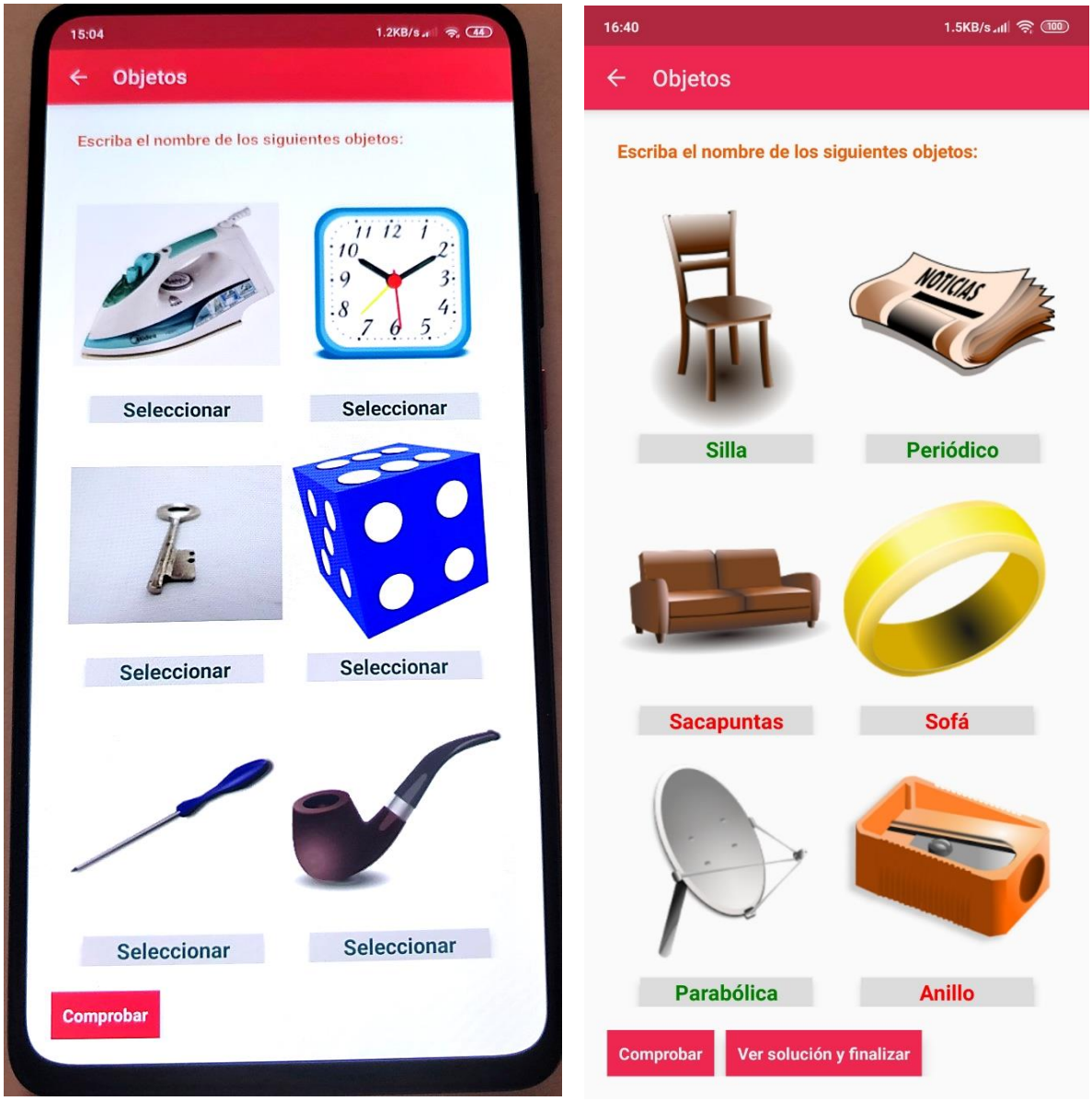

Figura 84 - Lenguaje Objetos

Una vez seleccionados todos los nombres, el tutelado deberá pulsar el botón *Comprobar* para ver si ha seleccionado los nombres correctamente. En este punto el tutelado puede volver a intentar seleccionar los nombres incorrectos que aparecen en rojo o puede finalizar el juego y ver la solución correcta.

Al finalizar el juego, se muestra una pantalla con los resultados del juego, en los que se mide el tiempo empleado en la resolución del ejercicio, se indica el número de respuestas correcta e incorrectas y el número de intentos tal y como se muestra en la Figura 85.

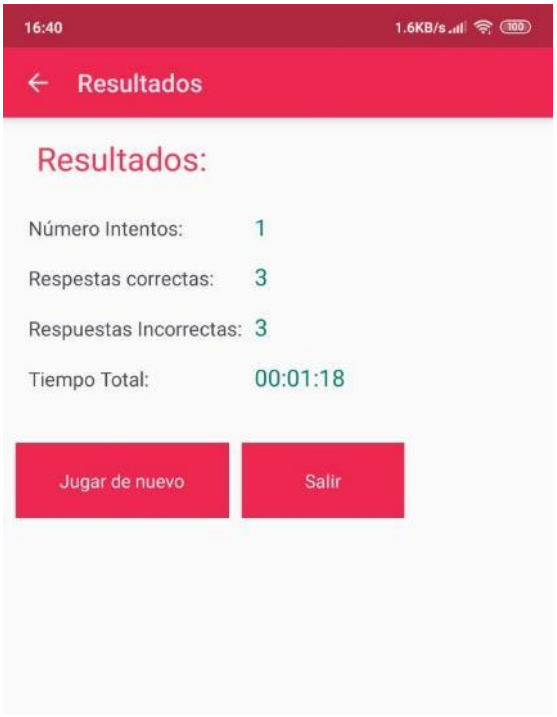

Figura 85 - Resultados Objetos

6.3.1.4 Memoria

La pérdida de la memoria a corto o largo plazo ejerce un efecto debilitante sobre la calidad de vida de una persona mayor y pone bajo presión a sus seres queridos y familiares. En esta categoría se pretende ayudar a las personas mayores a mejorar su memoria poniendo en práctica juegos diseñados para aumentar la agudeza y la percepción mental mejorando así las capacidades de memorización del tutelado.

- Juego: ***PAREJAS***

Este juego consiste en una rejilla compuesta por figuras iguales e irreconocibles, al pulsar sobre una de estas figuras se voltea y se muestra una imagen de algo reconocible (objeto, persona, animal…). La idea del juego es que el tutelado vaya volteando las figuras para encontrar parejas de imágenes iguales. Por ejemplo, si se voltea una figura y se encuentra un balón, si ya había aparecido ese balón en otro sitio, entonces el tutelado debe seleccionar esa figura, de la cual debe recordar la ubicación donde ya había aparecido, y así formar una pareja de balones. Deberá repetir el proceso hasta formar todas las parejas. El número de figuras de la rejilla varía en función del nivel de dificultad elegido. En la figura 86 se muestra el aspecto de una partida en curso.

Una vez encontradas todas las parejas, aparecerá la pantalla de resultados en la cual se muestra el tiempo empleado en la resolución del ejercicio, se indica el número de respuestas correcta (relevante en caso de que el tutelado finalice la partida antes de revelar todas las parejas) y el número de intentos (Cada dos figuras reveladas corresponde a un intento) tal y como se muestra en la Figura 87.

Figura 86 – Memoria Parejas

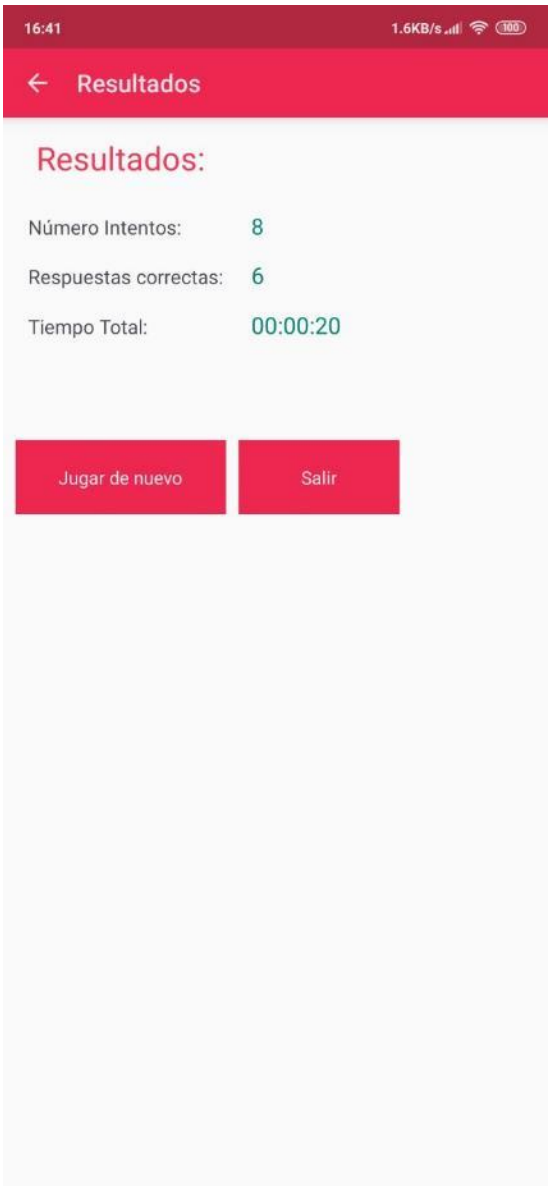

Figura 87 - Resultados Parejas

6.3.1.5 Orientación

En esta categoría se compone de ejercicios de estimulación cognitiva para estimular las capacidades básicas de orientación temporal para mantener a la persona en contacto con la realidad que le rodea. El objetivo es conseguir que la persona con demencia esté orientada el mayor tiempo posible. Se trabajan aspectos más recientes, por ejemplo, el día, mes, año y estación.

- Juego: ***FESTIVIDADES***

Este juego consiste en una serie de preguntas test sobre las fechas más importantes del año, las cuales deben ser respondidas por el tutelado eligiendo a qué festividad corresponde cada fecha. Las fechas varían según el nivel de dificultad elegido, correspondiendo las fechas más conocidas a los niveles fáciles y las menos conocidas a los más difíciles. En la figura 88 se muestra un ejemplo de una partida.

Una vez respondidas todas las preguntas sobre fechas, el jugador puede finalizar la partida y ver los resultados en la siguiente pantalla en la que se muestra el tiempo total invertido por el tutelado en responder las preguntas y también se muestra el porcentaje de aciertos.

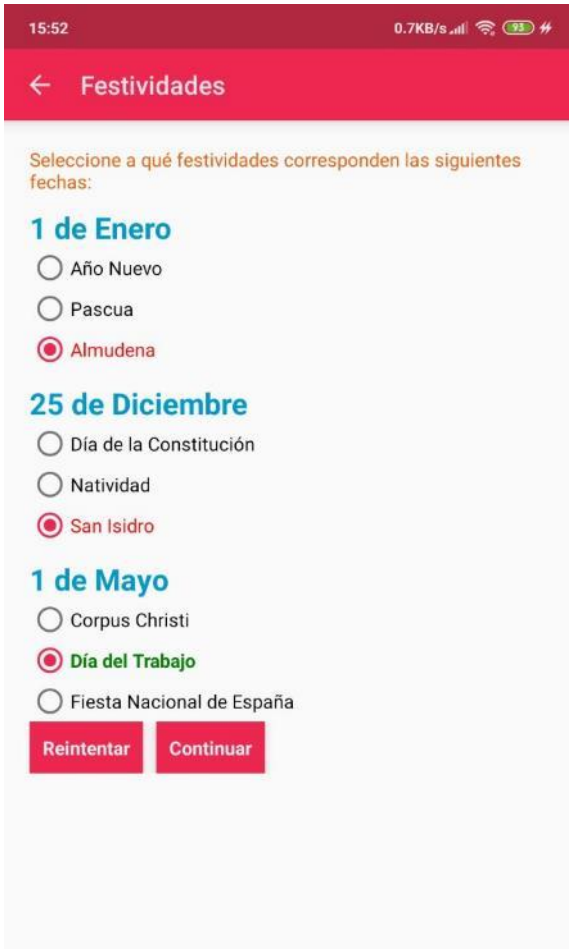

Figura 88 - Orientación Festividades

6.3.1.6 Percepción

La percepción es la capacidad mental que nos permite integrar y/o reconocer aquello que nos llega a través de nuestros sentidos. Nos permite reconocer aquellos objetos a los que prestamos atención y/o crear patrones propios de conocimiento, por lo tanto, debe producirse un encuentro entre la información sensorial y los archivos de memoria, dando paso a la percepción o interpretación de la realidad (CPDC, 2008).

El objetivo de los juegos de esta categoría es estimular la capacidad del tutelado para identificar los objetos de su entorno, mejorando así su habilidad de interpretar el significado de la información que recibe visualmente y detectar esos objetos con mayor rapidez.

❖ Juego: ***LABERINTOS***

En este juego se le muestra un laberinto al tutelado y se le pide que indique cuál es el camino correcto que debe seguir un sujeto para llegar s su objetivo al otro lado del laberinto. La dificultad de los laberintos varía en función del nivel de dificultad. En la Figura 89 se muestra la lista de la cual el tutelado puede elegir el laberinto a resolver.

Una vez encontrada la salida correcta y pulsada la opción correspondiente, se muestra en la siguiente pantalla el tiempo empleado en la resolución del ejercicio y se indica si la respuesta es correcta o no.

Figura 89 – Percepción Laberintos

6.3.1.7 Razonamiento

La capacidad de razonamiento nos permite secuenciar y organizar la información, suprimir los datos irrelevantes y nos da la posibilidad de comprender el lenguaje abstracto y la información visual. El objetivo de los juegos de esta categoría es el de estimular la capacidad de razonamiento del tutelado para ayudarle en sus actividades de vida diaria y mejorar sus funciones cognitivas (CPDC, 2008).

- Juego: ***COMPLETA LA SERIE***

En este juego se le muestra una serie de figuras al tutelado que son diferente entre sí y colocada de tal manera que forman una serie que se repite. Se le pregunta al tutelado que elija de otra lista de figuras cuál es la figura que completa o continua la serie propuesta.

En la figura 90 se muestra el aspecto de la partida en curso. Una vez seleccionada la figura se pasa a la pantalla de resultados donde se indica el tiempo total empleado en la resolución del ejercicio y se indica si la respuesta es correcta o no.

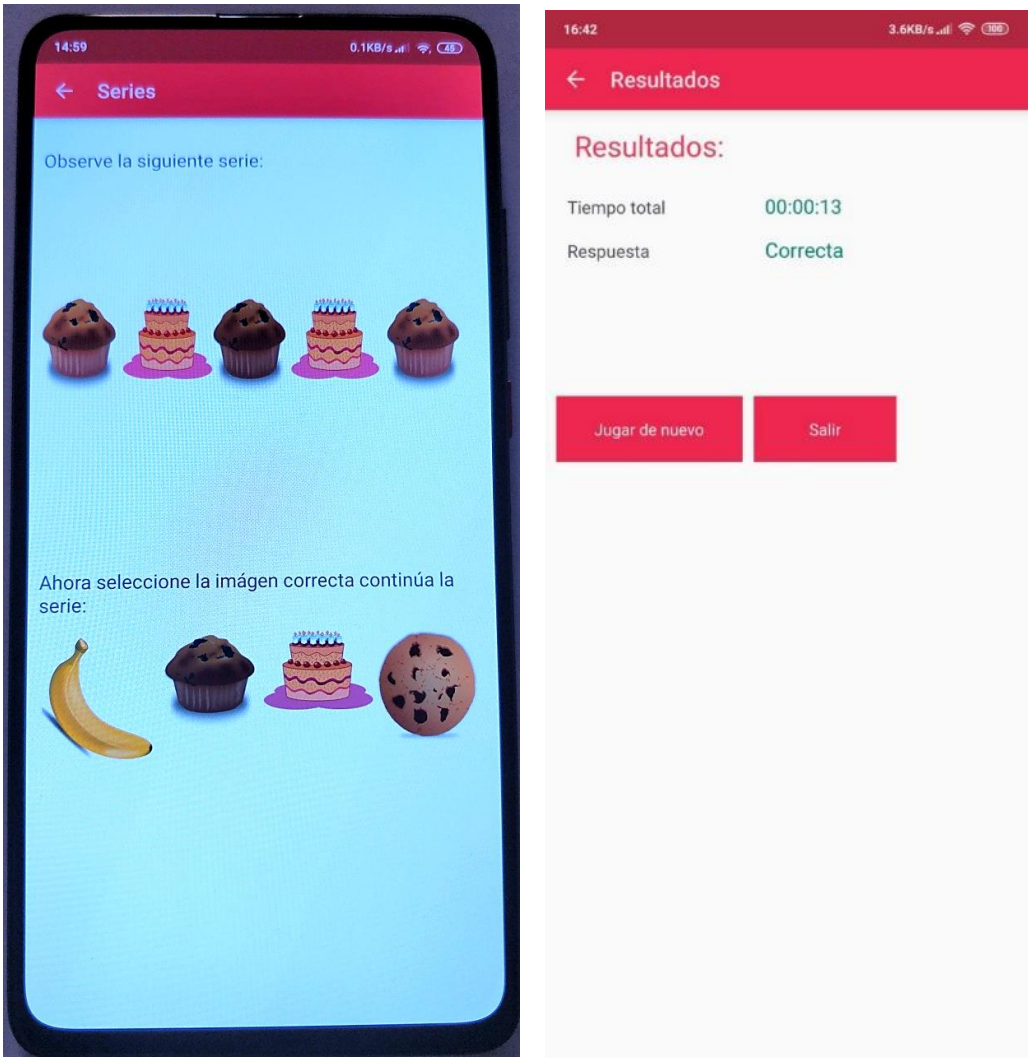

Figura 90 - Razonamiento Completa la Serie

6.4 Resultados

Esta sección está pensada para ayudar a voluntario a ver con cierto detalle en qué habilidades destaca el tutelado y también en cuales flaquea. Esto le permitirá al voluntario ofrecer un mejor programa de juegos al tutelado para ejercitar aquellas facultades donde más necesita mejorar el tutelado. Para ello, esta funcionalidad de la aplicación ofrece un resumen de los resultados de las partidas jugadas por el tutelado, proporcionando así una visión general de las estadísticas de cada uno de los juegos. Para ello se ha hecho uso de gráficas y cálculos de varias estadísticas de las partidas jugadas.

Al acceder a esta sección desde el menú lateral, se muestra la lista de los tutelados asignados al voluntario que está utilizando la aplicación. Al seleccionar un tutelado de la lista, se accede a la pantalla de resultados de las partidas jugadas. En esta pantalla también se muestra la lista de juegos disponibles que, al seleccionar uno de ellos , se mostrarán más detalles del juego individualmente. En la gráfica se muestra el porcentaje de fallos y aciertos del juego seleccionado y debajo de la gráfica se muestran más detalles de otras estadísticas del juego.

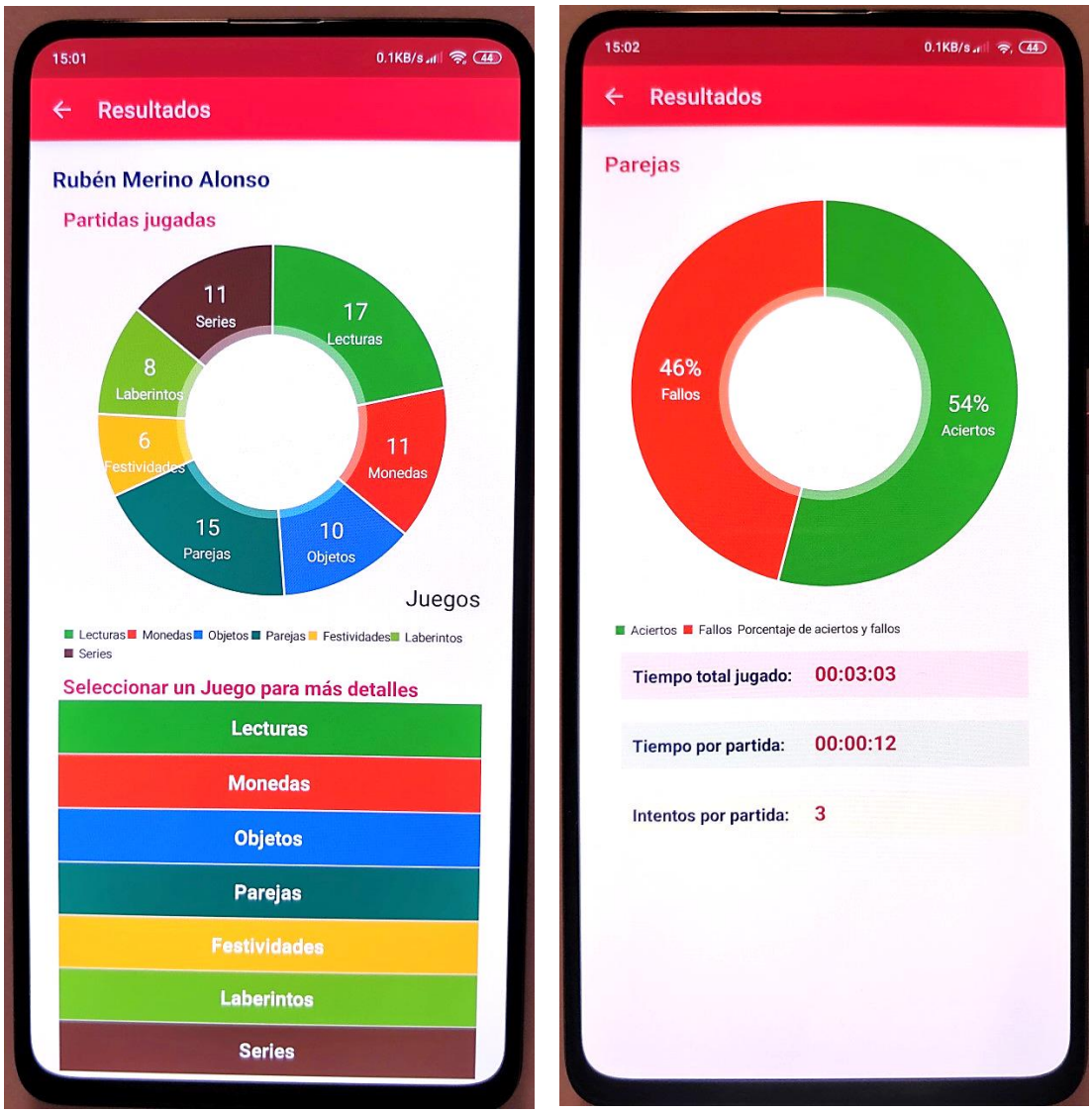

Figura 91 - Resultados de partidas jugadas y estadísticas de cada juego

Capítulo 7 – Conclusiones y Perspectivas Futuras

Capítulo 7 – Conclusiones y Perspectivas Futuras

7.1 Presupuesto económico

Para llevar a cabo el desarrollo de la aplicación y la realización del proyecto, se ha tenido que hacer uso de ciertos recursos como son los programas de software y soporte informático. Por suerte, dichos recursos ya se encontraban disponibles previamente y además todos los recursos gráficos y de texto usado en la app son de coste cero al ser de libre distribución, lo cual ha supuesto una reducción importante. Teniendo en cuenta lo anterior, hay que analizar otros factores que intervienen en el desarrollo de la aplicación en sí.

A continuación, se analizan los factores más que influyen en el precio de una aplicación (Porras, 2018):

- Coste por hora del programador

  Es uno de los factores más importantes y puede variar desde unos 30€ por hora (Autónomo/freelance con poca experiencia) hasta alrededor de 120 € por hora (consultoras y agencias especializadas con experiencia).

- Acceso a datos de la App

  Dependerá de si la aplicación necesita conectarse a servidores para realizar búsquedas, actualizar su información en tiempo real y mostrar los resultados, etc.

- Geo-posicionamiento para la aplicación

  Otro factor a tener en cuenta es si los usuarios de la aplicación necesitan información dependiendo de su localización. Por ejemplo, se podría mostrar información de la tienda más cercana.

- Realidad aumentada para la App

  Por ejemplo, en casos en los que se quiere que los productos que se ofrecen a través de la aplicación se muestren en una imagen en 3D en un catálogo digital

- Complejidad de la aplicación

Dependiendo del uso que se quiera hacer de la aplicación, existen aplicaciones sencillas que se desarrollan en pocas horas y aplicaciones complejas como por ejemplo juegos o redes sociales, que requieren de miles de horas de trabajo.

- El gestor de contenidos de la App

Depende de si los contenidos son estáticos o se pueden actualizar de forma dinámica. El gestor de contenidos o CMS es un factor determinante, y en el caso en que estos sean dinámicos tendrías dos opciones:

  - Usar un gestor de contenidos con el que se tenga que integrar la aplicación.
  - Crear un gestor a medida para modificar los contenidos de la aplicación.

- Pasarela de pago para la App

La aplicación en muchos casos puede requerir de una pasarela pago para vender productos a través de la App. Esto dependerá de las diversas opciones de pago que ofrezca la aplicación, si necesita integrar el pago a través de las tiendas App u otras pasarelas de pago, etc.

- Registro de usuarios en la aplicación

Si la aplicación requiere incluir un registro de usuarios, ésta se encarecerá. Ello dependerá de la base de datos que se lleve a cabo en cuanto a tamaño y tipo de datos que acumule.

- Envío de notificaciones *push*

Es interesante incluir la posibilidad del envío de mensajes personalizados a los usuarios a través de la aplicación que han descargado. Si este es el caso de la aplicación que quieres crear, tendrás que tener en cuenta el tipo de notificaciones, si es necesario crear una interfaz de gestión de notificaciones o si se debe integrar con un sistema existente, etc. Esta mejora en el diseño de la aplicación conlleva un sobrecoste en el desarrollo de la misma que deberás de prever dentro de tu presupuesto.

- Diseño gráfico para la aplicación

Lógicamente no es lo mismo un diseño sencillo con menús y pestañas a modo de ficha informativa basada en una plantilla, que aplicaciones que incluyan opciones de interacción para los usuarios más avanzadas y complejas.

- El número de plataformas dónde será accesible

  Las plataformas en las que se vaya a implementar la aplicación y el número de ellas en las que deba funcionar son determinantes para calcular el coste. Por lo general, lo normal es crear tu App para Android, iOS y Windows Phone, que son los sistemas operativos para dispositivos móviles predominantes actualmente.

- Integración con otros sistemas

  El desarrollo puede complicarse sensiblemente en el caso de tener que integrar la aplicación con sistemas existentes en la empresa (bases de datos, gestión de usuarios, gestor de contenidos, envío de notificaciones, etc.). Generalmente, una implementación de sistemas para una aplicación suele ser un proceso muy técnico y costoso que suele suponer un coste importante en el desarrollo de la App.

- Tipo de aplicación

  Las aplicaciones nativas están desarrolladas para una plataforma o sistema operativo concreto. Por este motivo, este tipo de aplicaciones ofrecen mayor calidad, prestaciones y rendimiento, pero claro está, a un mayor coste. Este rango es una aproximación sobre el precio que podría tener por 300h de desarrollo por parte del programador. Este tipo de aplicaciones requieren de una mayor implicación en su diseño y desarrollo, ya que necesitan de una base y diseños sólidos que sean ampliables con vistas a un futuro.

  Las aplicaciones híbridas están diseñadas y desarrolladas para ser compatibles con distintas plataformas y sistemas operativos para dispositivos móviles. el coste es notablemente inferior y, Por lo general, el precio para el desarrollo de una aplicación híbrida, suele oscilar entre un 65% y un 80% menos del coste de desarrollo de una aplicación nativa para Android o iOS.

Una vez vistos los factores más importantes a la hora de desarrollar la aplicación y teniendo en cuanta el coste de mano de obra y las características de la aplicación, se puede estimar que el coste total de la aplicación es de **7000 €**, ya que se han estimado **350 horas** de trabajo real de programación. Se ha tomado como referencia el salario por hora de un programador junior sin experiencia (20 €/hora).

7.2 Conclusiones Finales

En la actualidad, las TIC están presentes en una gran variedad de ámbitos como son la educación, la política, la seguridad o la sanidad. Es en este último donde la aplicación de las TIC ha supuesto un gran avance en la forma en la que el paciente y el profesional de la salud interactúan y en el tratamiento de la información del paciente y de cualquier aspecto relacionado con la salud en general.

Los avances de las nuevas tecnologías y su aplicación en el ámbito de la salud, ha dado lugar a lo que se conoce como *eHealth*, y dentro de este sector se encuentra otro sector conocido como *mHealth*, que, gracias a su versatilidad y flexibilidad, ha supuesto un cambio radical en la forma en la que interactúan los pacientes y los profesionales de la salud. Todo ello gracias a la conjunción de internet y movilidad se aplica al ámbito de la salud. El mayor representante del sector *mHealth* son las apps para dispositivos móviles, pues un dispositivo móvil es un medio de comunicación que es accesible a millones de personas y supone una flexibilidad y movilidad que ningún otro medio tecnológico es capaz de ofrecer. Por lo tanto, las *apps* para dispositivos móviles se podrían considerar uno de los mejores representantes de la aplicación de las TIC al ámbito de la salud.

Es pues, dentro de este contexto, donde se desarrolla el presente TFG, con el desarrollo de una aplicación móvil para Android para realización de juegos de rehabilitación cognitiva y de las actividades básicas de la vida diaria. En este caso, en el papel del paciente se encuentra el **tutelado** y en el papel del asistente socio-sanitario se encuentra el **voluntario,** cuyo cometido es el de asistir al tutelado en el uso de la aplicación para la realización de las actividades o juegos dirigidos a ayudar al tutelado con su rehabilitación cognitiva y las actividades básicas de vida diaria. Además, otro de los cometidos del voluntario es el de realizar un seguimiento del estado del tutelado y para ello hará uso de la app que tiene el objetivo de facilitar dicha tarea.

Una vez establecidos los objetivos del TFG y realizado las investigaciones necesarias para la adquisición del conocimiento necesario para la realización del TFG y el desarrollo de la aplicación Android. Dichos conocimientos se han tenido que adquirir desde cero, lo cual supuso ciertos desafíos, aunque el conocimiento adquirido ha sido valioso y de gran utilidad de cara al futuro. El desarrollo de aplicaciones para dispositivos móviles es uno de los sectores de mayor crecimiento en la actualidad, por lo que el desarrollo de este TFG supone un importante valor añadido al ser dirigido específicamente a este campo tan cotizado en el mercado laboral. En conclusión, la experiencia y conocimientos adquiridos, son de gran valor e importancia para el desarrollo, tanto profesional como personal.

En cuanto a los detalles del desarrollo de la aplicación en sí, se ha procurado utilizar las herramientas oficiales de desarrollo para Android y se ha seguido, en la medida de lo posible, las líneas de diseño y desarrollo marcadas por Google para

desarrolladores. Esto es importante de cara a la ampliación de las funciones de la app en futuras versiones, facilitando así el proceso de mejora y diagnóstico de problemas para futuros desarrolladores que trabajen en la mejora de la app.

Una vez realizadas las pruebas para comprobar que la aplicación cumplía con los requisitos iniciales establecidos, se elaboró un manual de usuario cuyo objetivo es el de ser una guía de referencia para todo aquel usuario que pretenda usarla, mostrando de forma sencilla el funcionamiento de la misma.

7.3 Perspectivas Futuras de Ampliación

En esta primera versión de la aplicación se ha procurado cubrir las funciones básicas necesarios para llevar a cabo los objetivos del TFG. Dichas funcionalidades están enfocadas para los usuarios de la aplicación, es decir, los tutelados y los voluntarios. Como es lógico, siempre habrá espacio para nuevas funcionalidades y mejoras de rendimiento y diseño. Puesto que este es un sector en constante evolución, es de prever que habrá avances y nuevas funcionalidades, tanto a nivel de funcionamiento, rendimiento y diseño visual. Todas estas características tienen la posibilidad de ser incorporadas a la aplicación y así mantenerla actualizada en el futuro.

A continuación, se describen algunas posibles funcionalidades y mejoras, ya sea como nuevos añadidos o mejoras a las características actuales:

- Añadir nuevos juegos y actividades para disponer de una mayor variedad que se adapte a los distintos tutelados y sus necesidades.
- Juegos y actividades personalizadas exclusivamente para cada tutelado y posibilidad de añadir dichos juegos de manera dinámica sin necesidad de actualizar la aplicación. Para ellos, se haría uso de plantillas, que el administrador o medico podrá usar para diseñar o personalizar el juego, se lo asignará al paciente en la plataforma web y después estas plantillas se descargarán en la aplicación para cada paciente.
- Añadir el concepto de gamificación como incentivo para los tutelados a la hora de hacer las actividades que se les asignen, promoviendo un ambiente de competitividad y de recompensas. Lo cual, dará a los juegos un toque más entretenimiento y no simplemente como algo de obligada realización.

- Añadir un apartado para los voluntarios a modo de red social que facilite la interacción entre ellos, ofreciéndoles la posibilidad de agilizar sus tareas de voluntariado y hacer su trabajo sea más colaborativo y eficiente.
- Mejorar el aspecto gráfico de la aplicación y actualizarla para incorporar el nuevo *material design 2.0.*
- Posibilidad de registrar a un nuevo Voluntario desde la propia aplicación, el registro se llevaría a cabo mediante un formulario. La aceptación y verificación

de los datos del voluntario y la asignación de tutelados se llevará a cabo por el administrador por temas de seguridad.

- Posibilidad de cambiar o recuperar la contraseña desde la aplicación.
- Funciones especiales de administrador desde la app, para aceptar nuevos voluntarios, borrar usuarios o modificar seguimientos realizados por cualquiera de los voluntarios, similar a los que pueda realizar desde la plataforma web.
- Mejorar las funcionalidades offline de la app para que sea completamente funcional en caso de que no se disponga de conexión y una mejor sincronización de los datos entre servidor web y aplicación.

Bibliografía y referencias

Bibliografía y referencias

ABC Color (2013), *¿Qué son los Smart TV?.* http://www.abc.com.py/edicion-impresa/suplementos/escolar/que-son-los-smart-tv-611020.html . Último acceso: septiembre 2019.

Academiaandroid (2014). *IDE: Entornos Integrados de Desarrollo para Android.* https://academiaandroid.com/ide-entornos-integrados-de-desarrollo-para-android/ . Último acceso: septiembre 2019

Android (2019). *Android developers: Android History*. https://www.android.com/intl/es_es/history/. Último acceso: septiembre 2019.

Android Ayuda (2017). *El plan de Google para acabar con la fragmentación de Android*. https://androidayuda.com/2017/12/02/project-treble-fragmentacion-android . Último acceso: septiembre 2019.

Android Developers (2019a). *Distribution dashboard*. https://developer.android.com/about/dashboards/ . Último acceso: septiembre 2019.

Android Developers (2019b). *Platform Versions.* http://developer.android.com/resources/dashboard/platformversions. Html . Último acceso: septiembre 2019.

Android Developers (2019c). *Configure your build*. https://developer.android.com/studio/build/. Último acceso: septiembre 2019.

Android Developers (2019d). *Publish your App*. https://developer.android.com/studio/publish/. Último acceso: septiembre 2019.

Android Developers (2019e). *Actividades.* https://developer.android.com/guide/components/activities?hl=es-419. Último acceso: septiembre 2019.

Android Developers (2019f). *Android Studio.* https://developer.android.com/studio/intro/. Último acceso: septiembre 2019.

Android Developers (2019g). *Arquitectura de la plataforma.* https://developer.android.com/guide/platform/?hl=es-419. Último acceso: septiembre 2019.

Android Developers (2019h). Material Design for Android. https://developer.android.com/guide/topics/ui/look-and-feel. Último acceso: septiembre 2019.

Android Developers (2019i). *Introducción a Android Studio.* https://developer.android.com/studio/intro/. Último acceso: septiembre 2019.

Apple (2019). *El nuevo iPadOS crea experiencias únicas diseñadas para el iPad.* https://www.apple.com/es/newsroom/2019/06/the-new-ipados-powers-unique-experiences-designed-for-ipad/ . Último acceso: noviembre 2019.

Ararteko (2007). *Atención Sociosanitaria: Una Aproximación al Marco Conceptual y a Los Avances Internacionales y Autonómicos.* www.ararteko.net/RecursosWeb/DOCUMENTOS/1/2_1357_3.pdf . Último acceso: agosto 2019.

Areatecnologia (2015), *Sistemas Operativos Móviles.* http://www.areatecnologia.com/informatica/sistemas-operativos-moviles.html . Último acceso: septiembre 2019.

Arquitecturajava (2016), *Qué es REST.* https://www.arquitecturajava.com/que-es-rest/. Último acceso: septiembre 2019.

Babel (2016), *¿Qué futuro se presenta con Windows 10 Mobile?.* https://babel.es/es/blog/blog/mayo-2016/%C2%BFque-futuro-se-presenta-con-windows-10-mobile . Último acceso: septiembre 2019.

Cabo Salvador, J. (2015). Sanidad del Futuro: e-Health, m-Health, e Inteligencia Ambiental. https://www.udima.es/es/sanidad-futuro-Health-inteligencia-ambiental. Último acceso: noviembre 2019.

CPDC (2008). Cuaderno de ejercicios de estimulación cognitiva. http://archivosweb.esteve.com/pacientes/deterioro-cognitivo/cuaderno1-de-estimulacion-cognitiva.pdf . Último acceso: noviembre 2019.

Cuello, J. y Vittone J. (2013). *Diseñando apps para móviles* (1ª Ed.). www.appdesignbook.com. Último acceso: noviembre 2019

Dispositivos Wereables (2014), *¿Que es Wearable? – Los dispositivos vestibles.* http://www.dispositivoswearables.net/. Último acceso: agosto 2019.

Ditrendia (2018), *Informe ditrendia: Mobile en España y en el Mundo 2018.* http://alturl.com/9p2pr . Último acceso: Octubre 2019.

Domínguez Chávez, J. (2016). *Curso de Programación Android con Java*. https://www.researchgate.net/publication/302112967_Curso_de_Programacion_Android_con_Java . Último acceso: agosto 2019.

El Androide Libre (2013). *Razones para elegir una Tablet frente a un Portátil*. https://elandroidelibre.elespanol.com/2013/03/10-razones-por-las-que-elegir-una-tablet-antes-que-un-pc-o-portatil.html. Último acceso: agosto 2019.

Evelyn Porras Z. (2012a). *Ingeniería de sistemas. Sistemas operativos móviles: Android*. http://eve-ingsistemas-u.blogspot.com/2012/04/sistemas-operativos-moviles-android.html. Último acceso: agosto 2019.

Evelyn Porras Z. (2012b). *Ingeniería de sistemas. Sistemas operativos móviles: iOS*. http://eve-ingsistemas-u.blogspot.com/2012/04/sistemas-operativos-moviles-ios.html. Último acceso: agosto 2019.

Fundamay (2019). *Programa de Voluntariado.* https://fundamay.org/voluntariado/. Último acceso: noviembre 2019.

Iconos (2019). *Conceptos básicos: definición de web app y ejemplos.* https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-una-web-app-y-que-clases-hay/. Último acceso: noviembre 2019.

InnovaAge (2018). *Apps Hibridas vs Nativas vs Generadas.* https://www.innovaportal.com/innovaportal/v/696/1/innova.front/apps-hibridas-vs-nativas-vs-generadas-que-decision-tomar. Último acceso: octubre 2019.

Intellipaat (2019). *iOS Architecture.* https://intellipaat.com/blog/tutorial/ios-tutorial/ios-architecture/. Último acceso: diciembre 2019.

Jadad, A. y Lorca, J. (2009). *Telemedicina asíncrona: ¿Una amenaza o la salvación del sistema sanitario en la era de la e-salud?. Revista eSalud*. Último acceso: agosto 2018.

Jahoda, P. (2014). MPAndroidChart. https://github.com/PhilJay/MPAndroidChart . Último acceso: noviembre 2019.

Ecma (2017). The JSON Data Interchange Syntax. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf Último acceso: noviembre 2019.

Larescv (2018). Tutela y curatela de una persona mayor dependiente, ¿cuándo se debe dar el paso de la incapacitación?. https://www.larescvalenciana.org/proteccion-de-las-personas-mayores-tutelas-e-incapacidad/. Último acceso: diciembre 2019.

Lorenzo Otero, J. y Fontán Scheitler, L., (2001). *La rehabilitación de los trastornos cognitivos*. Revista Med Uruguay, http://www.rmu.org.uy/revista/2001v2/art8 . Ultimo Acceso: agosto 2019.

Medium (2018). *UWP Apps and How UWP Applications Development Is Different.* https://medium.com/existek/uwp-apps-and-how-uwp-applications-development-is-different-e8934ebf2f6a . Ultimo Acceso: diciembre 2019.

Morillo Pozo, J. D. (2012). *Introducción a los dispositivos móviles*. https://www.exabyteinformatica.com/uoc/Informatica/Tecnologia_y_desarrollo_en_dispositivos_moviles/Tecnologia_y_desarrollo_en_dispositivos_moviles_(Modulo_2).pdf . Ultimo Acceso: noviembre 2019.

Microsoft (2019a). *Finalización del soporte de Windows 10 Mobile*. https://support.microsoft.com/es-es/help/4485197/windows-10-mobile-end-of-support-faq . Ultimo Acceso: diciembre 2019.

Microsoft (2019b). *Desarrollar aplicaciones para UWP*. https://docs.microsoft.com/es-es/visualstudio/cross-platform/develop-apps-for-the-universal-windows-platform-uwp?view=vs-2019 . Ultimo Acceso: diciembre 2019.

Microsoft (2019c). *¿Qué es una aplicación para UWP?* https://docs.microsoft.com/es-es/windows/uwp/get-started/universal-application-platform-guide . Ultimo Acceso: diciembre 2019.

Porras, M. (2018). *Cuánto cuesta una App móvil y cómo desarrollarla.* https://aulacm.com/precio-desarrollar-app-aplicacion-movil/. Último Acceso: octubre 2019.

Quees (2015). *Wearable - Explicación y definición de wearable*. https://www.quees.info/que-es-wearable.html. Último acceso: agosto 2019.

Richter, F. (2019). *The Smartphone Duopoly.* https://www.statista.com/chart/3268/smartphone-os-market-share/. Último acceso: Octubre 2019.

Openwebinars (2018). *¿Qué es REST? Conoce su potencia.* https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/. Último Acceso: noviembre 2019.

Sanz, D., Saucedo, M. y Torralbo, P. (2012). *Introducción a Android* http://www.it-docs.net/ddata/18.pdf. Último acceso: Octubre 2019.

Schäfers, M. (2015). AdvancedMaterialDrawer. https://github.com/madcyph3r/AdvancedMaterialDrawer . Último acceso: noviembre 2019.

Statista (2019). *Number of apps available in leading app stores 2019*. https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores . Último acceso: agosto 2019.

Swiftlatino (2017). *iOS*. https://swiftlatino.com/ios/ . Último acceso: noviembre 2019.

Tablet Area (2010). *¿Qué es un Tablet?.* http://www.tabletarea.com/caracteristicas.html. Último acceso: agosto 2019.

The PHP Group (2019). *¿Qué es PHP?.* https://www.php.net/manual/es/intro-whatis.php . Último acceso: agosto 2019.

Tomás Gironés J. (2013). *El Gran Libro de Android* (2ª Ed.). Barcelona, España: MARCOMBO, S A. pp. 21

Universidad Carlos III (2015). *Software De Comunicaciones: Android*. https://sites.google.com/site/swcuc3m/home/android . Último acceso: agosto 2019.

Universidad Politécnica de Valencia (2018a), *Las versiones de Android y niveles de API*. http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo/146-las-versiones-de-android-y-niveles-de-api. Último acceso: agosto 2019.

Universidad Politécnica de Valencia (2018b), C*omponentes de una aplicación*. http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo/149-componentes-de-una-aplicacion. Último acceso: agosto 2019.

Vicente Fuentes, F. (2013). *La Atención Sociosanitaria*. http://www.riicotec.org/InterPresent2/groups/imserso/documents/binario/07atencionsociosanitariafvicen.pdf. Último acceso: agosto 2019.

WHO (2019). eHealth at WHO*.* https://www.who.int/ehealth/about/en/. Último acceso: agosto 2019.

WHO (2011). mHealth: New horizons for health through mobile technologies*.* https://www.who.int/goe/publications/goe_mhealth_web.pdf. Último acceso: noviembre 2019.

WHO (2015). World report on ageing and health*.* https://www.who.int/ageing/publications/world-report-2015/es/. Último acceso: noviembre 2019.