Universidad de Valladolid

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN
MENCIÓN EN TELEMÁTICA

APLICACIÓN ANDROID PARA EL ENTRENAMIENTO COGNITIVO DE PERSONAS BAJO TUTELA JURÍDICA CON DISCAPACIDAD INTELECTUAL O DEL DESARROLLO

AUTOR:

D. RAÚL VELASCO CAMINERO

TUTORA:

DRA. DÑA. MÍRIAM ANTÓN RODRÍGUEZ

VALLADOLID, ENERO de 2017

TÍTULO: **Aplicación Android para el entrenamiento cognitivo de personas bajo tutela jurídica con discapacidad intelectual o del desarrollo**

AUTOR: **D. Raúl Velasco Caminero**

TUTORA: **Dra. Dña. Míriam Antón Rodríguez**

DEPARTAMENTO: **Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE: **Míriam Antón Rodríguez**

VOCAL: **Mario Martínez Zarzuela**

SECRETARIO: **David González Ortega**

SUPLENTE: **Francisco Javier Díaz Pernas**

SUPLENTE: **Mª Ángeles Pérez Juárez**

FECHA:

CALIFICACIÓN:

Resumen

El objetivo principal de este proyecto consiste en el desarrollo de una aplicación para la plataforma *Android* con el fin de tratar de mejorar las capacidades cognitivas de los tutelados a través de juegos desarrollados específicamente para tal fin. Aunque esta es la funcionalidad principal que se ofrece, la aplicación va más allá. Permite llevar un seguimiento de cada tutelado y crear y ver los detalles de los mismos. También permite ver estadísticas de los tutelados en los distintos juegos para ver su desarrollo y mejora. La aplicación es un sistema de seguimiento y rehabilitación donde los voluntarios introducen sus credenciales para poder acceder a todas las funcionalidades citadas anteriormente que ofrece la aplicación.

El sistema está desarrollado en *Java* utilizando el *IDE Android Studio* y puede utilizarse en cualquier dispositivo que disponga de *Android 4.0* o superior como sistema operativo.

Palabras clave: *Android, Android Studio, Estimulación Cognitiva, Tutelado, Discapacidad Intelectual, m-Health*

Abstract

The main aim of this project is to develop an *Android* application which tries to improve the cognitive abilities of the wards through enjoyable games that has been developed for that goal. Even though that is the main use that is offered, this application goes further. It allows to carry on the monitoring of each ward and create and view their details. It also allows to view the ward´s statistics in the different games to see how they are improving and getting better. This application is a monitoring and rehabilitation system where the volunteers insert their credentials in order to access all the functionalities previously mentioned, that the application offers.

The system is developed in *Java* using the *IDE Android Studio* and it can be used in any device that is provided with *Android 4.0* or higher as its operating system.

Keywords: *Android, Android Studio, Cognitive Stimulation, Ward, Intellectual Disabilities, m-Health*

A mis padres y hermana, por todo su apoyo, tanto en los buenos como en los malos momentos, y por hacer posible que haya llegado hasta aquí.

A mi tutora Miriam, por darme esta oportunidad y por la ayuda recibida sin importar el momento.

A todos los que me han apoyado y ayudado en la realización de este proyecto.

Contenido

ÍNDICE DE ILUSTRACIONES

CAPÍTULO 1

1 INTRODUCCIÓN

Desde hace unos años atrás hasta nuestros días, el incremento de la esperanza de vida en nuestra sociedad es un hecho evidente. Este aumento en la esperanza de vida de la población trae consigo un crecimiento acelerado del envejecimiento de la población. Esto es debido a la mejora en la calidad de vida en las personas y a los avances, tanto médicos como tecnológicos, en la medicina a lo largo de estas últimas décadas. Este envejecimiento de la población trae consigo la pérdida de las capacidades cognitivas, específicamente en memoria, atención y velocidad de procesamiento y las capacidades funcionales como la pérdida de movilidad del individuo. (Maroto)

De acuerdo con diversos estudios científicos, a través del entrenamiento cognitivo, se pueden observar ciertos efectos positivos que compensan el deterioro cognitivo con programas de entrenamiento ya sean generales o específicos para funciones como la memoria, el razonamiento y, en definitiva, todas las funciones cognitivas que se deterioran con el paso de los años. La estimulación cognitiva puede ser aplicada a cualquier persona, no solo a personas de avanzada edad que evidencian algún tipo de pérdida de capacidades cognitivas, sino también a personas que quieran mejorar sus capacidades para ser más hábiles. El entrenamiento cognitivo también es de vital importancia en personas que sufren algún tipo de enfermedad degenerativa como el Alzheimer o en personas que han sufrido algún tipo de enfermedad que disminuya sus capacidades cognitivas como un derrame cerebral. (Prieto, 2008)

Por otro lado, los avances tecnológicos a día de hoy son más que evidentes. Uno de los avances más significativos en la vida cotidiana de las personas y que más uso diario se hace de ello es el surgimiento de los *smartphones* o teléfonos inteligentes y las *tablets*. Estos dispositivos ofrecen las funcionalidades de un teléfono móvil convencional al que se le suman las funcionalidades de una microcomputadora.

Hoy en día el uso de estos dispositivos es algo rutinario y habitual y esto, unido a las funcionalidades que ofrecen, hace plantearse el hecho de beneficiarse de ello creando aplicaciones útiles que mejoren de forma significativa la calidad de vida de las personas más necesitadas, así como la tarea que desempeñan las personas encargadas de su cuidado.

Las ventajas de estos dispositivos móviles para implementar sistemas de seguimiento y estimulación cognitiva son infinitas. Su conectividad permitiría a especialistas y terapeutas hacer un seguimiento de los pacientes y, a su vez, consultar la situación de ellos siempre que dispongan de uno de los dispositivos citados anteriormente con la aplicación correctamente instalada.

Por lo expuesto anteriormente, este proyecto combina el uso de las nuevas tecnologías junto con los análisis y ensayos médicos proporcionando una herramienta muy útil tanto para la rehabilitación cognitiva y la prevención del desarrollo de enfermedades neurodegenerativas del paciente, así como un mayor control y seguimiento por parte de la persona encargada del seguimiento y cuidado de los pacientes. (Ball & Lillis, 2001)

1.1 OBJETIVOS

El objetivo principal de este proyecto es básicamente el diseño y desarrollo de aplicación *Android* para la rehabilitación o estimulación cognitiva de las personas que necesiten este tipo de ayuda. A su vez, la aplicación también permite a la persona encargada del cuidado de los pacientes un seguimiento de los mismos, así como el progreso que realizan mediante la visualización en gráficas de distintos parámetros de interés. Este objetivo general conlleva una serie de objetivos secundarios que se detallan a continuación:

- Búsqueda, diseño y desarrollo de juegos o ejercicios de diferentes categorías (percepción, memoria…) similares a los que pueda tener cualquier programa de rehabilitación cognitiva.
- Diseño y desarrollo de la interfaz de usuario para que sea intuitiva y fácil de usar por personas mayores que no estén acostumbradas al uso de este tipo de tecnologías, así como dotar a la aplicación de todas las funcionales que se necesiten.
- Desarrollo de la lógica de la aplicación para ofrecer todas las funcionales necesarias teniendo en cuenta el posible crecimiento y expansión de la misma.
- Diseño y desarrollo de un modelo de almacenamiento de los datos necesarios para el correcto funcionamiento de la aplicación.
- Extraer conclusiones acerca de la aplicación desarrollada.

1.2 FASES Y MÉTODOS

Para lograr los objetivos anteriormente descritos y la aplicación como producto final hay que seguir las siguientes fases:

- Fase de documentación: Estudio de los principales conceptos de *Android* como su entorno de desarrollo, *Android Studio*, y el lenguaje de programación en el que se va a desarrollar la aplicación, *Java*. Estudio de la comunicación entre la aplicación y el servidor, donde se van a almacenar todos los datos necesarios para la aplicación. Estudio del formato en los que la aplicación obtiene los estos datos (JSON, XML…) y, en definitiva, todos los aspectos que se necesitan para el desarrollo de la aplicación. Por otro lado, también es importante el estudio de los juegos más adecuados para cada tipo de categoría que se va a implementar en la aplicación.
- Fase de razonamiento: En base a los ejemplos que tenemos de ejercicios de estimulación cognitiva y los conocimientos adquiridos en la fase previa, se buscan y prueban diversas formas de implementar las distintas funcionalidades en *Android*.
- Búsqueda, selección y adecuación de las imágenes que se utilizan en la aplicación. Se utilizan imágenes libres de derechos de autor que su tamaño no sea excesivo para que la aplicación sea ligera y funcione correctamente.
- Implementación en el lenguaje de programación *Java* del sistema diseñado en las etapas anteriores.
- Implementación de la interfaz de usuario. Esta interfaz tiene que ser intuitiva y fácil de utilizar puesto que la aplicación está pensada para personas mayores que no tienen experiencia en el manejo de este tipo de dispositivos.
- Prueba y depuración de la aplicación el distintos *smartphones* y *tablets* para comprobar el correcto funcionamiento y visualización de todos los elementos que forman la aplicación.

1.3 MEDIOS

Para la implementación de la aplicación se está consultando asiduamente la página web oficial de *Android*, `https://developer.android.com/`, donde se encuentra la API de *Android* y guías para la correcta implementación de la aplicación. A mayores también se están consultando manuales y libros donde se muestran ejemplos útiles para nuestra aplicación.

Para el desarrollo y la depuración de la aplicación que se está desarrollando se están utilizando los siguientes equipos y componentes *software*:

- *Android Studio 2.1*.
- *GIMP*
- Sony Vaio de 15,4 pulgadas, Windows 10 como sistema operativo, procesador i5 de segunda generación, 4 GB de memoria *RAM DDR3* y disco duro *HDD* de 500 GB.
- Sony Xperia U con sistema operativo *Android 4.0.3*.
- Otros dispositivos *Android* de distintos tamaños de pantalla con *Android 4.0* o superior para comprobar en correcto funcionamiento en todos ellos.

CAPÍTULO 2

2 ESTUDIO DE LAS DISTINTAS TECNOLOGÍAS

El desarrollo de la tecnología móvil ha sufrido un vertiginoso e inimaginable desarrollo en los últimos 30 años. La idea que surgió en combinar las funcionalidades de una PDA (*Personal Digital Assistant*) con las de un teléfono móvil convencional se ha convertido en algo indispensable para la mayoría de las personas hoy en día con la infinidad de usos que estos dispositivos ofrecen.

El primer teléfono móvil en usar el término *smartphone* fue el Ericsson GS88 el cual era más avanzado y poseía funciones de correo electrónico, negación web, reloj mundial, un teclado QWERTY físico, modo avión, puerto infrarrojo, conexión a PC… que se puede observar en la Ilustración 1. (Sager, 2012)

Ilustración 1. *Ericsson GS88*

Es posible que el *boom* de los *Smartphones* empezase con el sistema operativo Windows Pocket PC (2000) y los teléfonos y dispositivos que llegaron al mercado con este sistema operativo como los de la marca HTC, aunque, sin duda, el evento que cambió la percepción de lo que era un *Smartphone* fue el anuncio del iPhone y de iOS en 2007, revolucionando la industria de la telefonía móvil y de los *Smartphones*. Este nuevo sistema operativo dio paso a *Android*, principal competidor de *Apple* y líder del mercado actualmente, lanzado unos meses después del anuncio del *iPhone*. Aunque en la actualidad los 2 sistemas operativos que más se utilizan son *iOS* y *Android* también cabe mencionar que existen otros sistemas operativos para los *Smartphones* como *Windows Phone* o *Blackberry*.

A continuación, en la Ilustración 2, se muestra el uso de los distintos sistemas operativos disponibles en la actualidad para estos dispositivos. Como se puede observar, el sistema más utilizado es *Android*, que cada año aumenta el número de usuarios con respecto a otros como *iOS*. Estos 2 sistemas operativos son los que, actualmente, lideran el mercado.

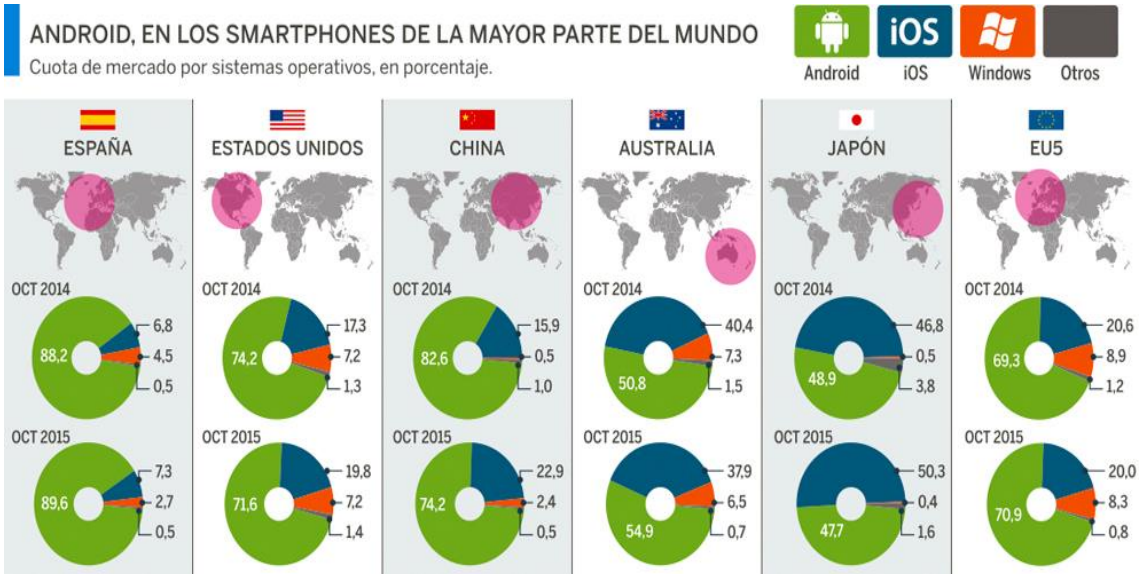

Ilustración 2. *Comparación de los distintos sistemas operativos para Smartphone (Octubre 2015)*

Algo similar ocurre con la evolución y desarrollo de las *tablets*. En 1972, Alan Key desarrolló el concepto de *Dynabook* que consistía básicamente en un ordenador para los niños de todas las edades con el objetivo de acercar a los niños al mundo digital, aunque la tecnología de la época no le daba posibilidad de construir un dispositivo funcional. (Toshiba)

En 1987, Apple Computer presentó un video conceptual acerca del *Knowledge Navigator*, una tableta futurista que respondía ante comandos de voz. (Apple, 1987)

Sin embargo, los primeros dispositivos verdaderos, solo aparecieron a principios del siglo XXI. Microsoft lanzó la *Microsoft Tablet PC* que licenciaba a varios fabricantes las tabletas, aunque, a pesar del relativamente poco éxito que logró, llevó a crear un nicho de mercado en hospitales y negocios móviles. Aunque no fue hasta 2010, cuando la empresa Apple presentó el iPad, basado en su exitoso iPhone, que alcanzó el éxito comercial. Al igual que ocurrió en el mercado de los *Smartphones* con el lanzamiento del Iphone, Apple ha marcado un antes y un después en el mundo de las *tablets*.

En la actualidad prácticamente todos los fabricantes de equipos electrónicos han incursionado en la producción de *tablets* (por ejemplo, Apple, Google, Polaroid, Samsung, Sony, Toshiba, Acer, Hewlett-Packard y Microsoft, por mencionar algunos), lo cual ha generado que el mercado se vea inundado de una inmensa cantidad de *tablets* con diferentes tamaños, aplicaciones, precios y sistemas operativos.

Como se puede observar en la Ilustracón 3, el sistema operativo más utilizado en las *tablets* es *Android*, al igual que ocurría en los *Smartphones*. Ambos sistemas operativos copan la mayor parte de mercado en cuanto a estos dispositivos se refiere.

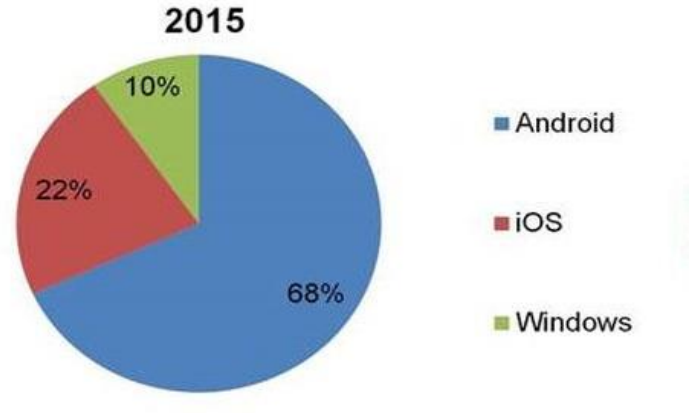

Ilustración 3. *Comparación de los distintos sistemas operativos para Tablet*

En cuanto al sistema operativo, cabe mencionar que Apple tiene una plataforma propia, cuya filosofía se centra en un ecosistema robusto y cerrado, mientras que *Android* es un sistema operativo muy extendido entre los usuarios, cómodo, versátil y práctico al que muchos están acostumbrados.

A lo largo de este capítulo se van a analizar los principales sistemas operativos que utilizan tanto los *Smartphones* como las *tablets*.

2.1 ANDROID

Android es una plataforma libre para el desarrollo de aplicaciones móviles basada en el núcleo operativo Linux. Inicialmente fue desarrollada por *Android, Inc.*, a la cuál *Google* respaldó económicamente y más tarde compró en el 2005. En 2007 se fundó la Open Handset Alliance (OHA), liderada por *Google* con otros 34 miembros entre los que se incluían fabricantes de dispositivos móviles, desarrolladores de aplicaciones, algunos operadores de comunicaciones y fabricantes de chips. Al mismo tiempo que se anunciaba la formación de esta fundación, la OHA presentó *Android*. Algunos de sus miembros son *Google, HTC, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile* y *Nvidia* entre otros. (UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA, 2016)

En la actualidad, *Android* ha evolucionado hasta convertirse en la plataforma líder frente a otras como *iOS, Windows Phone, BlackBerry, Palm, Java Mobile Edition* o *Linux Mobile*. La última versión lanzada al mercado es Nougat 7.0.

Android es una pila de software para dispositivos móviles que incluye un sistema operativo, middleware y diversas aplicaciones. El Android SDK aporta las herramientas y APIs necesarias para empezar a desarrollar aplicaciones para la plataforma Android usando el lenguaje de programación Java. En la Ilustración 4 se puede observar la arquitectura de 4 capas que forma este sistema operativo. (Lee, 2013)

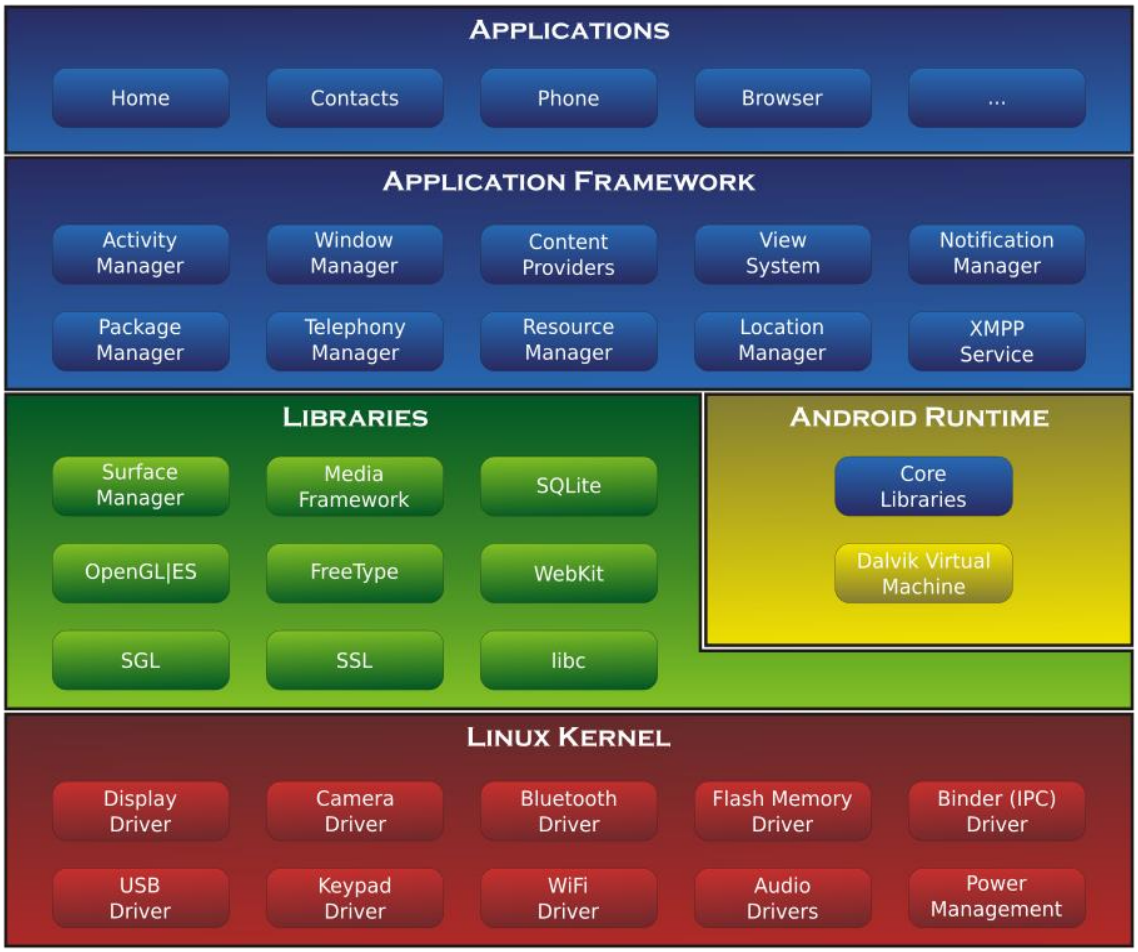

Ilustración 4. *Arquitectura del sistema Android*

2.1.1 ARQUITECTURA DEL SISTEMA *ANDROID*

A continuación, se detallan las características, especificaciones y componentes de cada capa:

- ***Applications:*** Esta capa está formada por el conjunto de aplicaciones instaladas en una máquina *Android*. Normalmente están escritas en *Java* utilizando el Android SDK, pero también existe otra opción consistente que es utilizando *C/C++*. Para esta opción se puede utilizar el Android NDK (*Native Development Kit*). Las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros.
- ***Application FrameWork:*** Los desarrolladores tienen acceso completo a los mismos APIs del *Framework* usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del *Framework*). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario. Los servicios más importantes que incluye son:

- *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.
- *Content Providers:* 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.
- *Activity Manager:* Conjunto de API 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.
- *Telephone Manager:* Incluye todas las API vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).
- *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.
- *XMPP Service:* Colección de API para utilizar este protocolo de intercambio de mensajes basado en XML.

- ***Libraries:*** La siguiente capa se corresponde con las librerías utilizadas por Android. Éstas han sido escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android.
  - *Librería libc:* Incluye todas las cabeceras y funciones según el estándar del lenguaje C. 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…).
  - *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*.
- ***Android Runtime:*** Al mismo nivel que *Libraries* de *Android* se sitúa el entorno de ejecución. Éste lo constituyen las Core Libraries, que son librerías con multitud de clases *Java* y la máquina virtual *Dalvik*.
- ***Linux Kernel:*** *Android* utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde *Android* es crear las librerías de control o drivers necesarios dentro de este kernel de Linux embebido en el propio *Android*.

2.1.2 VENTAJAS Y DESVENTAJAS DE *ANDROID*

A continuación, se van a presentar las mayores virtudes de este sistema operativo, así como sus puntos débiles para tener una perspectiva más global de este sistema operativo. (Santa María, 2016)

Entre las principales ventajas del empleo de *Android* se encuentran:

- El código de Android es abierto: Google liberó Android bajo licencia Apache. Cualquier persona puede realizar una aplicación para *Android* sin necesidad de pagar ninguna cuota anual como ocurre con *iOS*.
- Al igual que sucede con *iOS*, hay una gran comunidad de desarrolladores, por tanto, es sencillo encontrar información para programadores noveles en la red.
- Hoy día hay más de 650.000 aplicaciones disponibles para teléfonos Android, aproximadamente 2/3 son gratis. Además, la libertad de código permite adaptar Android a bastantes otros dispositivos además de teléfonos móviles. Está implantado en Tablets, GPS, relojes, microondas… incluso existe una versión de *Android* para PC.
- El lenguaje de programación empleado es *Java.* Está ampliamente extendido, además en caso de que se desconozca, es sencillo de comprender por personas con conocimientos en programación orientada a objetos.
- La participación de terceros en el desarrollo del sistema operativo conlleva a la aparición y creación de multitud de APIs.
- El sistema Android es capaz de hacer funcionar a la vez varias aplicaciones y además se encarga de gestionarlas, dejarlas en modo suspensión si no se utilizan e incluso cerrarlas si llevan un periodo determinado de inactividad. De esta manera, se evita un consumo excesivo de batería. Esta es una de sus mayores ventajas por la rapidez con la que carga una aplicación abierta previamente.

No todos son halagos para este sistema operativo. Entre sus principales inconvenientes se encuentran:

- *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.

- A pesar de ser una ventaja el ser un sistema multitarea, el hecho de tener varias aplicaciones abiertas hacen que el consumo de la batería aumente y como no todas las aplicaciones *Android* las cierra hay que instalar una aplicación para que las cierre. En la Market de *Android*, el *PlayStore*, hay una buena cantidad de aplicaciones para este fin, así que el problema es solucionable, pero debería venir preinstalado de fábrica.
- 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.
- *Google* no revisa las aplicaciones que se suben a *Google Play* por lo que puede haber bastantes aplicaciones que no tengan mucha funcionalidad para el usuario o que esas aplicaciones no funcionen correctamente.

2.2 iOS

iOS es un sistema operativo móvil propietario de la multinacional *Apple Inc*. *Apple* reveló por primera vez la existencia de *iOS*, por aquel entonces llamado *iPhone OS*, en la *MacWorld Conference* de enero del 2007, aunque el sistema no tuvo nombre oficial hasta marzo del 2008, fecha en la que salió la primera versión beta del *iPhone SDK (Start Development Kit)*. Originalmente fue desarrollado para el *iPhone*, aunque después se ha usado en dispositivos como el *iPod touch*, el *iPad* y el *Apple Tv*. *Apple* no permite la instalación de *iOS* en hardware de terceros. (Apple Inc., 2016)

En junio del 2010 Steve Jobs, CEO de *Apple*, anunció que *iPhone OS* pasaría a llamarse *iOS*. *iOS* está basado en el sistema operativo Mac OS X, que a su vez está basado en *Darwin BSD* y, por lo tanto, es un sistema operativo Unix. En concreto ambos sistemas operativos comparten el mismo núcleo *Mach/FreeBSD*, y utilizan como lenguajes de programación principales C y Objective-C.

Hasta junio del 2014, las aplicaciones nativas para *Mac OS X* e *iOS* se programaban únicamente en *Objective - C*. Se trata de un lenguaje orientado a objetos, muy dinámico y en constante mejora por parte de *Apple*. También cuenta con otras características que lo hacen muy robusto, como es su compatibilidad con *C/C++*.

En *WWDC 2014* (WorldWide Conference Developer), *Apple* presento *Swift*, un nuevo lenguaje de programación para sus aplicaciones. *Swift* es un lenguaje multiparadigma y está diseñado para integrarse con los *frameworks Cocoa* y *Cocoa Touch*. Puede usar cualquier biblioteca programada en *Objective-C* y llamar a funciones de *C*. También es posible desarrollar código en *Swift* compatible con *Objective-C* bajo ciertas condiciones. *Swift* tiene la intención de ser el lenguaje del futuro para *iOS* siendo un lenguaje seguro, de desarrollo rápido y conciso.

La interfaz de usuario de *iOS* está basada en el concepto de manipulación directa, usando gestos multitáctiles (eventos *multi - touch*). Los elementos de control consisten en deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee de una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto

de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo o rotarlo en tres dimensiones.

La arquitectura de *iOS* se divide en cuatro capas de abstracción, tal y como se muestra en la Ilustración 5. La capa *Cocoa Touch* es la de más alto nivel y la de *Core OS* la de más bajo nivel. En general, las dos capas superiores ofrecen servicios y tecnologías más sofisticadas que el resto de capas. (UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA, 2016)

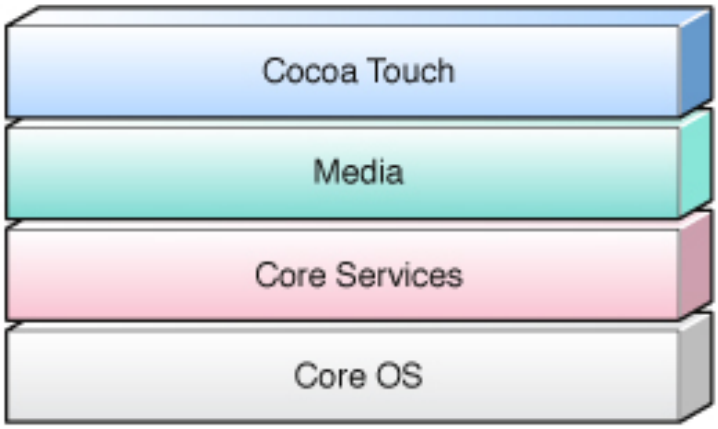

Ilustración 5. *Arquitectura del sistema iOS*

2.2.1 ARQUITECTURA DEL SISTEMA *iOS*

A continuación, se detallan las características, especificaciones y componentes de cada capa:

- ***Cocoa Touch***: Es la capa más importante para el desarrollo de aplicaciones *iOS*. Posee un conjunto de *frameworks* que proporciona el API de Cocoa para desarrollar aplicaciones. Se podría decir que Cocoa Touch proviene de Cocoa, la API ya existente en la plataforma MAC. Esta capa está formada por dos *frameworks* fundamentales:
    - *UIKit*: contiene todas las clases que se necesitan para el desarrollo de una interfaz de usuario.
    - *Foundation Framework*: define las clases básicas, acceso y manejo de objetos,
    - servicios del sistema operativo.

  Cabe destacar la presencia de los siguientes servicios:

    - Eventos *multi-touch*.
    - *Multi-tasking.*
    - Acelerómetro y giroscopio.
    - Geranquía de las vistas.
    - Localización e internalización: *Framework* que permite adoptar diferentes idiomas y regiones sin necesidad de realizar cambios en el código del programa.
    - Soporte para cámara.
- ***Media***: Provee los servicios de gráficos y multimedia a la capa superior. Algunos son:

    - *OpenAl* (*Open Audio Library*).
    - Mezcla de audio y grabación.
    - Reproducción de video.
    - Formatos de archivo de imágenes.
    - *Quartz*: *Framework* para manipular gráficos 2D.
    - *Core Animation*: *Framework* para la visualización de datos.
    - *OpenGL*: *Framework* para la manipulación de gráficos 3D.
- ***Core Services***: Contiene los servicios fundamentales del sistema que usan todas las aplicaciones como pueden ser:
    - *Networking.*
    - Base de datos *SQLite*.
    - *Core Location*: *Framework* que permite un fácil acceso al GPS.
    - Threads.
    - Soporte para XML.
- ***Core OS***: Actúa como núcleo del sistema operativo. Controla el sistema de memoria virtual, los hilos, los ficheros del sistema, la red y gestiona la comunicación con los marcos de la capa *Core Services*. Implementa:
    - *TCP/IP*.
    - *Sockets.*
    - Gestión de la batería.
    - *File System*: Sistema de ficheros.
    - Seguridad.

2.2.2 VENTAJAS Y DESVENTAJAS DE *iOS*

Al igual que se ha mostrado en el sistema *Android*, se van a exponer los puntos fuertes de este sistema y sus debilidades para tener una visión general y completa de este sistema. (Aguayo Sánchez)

En cuanto a las virtudes de este sistema cabe destacar:

- El *framework Cocoa Touch* es fácil de usar.
- Existe una gran comunidad de desarrolladores en torno a *iOS.*
- La optimización de código es mejor que en Android porque *iOS* está diseñado sólo para los dispositivos de *Apple.*
- En *iOS* existe un proceso de aprobación en el App Store, en el cual las aplicaciones son revisadas antes de que se publiquen. De este modo se puede descargar contenido de manera segura sin límites.
- Similitud a nivel sistema operativo, a diferencia de *Android* aquí no importa si usas un *tablet*, un iPod o un teléfono, todo luce igual. Lo cual sin duda es un factor muy agradecido por todos aquellos que se están iniciando en el mundo de la tecnología, principalmente para personas mayores.
- *Siri*, el asistente de voz de *iOS* que permite conocer bastantes cosas con solo preguntárselo con la voz. Además, suele ser de bastante utilidad cuando se necesita que el equipo nos recuerde algo, poner alguna alarma o fijar algún evento en el calendario sin siquiera tener que desbloquear el teléfono.
- Multitarea real a partir de *iOS 7.*

En cuanto a los principales inconvenientes se encuentran:

- Los lenguajes de programación de *iOS (swift y objective-c)* no están tan extendidos como el lenguaje empleado por *Android*, *Java*.
- Apple no permite modificar la API de cualquier componente de su *framework*, lo que resta libertad y capacidad de innovación al desarrollador.
- Apple sólo permite desarrollar aplicaciones en ordenadores de su propia marca bajo Mac OS X. Para subir una aplicación al AppStore es necesario disponer del número de serie del ordenador Mac.
- Los desarrolladores necesitan abonar actualmente una cuota de 99$ anuales para contar con las herramientas de desarrollo y poder subir aplicaciones a la tienda. Los desarrolladores que abonen esta cuenta entrarán a formar parte del iPhone Developer Program.
- Muy poca personalización visual, esto a diferencia de lo que se puede hacer en *Android*, claro. A pesar de que con *iOS 7* llega la manera de cambiar algunos colores dentro del sistema operativo con solo cambiar nuestro fondo de pantalla, la personalización en general del sistema operativo es muy pobre.

2.3 WHIDOWS PHONE

Es un sistema operativo móvil desarrollado por *Microsoft*, como sucesor de *Windows Mobile*. A diferencia de su predecesor está enfocado en el mercado de consumo en lugar de en el mercado empresarial. Es competidor directo de *Android* e *iOS*, aunque todavía está muy por detrás de ellos.

Windows Phone 8.1, anunciado en abril de 2014 en la conferencia Build para desarrolladores, es la última versión y definitiva de *Windows Phone 8*. Esta versión trae una gran cantidad de novedades que ponen a *Windows Phone* al nivel de *Android* y de *iOS*. *Windows Phone 8* sustituye a la arquitectura previamente basada en *Windows CE* con uno basado en el kernel de *Windows NT*, permitiendo a los desarrolladores portar aplicaciones fácilmente entre las dos plataformas. Los dispositivos *Windows Phone 8* admiten resoluciones de pantalla mayores, disponen de un potente procesador gráfico, albergan procesadores multi-núcleo, *RAM* aumentada, NFC (que principalmente se puede utilizar para compartir contenidos y realizar pagos) entre otros. (Microsoft, 2016)

Debido a la evidente fragmentación de sus sistemas operativos, *Microsoft* anunció en enero de 2015 que dará de baja a *Windows Phone*, para enfocarse en un único sistema más versátil denominado *Windows 10.* Este sistema es el sucesor de *Windows Phone 8.1* y tiene como objetivo principal la integración y unificación más estrecha con su homólogo para PC de *Windows 10*, y proporcionar una plataforma para los teléfonos inteligentes, y pequeñas *tablets*.

Como ya se ha dicho anteriormente, *Windows Phone 8* comparte núcleo con *Windows 8 (Windows NT).* Este hecho hace que sus modelos de aplicación tengan similitudes y nos sea sencillo compartir código entre ambas plataformas.

Aunque no es necesario tener una cuenta de desarrollador para descargar el SDK y empezar a desarrollar aplicaciones, la cuenta es necesaria para desbloquear nuestro terminal móvil y para publicar nuestras apps en el *Store de Windows Phone*.

Existen las siguientes opciones para adquirir una cuenta de desarrollador:

- Incluida si tienes una suscripción MSDN.
- Gratis para estudiantes con la suscripción *Dreamspark*.
- 80€/año para desarrolladores individuales.

A continuación, en la Ilustración 6, se puede observar la arquitectura que se utiliza en Windows Phone 8 con sus 4 capas. (Gómez, 2016)

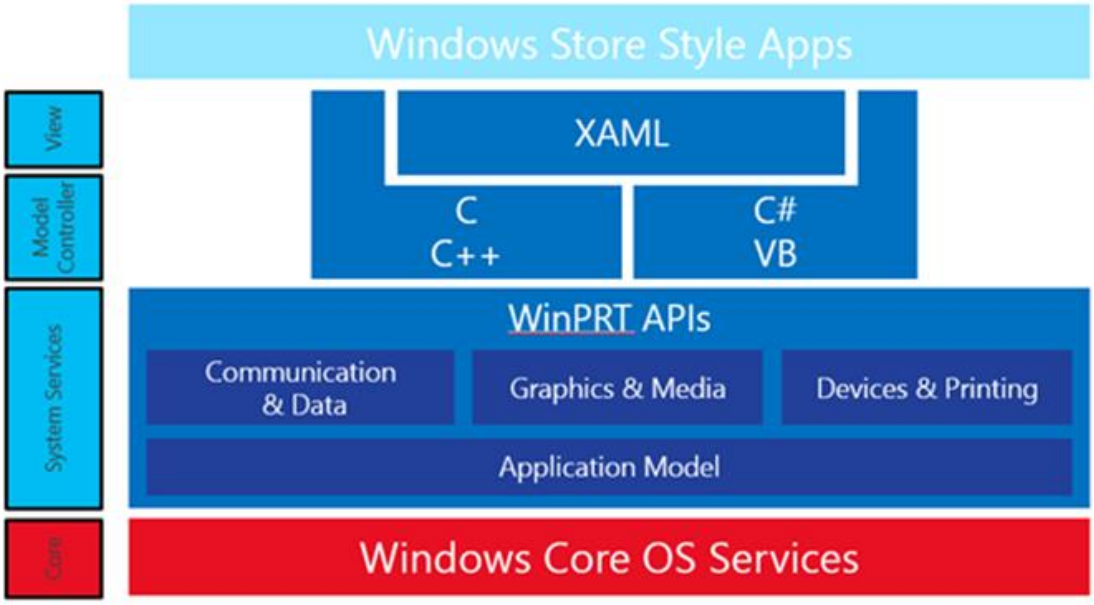

Ilustración 6. *Arquitectura del sistema Windows Phone 8*

2.3.1 ARQUITECTURA WINDOWS 8

Con la arquitectura de Windows Phone 8, Microsoft pretende poner en marcha una revolución que puede modificar el panorama del desarrollo de aplicaciones para siempre. Ofrecer a los desarrolladores web la posibilidad de crear aplicaciones nativas para Windows 8 con las mismas herramientas, lenguajes, estándares y buenas prácticas a las que están acostumbrados. Esta puerta permitirá trascender los límites de la Web y facilitará la creación de una gran cantidad de aplicaciones atractivas que pueden marcar la diferencia entre el éxito y el fracaso en Windows 8.

La programación de las nuevas Aplicaciones Metro con HTML/CSS/Javascript no van a tomar simplemente estos lenguajes, van a basarse en los estándares y buenas prácticas que ya forman parte intrínseca de la Web, o bien están en fase de desarrollo en el W3C con vistas a una próxima incorporación. (Serna, Julián, & Landa)

Esta decisión no sólo facilita la reutilización de los conocimientos de los desarrolladores web, sino que también abre la puerta a poder utilizar esquemas híbridos de desarrollo, con aplicaciones que empiezan en un entorno exclusivamente web con un conjunto de

funcionalidades reducido y que, una vez probadas, facilitan la descarga de una aplicación Metro nativa en Windows 8 con una oferta de servicios para el usuario mucho más potente.

A continuación, se detallan las características, especificaciones y componentes de cada capa:

- ***View***: contiene todas las clases que se necesitan para el desarrollo de una interfaz de usuario. La interfaz de usuario se puede desarrollar usando *XAML* (lenguaje de marcado de aplicaciones extensible) creado por Microsoft o bien utilizando *HTML5/CSS*. En esta capa de la aplicación se ejecutan todas las aplicaciones del sistema operativo.
- ***Model Controller***: capa que ofrece un *framework* que usa .NET 4.5 que utiliza *C/C++* o *C#* para comunicarse con la interfaz basada en *XAML* o *Javascript* para comunicarse con la interfaz basada en *HTML5/CSS*. En la capa de modelos se encuentran los modelos de aplicación, los modelos de interfaces de usuario y la integración a la nube, en esta capa se provee las herramientas base para el uso del sistema operativo.
- ***System Services***: contiene un conjunto de APIs que soportan el desarrollo en *C++/CX* como en *Javascript*.
- ***Core***: consiste en kernel CE de Microsoft. Se encarga de las tareas de bajo nivel como procesos, threads (hilos) y la administración de memoria del sistema. Contiene los drivers básicos de los dispositivos.

2.3.2 MODELOS DE DESARROLLO *WINDOWS PHONE 8*

La forma más común de desarrollar aplicaciones para *Windows Phone* es con un proyecto ***XAML & C#/VB***. La interfaz de usuario se define con *XAML* y la lógica con *C#* o *Visual Basic .NET*. Con este tipo de proyecto tenemos acceso a las *APIs .NET* para *Windows Phone* y *Windows Phone Runtime*, ambas con código administrado.

Otro modelo de desarrollo disponible para *Windows Phone*, usando código administrado, es ***juegos XNA***. Disponemos de la misma funcionalidad que teníamos con *Windows Phone 7*. La lógica de nuestro juego será escrita en *C#* o *Visual Basic .NET*. La gran limitación de este modelo de desarrollo es que sólo dispondremos de las *APIS* de *Windows Phone 7*, y no las de la versión 8. (Serna, Julián, & Landa)

Un tercer modelo de aplicación es el que usaremos para desarrollar aplicaciones ***Direct3D***. Este tipo de proyectos usarán exclusivamente código nativo, y su interfaz de usuario se definirá con *Direct3D*. Su uso más común es para el desarrollo de juegos.

Este modelo de aplicación nos proporciona enormes beneficios: será relativamente fácil portar un juego de *PC* a su equivalente en *Windows Phone*, tendremos acceso a las APIs de *Windows Phone Runtime* y podremos compartir componentes de código nativo como librerías gráficas, motores de juegos….

El cuarto modelo de aplicación es un híbrido entre *XAML/código administrado* y código nativo. Las aplicaciones de código administrado también pueden interactuar con librerías nativas. Para ello, podemos añadir tanto componentes *Windows Phone Runtime* como librerías *C++* de tipo *Dynamic Link Library*. En funciones de nuestra aplicación que requieran gran capacidad de computación (procesado de imágenes, renderizado de documentos, etc.) se recomienda usar código nativo para un mayor rendimiento.

El quinto modelo de aplicación que tenemos disponible en *Windows Phone 8* es el de aplicaciones ***HTML5***. *Windows Phone 8* incluye *Internet Explorer 10*, que provee un excelente soporte para *HTML5* y un nuevo motor *JavaScript* cuatro veces más rápido que en la versión anterior.

Ilustración 7. *Modelos de desarrollo de aplicaciones Windows Phone 8*

2.3.3 VENTAJAS Y DESVENTAJAS DE *WINDOWS PHONE*

En cuanto a las ventajas que nos ofrece este sistema operativo cabe destacar:

- Proporciona una interfaz intuitiva superando incluso a la que ofrece *iOS*. Podemos acceder a todas las listas de programas con solo presionare la flecha del extremo superior derecho y así, a la vez, acceder a la configuración.
- Windows Live ID: Podemos fácilmente configurar nuestra cuenta de correo hotmail, live o cualquier subdirección montada sobre los servidores de Windows Live, y así mantener sincronizada nuestra cuenta de correo, nuestros contactos y demás servicios Microsoft.
- *Windows Phone*, a través de nuestra cuenta de Windows Live ID, nos permite hacer un borrado remoto de los archivos en caso de haber extraviado el terminal.

Y en cuanto a las desventajas que ofrece este sistema cabe destacar:

- Existe un menor número de aplicaciones disponibles en el Market de Windows Phone respecto a otros sistemas operativos como *iOS* o *Android*.
- Muy poca personalización visual al igual que ocurría con *iOS*.
- No es compatible con *Flash*.
- No permite multitarea con aplicaciones de terceros.

2.4 BLACKBERRY

BlackBerry OS es un sistema operativo móvil de código cerrado desarrollado por *BlackBerry*, antigua *Research In Motion (RIM)*; para los dispositivos *BlackBerry*. El sistema permite multitarea y tiene soporte para diferentes métodos de entrada adoptados por RIM para su uso en computadoras de mano, particularmente la *trackwheel, trackball, touchpad* y pantallas táctiles. Su desarrollo se remonta a la aparición de los primeros *handheld* en 1999. Estos dispositivos permiten el acceso a correo electrónico, navegación web y sincronización con programas como *Microsoft Exchange* o *Lotus Notes* aparte de poder hacer las funciones usuales de un teléfono móvil. (Campo & García Rubio, 2014)

Se caracteriza por ser un sistema operativo atractivo y algo diferente a *Android* e *iOS*, además, sus iconos son sencillos. Casi todas sus aplicaciones piden permisos para sincronizarse con *BlackBerry Messenger*, y es que esta aplicación es el centro del teléfono.

El *OS BlackBerry* está claramente orientado a su uso profesional como gestor de correo electrónico y agenda. Desde la cuarta versión se puede sincronizar el dispositivo con el correo electrónico, el calendario, tareas, notas y contactos de *Microsoft Exchange Server* además es compatible también con *Lotus Notes* y *Novell GroupWise*.

BlackBerry Enterprise Server (BES) proporciona el acceso y organización del correo electrónico a grandes compañías identificando a cada usuario con un único BlackBerry PIN. Los usuarios más pequeños cuentan con el software BlackBerry Internet Service, programa más sencillo que proporciona acceso a Internet y a correo *POP3 / IMAP / Outlook Web Access* sin tener que usar *BES*.

Al igual que en el *SO Symbian* desarrolladores independientes también pueden crear programas para *BlackBerry*, pero en el caso de querer tener acceso a ciertas funcionalidades restringidas necesitan ser firmados digitalmente para poder ser asociados a una cuenta de desarrollador de *RIM*.

El mayor inconveniente es que todavía su tienda de aplicaciones, *BlackBerry World*, no cuenta con todas las aplicaciones que nos gustaría.

La última versión disponible es la *BlackBerry 10*, en concreto 10.3.1.

Esta última versión reúne un nuevo aspecto y funcionamiento junto con nuevas prestaciones de productividad potentes diseñadas para aumentar la productividad, colaboración y comunicación. Añade muchas mejoras, entre las que destaca la compatibilidad con aplicaciones *Android* y la sincronización de todos nuestros documentos y datos entre distintos dispositivos como el PC o la *Tablet*.

A continuación, en la Ilustración 8, se muestra la arquitectura de 4 capas que conforma este sistema operativo. En la sección siguiente se analizan con detalle estas 4 capas con los componentes que la formas y las funcionalidades que ofrece. (UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA, 2016)

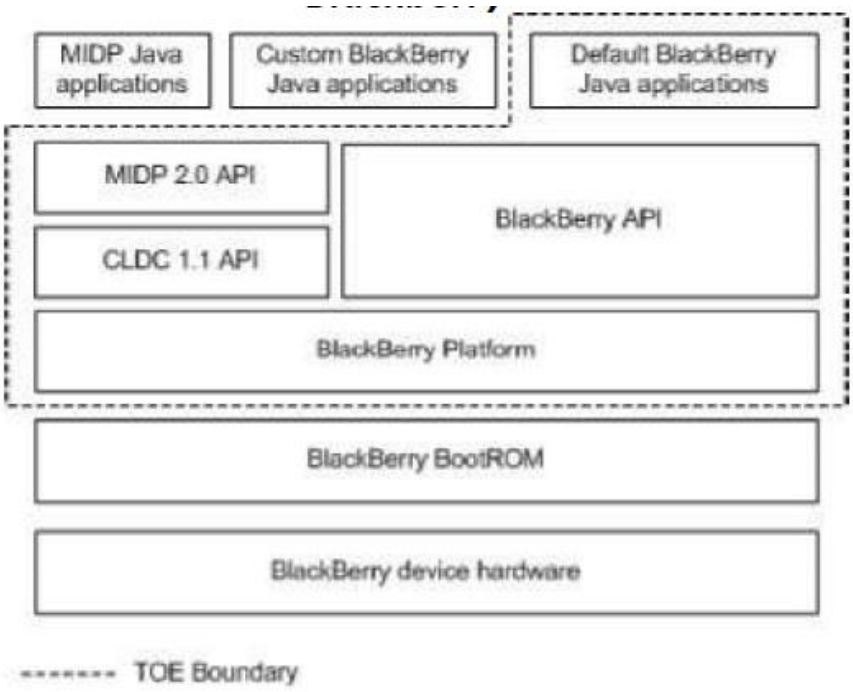

Ilustración 8. *Arquitectura de Blackberry OS*

2.4.1 ARQUITECTURA DEL SISTEMA *BLACKBERRY OS*

A continuación, se detallan las características, especificaciones y componentes de cada de las 4 capas:

- ***Backberry Device Hardware***: Esta capa del sistema operativo es la encargada de trabajar con el hardware de los dispositivos móviles.
- ***Blackberry Bootroom***: Esta capa de la arquitectura de la plataforma es la encargada de realizar un arranque seguro del hardware y del sistema operativo, este inicia en la memoria flash y verifica la firma del dispositivo en la memoria ROM del equipo para verificar que se encuentre correctamente asignado, esta es una medida de seguridad de los dispositivos BlackBerry para ejecutar los procesadores y los sistemas operativos.
- ***Blackberry Platform & Blackberry Api***: estas capas pueden ir fusionadas. Se encargan de proveer toda la plataforma del sistema operativo de *BlackBerry* como, por ejemplo, la identificación y la seguridad entre otras. A mayores provee todas las funciones y servicios de la plataforma de *Blackberry* a los desarrolladores de aplicaciones de la plataforma.
- ***Aplicaciones***: La capa de aplicaciones contiene las aplicaciones desarrolladas por defecto de *BlackBerry*, las aplicaciones *Java* caracterizadas, las aplicaciones *MIDP* y las aplicaciones desarrolladas en otras plataformas que provee *BlackBerry*.

2.4.2 VENTAJAS Y DESVENTAJAS DE *BLACKBERRY OS*

En cuanto a las ventajas que nos ofrece este sistema operativo cabe destacar:

- *Blackberry 10* se caracteriza por ser un sistema operativo realmente atractivo y algo diferente a *Android* e *iOS*, algo que también se agradece, por eso de salirse un poco de la norma. Sus iconos son sencillos y podemos personalizar de manera fácil muchos parámetros de la interfaz para adaptarlo a nuestro gusto.
- Todas las aplicaciones piden permiso para sincronizarse con *BlackBerry Messenger*, y es que esta aplicación es el centro del teléfono.
- Su sistema operativo se caracteriza por ser bastante ligero y funcionar con tremenda fluidez. Algo que puede parecer de lo más normal hoy en día, pero recordamos que, tras el uso continuado de un terminal y la instalación de muchas aplicaciones, transferencia de música, imágenes y vídeos a la memoria interna…, este se ralentiza como cualquier ordenador resultando cualquier tarea mucho más lenta e incómoda para el usuario. Esto es algo que en *Blackberry 10* difícilmente ocurre.
- Ofrece la posibilidad de separar el teléfono en dos pestañas, una para la vida personal y otra para la vida laboral, una manera de tener todo en un mismo teléfono y no tener uno para el trabajo y otro para el hogar.
- Una cualidad a destacar en *Blackberry 10* es la seguridad, un aspecto que la empresa canadiense cuida muchísimo. Y es por ello que en el Pentágono tan sólo han aprobado como seguros los dispositivos *BlackBerry Z10* y *Q10*, así como los Samsung con Android que Knox.

Y en cuanto a las desventajas que ofrece este sistema cabe destacar:

- El mayor inconveniente de *Blackberry 10* es que su tienda de aplicaciones, *BlackBerry World*, no cuenta con todas las aplicaciones que nos gustaría. El mercado de aplicaciones es bastante más limitado que en *Android* o *iOS*.
- Otro gran inconveniente es que, en torno al 20 por ciento de las aplicaciones existentes en el *BlackBerry World*, no son nativas y han sido introducidas adaptando ligeramente la versión de *Android* para que funcione más o menos bien en dispositivos BlackBerry. (Flores Galea)

2.5 SYMBIAN OS

Es un sistema operativo abierto y estándar para dispositivos de telefonía móvil. *Symbian OS* es actualmente un sistema operativo multitarea de 32 bits basado en ROM con una arquitectura de micro-kernel altamente modular que ofrece numerosas APIs (Application Programming Interfaces) para el desarrollo de aplicaciones de comunicaciones y soporta los principales estándares de la industria inalámbrica *WAP*, *XHTML*, *J2ME*, *MIDP*, *MMS*, *Bluetooth*, *GPRS*, *CDMA*, *SyncML*, *IPv6*, *IPsec*, etc. (Symbian, 2010)

Para la programación de aplicaciones se pueden utilizar distintos lenguajes: *Visual Basic*, *Java*, *OPL* y *C++*, siendo este último el lenguaje nativo de *Symbian* y el que proporciona acceso a un mayor número de funcionalidades. Existen diferentes SDKs (Software Development Kit) para el desarrollo. El SDK proporciona las herramientas y la documentación necesarias para el desarrollo de aplicaciones en *Symbian* y un emulador del terminal móvil para PC. Los distintos

SDKs están ligados a diferentes plataformas. Cada una de estas plataformas proporciona una interfaz de usuario y un conjunto de aplicaciones del sistema para mensajería, telefonía, multimedia, agenda y otras tareas, que permite a los diferentes fabricantes personalizar sus entornos de desarrollo. Estas aplicaciones hacen uso de los motores de aplicación genéricos proporcionados por *Symbian OS*. Las principales plataformas existentes son *UIQ*, *Nokia Serie 60* y *Nokia Communicator*.

El objetivo del *Symbian* era crear un sistema operativo para terminales móviles que pudiera competir con el de Palm o el Windows Mobile de Microsoft y posteriormente *Android* de Google, *iOS* de Apple, *Windows Phone* de Microsoft y *BlackBerry OS* de Blackberry.

Las características principales de este sistema operativo son:

- Uso eficiente de todos los recursos de la máquina (especialmente batería, RAM y ROM).
- Acceso inmediato a los datos.
- Ejecución de múltiples tareas (multiprogramación).
- Manejo fiable de los datos incluso en caso de fallo en la comunicación o falta de recursos, como memoria, disco o batería.
- Solo usa procesador ARM que, a diferencia de otros procesadores como Intel, este consume menor cantidad de energía lo que hace que la batería dure más.
- Adaptabilidad al hardware específico y a las pilas de telefonía de los fabricantes.
- Consistencia en la comunicación entre los datos propios del dispositivo y otros.
- Memoria RAM máxima de 2GB.

A continuación, en la Ilustración 9, se puede observar la arquitectura que se utiliza Symbian OS con sus 5 capas. En la sección siguiente se detallan los componentes y funcionalidades que posee cada capa. (Actas de las XIII Jornadas de Concurrencia y Sistemas Distribuidos, 2005)

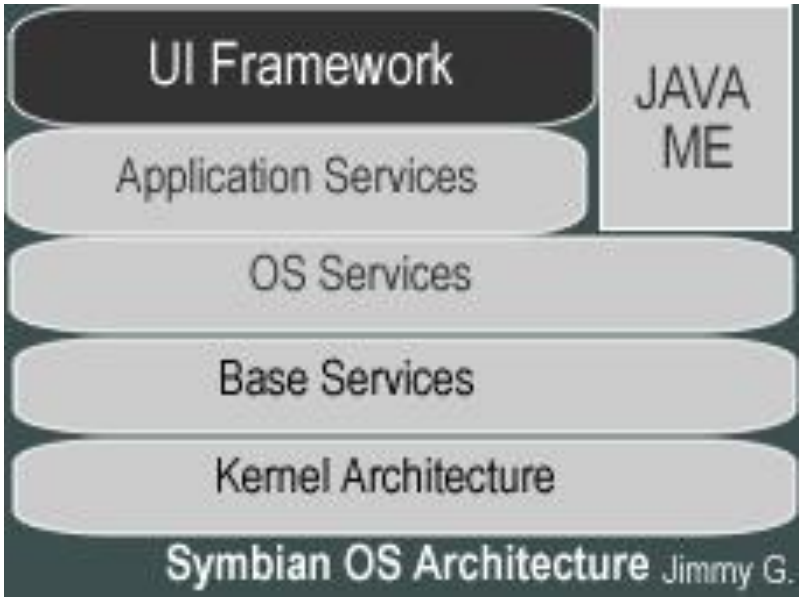

Ilustración 9. *Arquitectura del sistema Symbian OS*

2.5.1 ARQUITECTURA DEL SISTEMA *SYMBIAN OS*

Presenta una arquitectura por capas, cuyo objetivo principal es la organización jerárquica del sistema operativo.

A continuación, se detallan las características, especificaciones y componentes de cada de las 5 capas que forman este sistema operativo:

- ***Kernel Architecture***: En realidad la arquitectura correcta es la microkernel. El microkernel se encarga de lo mínimo necesario para que el sistema operativo opere correctamente, es decir, el microkernel se encarga de la gestión, de la planificación, comunicación de procesos y otras tareas elementales y los otros servicios como gestión de la memoria, etc (que se encargaba básicamente el kernel) ahora se ejecutan como procesos. La ventaja de esta arquitectura microkernel es que cuando ocurre un fallo en el sistema, este fallo solo afecta al proceso, aplicación o módulo donde ocurrió el fallo y no afecta a todo el sistema. Es de mucha utilidad en los *smartphones* pues éstos tienen diversas aplicaciones que molestarían a los usuarios si al fallar una aplicación su teléfono móvil se colgara o perdiera comunicación.
- ***Base Services***: Constituye el núcleo de *Symbian* y está formada por las librerías de usuario, el microkernel y los controladores de dispositivos (drivers).
- ***OS Services***: Son los servicios principales del sistema. Entre ellos destacan el servicio de comunicación, que proporciona el marco de trabajo y los servicios del sistema para las comunicaciones y el establecimiento de conexiones de red. También está el servicio de mensajería que proporciona el soporte para los protocolos de envío y recepción de SMS, MMS, correo electrónico… y el servicio de telefonía, entre otros.
- ***Application Services***: Se encuentra *Java ME*, que es un conjunto de interfaces de programación de aplicación para que las diferentes empresas que producen equipos que soportan este sistema operativo puedan modificar las aplicaciones a los requisitos de sus equipos creando así diversas plataformas del sistema operativo.
- ***UI Framework***: Es la capa que permite la interacción entre el usuario y la plataforma del sistema.

2.5.2 VENTAJAS Y DESVENTAJAS DE *SYMBIAN OS*

En cuanto a las ventajas que nos ofrece este sistema operativo cabe destacar:

- Instalación de programas y juegos.
- Sistema operativo fiable (presencia desde hace 10 años en el mercado).
- Un sistema multitarea bien desarrollado.
- Mayor duración de la batería. Tiene más autonomía.
- Variedad de dispositivos disponibles.
- Fuerte énfasis en las funciones básicas de telefonía y multimedia en sus dispositivos.
- Actualizaciones constantes durante muchos años.
- Es compatible con terminales de todas las gamas.
- Poca vulnerabilidad.

Y en cuanto a las desventajas que ofrece este sistema cabe destacar:

- El precio de los móviles que incluyen *Symbian* suele ser más caro que el de los modelos que no lo llevan.

- El equipo a veces tarda en responder.
- Interfaz poco estética o rústica.

2.6 COMPARATIVA ENTRE SISTEMAS OPERATIVOS MÓVILES

Android e *iOS* se postulan como grandes referentes en el mundo de los *smartphones*, pero hay otros sistemas operativos que tratan de ofrecer alternativas cada vez más capaces y atractivas como, por ejemplo, *Windows Phone* y *Blackberry RIM*.

A diferencia de *iOS*, *Android* cuenta con numerosos dispositivos en el mercado, desde los más simples y económicos hasta los más complejos y costosos. Por esta razón, la plataforma para *smartphones* de Google, actualmente acapara el 70% de la cuota de mercado global.

La plataforma de *smartphones* de Apple ha sido un referente desde el lanzamiento del primer *iPhone* en el año 2007. Presume de 800.000 aplicaciones en su *App Store* y de una cuidada interfaz, que ha servido de inspiración para el resto de sistemas operativos móviles.

Otra ventaja de *Android* sobre *iOS* es la posibilidad de reproducir vídeos en alta definición en alguna pantalla o TV gracias a varios modelos que poseen puertos *HDMI*. El *iPhone*, además de no poseer este puerto, tampoco tiene slot para tarjetas de memoria para poder expandir su capacidad con una simple *microSD*.

Por otra parte, los usuarios de *iOS* tienen la ventaja de tener siempre la última versión del sistema gracias a una actualización constante y automática.

Entrando un poco más en las especificaciones técnicas y en los componentes esenciales que forman estos sistemas operativos, un componente a tener en cuenta es el *kernel* (núcleo) que utilizan. *Android* usa un *kernel* Linux, con una mezcla especial de *Java*. El *iPhone* se basa en *OS X*, que a su vez es una variante de *Unix*, uno de los sistemas operativos más poderosos en el mundo de la informática. *S60* y *Windows Mobile* son sistemas operativos muy maduros y estables, aunque la edad no siempre es una ventaja. Por último, *RIM* usa un *kernel* propio, que al igual que *Android*, tiene un motor *Java*, y aunque han mejorado la interfaz notablemente, suele mostrar algunas limitaciones propias de su edad.

En la Ilustración 10 podemos ver algunas de las principales características de los sistemas operativos estudiados anteriormente.

COMPARATIVA PRINCIPALES S.O. MÓVILES

	iOS 10	Nougat 7.0	Symbian 9.5	Windows 10 Mobile	BlackBerry 10
Compañía	Apple	Google	Symbian Foundation*	Microsoft	Blackberry
Núcleo SO	Mac OS x	Linux	Mobile OS	Windows NT	QNX (tipo Unix)
Familia CPU	ARM	ARM	ARM	ARM	ARM/MIPS
Arquitectura	64bit	64 bit	32bit	32bit	32bit
Lenguaje programación	Objetive-c /Swift	Java	C++	C#	C++
Licencia software	Propietaria	Libre	Libre	Propietaria	Propietaria
HTML5	Sí	Sí	No	Sí	Sí
Tienda Apps	App Store	Google Play	Ovi Store	Tienda Windows	BlackBerry World
Cuenta desarrollador	99$/año	25$ una vez	Gratis	19$ una vez	Gratis
Memoria externa	No	Sí (SD/microSD)	Sí (SD/microSD)	Sí (SD/microSD)	Sí (SD/microSD)
Apps nativas	Sí	Sí	Sí	Sí	Sí
Asistente virtual	Siri	Google Now	----------	Cortana	BlackBerry Assistant
IDE	Xcode	Android Studio	Eclipse	Windows App Studio	Momentics
Plataforma desarrollo	Mac	Windows/Mac /Linux	Windows/Mac /Linux	Windows 10	Windows/Mac /Linux

Ilustración 10. *Comparativa de las principales plataformas móviles (Noviembre 2016)*

*Nokia, Sony Mobile Communications, Psion, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic, Sharp…

PRINCIPALES CARACTERÍSTICAS S.O. MÓVILES

	iOS 10	Nougat 7.0	Symbian 9.5	Windows 10 Mobile	BlackBerry 10
Multiventana*	Sí	Sí	No	Sí	No
Eliminar apps preinstaladas	Sí	Sí	No	No	No
Bloqueo notificaciones	Sí	Sí	No	Sí	Sí
Interacción con notificaciones desde bloqueo	Sí	Sí	No	No	No
Aviso llamadas *spam*	Sí	Sí	No	Sí	No
Sincronización escritorio	Sí	No	No	Sí	Sí

Ilustración 11. *Principales características de los S.O. móviles (Noviembre 2016)*

*Multiventana: varias apps en la pantalla al mismo tiempo

2.7 ELECCIÓN DE LA PLATAFORMA PARA EL DESARROLLO DE LA APP

Tras realizar un estudio de los principales sistemas operativos móviles conociendo en profundidad sus mayores cualidades y ventajas que nos pueden ofrecer, así como los mayores inconvenientes y puntos débiles que poseen, finalmente la plataforma de desarrollo elegida para el desarrollo e implementación de la aplicación es *Android*.

A continuación, se presentan los principales motivos por los que se ha elegido la plataforma *Android*:

- El principal motivo para utilizar esta plataforma es el uso de la misma. El 51,9% de todos los terminales móviles en uso utilizan *Android* frente al 42,9% que utilizan *iOS*, su inmediato competidor.
- Otro motivo a tener en cuenta es la inmensa variedad de terminales que soportan este sistema operativo. Existen una infinidad de dispositivos de todos los tamaños y gamas. Tenemos terminales muy básicos por un precio bastante asequible hasta terminales de alta gama con un precio bastante superior.
- El último motivo es la existencia de una aplicación similar para la plataforma *iOS*.

Una vez selecciona la plataforma, hay que decidir a partir de que versión de *Android* va a estar disponible la aplicación. A continuación, se muestran las versiones de *Android* que más se utilizan con su correspondiente número de versión:

- ***KitKat***: 4.4
- ***Lollipop***: 5.0
- ***Marshmallow***: 6.0
- ***Froyo***: 2.2
- ***Gingerbread***: 2.3
- ***Ice Cream Sandwich***: 4.0
- ***Jelly Bean***: 4.3

Si se observa la Ilustración 12, las versiones que más se utilizan son *KitKat*, *Lollipop* y *Jelly Bean* todas superiores a 4.0. Teniendo en cuenta este dato, la aplicación se podrá ejecutar en Android 4.0 y superiores. No merece la pena dar soporte a versiones anteriores puesto que ofrecen muchas menos funcionalidades (sobre todo en la interfaz de usuario) que no podríamos utilizar y el número de dispositivos que no van a poder utilizar la aplicación es ínfimo. Por supuesto siempre tienen la opción de actualizar la versión de *Android* que utilizan en sus dispositivos. Los beneficios que se obtienen por dar soporte a partir de la versión 4.0 son considerables tanto a nivel de interfaz de usuario como al uso de las librerías siendo estás mucho más completas y están mucho más optimizadas. En cuanto a las librerías que se van a utilizar para la conexión del servidor son más seguras y eficientes.

En conclusión, la aplicación se va a realizar para la plataforma *Android* dando soporte a versiones de Android 4.0 y superiores por los motivos expuestos anteriormente.

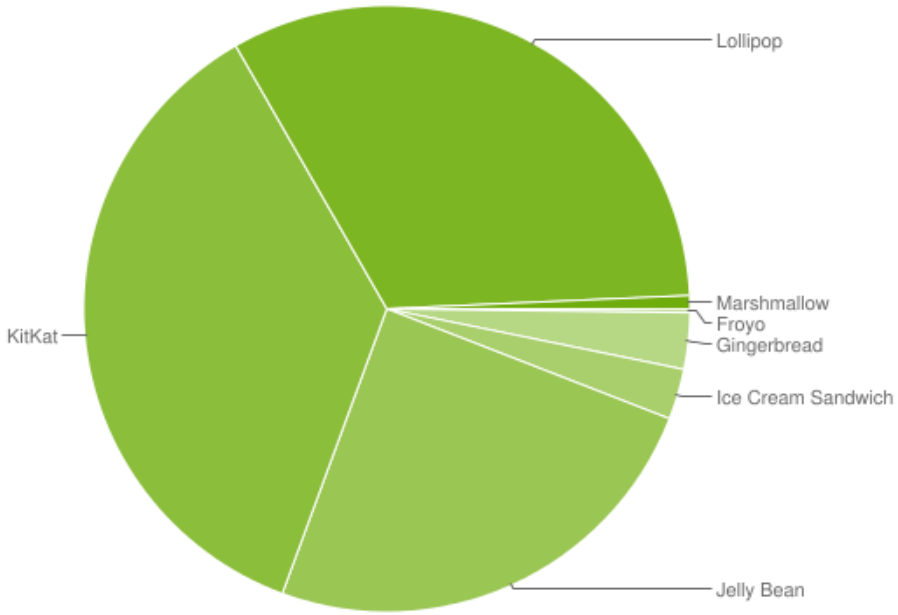

Ilustración 12. *Versiones de Android utilizadas (Diciembre 2015)*

CAPÍTULO 3

3 DESARROLLO DE LA APLICACIÓN

En este capítulo se explica de forma detallada el diseño y desarrollo de la aplicación. Primero se muestra la configuración de *Android Studio*, el *IDE* (*Integrated Development Enviroment* o Entorno de Desarrollo Integrado) que utilizaremos para desarrollar la app. Tras esto se explica el flujo de la app utilizando diagramas de flujo para ver el comportamiento de la misma y los patrones de diseño seguidos para crearla. También se explica la arquitectura cliente-servidor que se va a utilizar para obtener todos los datos del servidor para que todo esté centralizado y la app sea más ligera y ocupe menos memoria en el terminal. (Leuzas, 2015)

3.1 ANDROID STUDIO

Android Studio es un entorno de desarrollo integrado para la plataforma *Android*. Fue anunciado el 16 de mayo de 2013 en la conferencia *Google I/O*, y reemplazó a *Eclipse* como el *IDE* oficial para el desarrollo de aplicaciones para *Android*. La primera versión estable fue publicada en diciembre de 2014. Anteriormente se usaba *Eclipse* y la *SDK* de Android para desarrollar aplicaciones. (Android Studio website, 2016)

Está basado en el software *IntelliJ IDEA* de JetBrains, y es publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux.

Android Studio está disponible para Windows 2003, Vista, 7, 8, 10 y GNU/Linux, tanto plataformas de 32 como de 64 bits, Linux con *GNOME* o *KDE* y 2 GB de memoria RAM mínimo y Mac OS X, desde 10.8.5 en adelante.

Entre sus características destacan:

- Renderización en tiempo real.
- Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Soporte para construcción basada en *Gradle*.
- Refactorización especifica de Android y arreglos rápidos.
- Herramientas *Lint* para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones, y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para *Android Wear*.

Este *IDE* requiere de unos requisitos de sistema un poco elevados:

- 2 GB de RAM (4 GB recomendados).
- 400 MB de espacio en disco.
- 1 GB para *Android SDK*.
- Monitor de 1280x800 mínimo.
- *Java Development Kit 7* o superior.

También dispone de la aceleración del emulador (*HAXM*) en sistemas Windows, se requiere un procesador Intel con soporte para *VT-x*, *EM64T* y funcionalidad *Execute Disable Bit*.

En la Ilustración 13 se muestra la interfaz de usuario de *Android Studio*.

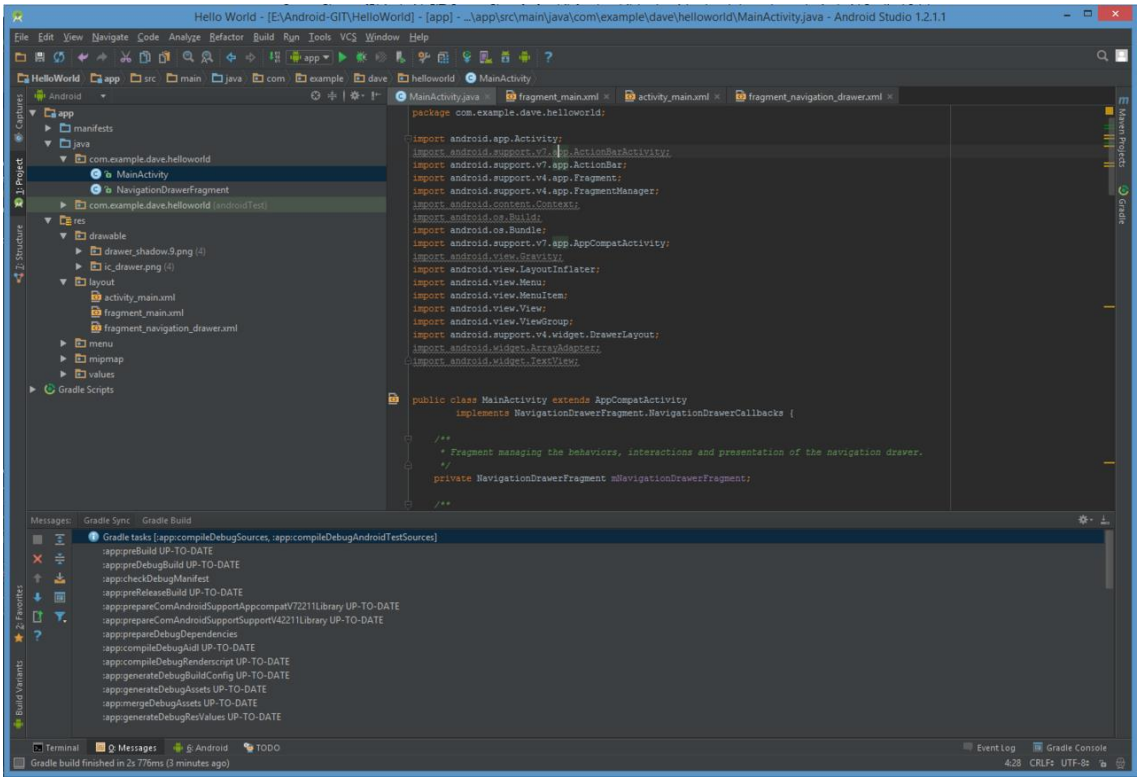

Ilustración 13. *Interfaz de Usuario de Android Studio*

3.2 MODELO CLIENTE-SERVIDOR

El término cliente servidor es aplicado a la arquitectura de software de una aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Este modelo es una relación entre procesos corriendo en máquinas separadas que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma. (Rodger, 2012)

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

El cliente es el remitente de una solicitud, hace una petición que puede convertirse en múltiples requerimientos de trabajo.

El servidor es el receptor de la solicitud enviada por el cliente. Normalmente el servidor es una máquina bastante potente que actúa de depósito de datos y funciona como un sistema gestor de base de datos.

La red cliente-servidor es una red de comunicaciones en la cual los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de sólo lectura y los que, por el contrario, pueden ser modificados, etc. Este tipo de red puede utilizarse conjuntamente en caso de que se esté utilizando en una red mixta. (Shulin, 2014)

Algunas de las características de la arquitectura modelo cliente-servidor son:

- Las funciones de cliente y servidor pueden estar en plataformas separadas o en la misma plataforma.
- El cliente y el servidor pueden actuar como una sola entidad y también como entidades separadas realizando actividades o tareas independientes.
- Un servidor da servicio a múltiples clientes en forma concurrente.
- Cada plataforma puede ser escalaba independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores se realizan de una manera transparente al usuario final.

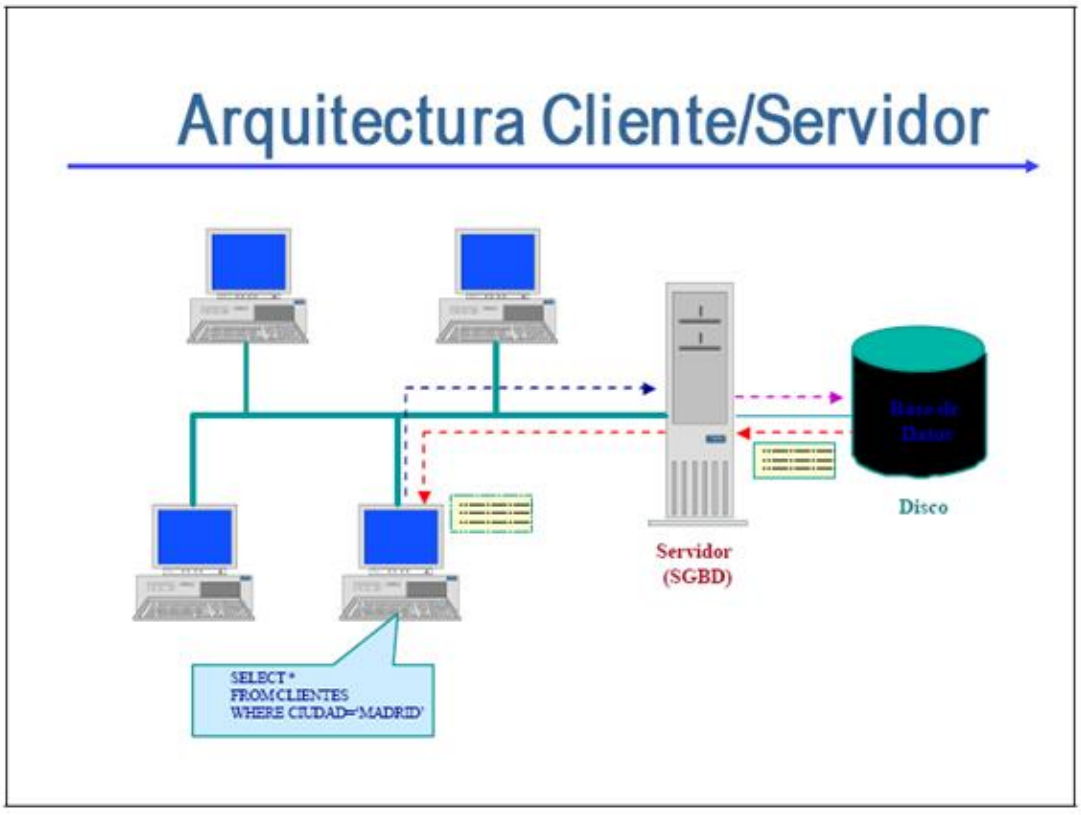

Ilustración 14. *Modelo Cliente-Servidor* (Díaz F. , 2006)

3.3 PARTE DEL CLIENTE

En esta sección se explica toda la lógica de nuestra aplicación mediante el uso de diagramas de flujo. De esta forma se tiene una visión global del funcionamiento de la aplicación. También se explica la conexión con el servidor y los datos que se obtienen del mismo.

Destacar que se ha creado un usuario por defecto para poder utilizar algunas funcionalidades de la aplicación en caso de no disponer de conexión a internet como los juegos mediante el uso de preferencias que ofrece *Android*, pero no se almacenan en local nuevos seguimientos, nuevos tutelados ni las estadísticas de la partida ya que tendríamos que mantener una base de datos en la aplicación utilizando *SQLite* lo que implicaría que la aplicación fuese más lenta y ocupase más memoria en el terminal lo cual no compensa. Además, cuando tuviese conexión a internet, habría que sincronizar todos esos datos almacenados en la aplicación guardándolos en el servidor para que estuviesen disponibles en cualquier otro terminal.

A mayores y utilizando las preferencias de nuevo, se han creado un par de tutelados por defecto y unos seguimientos para observar el funcionamiento de los mismos cuando no se dispone de conexión a internet.

3.3.1 INICIO EN LA APLICACIÓN

Nada más iniciar la aplicación el usuario tiene que introducir su usuario y contraseña. En cuanto el usuario pulsa Enviar esos datos se envían por *POST* al servidor. Si el usuario y la contraseña coinciden con el usuario almacenado en la base de datos se devuelve un mensaje de confirmación. En caso de no existir, se envía un mensaje de error. Esta primera parte de inicio de sesión del usuario se describe en la Ilustración 15. Si los datos de sesión son correctos, se carga la página principal desde la que se puede acceder a los seguimientos, a los juegos, a los tutelados y a las estadísticas.

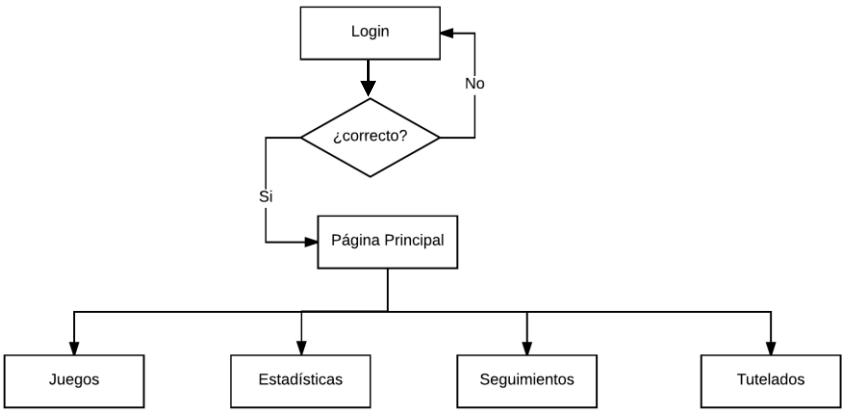

Ilustración 15. *Diagrama de flujo del inicio de la app*

3.3.2 OPCIÓN *SEGUIMIENTOS* DE LA APLICACIÓN

En esta sección se desarrolla la opción de seguimientos que aparece en la pantalla principal de la aplicación anteriormente mencionada.

Esta opción `seguimientos` es la encargada de mostrar los seguimientos que un tutelado tiene. También permite crear nuevos seguimientos para el tuteado que deseemos. En la Ilustración 16 mostrada posteriormente se muestra el diagrama de flujo que sigue esta opción `seguimientos`.

Inicialmente, se cargan los tutelados que tiene ese usuario. Para ellos se utiliza la clave `cargarTutelados.java`. En caso de que el terminal no tenga conexión a internet, se cargan 2 tutelados que están almacenados en las preferencias que ofrece *Android* como se indicó anteriormente, pero no son los tutelados reales, solo se han creado para observar el funcionamiento de esta opción.

En caso de disponer conexión a internet, se realiza una solicitud al servidor para obtener los tutelados que realmente tiene ese usuario. Una vez recibidos todos los tutelados se muestran en un *ListView* (apariencia como si fuese una lista) que ofrece *Android*. (Tomás, 2014)

Los datos enviados por el servidor están en formato *JSON*. Una vez recibidos en el cliente, se obtienen los datos que contienen utilizando la clase `JSONParser.java` creada para ello.

Una vez mostrados todos los tutelados, se puede, o bien ver todos los seguimientos de ese tutelado o crear un nuevo seguimiento para ese tutelado. Para mostrar todos los seguimientos de ese tutelado se utiliza la clase `todosSeguimientos.java`. Si por el contrario queremos crear un nuevo seguimiento, se utiliza la clase `nuevoSeguimiento.java`.

Si mostramos todos los seguimientos, tenemos que realizar otra petición al servidor para obtener todos los seguimientos que tenga el tutelado seleccionado. En caso de no disponer de conexión a internet, se obtienen un par de seguimientos que están almacenados en la aplicación mediante preferencias como ocurría en el caso de listar todos los tutelados. Una vez mostrados todos los seguimientos se pueden mostrar los detalles del mismo, realizando otra petición al servidor como en los casos anteriores. Para mostrar los detalles de un seguimiento se utiliza la clase `verDetallesSeguimiento.java`.

En caso de crear un nuevo seguimiento es necesario disponer de conexión a internet. En caso de no disponer de conexión no se podrá crear ese nuevo seguimiento. Para crearlo el usuario introduce todos los datos que se necesitan y, una vez introducidos, se envían al servidor. Si todo ha ido bien, se muestra un mensaje de confirmación.

Básicamente estas son las funcionalidades que ofrece esta opción.

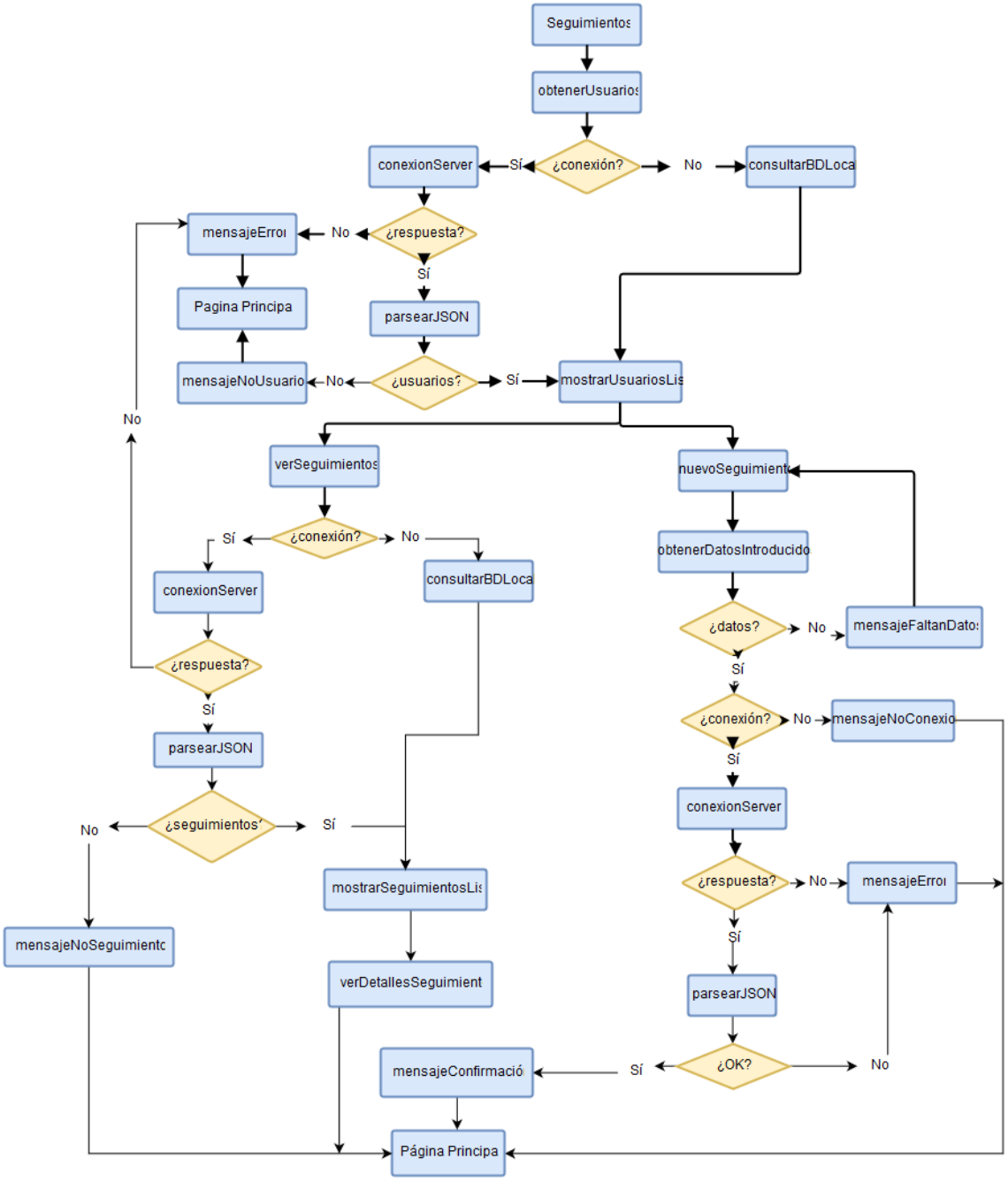

Ilustración 16. *Diagrama de flujo de la opción Seguimientos*

3.3.3 OPCIÓN *ESTADÍSTICAS* DE LA APLICACIÓN

En esta sección se desarrolla la opción `estadísticas` que se encuentra en la página principal de la aplicación.

Esta opción permite visualizar en una gráfica parámetros de interés (tiempo de juego, respuestas correctas…) de las partidas que un tutelado ha jugado. Con estos resultados se puede observar si el tutelado adquiere ciertas habilidades y mejorías al realizar los juegos o si, por el contrario, se estanca y se necesita cambiar el tipo de juego o la terapia.

Inicialmente, al igual que ocurría en la opción `seguimientos`, primero se muestran todos los tutelados siguiendo el mismo procedimiento. Una vez que seleccionamos el tutelado se muestran todos los parámetros para mostrar la gráfica correspondiente. Para ello se utiliza la clase `estadística.java`.

Una vez que el usuario ha introducido los datos que desea visualizar, se envían al servidor para obtener todas las estadísticas que se van a visualizar. Estos datos se reciben en formato *JSON* como todos los datos recibidos del servidor. Se interpretan utilizando la clase `JSONParser.java`.

Una vez obtenidos los datos, se utiliza la clase `verEstadisticas.java`. Utilizando una librería para representar gráficas, se representan todos los datos en formato de gráfica (*LineChart*).

En la Ilustración 17 se muestra el diagrama de flujo de esta opción. Se refleja todo lo expuesto anteriormente.

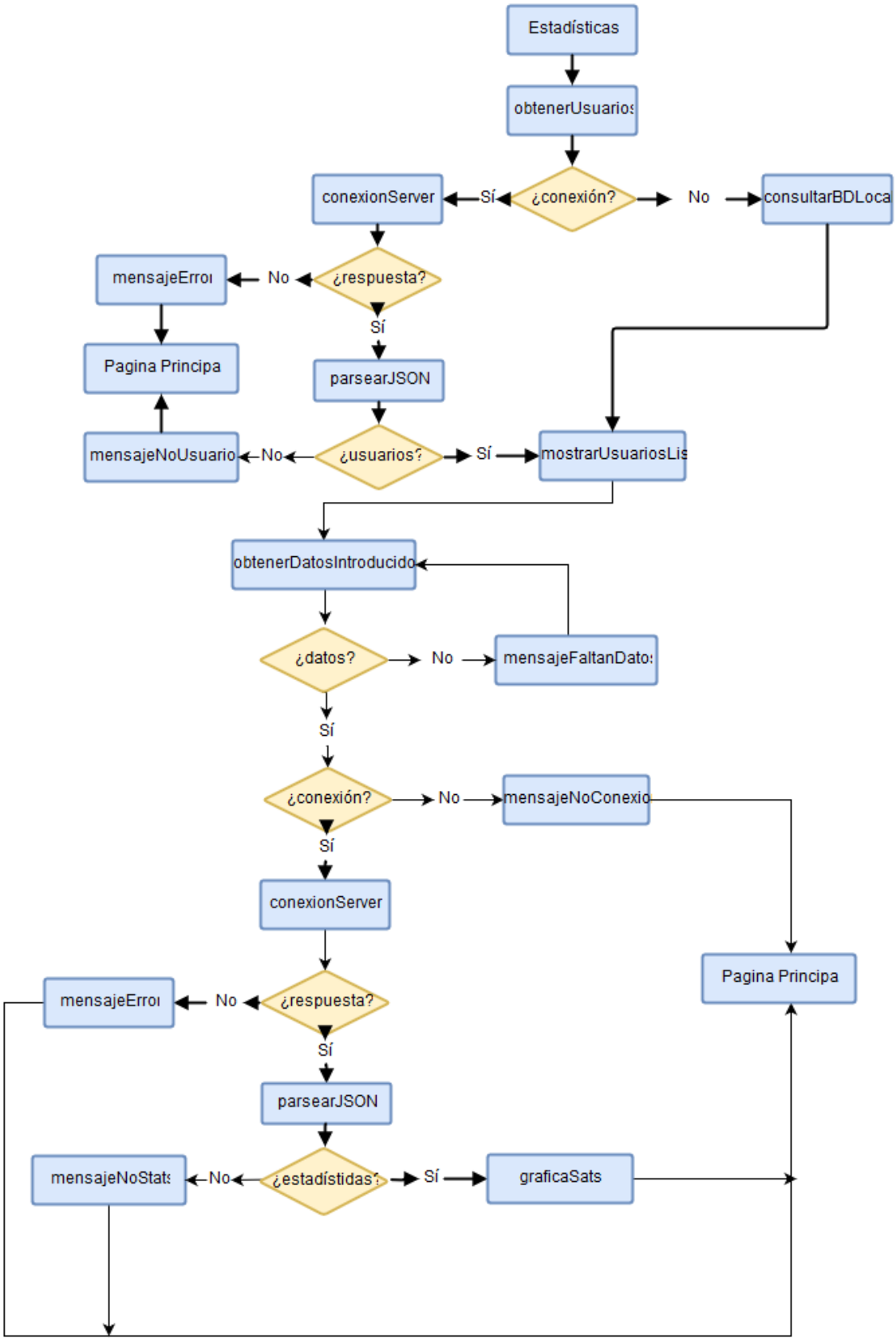

Ilustración 17. *Diagrama de flujo de la opción Estadísticas*

3.3.4 OPCIÓN *TUTELADOS* DE LA APLICACIÓN

En esta sección se desarrolla la opción `tutelados` que se encuentra en la página principal de la aplicación.

Esta opción permite visualizar los tutelados que tiene un usuario a su cargo, así como ver la información detallada de cada tutelado. En este perfil del tutelado aparecen datos interesantes desde el punto de vista del voluntario como dirección del domicilio, alergias detectadas, teléfono del tutelado, observaciones de salud y más datos relacionados. A mayores también se permite crear nuevos tutelados a cargo del usuario.

Para la creación de nuevos tutelados se utiliza la clase `nuevoTutelado.java`. Para crear un nuevo tutelado el usuario introduce todos los datos que le indica la aplicación. Una vez introducidos todos los datos, se envían los datos al servidor mediante una petición POST que los contiene. El servidor envía un mensaje que indica cómo ha ido la operación (OK o ERROR) y este mensaje se muestra al usuario.

Para mostrar los detalles de un tutelado primer se muestran todos los tutelados que tiene ese usuario a su cargo utilizando la clase `verTutelados.java` como se realiza en las otras opciones. Una vez que se obtienen y muestran todos los tutelados, se selecciona el tutelado del que se van a mostrar los detalles. Para mostrar los detalles se utiliza la clase `detallesTutelado.java`. En esta clase se envía una petición POST al servidor para obtener los detalles del tutelado apoyándose en la clase `JSONParser.java` para realizar esa petición. Una vez recibida la respuesta del servidor, y si todo ha ido bien, se muestran los detalles al usuario.

En la Ilustración 18 se expone, mediante un diagrama de flujo, el funcionamiento de esta opción. Se detallan todas las operaciones que se realizan en las distintas funcionalidades que ofrece y que están explicadas anteriormente.

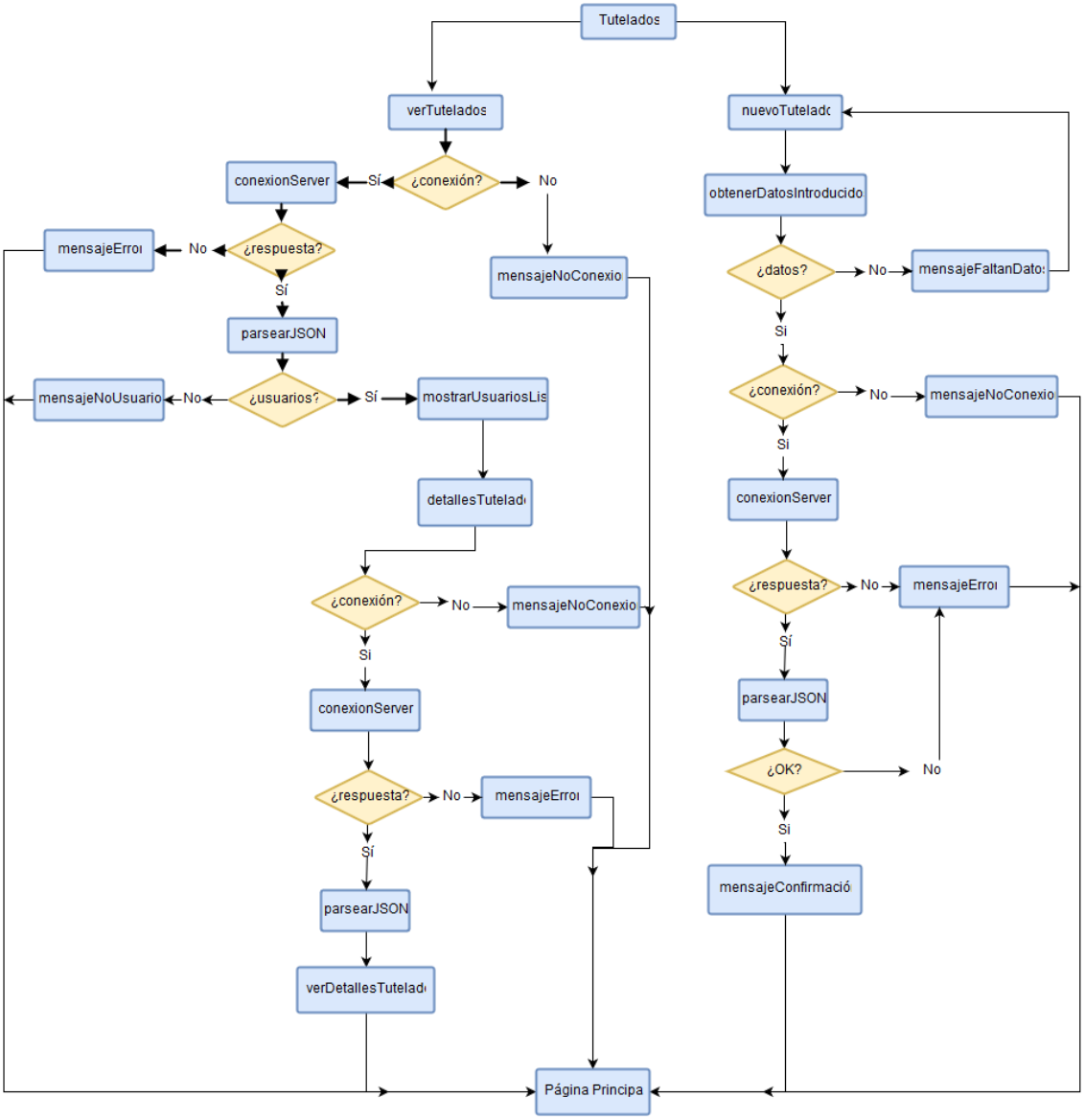

Ilustración 18. *Diagrama de flujo de la opción Tutelados*

3.3.5 OPCIÓN *JUEGOS* DE LA APLICACIÓN

En esta sección se desarrolla la opción `juegos` que se encuentra en la página principal de la aplicación.

En esta sección se encuentran básicamente todos los juegos que dispone la aplicación. Como en todas las opciones que ofrece la aplicación, primero se selecciona el tutelado que va a jugar. Como en los casos anteriores, si no se dispone de conexión se consulta la base de datos local para que se puedan ver todos los juegos que están disponibles sin tener que tener conexión a internet en el dispositivo.

Una vez seleccionado el tutelado, se selecciona la categoría a la que se quiere jugar y, tras esto, el juego al que se quiera jugar.

En la pantalla en la que se está jugando se muestran 2 opciones: comprobar y mostrar resultados.

La opción `comprobar` permite mostrar las opciones correctas e incorrectas. Por ejemplo, en el juego Matemáticas que consiste en completar una serie de operaciones aritméticas (sumas, restas y multiplicaciones dependiendo del nivel de juego), si se pulsa el botón `comprobar` se muestran las respuestas correctas y las incorrectas.

La opción `mostrar Resultados` carga una nueva pantalla (*Activity*) y se muestran las estadísticas de ese juego como tiempo jugado, respuestas correctas, respuestas incorrectas, intentos…. Además, si el dispositivo dispone de conexión a internet, estas estadísticas se almacenan en el servidor para obtener posteriormente las estadísticas de cada juego y observar la evolución del tutelado.

En la Ilustración 19 se expone, mediante un diagrama de flujo, el funcionamiento de esta opción. Se detallan todas las operaciones que se realizan en las distintas funcionalidades que ofrece y que están explicadas anteriormente.

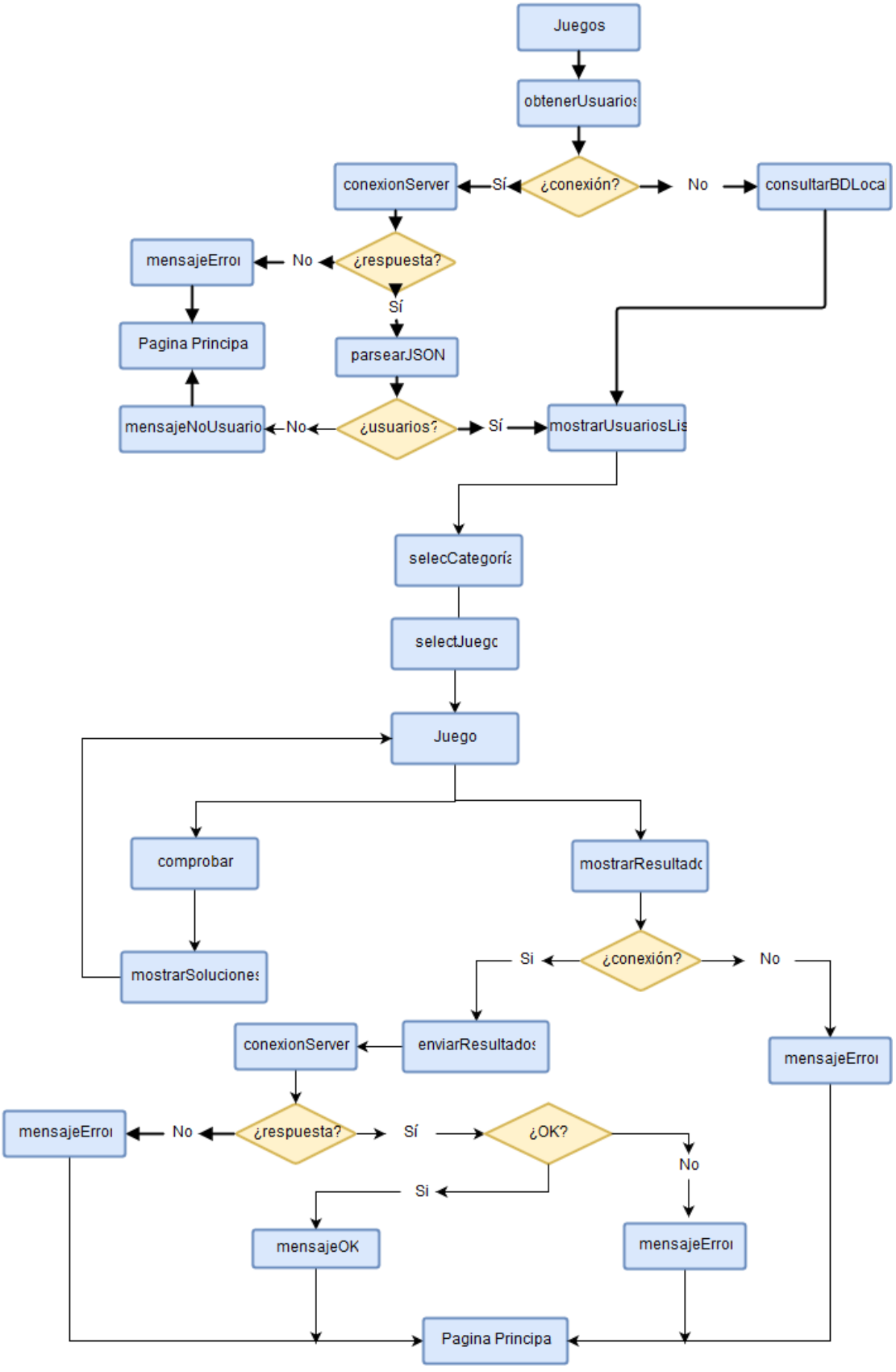

Ilustración 19. *Diagrama de flujo de la opción Juegos*

3.4 PARTE DEL SERVIDOR

El servidor es el encargado de mantener todos los datos necesarios para el correcto funcionamiento de la aplicación. Estos datos se almacenan en una base de datos que sigue un modelo relacional vinculando tablas entre sí. En la Ilustración 20 se muestra el modelo de base de datos creado. (Welling, 2003)

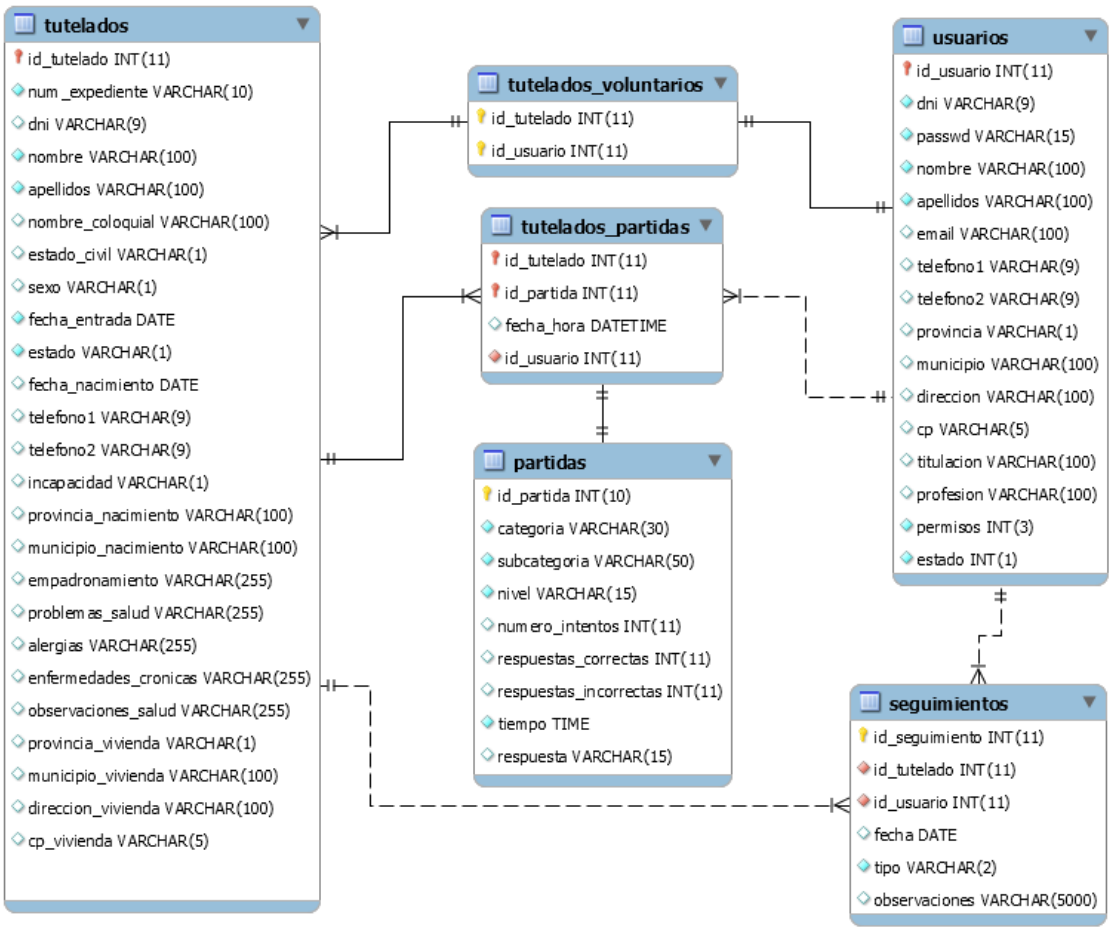

Ilustración 20. *Estructura de la base de datos*

En las sucesivas secciones se explican los contenidos y las relaciones que existen entre las distintas tablas que conforman nuestra base de datos.

3.4.1 TABLA *TUTELADOS*

Esta tabla contiene toda la información directa e indirecta de los tutelados, es decir, habrá datos de interés para el usuario, y datos útiles para la aplicación.

En la Ilustración 21 se muestran todos los campos de esta tabla con sus correspondientes tipos de datos asociados y si ese campo puede estar vacío o no. Como clave primaria de la tabla se utiliza el campo `id_tutelado` auto incremental, esto es, se asigna automáticamente cuando se crea un nuevo tutelado en la base de datos.

NOMBRE	TIPO	NULO	PREDETERMINADO	EXTRA
id_tutelado	int(11)	No	Ninguna	AUTO_INCREMENT
num_expediente	varchar(10)	No	Ninguna	
dni	varchar(9)	Si	NULL	
nombre	varchar(100)	No	Ninguna	
apellidos	varchar(100)	No	Ninguna	
nombre_coloquial	varchar(100)	Si	NULL	
estado_civil	varchar(1)	Si	NULL	
sexo	varchar(1)	Si	NULL	
fecha_entrada	date	No	Ninguna	
estado	varchar(1)	No	Ninguna	
fecha_nacimiento	date	Si	NULL	
telefono1	varchar(9)	Si	NULL	
telefono2	varchar(9)	Si	NULL	
incapacidad	varchar(1)	Si	NULL	
provincia_nacimiento	varchar(100)	Si	NULL	
municipio_nacimiento	varchar(100)	Si	NULL	
empadronamiento	varchar(255)	Si	NULL	
problemas_salud	varchar(255)	Si	NULL	
alergias	varchar(255)	Si	NULL	
enfermedades_cronicas	varchar(255)	Si	NULL	
observaciones_salud	varchar(255)	Si	NULL	
provincia_vivienda	varchar(1)	Si	NULL	
municipio_vivienda	varchar(100)	Si	NULL	
direccion_vivienda	varchar(100)	Si	NULL	
cp_vivienda	varchar(5)	Si	NULL	

Ilustración 21. *Tabla TUTELADOS*

3.4.2 TABLA *USUARIOS*

Esta tabla contiene toda la información de los usuarios del sistema, pudiendo ser voluntarios que tutelan a sus tutelados. Destacar que a los tutelados no se les entiende como usuarios del sistema por lo que se encuentran en otra tabla de la base de datos. En la Ilustración 22 se muestran los campos que tiene esta tabla con el tipo de dato de cada campo y si puede estar vacío o no. Esta tabla utiliza como clave primaria el campo `id_usuario` auto incremental como en la tabla `Tutelados`. Cuando el usuario inicia sesión en la aplicación utiliza los campos `email` y `passwd`. Si coinciden los datos que el usuario envía con los que están almacenados en la base de datos se crea la sesión para ese usuario.

NOMBRE	TIPO	NULO	PREDETERMINADO	EXTRA
id_usuario	int(11)	No	Ninguna	AUTO_INCREMENT
dni	varchar(9)	No	Ninguna	
passwd	varchar(15)	No	Ninguna	
nombre	varchar(100)	No	Ninguna	
apellidos	varchar(100)	No	Ninguna	
email	varchar(100)	Si	NULL	
telefono1	varchar(9)	Si	NULL	
telefono2	varchar(9)	Si	NULL	
provincia	varchar(1)	Si	NULL	
municipio	varchar(100)	Si	NULL	
direccion	varchar(100)	Si	NULL	
cp	varchar(5)	Si	NULL	
titulacion	varchar(100)	Si	NULL	
profesion	varchar(100)	Si	NULL	
permisos	int(3)	No	10	
estado	int(1)	No	1	

Ilustración 22. *Tabla USUARIOS*

3.4.3 TABLA *TUTELADOS_USUARIOS*

Esta tabla posee la relación tutelados-usuarios. Es decir, asocia a cada usuario todos los tutelados que le pertenecen. Esta tabla relacional necesita 2 claves primarias que son `id_usuario` e `id_tutelado`. En la Ilustración 23 se muestran los campos de esta tabla con los tipos de datos como en las anteriores tablas.

NOMBRE	TIPO	NULO	PREDETERMINADO	EXTRA
id_tutelado	int(11)	No	Ninguna	
id_usuario	int(11)	No	Ninguna	

Ilustración 23. *Tabla TUTELADOS_USUARIOS*

3.4.4 TABLA *TUTELADOS_PARTIDAS*

Esta tabla asocia las partidas con los tutelados. El usuario posee uno o varios tutelados y ellos a su vez pueden tener una o varias partidas. El funcionamiento es similar a la tabla `Tutelados_Usuarios`. Relaciona a cada tutelado con todas las partidas que ha jugado. Las claves primarias de esta tabla son `id_partida` e `id_tutelado`. A mayores esta tabla almacena la fecha y la hora en la que se guardó la partida por si en algún momento puede ser de interés conocer este dato.

NOMBRE	TIPO	NULO	PREDETERMINADO	EXTRA
id_tutelado	int(11)	No	Ninguna	
id_partida	int(11)	No	Ninguna	
fecha_hora	datetime	Si	NULL	
id_usuario	int(11)	Si	NULL	

Ilustración 24. *Tabla TUTELADOS_PARTIDAS*

3.4.5 TABLA *PARTIDAS*

Esta tabla almacena toda la información referente a las partidas que juegan los voluntarios. Contiene toda la información relacionada con las partidas que se juegan. Cada juego utiliza unos campos u otros dependiendo de los datos que se quieran almacenar. La clave primaria de la tabla es `id_partida` auto incremental, esto es, se asigna automáticamente cuando se crea una nueva partida en la base de datos.

Destacar que el campo `categoría` almacena la categoría a la que pertenece el juego (razonamiento, cálculo, percepción…), el campo `subcategoría` almacena el nombre del juego (Da El Cambio, Laberintos…) y el campo `nivel` almacena la dificultad en la que se juega la partida, es decir, fácil, medio o difícil.

NOMBRE	TIPO	NULO	PREDETERMINADO	EXTRA
id_partida	int(10)	No	Ninguna	AUTO_INCREMENT
categoria	varchar(30)	No	Ninguna	
subcategoria	varchar(50)	No	Ninguna	
nivel	varchar(15)	No	Ninguna	
numero_intentos	int(11)	Si	NULL	
respuestas_correctas	int(11)	Si	NULL	
respuestas_incorrectas	int(11)	Si	NULL	
tiempo	time	No	Ninguna	
respuesta	varchar(15)	Si	NULL	

Ilustración 25. *Tabla PARTIDAS*

3.4.6 TABLA *SEGUIMIENTOS*

Posee información acerca de los tutelados. Cuando un usuario o voluntario entra en la aplicación con su email y contraseña, puede ver seguimientos de los tutelados que tiene asignados o insertar un nuevo seguimiento. Esta tabla es la que almacena esos datos que introduce el voluntario. Como clave primaria de la tabla se utiliza el campo `id_seguimiento` auto incremental, esto es, se asigna automáticamente cuando se crea un nuevo seguimiento en la base de datos. En el campo `observaciones` se almacena el comentario que el voluntario crea importante sobre el tutelado y el campo `fecha` almacena la fecha y la hora en la que se creó ese nuevo seguimiento.

En la Ilustración 26 se muestran todos los campos de esta tabla con sus correspondientes tipos de datos asociados y si ese campo puede estar vacío o no.

NOMBRE	TIPO	NULO	PREDETERMINADO	EXTRA
id_seguimiento	int(11)	No	Ninguna	AUTO_INCREMENT
id_tutelado	int(11)	No	Ninguna	
id_usuario	int(11)	No	Ninguna	
fecha	date	Si	NULL	
tipo	varchar(2)	No	2	
observaciones	varchar(500)	Si	NULL	

Ilustración 26. *Tabla SEGUIMIENTOS*

3.5 COMUNICACIÓN CLIENTE-SERVIDOR

Una vez descrito y entendido el funcionamiento tanto de la parte del cliente como la parte del servidor de nuestra aplicación, desarrollamos la comunicación entre ellos para que la aplicación funcione correctamente.

La comunicación entre el cliente y el servidor se lleva a cabo mediante el uso de servicios web *RESTful* con *HTTP*, más concretamente, utilizando peticiones *POST* y obteniendo en la respuesta la representación del recurso solicitado en formato *JSON*. (Potencier & Weaver, 2012)

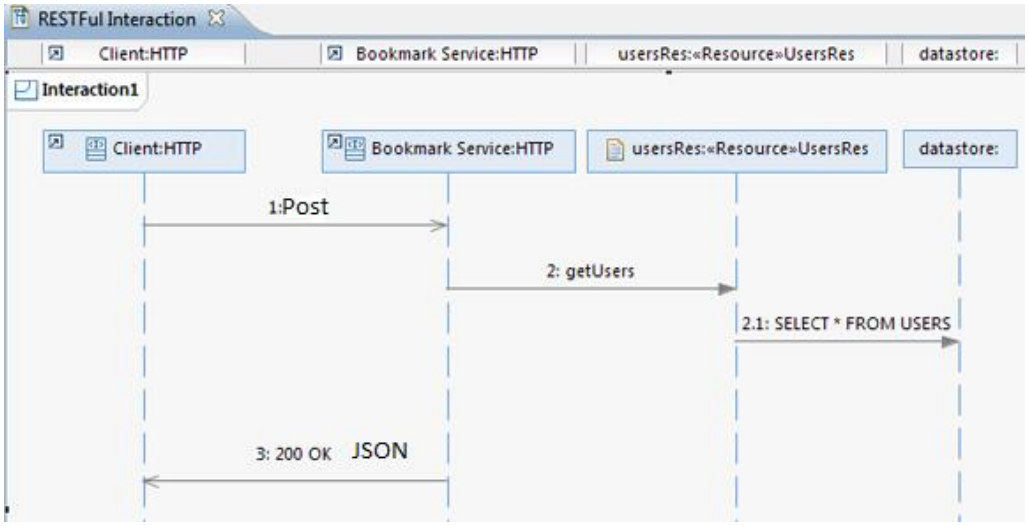

Ilustración 27. *Comunicación cliente-servidor*

Para realizar esta comunicación se hace uso de unos *Web Services* creados. Estos *Web Services* obtienen la petición por parte del cliente, se comunican con la base de datos, obtienen la información requerida por el cliente, transforman los datos a formato *JSON* y envían esos datos al cliente.

En la Ilustración 27 se observa el funcionamiento, de forma general, de la comunicación entre nuestro cliente y el servidor. (Potencier & Weaver, 2012)

Para poder realizar todas las comunicaciones necesarias se han desarrollado una serie de *Web Services* y se utilizarán unos u otros dependiendo de los datos que queramos obtener. Por ejemplo, se ha creado uno para el inicio de sesión del usuario, otro para la obtención todos los seguimientos de un tutelado… todos ellos escritos en *php*.

CAPÍTULO 4

4 MANUAL DE USUARIO

Una vez descrita la parte técnica de desarrollo e implementación de la aplicación, se va a explicar el funcionamiento de la misma.

La aplicación ofrece una serie de juegos diseñados específicamente para que ayuden a la rehabilitación cognitiva de aquellas personas que lo necesiten. También ofrece herramientas para que los usuarios/voluntarios puedan llevar un seguimiento y observar las mejoras que van realizando mediante gráficas asociadas a los distintos parámetros del juego.

A continuación, se presenta el manual de usuario que contiene todos los usos que ofrece la aplicación e indica a los usuarios como tienen que utilizarla.

Cabe destacar que la aplicación está disponible en castellano e inglés.

4.1 PANTALLA INICIAL

Esta pantalla inicial consta de 2 campo de texto para que el usuario introduzca su e-mail y su contraseña para autentificarse en el sistema y poder acceder al resto de la aplicación. Si los datos son incorrectos se le notifica al usuario mediante un *toast* (mensaje que se muestra en la pantalla). En la Ilustración 28 se muestra la pantalla de inicio.

En caso de que el dispositivo no disponga de conexión a Internet, se le notifica al usuario mediante un *toast* y éste podrá acceder a la aplicación con una cuenta por defecto creada utilizando las preferencias que ofrece *Android* para que se pueda acceder a la aplicación. La cuenta por defecto es email `admin@admin.es` y la contraseña `admin`.

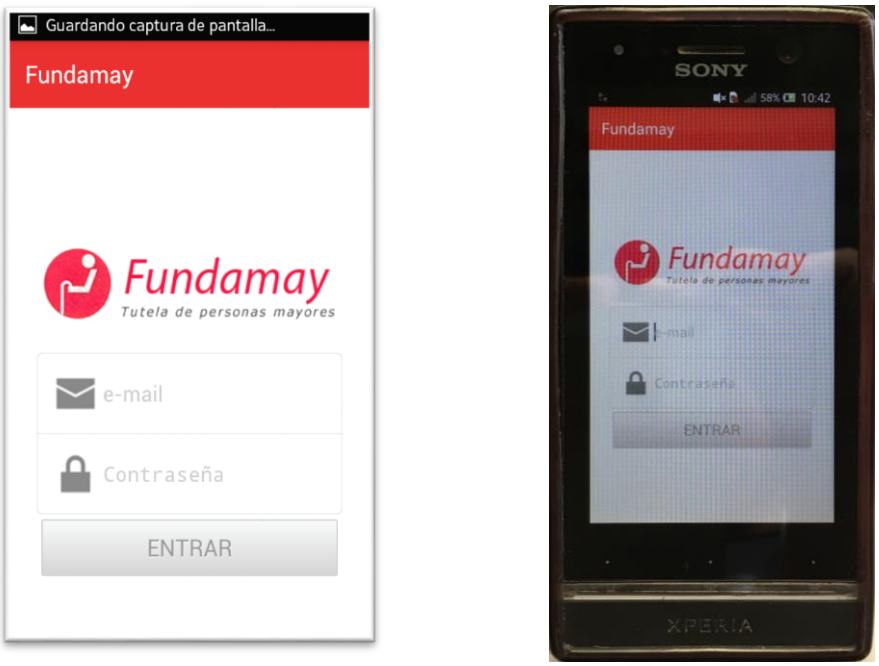

Ilustración 28. *Pantalla de inicio*

4.2 PANTALLA PRINCIPAL

En esta pantalla se muestran las opciones que el usuario puede elegir. Estas opciones son las siguientes: seguimientos, usuarios, juegos y estadísticas.

Estas opciones se muestran en distintos botones para que el usuario pulse la opción que desee ver. Esta pantalla también dispone de un menú que ofrece otras funcionalidades como la elección del nivel de dificultad de los juegos. Todas las opciones que dispone el menú se detalla en la sección correspondiente.

En la ilustración 29 se muestra esta pantalla principal.

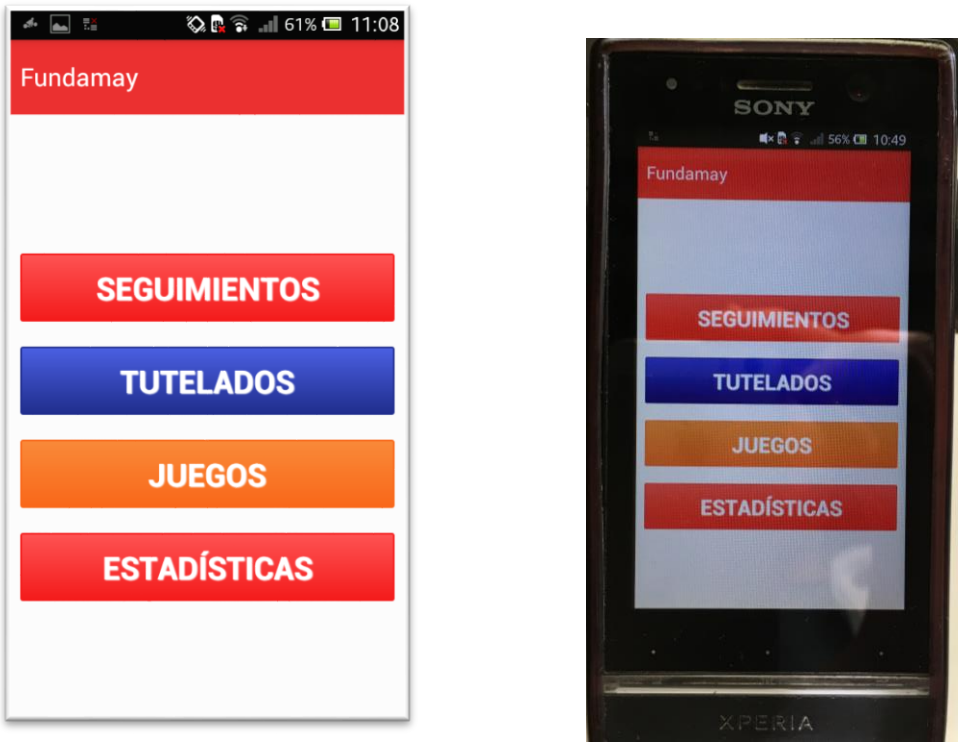

Ilustración 29. Pantalla principal

4.3 PANTALLA MOSTRAR TUTELADOS

Esta pantalla se muestra una vez que el usuario ha pulsado la opción `seguimientos`, la opción `juegos` o la opción `estadísticas`. En ella se muestran todos los voluntarios que el usuario tiene a su cargo. En caso de que este voluntario no tenga ningún tutelado a su cargo, se le notifica en la pantalla.

Los usuarios en un principio se obtienen de la base de datos. En caso de no disponer de conexión a internet en el dispositivo, se muestran los usuarios de ejemplo de la base de datos local y se notifica al usuario indicando que estos son tutelados de ejemplo para mostrar las funcionalidades que ofrece.

En la ilustración 30 se muestra esta pantalla.

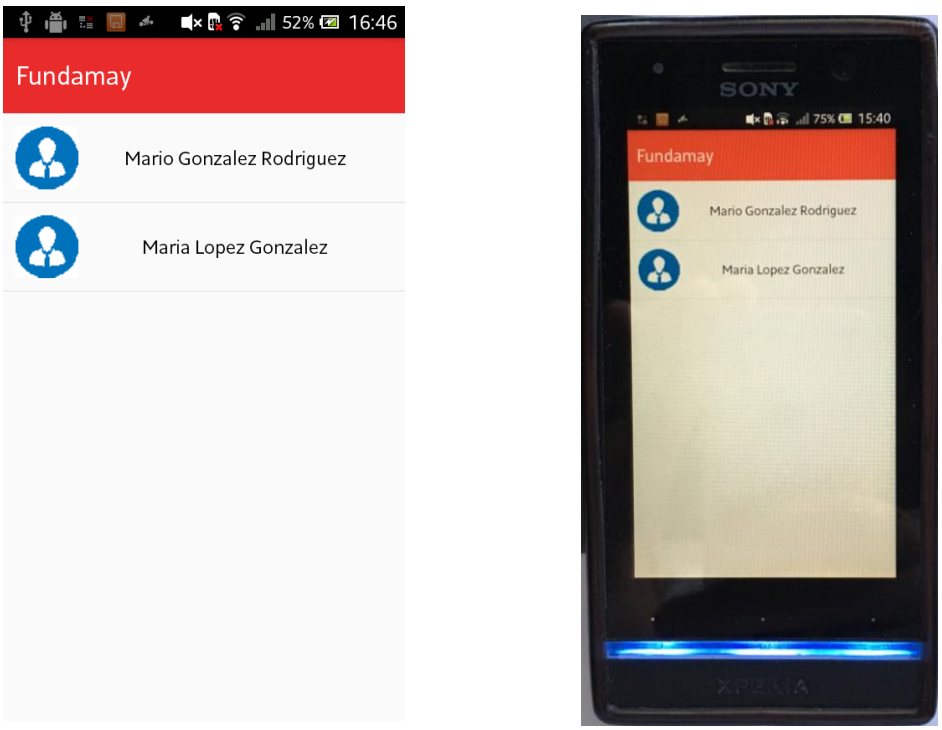

Ilustración 30. *Pantalla Mostrar Tutelados*

4.4 PANTALLA NUEVO SEGUIMIENTO

Una vez que se selecciona el tutelado en la pantalla `Mostrar Tutelados` se crea un diálogo con las opciones `nuevo seguimiento` y `ver seguimientos` como se muestra en la Ilustración 31.

Esta pantalla se muestra si el usuario pulsa en la opción nuevo seguimiento. Esta pantalla permite crear un nuevo seguimiento. El usuario tiene que cumplimentar todos los campos que se muestran y, una vez cumplimentados y, si el dispositivo posee conexión a internet, se envían los datos al servidor. Una vez que se obtiene la respuesta del servidor se muestra un mensaje indicando como ha ido la operación (correcto o error).

En caso de que el usuario no introduzca todos los datos o no disponga de conexión a internet para poderlos enviar al servidor también se le notificará mediante un mensaje.

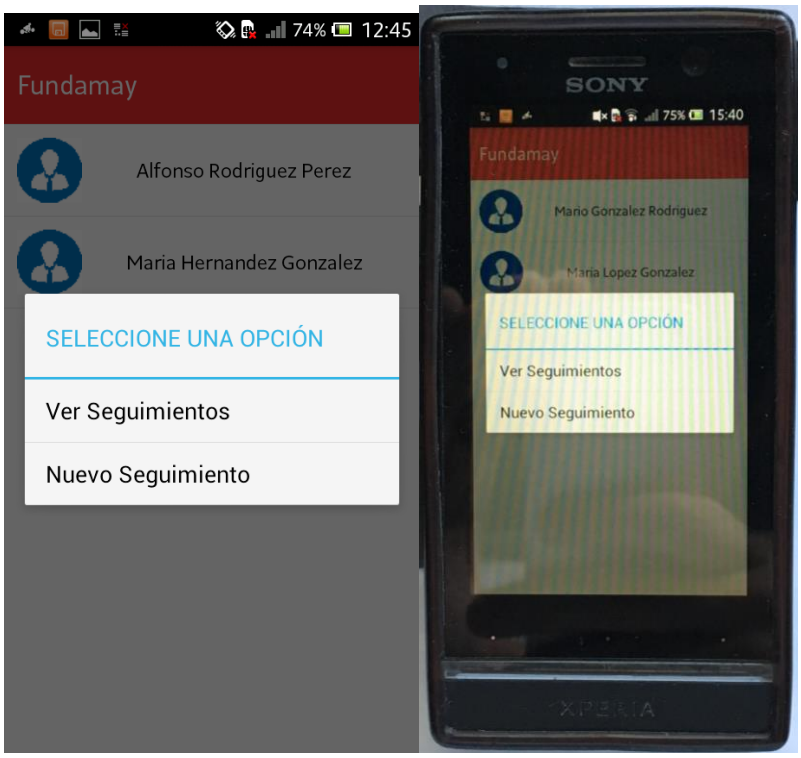

Ilustración 31. *Dialogo de selección en la opción Seguimientos*

En la Ilustración 32 se muestra la interfaz para insertar un nuevo seguimiento.

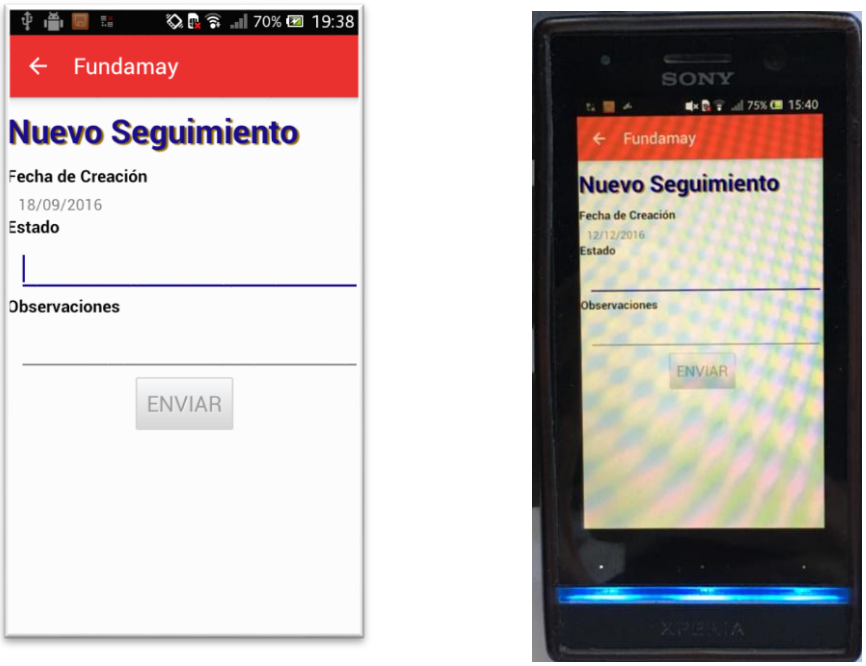

Ilustración 32. *Pantalla nuevo seguimiento*

4.5 PANTALLA VER SEGUIMIENTOS

En esta pantalla se muestran un resumen de los seguimientos que ese tutelado tiene. La interfaz es similar a la de `Mostrar Tutelados` como se muestra en la Ilustración 33.

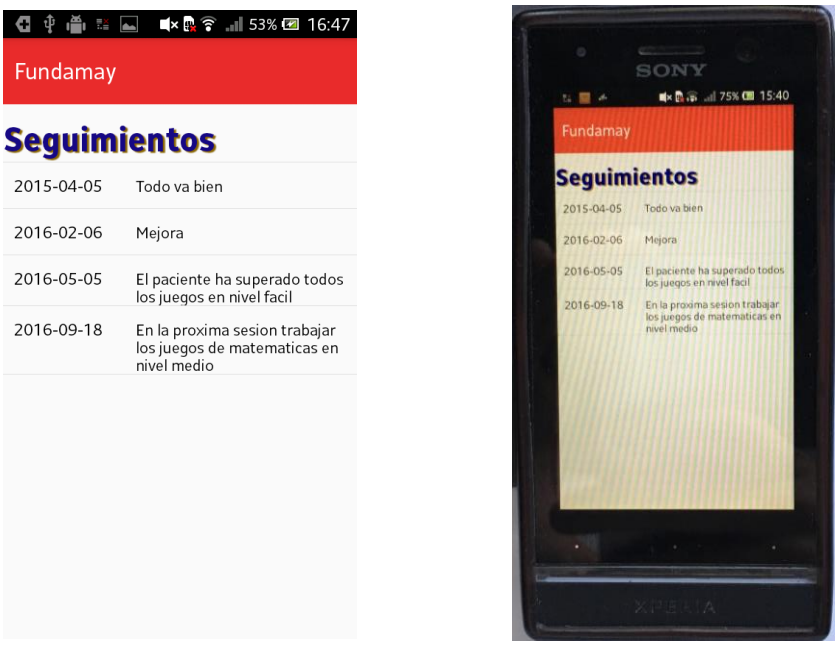

Ilustración 33. *Pantalla ver seguimientos*

Si se pulsa sobre un seguimiento, se crea un diálogo que permite ver los detalles de ese seguimiento como se puede observar en la Ilustración 34.

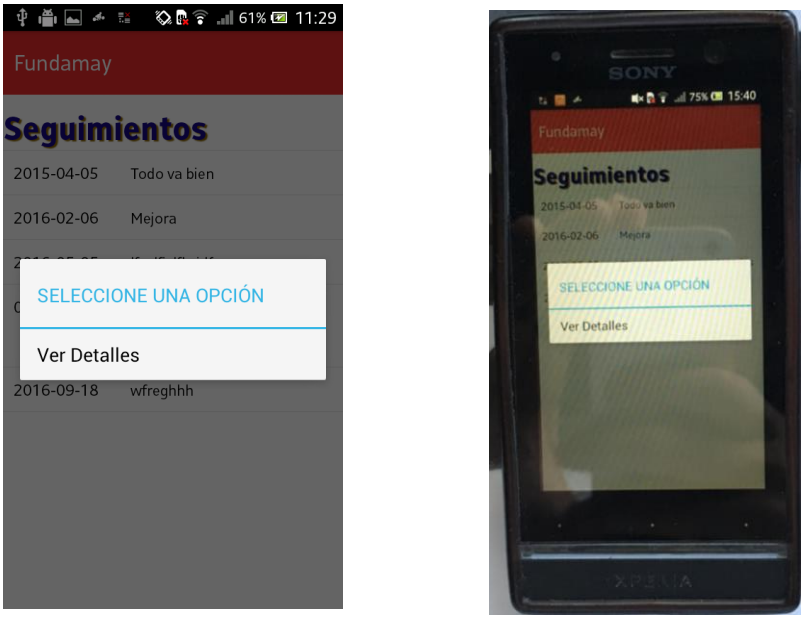

Ilustración 34. *Diálogo para ver los detalles de un seguimiento*

4.6 PANTALLA DETALLES SEGUIMIENTO

En esta pantalla se muestran todos los detalles relacionados en el seguimiento seleccionado como se muestra en la Ilustración 35.

Ilustración 35. *Pantalla detalles seguimiento*

4.7 PANTALLA NUEVO TUTELADO

Una vez descrito el funcionamiento de la opción `seguimientos` se va a explicar el funcionamiento de la opción `tutelados`.

Si el usuario pulsa la opción `tutelados` se muestra un diálogo con las opciones nuevo tutelado y ver tutelados como se muestra en la Ilustración 36.

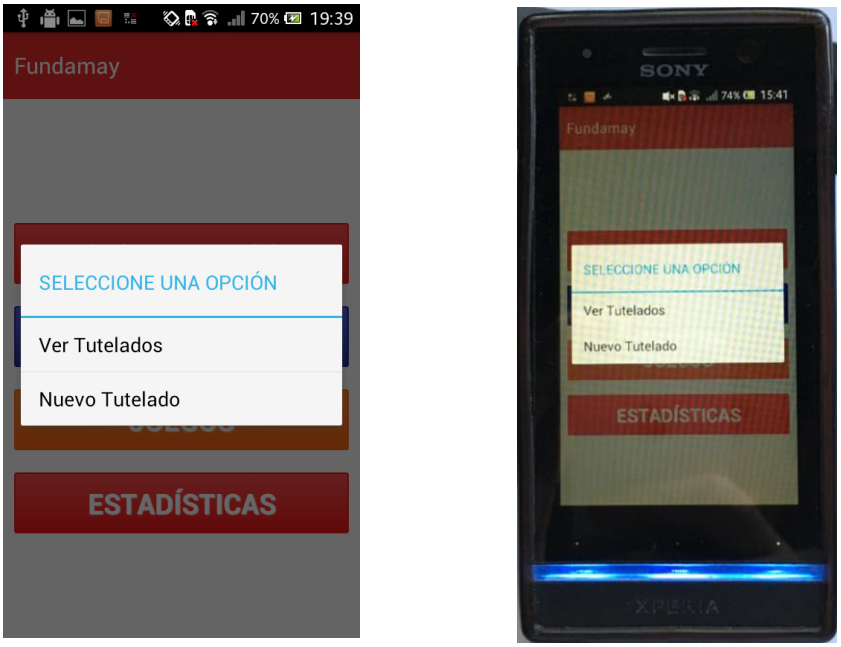

Ilustración 36. Dialogo con las opciones de tutelados

En la pantalla de nuevo tutelado se muestra un formulario que el usuario tiene que cumplimentar con todos los datos que se le piden. Si no introduce todos los datos se le notifica mediante un mensaje en la pantalla.

Una vez que el usuario ha introducido todos los datos se comprueba si el dispositivo tiene acceso a internet y, en caso afirmativo, se envían los datos al servidor y se notifica al usuario como ha ido la operación (correcto o error) mediante un mensaje en la pantalla. En caso de que no se disponga de conexión en el dispositivo, el tutelado no se almacenará en local y se notifica al usuario mediante un mensaje en la pantalla. En la Ilustración 37 se muestra como es la pantalla de nuevo tutelado.

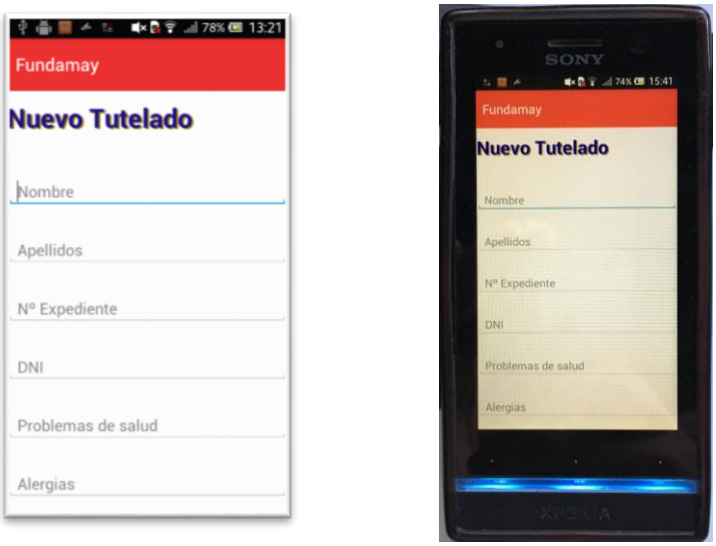

Ilustración 37. *Pantalla nuevo tutelado*

4.8 PANTALLA VER TUTELADOS

Si el usuario pulsa la opción ver tutelados del diálogo mencionado anteriormente se muestra la pantalla ver tutelados. En esta pantalla se muestran los tutelados que ese usuario tiene a su cargo. Si no dispone de ningún tutelado se muestra un mensaje indicándolo.

En caso de que el dispositivo no disponga de conexión a internet, se muestra un mensaje en la pantalla y no se muestra ningún tutelado, puesto que esta opción es para los tutelados que realmente tiene asignados.

Una vez que se muestran los tutelados, si el usuario pulsa sobre uno de ellos, aparece un diálogo con la opción ver detalles que se utiliza para mostrar los detalles de ese tutelado.

En la Ilustración 38 se muestra el diálogo que se crea cuando el usuario pulsa sobre un tutelado. Si pulsa sobre la opción ver detalles, se carga la pantalla detalles tutelado que se analiza en la sección siguiente.

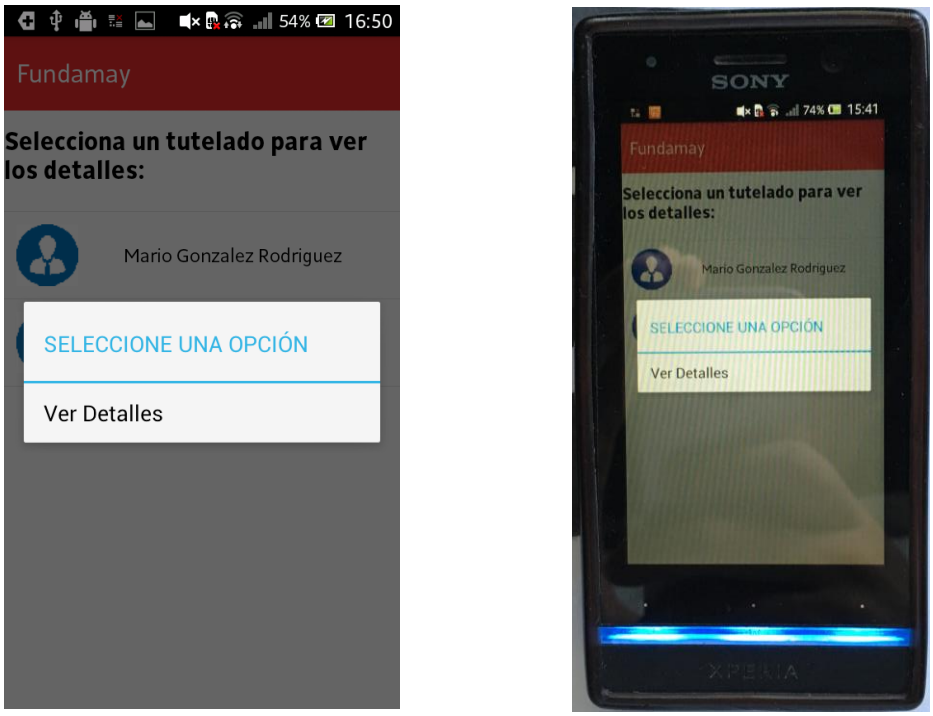

Ilustración 38. *Pantalla ver tutelados*

4.9 PANTALLA DETALLES TUTELADO

Esta pantalla muestra todos los detalles relacionados con el tutelado seleccionado por el usuario. Aporta datos de interés como el domicilio, el número de teléfono, alergias entre otros datos.

En la Ilustración 39 se muestra la pantalla de detalles tutelado con los datos que se ofrecen del tutelado.

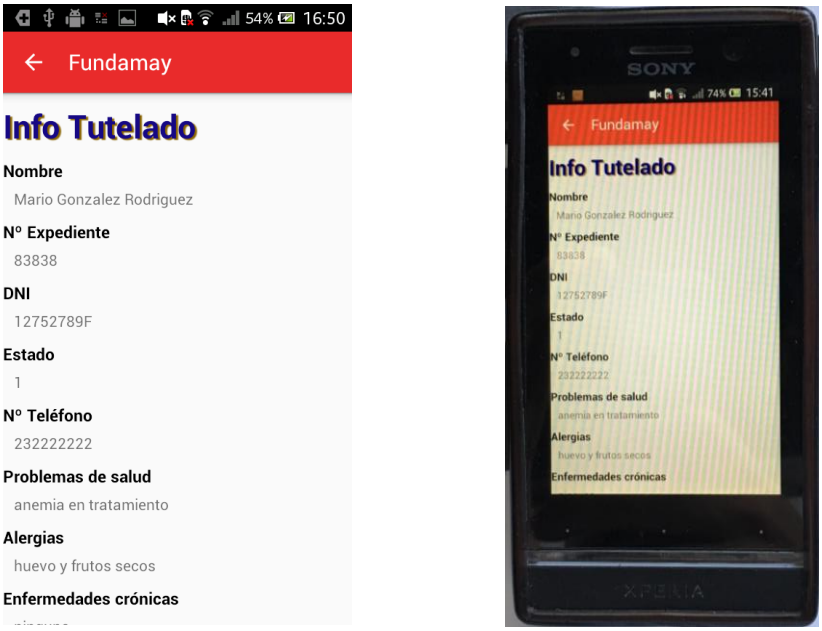

Ilustración 39. *Pantalla detalles tutelado*

4.10 PANTALLA ESTADÍSTICAS

La pantalla de estadísticas se muestra una vez que el usuario pulsa la opción `estadísticas` que aparece en la pantalla principal.

En esta pantalla se muestran todos los tutelados que tiene a su cargo ese usuario. Esta pantalla es igual a la de `Mostrar Tutelados` y contiene la misma información. Una vez que el usuario pulsa sobre un tutelado se carga la pantalla `Seleccionar Estadísticas`.

En la Ilustración 40 se muestra esta pantalla.

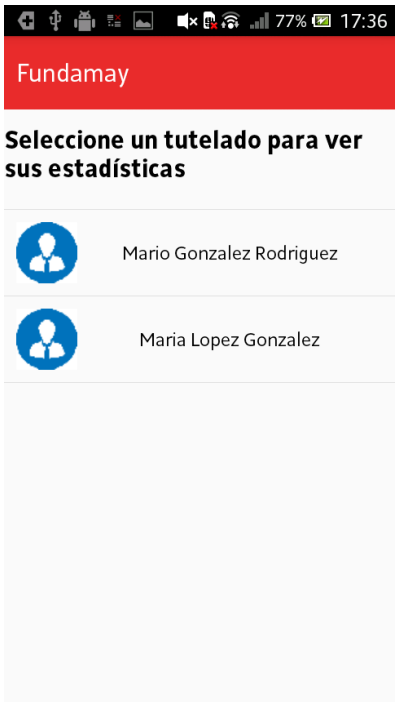

Ilustración 40. *Pantalla estadísticas*

4.11 PANTALLA SELECCIONAR ESTADÍSTICAS

En esta pantalla el usuario selecciona las estadísticas que desea visualizar. Para ello el usuario tiene que introducir los campos que se le indican.

Los campos a introducir son el nombre del juego del que se quieren obtener las estadísticas, el tipo de estadística que se quiere visualizar (tiempo, respuestas correctas, respuestas incorrectas e intentos), el nivel de dificultad del juego (fácil, medio y difícil) y el número de partidas que se desea visualizar. En caso de querer mostrar todas, basta con dejar el campo vacío.

Una vez que el usuario a introducido todos los datos se muestra la gráfica con las estadísticas obtenidas en la pantalla `Ver Estadísticas`.

En la Ilustración 41 se muestra esta pantalla.

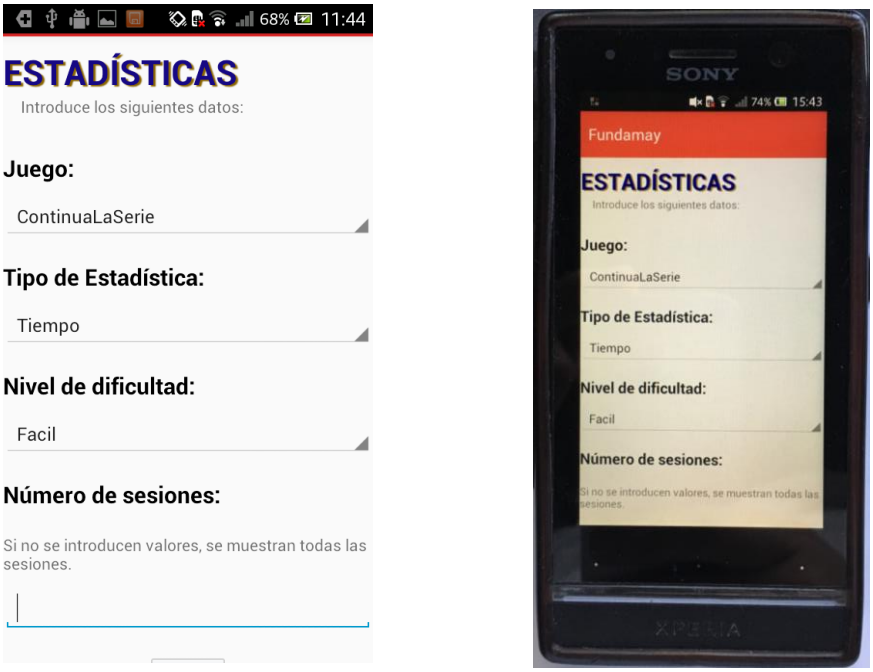

Ilustración 41. *Pantalla seleccionar estadísticas*

4.12 PANTALLA VER ESTADÍSTICAS

En esta pantalla se muestra una gráfica con los resultados obtenidos. En la Ilustración 42 se puede observar un ejemplo de ello.

Ilustración 42. *Pantalla ver estadísticas*

4.13 PANTALLA JUEGOS

La pantalla juegos se muestra una vez que el usuario a pulsado la opción `juegos` de la pantalla principal. Esta pantalla es similar a la pantalla estadísticas como se puede observar en la Ilustración 43.

En esta pantalla se muestran los tutelados que el usuario tiene a su cargo. En caso de no disponer de conexión a internet en el dispositivo, se obtienen un par de tutelados de ejemplo para que se puedan ver los juegos. En caso de obtener estos tutelados de ejemplo se muestra un mensaje en la pantalla avisando al usuario.

Si cuando termine el juego sigue sin disponer de conexión a internet, los resultados de la partida no se almacenan en ningún sitio (ni en el servidor ni en local).

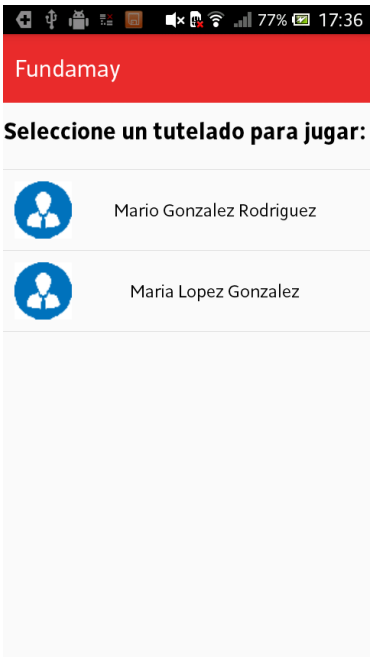

Ilustración 43. *Pantalla juegos*

4.14 PANTALLA CATEGORÍA JUEGOS

Una vez que el usuario ha seleccionado el tutelado que va a jugar se carga esta pantalla. En esta pantalla aparecen todas las categorías de juego disponibles.

Las categorías disponibles son las siguientes: cálculo, lenguaje, razonamiento, orientación, atención, percepción y memoria.

En la Ilustración 44 se muestran los botones de esta pantalla (uno por categoría) y, además hay un botón en la parte superior izquierda para volver a la pantalla anterior.

Ilustración 44. *Pantalla categoría juegos*

4.15 PANTALLA JUEGOS MEMORIA

Esta pantalla aparece una vez que usuario selecciona la categoría de memoria. En esta pantalla se muestran todos los juegos disponibles para esta categoría. Actualmente solo existe un juego para esa categoría como se muestra en la Ilustración 45.

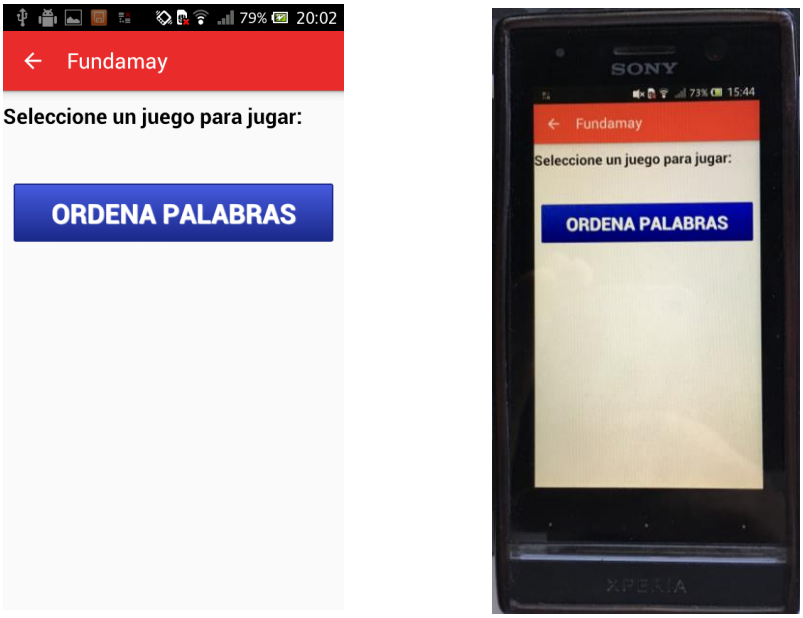

Ilustración 45. *Pantalla juegos memoria*

4.16 PANTALLA JUEGOS PERCEPCIÓN

Esta pantalla aparece una vez que usuario selecciona la categoría de percepción. En esta pantalla se muestran todos los juegos disponibles para esta categoría. Actualmente están disponibles los juegos `Imágenes Iguales` y `Formas` como se muestra en la Ilustración 46.

Ilustración 46. Pantalla juegos percepción

4.17 PANTALLA JUEGOS CÁLCULO

Esta pantalla aparece una vez que usuario selecciona la categoría de cálculo. En esta pantalla se muestran todos los juegos disponibles para esta categoría. Actualmente están disponibles los juegos `Matemáticas` y `Da el Cambio` como se muestra en la Ilustración 47.

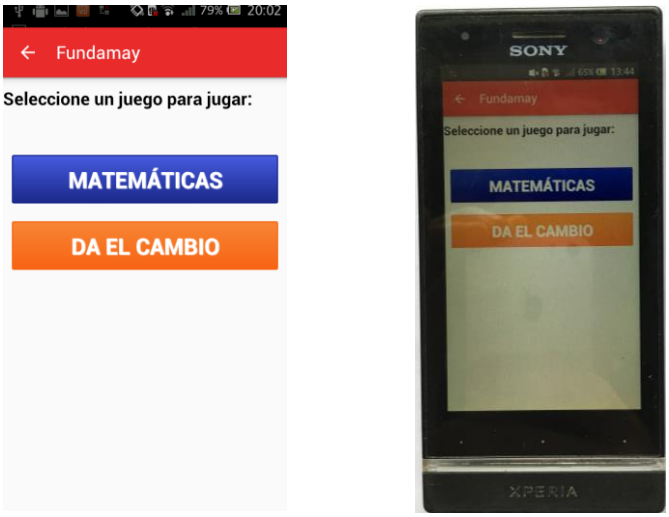

Ilustración 47. *Pantalla juegos cálculo*

4.18 PANTALLA JUEGOS LENGUAJE

Esta pantalla aparece una vez que usuario selecciona la categoría de lenguaje. En esta pantalla se muestran todos los juegos disponibles para esta categoría. Actualmente está disponible el juego `Nombrar Imágenes` como se muestra en la Ilustración 48.

Ilustración 48. *Pantalla juegos lenguaje*

4.19 PANTALLA JUEGOS RAZONAMIENTO

Esta pantalla aparece una vez que usuario selecciona la categoría de razonamiento. En esta pantalla se muestran todos los juegos disponibles para esta categoría. Actualmente están disponibles los juegos `Festividades` y `Continúa la Serie` como se muestra en la Ilustración 49.

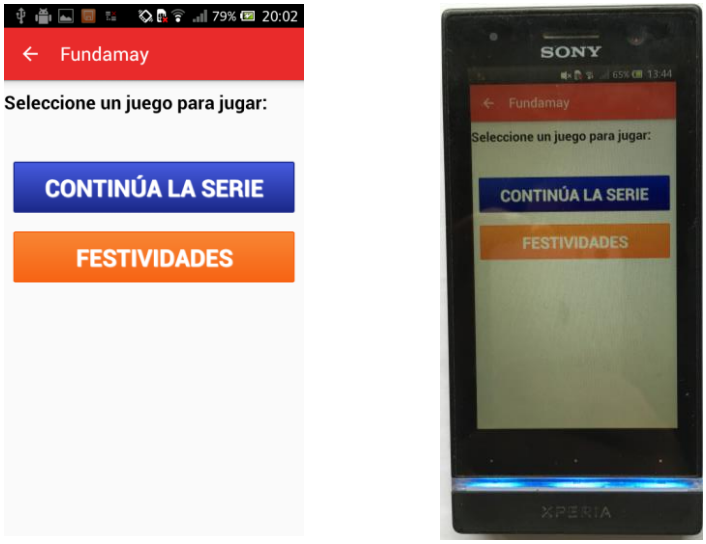

Ilustración 49. *Pantalla juegos razonamiento*

4.20 PANTALLA JUEGOS ORIENTACIÓN

Esta pantalla aparece una vez que usuario selecciona la categoría de orientación. En esta pantalla se muestran todos los juegos disponibles para esta categoría. Actualmente está disponible el juego `Laberintos` como se muestra en la Ilustración 50.

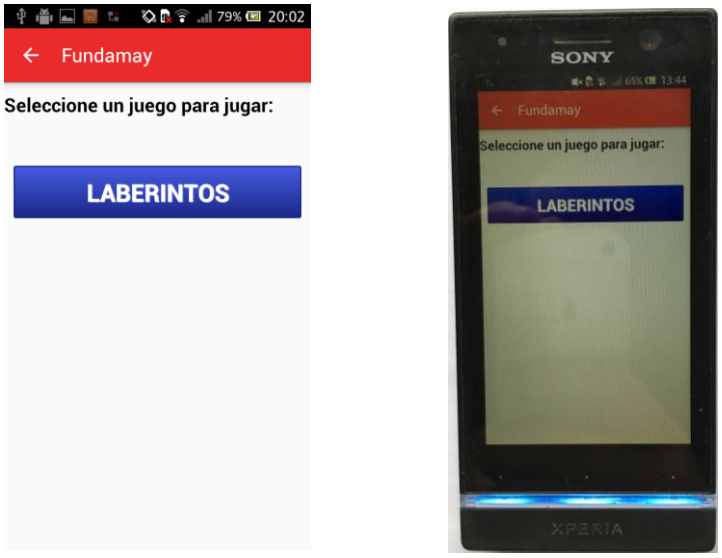

Ilustración 50. *Pantalla juegos orientación*

4.21 PANTALLA JUEGOS ATENCIÓN

Esta pantalla aparece una vez que usuario selecciona la categoría de atención. En esta pantalla se muestran todos los juegos disponibles para esta categoría. Actualmente está disponible el juego `Objeto Diferente` como se muestra en la Ilustración 51.

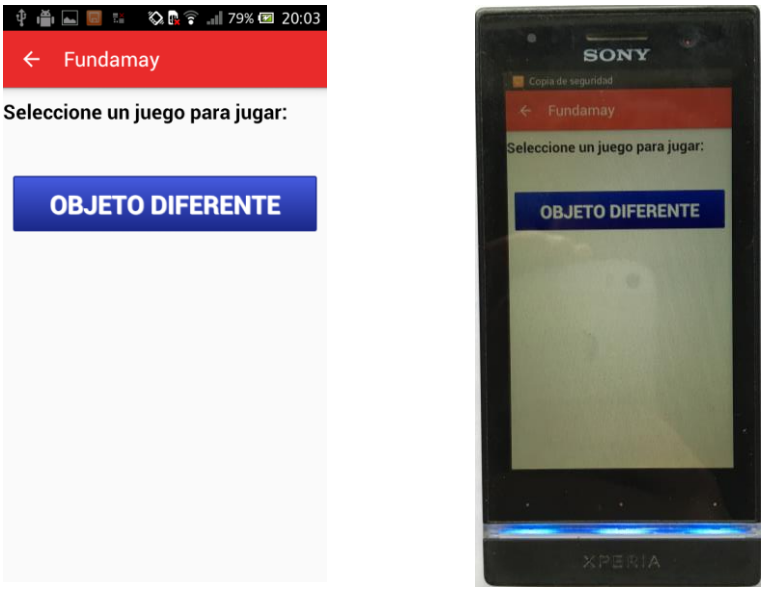

Ilustración 51. *Pantalla juegos atención*

4.22 PANTALLA JUEGO *ORDENA PALABRAS*

Esta pantalla aparece cuando se pulsa sobre dicho juego en la categoría de memoria. El juego consiste en mostrar cada palabra durante un par de segundos y luego ordenar esas palabras por orden de aparición. En la Ilustración 52 se muestra la interfaz de inicio del juego. Dependiendo del nivel de dificultad en el que se esté jugando, aparecen más o menos palabras (4 palabras nivel fácil, 6 palabras nivel medio y 10 palabras nivel difícil). Una vez que el jugador esté preparado se pulsa el botón `Empezar` para que se empiecen a mostrar las palabras.

Ilustración 52. *Pantalla de inicio del juego Ordena Palabras*

Una vez que el jugador ha pulsado el botón `Empezar` comienzan a salir palabras aleatorias. Una vez que han salido todas las palabras se muestran y el jugador introduce la posición en la que han aparecido. En la Ilustración 53 se muestra el funcionamiento descrito.

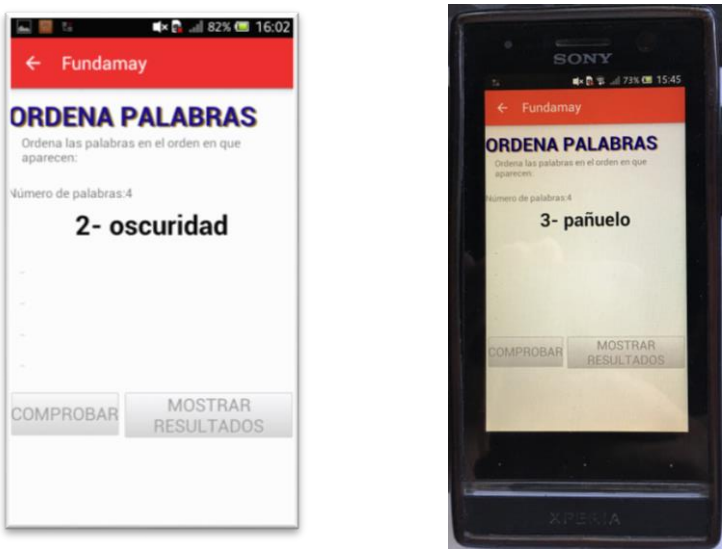

Ilustración 53. *Pantalla que muestra las palabras del juego Ordena Palabras*

Una vez que el usuario ha introducido la posición, tiene que pulsar el botón comprobar para ver los aciertos y los fallos que tiene como se muestra en la Ilustración 54. Si alguna opción está mal, se puede modificar y volver a pulsar el botón `Comprobar`. Esto se puede realizar tantas veces como se quiera, pero cada vez que se pulsa, el número de intentos se incrementa. Una vez que todas las respuestas son correctas o se quieran mostrar los resultados de la partida (se puede pulsar, aunque haya respuestas incorrectas) se pulsa el botón `Mostrar Resultados` y se carga la pantalla `Mostrar Resultados` que se describirá posteriormente.

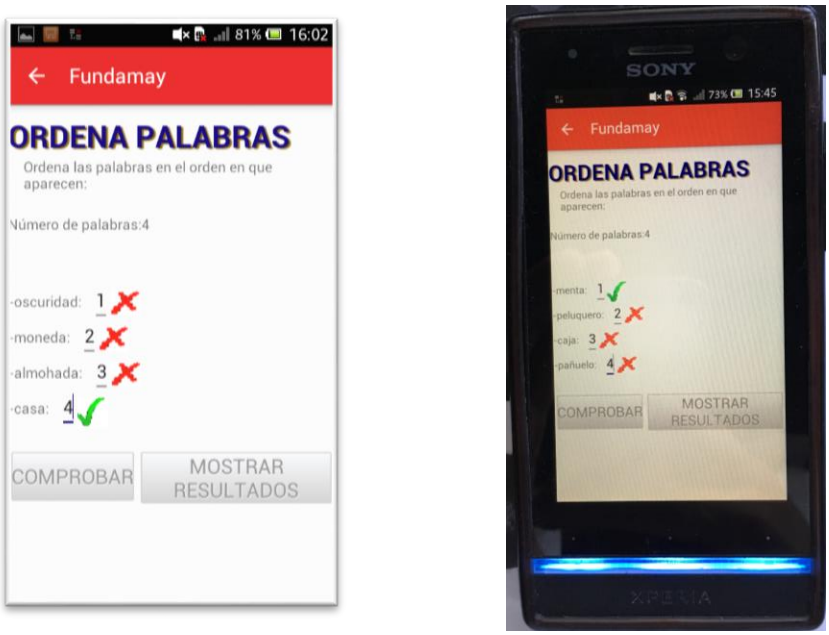

Ilustración 54. *Comprobar los resultados del juego Ordena Palabras*

4.23 PANTALLA JUEGO *IMÁGENES IGUALES*

Esta pantalla aparece cuando se pulsa sobre dicho juego en la categoría de percepción. El juego consiste en seleccionar todas las imágenes iguales a la que se muestra de ejemplo como se muestra en la Ilustración 55. Cuando el usuario selecciona una imagen se ponen los bordes de color verde para saber cual está marcada. Si vuelve a pulsar sobre ella se desmarca.

Una vez que el usuario ha seleccionado todas las imágenes que cree que son iguales tiene que pulsar el botón `Comprobar` como en el juego `Ordena Palabras`. Si son correctas se muestra un mensaje en la pantalla indicando que la solución es correcta. En caso de haber fallado alguna imagen se puede volver a intentar todas las veces que se quiera. Para ver los resultados tiene que pulsar el botón `Mostrar Resultados`.

Cuanto mayor es el nivel de dificultad del juego más imágenes posibles se muestran. En la Ilustración 55 el nivel del juego era fácil.

Ilustración 55. *Pantalla juego Imágenes Iguales*

4.24 PANTALLA JUEGO *FORMAS*

Esta pantalla aparece cuando se pulsa sobre dicho juego en la categoría de percepción. El juego consiste en indicar cuantas formas geométricas pedidas (pueden ser cuadrados, triángulos, círculos o rectángulos) hay en cada fila. Las figuras geométricas aparecen en distintos colores para mejorar la atención y la percepción del jugador. En la Ilustración 56 se muestra la pantalla de este juego en nivel fácil.

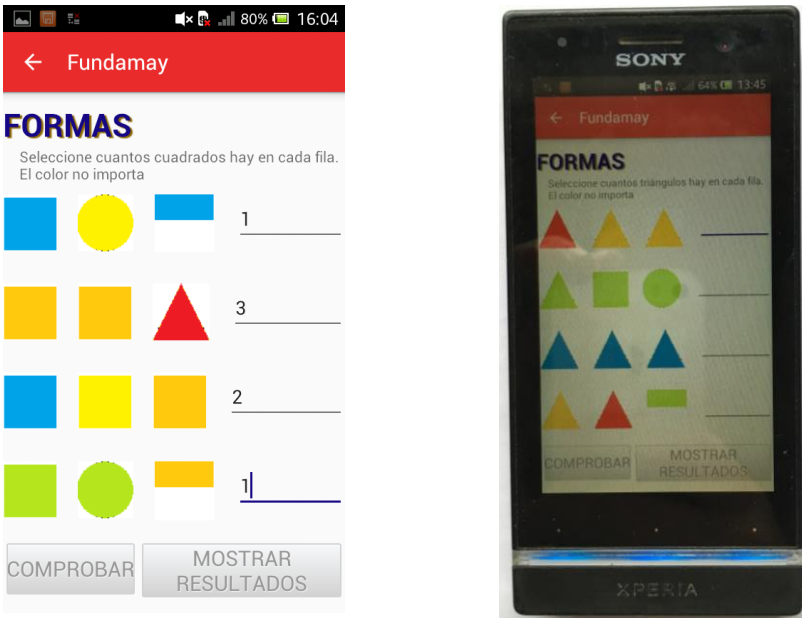

Ilustración 56. *Pantalla juego Formas*

Como en los juegos anteriores, cuanto mayor es la dificultad seleccionada, más filas aparecen y más opciones por fila aparecen.

Una vez que el usuario introduce todos los números en las filas tiene que pulsar el botón `Comprobar`. Si el usuario no introduce todos los datos se muestra un mensaje en la pantalla indicando que los introduzca todos. Una vez pulsado el botón, se notifica mediante un mensaje en la pantalla el número de filas incorrectas que se han introducido como se muestra en la Ilustración 57. En ningún caso se indican cuales están mal para que el juego sea más educativo. Como en los juegos anteriores, se puede pulsar el botón `Comprobar` las veces que se quiera. Una vez que el jugador no ha fallado ninguna fila puede pulsar el botón `Mostrar Resultado` para ver las estadísticas y datos de la partida jugada.

Ilustración 57. *Comprobación de las respuestas en el juego Formas*

4.25 PANTALLA JUEGO *MATEMÁTICAS*

Esta pantalla aparece cuando se pulsa sobre dicho juego en la categoría de cálculo. El juego consiste en resolver las operaciones aritméticas que se proponen como se muestra en la Ilustración 58. En el nivel fácil solo se incluyen sumas con números de un solo dígito. En el nivel medio ya se incluyen números de 2 dígitos con operaciones de suma y resta. Y en el nivel difícil se incluyen multiplicaciones a mayores de lo que ofrece le nivel medio.

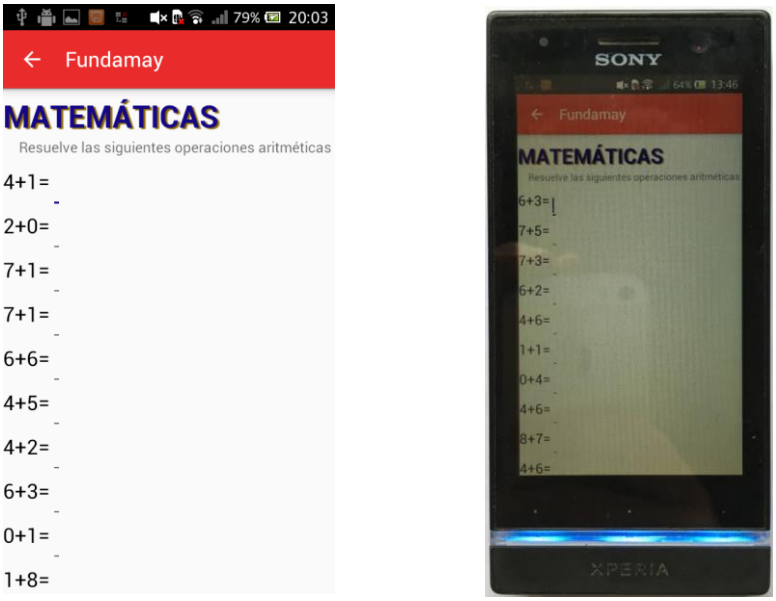

Ilustración 58. *Pantalla juego Matemáticas*

Una vez que el jugador ya ha introducido todos los resultados, se pulsa el botón `Comprobar` para ver si los resultados introducidos son correctos. Al lado de cada operación se muestra si es correcto el resultado o no como se muestra en la Ilustración 59. Las operaciones se pueden comprobar tantas veces como se desee. Para visualizar las estadísticas de la partida basta con pulsar el botón `Mostrar Resultados`. Este botón se puede pulsar, aunque haya respuestas erróneas, pero no es lo ideal.

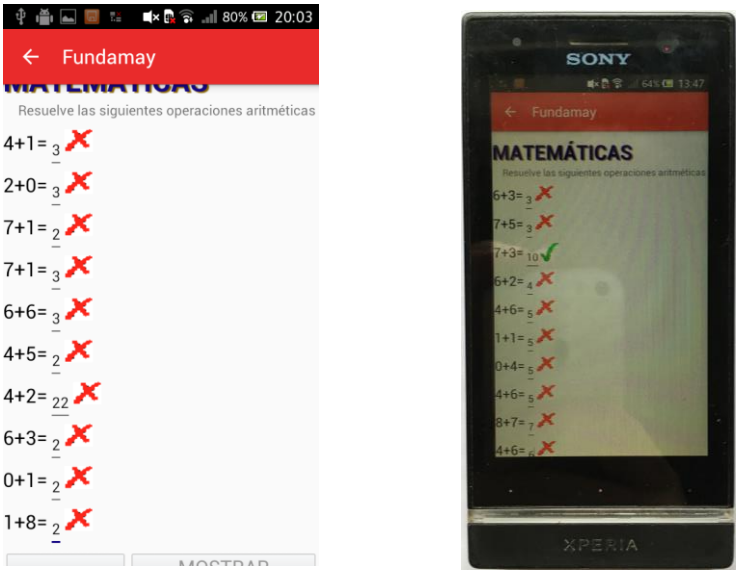

Ilustración 59. *Comprobación de los resultados en el juego Matemáticas*

4.26 PANTALLA JUEGO *DA EL CAMBIO*

Esta pantalla aparece cuando se pulsa sobre dicho juego en la categoría de cálculo. El juego consiste en dar el cambio correcto. En la Ilustración 60 se muestra la pantalla del juego. Cada moneda es un botón que suma la cantidad correspondiente al cambio total a devolver. Dependiendo del nivel de dificultad el billete del "pago" es de un valor. En el nivel fácil el billete es de 5€, en el nivel medio de 10€ y en el nivel difícil el billete es de 20€. También se dispone de un botón de *reset* en caso que el cambio se introduzca mal se pueda poner a 0 para volver a empezar.

Ilustración 60. *Pantalla juego Da El Cambio*

Una vez que el jugador ha introducido el cambio pulsando las monedas correspondientes, tiene que pulsar el botón `Comprobar` para ver si es correcto. El resultado se le notificará mediante un mensaje en la pantalla como muestra la Ilustración 61.

En caso de ser incorrecto lo puede volver a intentar pulsando el botón de *reset* e introduciendo otra vez el cambio. Este se puede repetir las veces que se desee. Una vez que el cambio sea correcto puede pulsar el botón `Mostrar Resultados` para ver las estadísticas de la partida jugada.

Ilustración 61. *Comprobación de los resultados en el juego Da El Cambio*

4.27 PANTALLA JUEGO *NOMBRAR IMÁGENES*

Esta pantalla aparece cuando se pulsa sobre dicho juego en la categoría de lenguaje. El juego consiste en introducir el nombre de los objetos que se muestran en las imágenes como se muestra en la Ilustración 62. Cuanto mayor es el nivel del juego, más imágenes aparecen y más difícil es el nombre de las mismas.

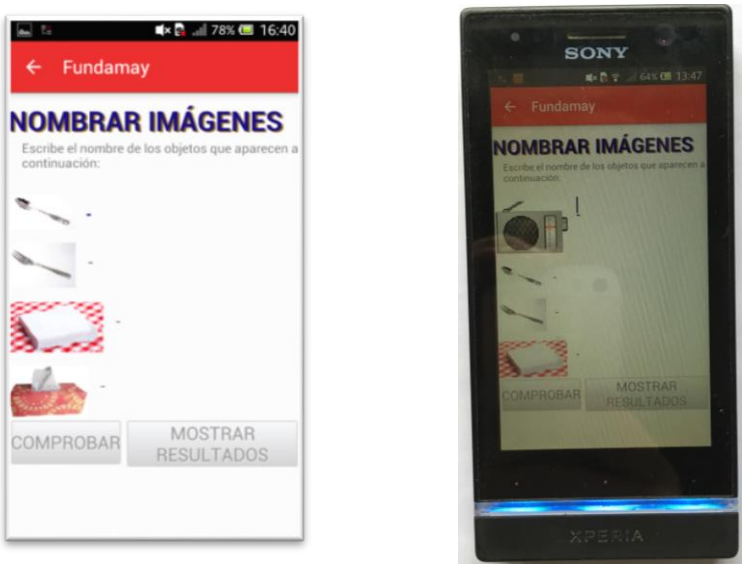

Ilustración 62. *Pantalla juego Nombrar Imágenes*

Como en el resto de los juegos, el jugador puede pulsar el botón `Comprobar` una vez que ha introducido todos los nombres de las imágenes para comprobar que son correctos. Si el nombre es correcto aparece un tic verde al lado de la palabra. En caso contrario, aparece una cruz roja. A mayores se muestra un mensaje en la pantalla indicando el número de objetos acertados correctamente como muestra la Ilustración 63. Una vez que todos los nombres son correctos, puede pulsar el botón `Mostrar Resultados` para ver las estadísticas de la partida.

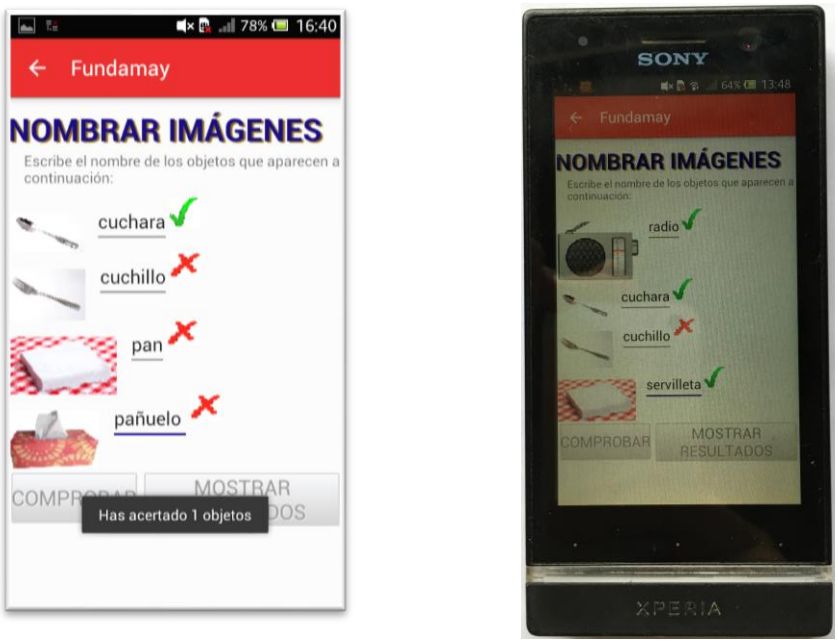

Ilustración 63. *Comprobación de los resultados en el juego Nombrar Imágenes*

4.28 PANTALLA JUEGO *CONTINÚA LA SERIE*

Esta pantalla aparece cuando se pulsa sobre dicho juego en la categoría de razonamiento. El juego consiste en adivinar el siguiente objeto de la serie que se muestra en la imagen superior. En la Ilustración 64 se muestra la pantalla de este juego. Las imágenes inferiores son las opciones disponibles para completar la serie. A medida que aumenta el nivel de dificultad del juego, las series son más variadas y más complejas.

Una vez que el jugador tiene clara su respuesta basta con que pulse sobre esa imagen. Si el fondo de la imagen se pone en rojo, indica que la respuesta es incorrecta. A mayores se muestra un mensaje en la pantalla indicando que la respuesta es errónea como se muestra en la Ilustración 65. En caso contrario, el fondo de la imagen se pone verde y se carga la pantalla `Mostrar Resultados` para ver las estadísticas de la pantalla de forma automática.

Ilustración 64. *Pantalla juego Continúa la Serie La Serie*

Ilustración 65. *Comprobación de los resultados en el juego Continúa*

4.29 PANTALLA JUEGO *FESTIVIDADES*

Esta pantalla aparece cuando se pulsa sobre dicho juego en la categoría de razonamiento. El juego consiste en acertar el día que se celebran las festividades propuestas (Navidad, Año Nuevo…) como se muestra en la Ilustración 66. Dependiendo del nivel de dificultad se muestran más o menos festividades. Cada festividad tiene 3 opciones posibles siendo solo una la correcta.

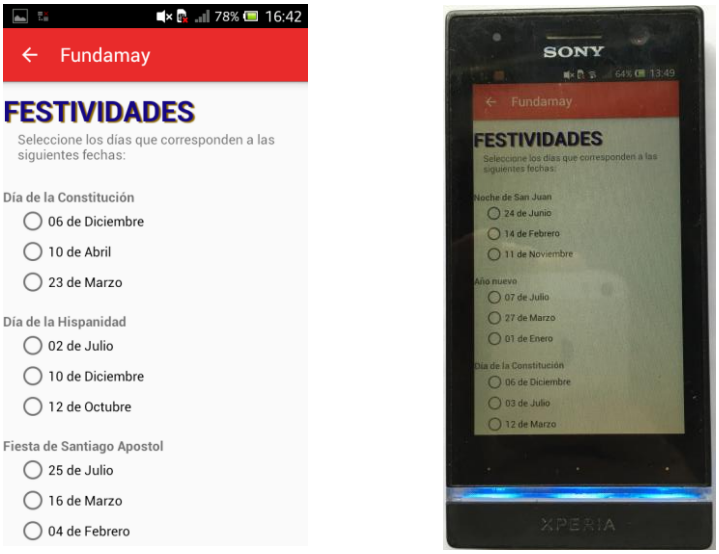

Ilustración 66. *Festividades Pantalla juego Festividades*

Una vez que el jugador ha completado todas las festividades tiene que pulsar el botón `Comprobar` para ver si los resultados son correctos. Si el resultado es correcto se muestra un tic verde al lado de la festividad acertada. En caso contrario, se muestra una X roja. Además, se muestra por pantalla un mensaje indicando el número de festividades acertadas como se muestra en la Ilustración 67. Una vez que todas las festividades han sido acertadas se pulsa el botón `Mostrar Resultados` para mostrar las estadísticas de la partida que se acaba de jugar.

Ilustración 67. *Comprobación de los resultados en el juego*

4.30 PANTALLA JUEGO *LABERINTOS*

Esta pantalla aparece cuando se pulsa sobre dicho juego en la categoría de orientación. El juego consiste en acertar que pieza de comida se va a comer el animal una vez que pase el laberinto como se muestra en la Ilustración 68. En el nivel fácil del juego el animal es un mono y solo tiene 2 trozos de fruta posibles que comerse. En los niveles medio y difícil el animal es un ratón y el número de trozos de queso posibles que se puede comer es mayor.

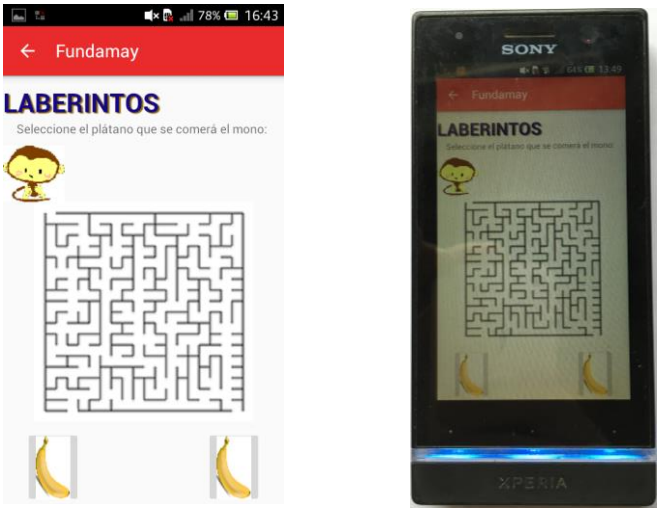

Ilustración 68. *Pantalla juego Laberintos*

El jugador tiene que seleccionar el trozo de comida que cree que se comerá el animal. Si la respuesta es incorrecta, el trozo de comida se pone en rojo y se muestra un mensaje en la pantalla indicando que la solución es errónea como se muestra en la Ilustración 69. En caso de ser correcta la respuesta, el trozo de comida se pone en verde y se carga la pantalla `Mostrar Resultados` para ver las estadísticas de la partida.

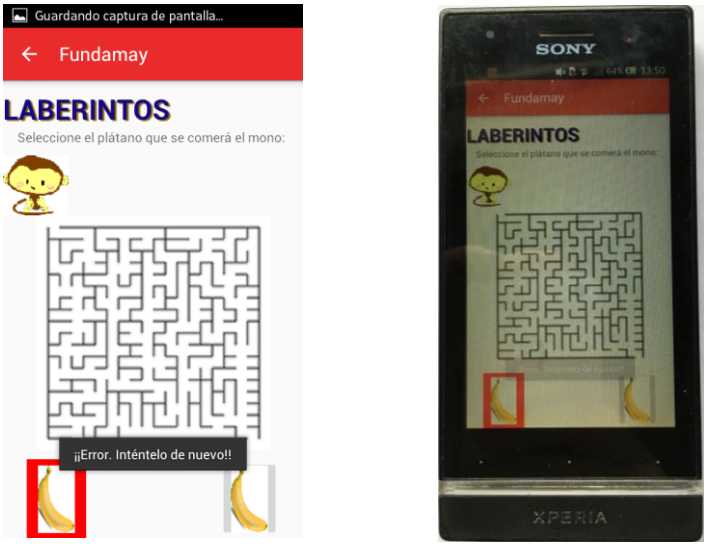

Ilustración 69. *Comprobación de los resultados en el juego Laberintos*

4.31 PANTALLA JUEGO *OBJETO DIFERENTE*

Esta pantalla aparece cuando se pulsa sobre dicho juego en la categoría de atención. El juego consiste en acertar la palabra que no se encuentra en la categoría del resto de palabras. Por ejemplo, si hay 3 palabras de la categoría lugares (isla, rio y desierto) y otro de la categoría comida (pan) la palabra "intrusa" sería la que pertenece a la categoría comida (pan). En la Ilustración 70 se muestra la pantalla de este juego. Cuanta mayor se la dificultad del juego, más ejercicios hay resolver.

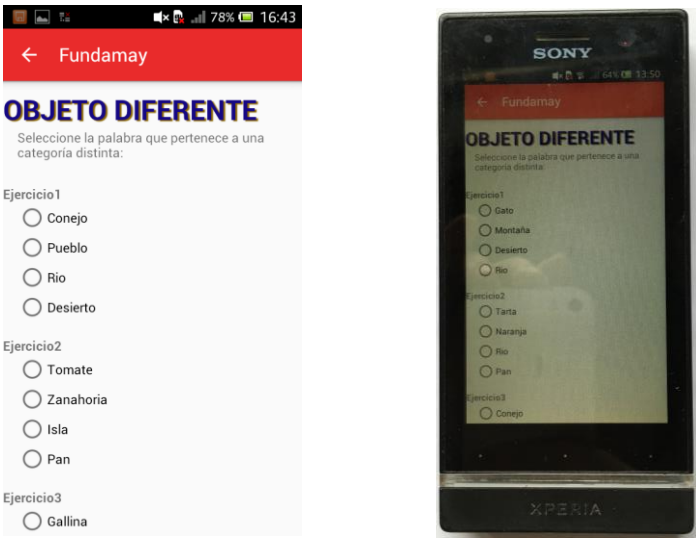

Ilustración 70. *Pantalla juego Objeto Diferente*

Una vez que el jugador ha completado todos los ejercicios propuestos, tiene que pulsar el botón `Comprobar` para ver si las respuestas son correctas o no. Al lado de cada ejercicio se indica si la solución es correcta o no. Si la solución es correcta se muestra un tic verde al lado del ejercicio. En caso contrario, se muestra una cruz roja. En la Ilustración 71 se muestra la comprobación de las respuestas explicada anteriormente.

Las soluciones se pueden comprobar tantas veces como sea necesario, aunque esto se verá reflejado en el número de intentos utilizados para completar con éxito el juego.

Una vez acertados todos los ejercicios, el jugador puede pulsar el botón `Mostrar Resultados` para visualizar todas las estadísticas de la partida que acaba de jugar en la pantalla `Mostrar Resultados`.

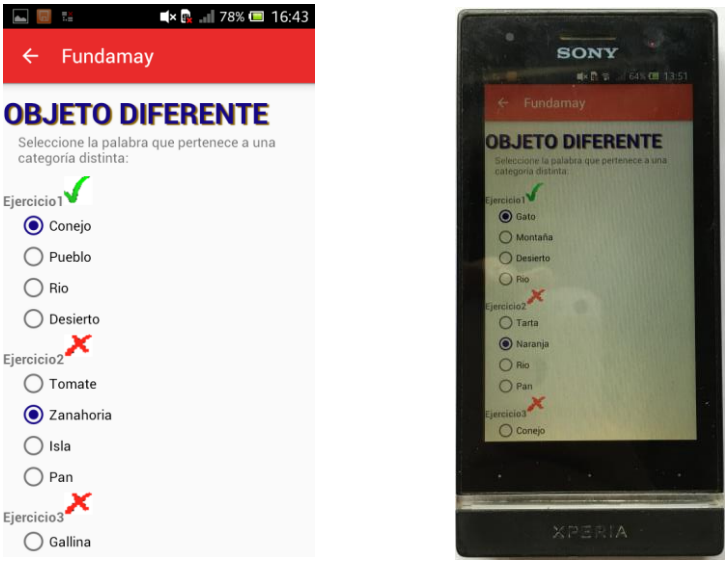

Ilustración 71. *Comprobación de los resultados en el juego Objeto Diferente*

4.32 PANTALLA MOSTRAR RESULTADOS

Esta pantalla se carga después de cada juego para mostrar las estadísticas de cada partida que se juega. En la Ilustración 72 se muestra la interfaz de esta pantalla. Las estadísticas que se muestran de la partida jugada son el número de intentos para conseguir acertar correctamente las pruebas del juego, el número de respuestas correctas, el número de respuestas incorrectas y el tiempo total de partida.

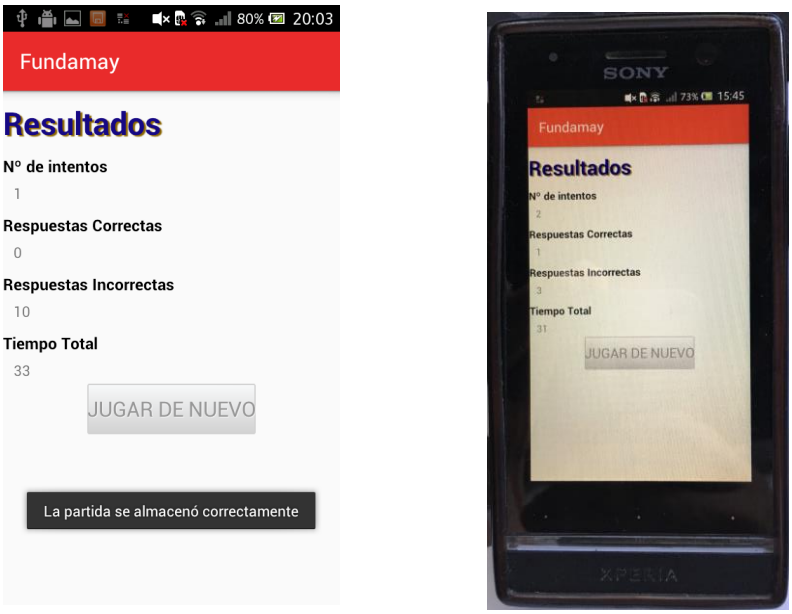

Ilustración 72. *Pantalla Mostrar Resultados*

Una vez que se carga esta pantalla, la aplicación intenta enviar estas estadísticas al servidor. En caso de no disponer de conexión a internet en el dispositivo, se muestra un mensaje en la pantalla y los datos de esa partida se perderían puesto que no se almacenan en local. Este proceso se muestra en la Ilustración 73. En caso de disponer de conexión a internet los datos de la partida se envían al servidor y se muestra un mensaje en la pantalla con el resultado de esta operación para informar al usuario. En la Ilustración 72 se muestra el mensaje indicando que la operación ha finalizado correctamente. En caso contrario, se mostraría un mensaje de error. Si se pulsa sobre el botón `Jugar de Nuevo` se carga de nuevo la partida jugada anteriormente.

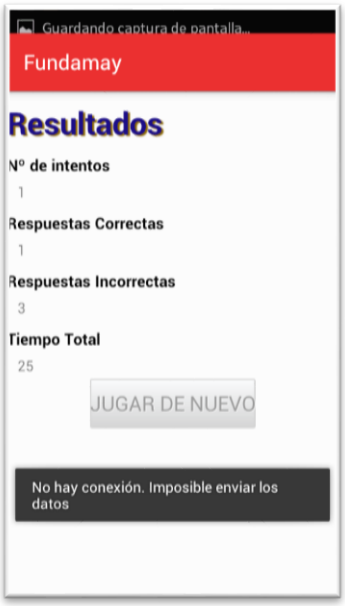

Ilustración 73. *Pantalla Mostrar Resultados sin conexión a internet*

4.33 PANTALLA MENÚ DE LA APLICACIÓN

Esta pantalla se carga cuando el usuario pulsa el botón menú de su terminal. En la Ilustración 74 se muestra el menú que dispone nuestra aplicación. Se pueden observar las distintas posibilidades que ofrece el menú. Desde cualquier pantalla anteriormente descrita se puede acceder directamente a la pantalla principal, a los seguimientos de un tutelado, a los tutelados disponibles para ese usuario y a los juegos que ofrece la aplicación. En definitiva, se puede acceder a todas las funcionalidades que ofrece la aplicación desde cualquier pantalla en la que nos encontremos.

También se puede configurar el nivel de dificultad de los juegos con las 3 dificultades disponibles (fácil, medio y difícil) como se muestra en la Ilustración 75.

Por último, dispone del botón `Salir`. Una vez que ya no queramos utilizar más la aplicación cierra la misma.

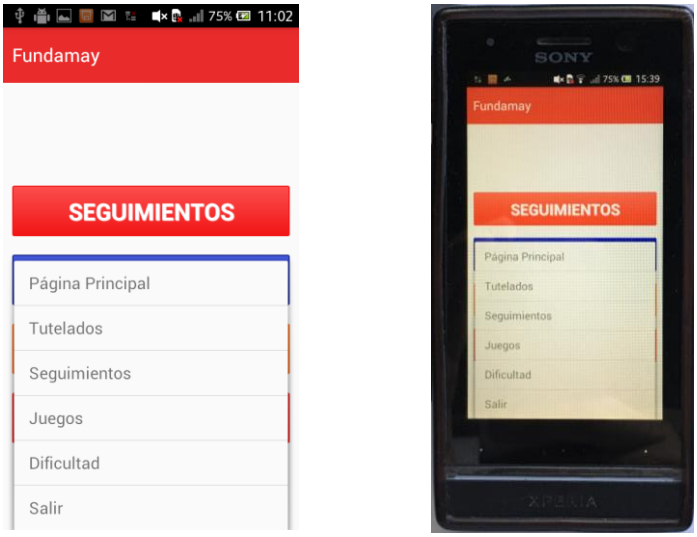

Ilustración 74. *Menú de la aplicación*

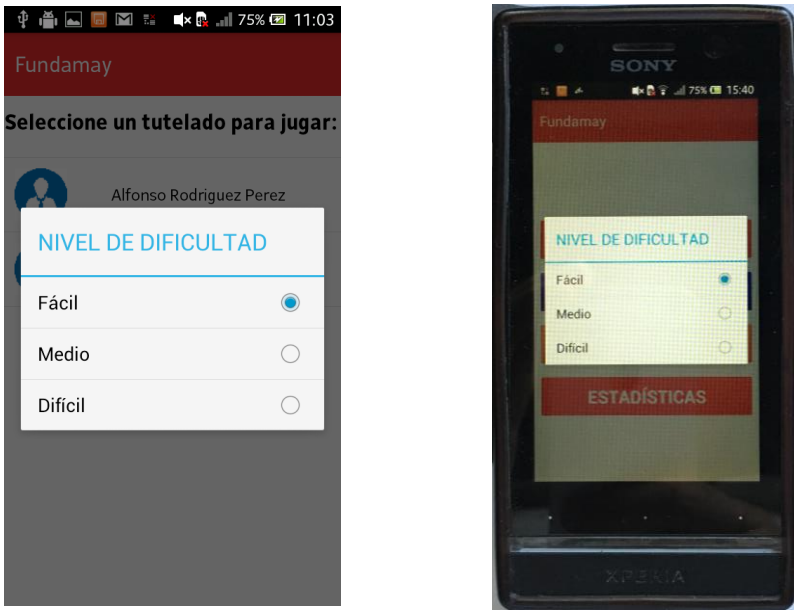

Ilustración 75. *Dificultad del nivel de los juegos*

4.34 REQUISITOS DEL SISTEMA

En cuanto a los requisitos del sistema, la aplicación requiere:

- Terminal Android 4.0 o superior.
- Conexión a internet bien mediante Wifi o bien utilizando la red móvil.
- Espacio disponible de 10 Mb o superior para instalar la aplicación.

CAPÍTULO 5

5 PRESUPUESTO ECONÓMICO

En este capítulo se estimará un presupuesto económico del proyecto realizado. A través de las diferentes fases seguidas en la realización de este proyecto, se irán incluyendo los diferentes gastos generados tanto en el desarrollo de software como el de hardware utilizado.

Antes de comenzar con el cálculo aproximado de presupuesto, indicar que existen numerosos criterios para establecer un precio a una aplicación. En algunas empresas, prima la mano de obra, en otras el tiempo dedicado a su desarrollo, y en otras la calidad del software usado.

Por ejemplo, según las funcionalidades que ofrezca la aplicación desarrollada, los precios son menores para las aplicaciones con funcionalidades básicas, y van aumentando el precio según se van introduciendo bases de datos con funcionalidades personalizadas, juegos, mejoras o modificaciones en el hardware del dispositivo, hasta llegar a las aplicaciones totalmente dinámicas, como pueden ser *Pages* o *Adobe Ideas*, cuyo su precio alcanza cifras elevadas.

Además, existen otros factores que incrementan el precio de las aplicaciones móviles. Incluir en una aplicación básica la funcionalidad de compra (utilizando alguna pasarela de pago), puede aumentar el precio de la misma en más de 3000€. Los servicios web, para llevar el contenido a un punto de acceso remoto para que pueda actualizar su aplicación con archivos *XML*, y los G*ame Center*, también hacen que su precio se incremente entre 1500 y 2500€ a mayores.

Teniendo en cuenta todos los factores expuestos anteriormente, se muestra una tabla con el presupuesto estimado para el desarrollo de la aplicación desde el punto de vista de un trabajador autónomo. En la Ilustración 76 se muestra la tabla con el presupuesto realizado.

En el presupuesto obtenido no se han incluido las horas de aprendizaje ni de desarrollo de aplicaciones en *Android* ni el aprendizaje de creación de *Web Services* para la comunicación entre la aplicación y el servidor. Las horas reflejadas en este presupuesto son las horas que se han necesitado para la fase de diseño e implementación de la aplicación (parte cliente) y de la parte del servidor.

A continuación, se muestra un desglose detallado del presupuesto realizado:

- Salario del programador, que es la única partida de ingresos y que representa en mayor coste del presupuesto. El coste por hora asociado es de 13 € (programador Junior) y el tiempo empleado para el desarrollo completo del sistema ha sido de 215 horas.
- Cuota mensual de la seguridad social en régimen de autónomo (267,04 €) utilizando la base mínima de cotización (893,10 €). En el presupuesto se ha prorrateado a 25 días trabajados (jornada laboral de 8 horas diarias).
- Licencia de Adobe Photoshop: Consiste en un programa para la creación, edición y retoque de imágenes, desarrollado por la compañía *Adobe Systems*. En un principio fue creado para la plataforma *Apple*, pero posteriormente se desarrolló para *Windows*. Sin duda se trata del programa de edición de fotografía más popular, ello se debe a las numerosas posibilidades de retoque y modificación de fotografías que ofrece. Ha sido utilizado para la modificación de todas las imágenes que utiliza la aplicación para que

se adapten a los diferentes tamaños de pantalla de los diversos dispositivos donde puede ser utilizada.

- Gastos generales correspondientes al lugar de trabajo. Estos son, el coste del alquiler, la luz, el agua, la conexión a Internet y el teléfono. La estimación de estos gastos fijos se ha prorrateado, igual que en el caso anterior, para 25 días.
- Programas software necesarios para el desarrollo del sistema. Estos programas software tienen licencia libre y se han añadido para tener un presupuesto más detallado, aunque el coste sea nulo.
- Coste del ordenador portátil. El ordenador tiene una depreciación y, según el Ministerio de Hacienda, se permite que anualmente se incluya como gasto de depreciación en la declaración de impuestos hasta un 25% del valor total.
  Esto implica que en 4 años el ordenador queda desgravado, por lo que al año se puede poner como gasto de depreciación del ordenador un 25% de su valor (200 €). De este importe, se incluirá en este presupuesto la parte proporcional correspondiente a un mes (16,67 €), y no a 25 días como en los casos anteriores, porque suponemos que cuando adquiramos un nuevo equipo éste tendrá un coste superior al actual (aplicando el criterio contable de coste de renovación).
  El ordenador ha sido utilizado tanto para el proceso de aprendizaje como para las fases de diseño y desarrollo de la aplicación.

Finalmente se ha obtenido un presupuesto final de 3289,20 € para el desarrollo de este sistema de estimulación cognitiva elaborado en este proyecto. De este importe total, se estima un beneficio aproximado para el programador de 2650 €.

ESTIMACIÓN PRESUPUESTO ECONÓMICO

CONCEPTO	DETALLE	COSTE
Coste mano de obra programador Junior	13€/hora * 215h	**2795 €**
Cuota autónomo	267,04€/mes = 8,9€/día → 25 días trabajados	**222,53 €**
Licencia Photoshop	12€/mes	**10 €**
Costes oficina y gastos generales	Alquiler + Luz + Agua + Conexión Internet/teléfono (25 días)	**245 €**
Software Libre	phpMyAdmin como gestor de bases de datos NotePad++ PHP 5.4.16 + MySQL + Apache 2.4.6 Android Studio	**0 €**
Google Play	Cuenta de desarrollador en Google Play	**25 €**
Coste ordenador portátil con Windows 10	Coste total: 800€. Amortización fiscal: 25%/año: 200€ → Al mes: 16,67€	**16,67 €**
	TOTAL	**3314,20 €**

Ilustración 76. *Estimación del presupuesto de la aplicación*

CAPÍTULO 6

6 CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo se extraen todas las conclusiones obtenidas en la realización de este proyecto desde un punto de vista profesional y personal. Se abordan distintos aspectos relacionados con las tecnologías actuales, así como diversos aspectos de la aplicación desarrollada y los pasos seguidos para la realización de la misma.

Otro aspecto que se trata en este capítulo es la línea de trabajo futura, en la que se proponen ideas para su posible desarrollo e implementación. De este modo, se puede desarrollar una aplicación mucho más completa que ofrezca más funcionalidades con el fin de mejorar el sistema.

6.1 CONCLUSIONES

A lo largo de este proyecto se han investigado y estudiado los principales y más importantes sistemas operativos móviles (*Android*, *iOS*, *Windows Phone*, *Backberry* y *Symbian OS*) existentes en el mercado actualmente para después elegir la opción que más se ajusta a nuestras necesidades y requisitos.

Finalmente se ha optado por el desarrollo de una aplicación nativa para el sistema operativo *Android* disponible tanto en castellano como en inglés. Esta aplicación soporta varias resoluciones de pantalla y está disponible tanto para *Smartphone* como para *Tablet*. Las ventajas de estos dispositivos son muchas y muy variadas y todas ellas han servido de motivación para el desarrollo del proyecto. Entre las más destacadas se encuentra la capacidad de movilidad que aportan estos terminales pudiendo realizar los tutelados sus tratamientos y los voluntarios el seguimiento de los mismos en cualquier lugar. Además, la facilidad de uso de estos dispositivos unida a la mejora en la calidad y el tamaño de la pantalla hacen que los usuarios finales de la aplicación puedan disfrutar de la comodidad que supone poder realizar sus tratamientos en cualquier lugar donde se encuentren. También destacar que existen dispositivos muy económicos para dar soporte a ésta y otras aplicaciones debido a la versatilidad de los mismos. Sin embargo, el mayor inconveniente que surge es la necesidad de tener conexión a internet en el dispositivo para utilizar todas las funcionalidades que ofrece la aplicación. De no disponer de esta conexión, la aplicación pierde parte de las funcionalidades y queda un poco limitada.

Aunque el sistema desarrollado puede servir para funcionar en su estado actual a un nivel local, todavía son necesarios cambios en él para que pueda ser utilizado al cien por cien de sus posibilidades, ya que la base de datos local no almacena los seguimientos ni los tutelados, sólo tiene los datos de inicio de sesión de los voluntarios. Esto puede ser una línea de desarrollo futura interesante.

Se ha realizado un análisis técnico y de funcionamiento de todas las funcionalidades que ofrece la aplicación tales como el inicio de sesión, los tutelados, los seguimientos, las estadísticas y los juegos.

Posteriormente se ha elaborado un manual de uso para que los usuarios puedan sacar el máximo partido a la aplicación y sepan utilizar todos los recursos que ofrece la misma. El funcionamiento es bastante sencillo e intuitivo y se encuentra bien detallado en la sección 4 de este documento.

En definitiva, destacar que se han alcanzado de forma satisfactoria los objetivos que se pretendían con la elaboración de este proyecto en el desarrollo de una aplicación para el desarrollo cognitivo. Por un lado, destacar el aprendizaje en profundidad del lenguaje de programación *Java* para el desarrollo de aplicaciones para *Android* y, por otro lado, el desarrollo de una aplicación de estimulación cognitiva que pueda facilitar el trabajo de los voluntarios a la vez que mejora las capacidades cognitivas de los tutelados. De este modo, el voluntario puede llevar un seguimiento de los voluntarios desde cualquier lugar y de forma cómoda y sencilla.

6.2 LÍNEAS FUTURAS DE DESARROLLO

El sistema de rehabilitación desarrollado cubre buena parte de las funciones que se pedían. Aun así, existen varias funcionalidades a añadir y además, la modificación de alguna de las funciones presentes puede facilitar el trabajo futuro.

Una línea de desarrollo futura interesante sería desarrollar esta aplicación para otras plataformas como *Windows Phone*. De este modo, la aplicación estaría disponible en la mayoría de terminales móviles existentes hasta el momento.

Otra línea futura de desarrollo es la mejora de la aplicación en diversos aspectos. A continuación, se presentan algunas ideas para su desarrollo en el ámbito de los juegos:

- Una mejora interesante puede ser la implementación de un botón de pausa en los juegos para que los temporizadores no actúen en caso de que el jugador deje el juego por algún motivo.
- Aunque existe al menos un juego por categoría puede ser interesante el desarrollo de más juegos con sus correspondientes niveles de dificultad.
- Aunque todos los juegos implementados tienen los 3 niveles de dificultad puede ser útil la ampliación de alguno de ellos como el de `Laberintos` que los modelos de laberinto son limitados y pronto se repetirán. Esto también se puede extrapolar al resto de juegos.

En cuanto a la base de datos:

- Se puede implementar una pequeña base de datos local utilizando *SQLite* y, en caso de no disponer de conexión a internet, se almacena en local hasta que el dispositivo tenga conexión y se pueda sincronizar la base de datos local con la del servidor. Posteriormente se pueden borrar los datos de la base de datos local para que no ocupe demasiada memoria la aplicación.
- Ampliación de la aplicación para poder gestionar todos los datos de los tutelados como datos económicos, datos jurídicos….
- Creación de nuevas estadísticas en la base de datos para los juegos.

En cuanto a la aplicación en general:

- En la opción `Tutelados` que ofrece, actualmente solo se muestran los tutelados que lleva ese voluntario, los datos de interés del tutelado como problemas alérgicos, número de teléfono, dirección del domicilio…y la creación de un tutelado. Se pueden implementar otras opciones como la eliminación de un tutelado y la modificación de los datos del mismo.
- En la opción `Estadísticas` se podrían mostrar varias estadísticas en una gráfica o valores medios de los datos mostrados para tener una referencia en la comparación de resultados.
- Desarrollo de la aplicación utilizando los `Fragments` que ofrece *Android* para la mejor visualización de los contenidos en caso de utilizar una *Tablet*.
- Posibles mejoras que surjan con el desarrollo de los dispositivos y del sistema operativo *Android*.
- Crear un sistema de notificaciones que muestre los cambios y avisos al usuario una vez que se registre en la aplicación.

6.3 EXPERIENCIA PERSONAL

A nivel personal el desarrollo de este proyecto ha sido una experiencia muy enriquecedora tanto a nivel personal como de conocimientos adquiridos.

Esto me ha servido para adquirir y complementar los conocimientos sobre el lenguaje de programación *Java*. También me ha servido para aprender a desarrollar aplicaciones para el sistema operativo *Android* y conocer más en profundidad el funcionamiento de otros sistemas operativos móviles. El desarrollo de aplicaciones para Android ha sido una cosa que siempre me ha interesado puesto que lo utilizo a diario y cada vez más se da el uso de este tipo de dispositivos por eso pienso que es importante saber desarrollar aplicaciones para *Android*.

Otro punto de interés es la utilización de bases de datos *MySQL*, siendo muy útiles para almacenar toda la información del sistema desarrollado. He tenido que aprender a comunicarme desde la aplicación con el servidor donde se encuentran almacenados todos los datos. Esta comunicación finalmente se llevó a cabo utilizando servicios *Restful* y desarrollando unos *Web Services* en *php*. Por lo tanto, además de aprender a desarrollar aplicaciones para *Android* también he aprendido a crear *Web Services* en *php* y la utilización de bases de datos *MySQL*.

Estoy contento con los resultados obtenidos en este proyecto y por todo lo aprendido al llevarlo a cabo. Aunque he aprendido bastantes cosas de *Android* quiero seguir profundizando en el desarrollo de aplicaciones con las novedades que ofrecen las nuevas versiones de *Android*, así como aprender *Swift*, el lenguaje de desarrollo de aplicaciones para *iOS*.

La realización de este proyecto no solo me ha aportado nuevos conocimientos a nivel académico, sino que también es gratificante poder ayudar a personas a mejorar su calidad de vida utilizando todos los conocimientos adquiridos a lo largo de los años. Si la aplicación logra ayudar a mejorar la calidad de vida de las personas que lo necesitan y facilitar el seguimiento de los mismos a los voluntarios sería una satisfacción personal por todo el esfuerzo empeñado en aprender los conocimientos necesarios para desarrollar este sistema.

7 BIBLIOGRAFÍA

(2005). En A. Díaz, F. Rivas, & P. Merino, *Actas de las XIII Jornadas de Concurrencia y Sistemas Distribuidos* (págs. 259-269). Thomson.

Aguayo Sánchez, J. M. (s.f.). *Desarrollo de aplicaciones iOS para iPhone & iPad: Essentials.* Madrid: 0xWord.

Android Developers. (10 de Mayo de 2016). Obtenido de Android Developers: https://developer.android.com/

Android Studio website. (15 de Mayo de 2016). Obtenido de https://developer.android.com/studio/intro/index.html

Apple (Dirección). (1987). *Knowledge Navigator* [Película].

Apple Inc. (3 de Abril de 2016). Obtenido de http://www.apple.com/

Ball, M., & Lillis, J. (2001). Ehealth: transforming the physician/patient relationship. *International Journal of Medical Informatics 61*, 1-10.

Campo, C., & García Rubio, C. (2014). *Sistemas Operativos de Dispositivos Moviles.* Madrid: Universidad Carlos III.

Díaz, F. (2006). EL MODELO CLIENTE/SERVIDOR. Segovia: Universidad de Valladolid.

Flores Galea, A. L. (s.f.). *Cómo...BlackBerry.* Creaciones Copyright.

Gómez, I. (3 de Mayo de 2016). *Arquitectura de la plataforma de desarrollo de windows phone 7*. Obtenido de http://docplayer.es/17596540-Arquitectura-de-la-plataforma-de-desarrollo-de-windows-phone-7.html

Lee, W.-m. (2013). *Android. desarrollo de aplicaciones ganadoras.* Madrid: ANAYA MULTIMEDIA.

Leuzas, L. (2015). *Diseño y Desarrollo de una Aplicación iOS de Seguimiento y Estimulación Cognitiva.* Universidad de Valladolid: Trabajo Fin de Grado.

Maroto, M. Á. (s.f.). *La memoria: Programa de estimulación y mantenimiento cognitivo.* Comunidad de Madrid: INSTITUTO DE SALUD PÚBLICA: CONSEJERÍA DE SANIDAD.

Microsoft. (5 de Marzo de 2016). *Microsoft*. Obtenido de http://windowsphone.com

Potencier, F., & Weaver, R. (2012). *Symfony 2.4, el libro oficial.* Symfony.

Prieto, M. (2008). *Mente activa : ejercicios para la estimulación cognitiva gerontológica.* Madrid: Pirámide.

Rodger, R. (2012). *Desarrollo de aplicaciones en la nube para dispositivos móviles.* Madrid: Anaya Multimedia.

Sager, I. (2012). *Before Iphone and Android Came Simon, the First Smartphone*. New York: Bloomberg, L.P.

Santa María, F. (15 de Abril de 2016). *StaffCreativa*. Obtenido de http://www.staffcreativa.pe/blog/android-ventajas-desventajas/

Serna, R., Julián, J. Y., & Landa, I. (s.f.). *Introducción al desarrollo en Windows y Windows Phone 8* . Krasis.

Shulin, Y. (2014). Research and implementation of Web Services in Android. IEEE CONFERENCE PUBLICATIONS.

Symbian, F. (2010). Symbian Completes Biggest Open Source Migration Project Ever.

Tomás, J. (2014). *El gran libro de Android avanzado*. Barcelona: Marcombo.

Toshiba. (s.f.). History of Notebook PC "Dynabook".

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA. (17 de Mayo de 2016). Obtenido de http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/leccin_13_desarrollo_de_aplicaciones.html

Welling, L. (2003). *PHP and MySQL Web Development*. Estados Unidos: Sams Publishing.

8 ANEXOS TÉCNICOS

8.1 ANEXO 1: 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*. Además del potente editor de códigos y las herramientas para desarrolladores de *IntelliJ*, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android, como las siguientes: (Android Developers, 2016)

- Sistema de compilación flexible basado en *Gradle*.
- 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, uso, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK.
- Soporte integrado para *Google Cloud Platform*, que facilita la integración de *Google Cloud Messaging* y *App Engine*.

8.1.1 REQUISITOS MÍNIMOS DEL SISTEMA

Los requisitos del sistema mínimos para que funcione correctamente son los siguientes, independientemente del sistema operativo sobre el que se instale: (Android Developers, 2016)

- 2 GB de memoria RAM como mínimo; se recomiendan 8.
- 2 GB de espacio mínimo disponible en el disco; se recomiendan 4 (500 MB para IDE + 1,5 GB para el Android SDK y la imagen de sistema del emulador).
- Resolución de pantalla mínima de 1280 x 800.
- Para el emulador acelerado: Sistema operativo de 64 bits y procesador Intel compatible con *Intel VT-x*, *Intel EM64T* (Intel 64) y la funcionalidad *Execute Disable (XD) Bit*.

Para instalarlo en Windows se necesita Microsoft Windows 7/8/10 (32 o 64 bits), para instalarlo en Mac se necesita Mac OS X 10.8.5 o versiones posteriores hasta 10.11.4 (El Capitán) y para sistemas Unix se necesita GNOME o KDE de escritorio[1], una distribución de 64 bits capaz de ejecutar aplicaciones de 32 bits y GNU C Library (glibc) 2.11 o versiones posteriores.

[1] Pruebas realizadas en Ubuntu 12.04, Precise Pangolin (distribución de 64 bits capaz de ejecutar aplicaciones de 32 bits)

8.1.2 INSTALACIÓN DE ANDROID STUDIO

Esta guía de instalación es válida para instalar Android Studio sobre Windows. Para poder utilizar Android Studio es necesario tener instalado Java en el ordenador. En caso de no tenerlo, será necesaria su instalación. Con la versión *Java 7 update 79* es suficiente, aunque conviene tener siempre la última versión Java instalada por temas de seguridad.

Para descargar el instalador de Android Studio basta con acceder a la siguiente URL: https://developer.android.com/studio/index.html?hl=es-419#tos-header. Una vez descargado solo hay que seguir los siguientes pasos:

1. Ejecutar el archivo `.exe` que se descargó.
2. Seguir las indicaciones del asistente de configuración para instalar Android Studio y las herramientas de SDK necesarias. En algunos sistemas de Windows, la secuencia de comandos de inicio no encuentra el destino de instalación del JDK. Si se produce este problema, debes configurar una variable de entorno que indique la ubicación correcta. Selecciona `Start menu > Computer > System Properties > Advanced System Properties`. Luego abre la pestaña `Advanced > Environment Variables` y agrega una nueva variable de sistema `JAVA_HOME` que apunte a tu carpeta de JDK. Por ejemplo, `C:\Program Files\Java\jdk1.8.0_77`.

8.1.3 CREAR LA PRIMERA APP CON ANDROID STUDIO

Lo primero que hay que hacer para crear una App es crear un proyecto en Android Studio. Para crear el proyecto hay que seguir los siguientes pasos: (Android Developers, 2016)

1. En Android Studio, crear un proyecto:
   - Si es la primera vez que se inicia Android Studio o no existe ningún proyecto cargado, aparecerá la ventana `Welcome to Android Studio` como se muestra en la Ilustración 77. Hacer clic en `Iniciar un nuevo proyecto de Android Studio`.
   - Si ya existe un proyecto abierto, seleccionar `File > New Project`.
2. En la pantalla New Project, ingresar los siguientes:
   - Nombre de la App: MyApp[2]
   - Dominio de la empresa: example.com

   Si se desea programar la app en C++, hay que seleccionar la casilla correspondiente. Android Studio completa el nombre del paquete y la ubicación, pero puedes editar estos datos si lo deseas.
3. En la pantalla `Target Android Devices`, se selecciona el SDK mínimo requerido[3]. También se selecciona el tipo del dispositivo donde se va a instalar la App como se muestra en la Ilustración 78. Conservar los valores predeterminados y hacer clic en `Next`.

[2] Nombre que queramos dar a la app. El campo *empresa* es irrelevante. Posteriormente estos valores se pueden modificar.

[3] El SDK mínimo requerido es la primera versión de Android que admite tu app, lo cual se indica a través del nivel de API. Para lograr la compatibilidad con la mayor cantidad posible de dispositivos, debes establecer esto en la versión más antigua disponible que permite a tu app proporcionar su conjunto de funciones fundamentales.

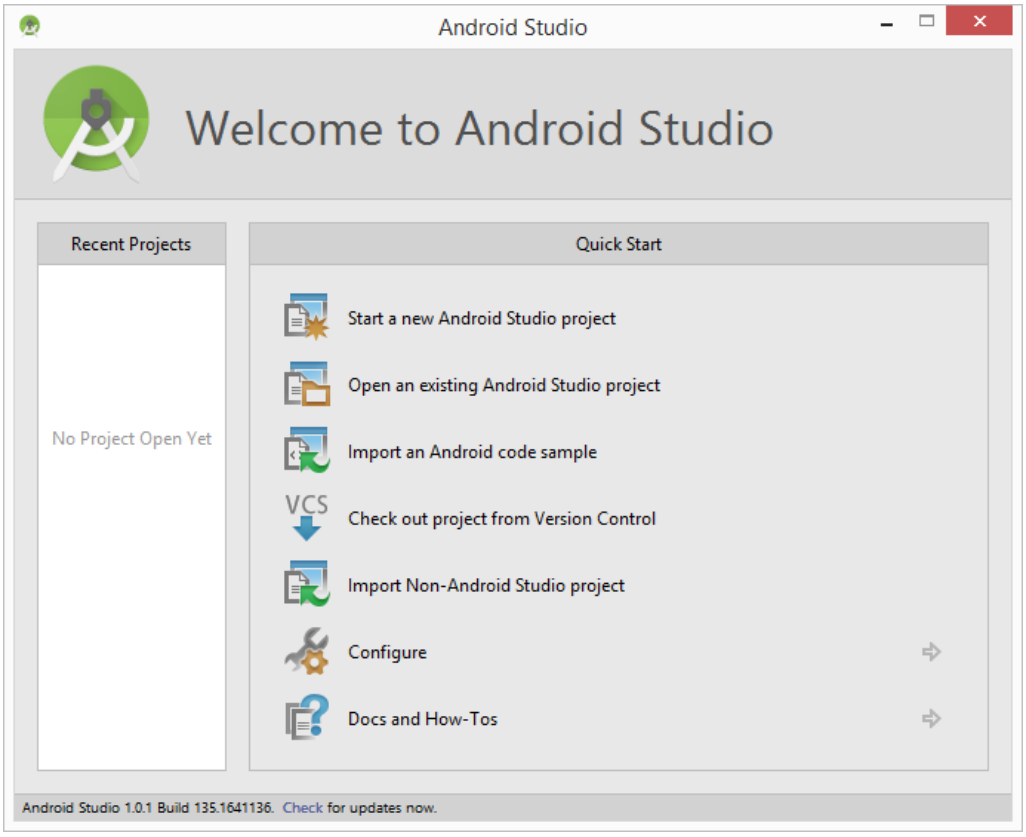

Ilustración 77. *Ventana* `Welcome to Android Studio`

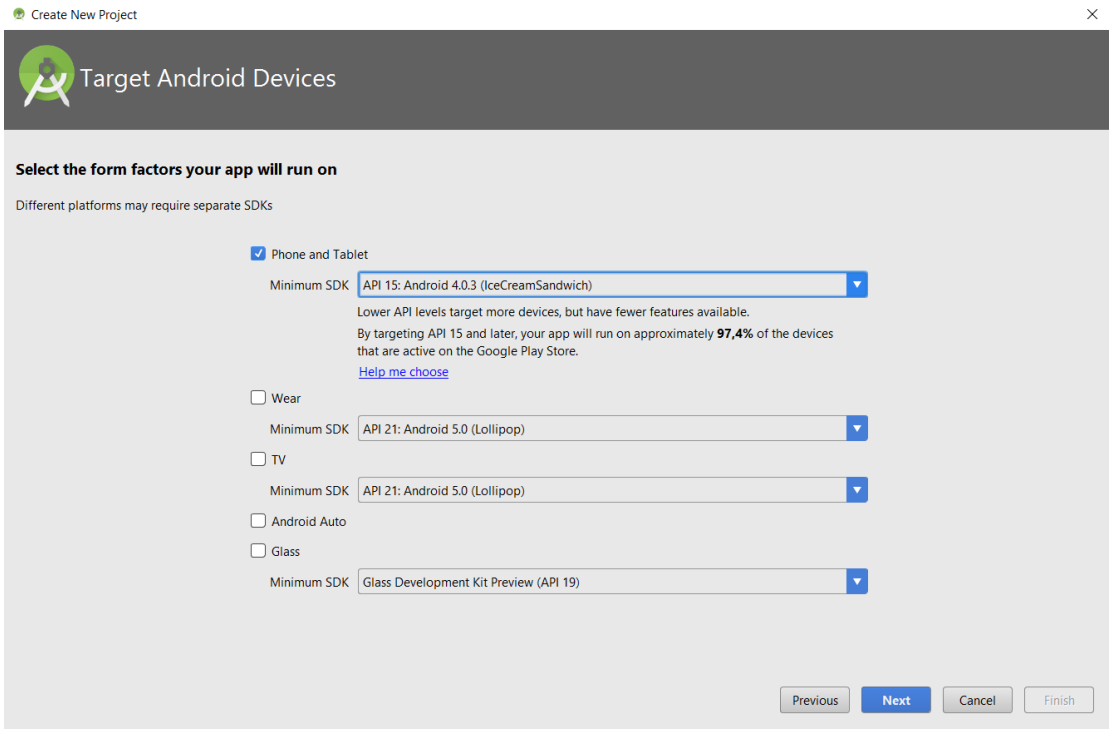

Ilustración 78. *Ventana* `Target Android Devices`

4. En la pantalla `Add an Activity to Mobile`, seleccionar `Empty Activity` y hacer clic en `Next`. En esta pantalla se pueden elegir distintas interfaces de usuario como se muestra en la Ilustración 79. En caso de necesitar una interfaz en concreto basta con seleccionarla y el *layout* correspondiente se genera automáticamente.
5. En la pantalla `Customize the Activity`, conservar los valores predeterminados y hacer clic en `Finish`.

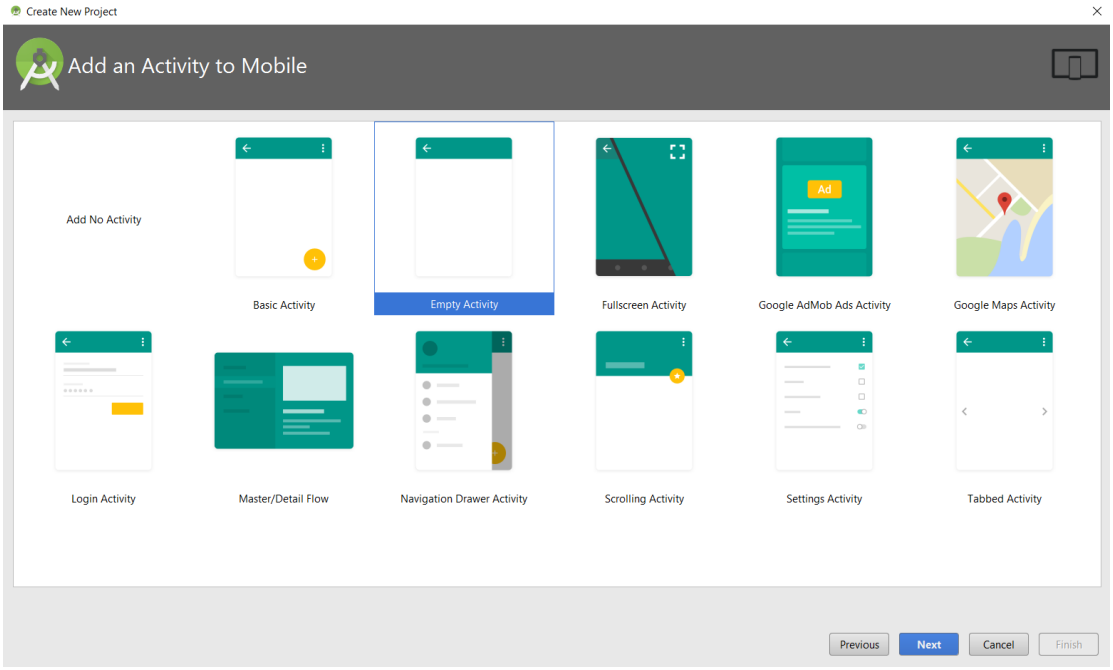

Ilustración 79. *Ventana* `Add an Activity to Mobile`

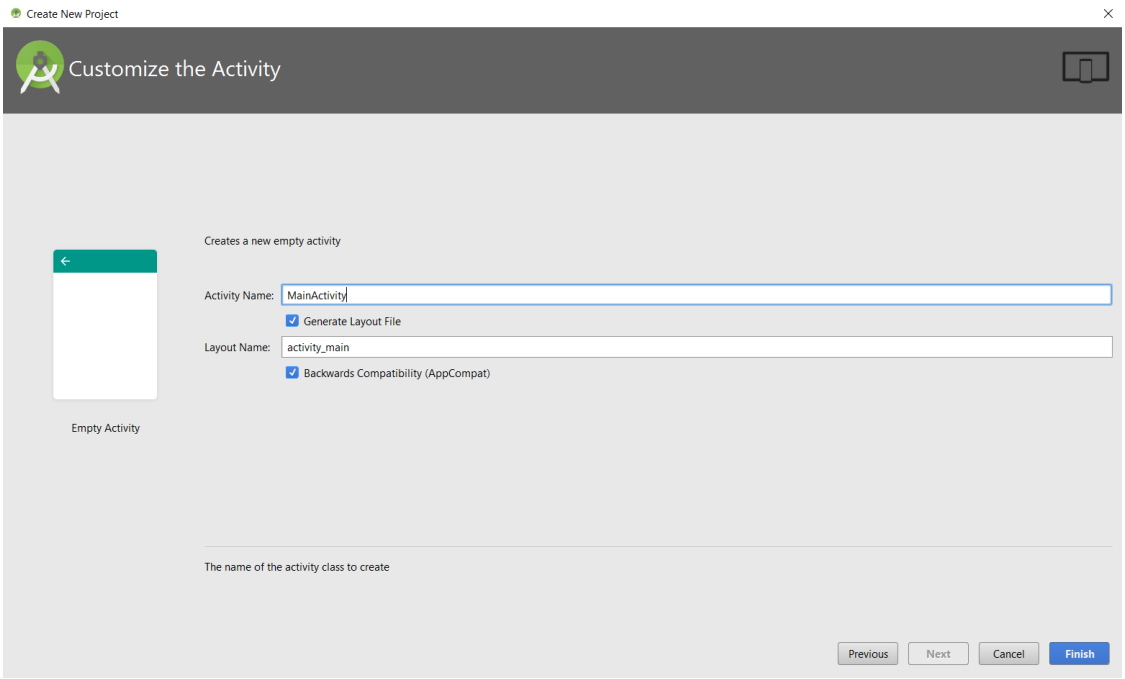

Ilustración 80. *Ventana* `Customize the Activity`

Después de procesar, Android Studio se abre y aparece una app `Hello World` con archivos predeterminados. Posteriormente se añadirán nuevos ficheros para crear unas funcionalidades de la App.

8.1.4 ESTRUCTURA DEL PROYECTO

Cada proyecto en Android Studio contiene uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen los siguientes:

- Módulos de apps para Android
- Módulos de bibliotecas
- Módulos de Google App Engine

De forma predeterminada, en Android Studio se muestran los archivos del proyecto en la vista de proyectos de Android, como se muestra en la Ilustración 81. Esta vista está organizada en módulos para que se pueda acceder rápidamente a los archivos de origen claves del proyecto.

Todos los archivos de compilación son visibles en el nivel superior de Secuencias de comando de *Gradle* y cada módulo de la aplicación contiene las siguientes carpetas:

- **manifests:** contiene el archivo `AndroidManifest.xml`.
- **java:** contiene los archivos de código fuente de Java, incluido el código de prueba JUnit.
- **res:** Contiene todos los recursos, como diseños XML, cadenas de IU e imágenes de mapa de bits.

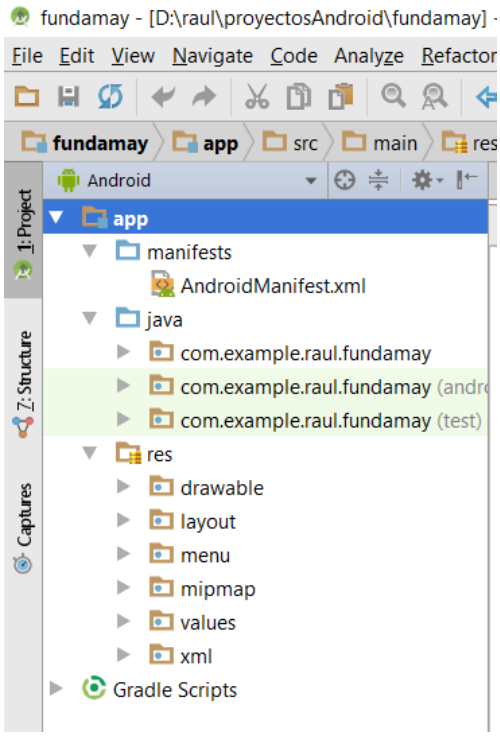

Ilustración 81. *Estructura de un proyecto en Android Studio*

8.1.5 INTERFAZ DE USUARIO

La ventana principal de Android Studio consta de varias áreas lógicas que se identifican en la Ilustración 82. (Android Developers, 2016)

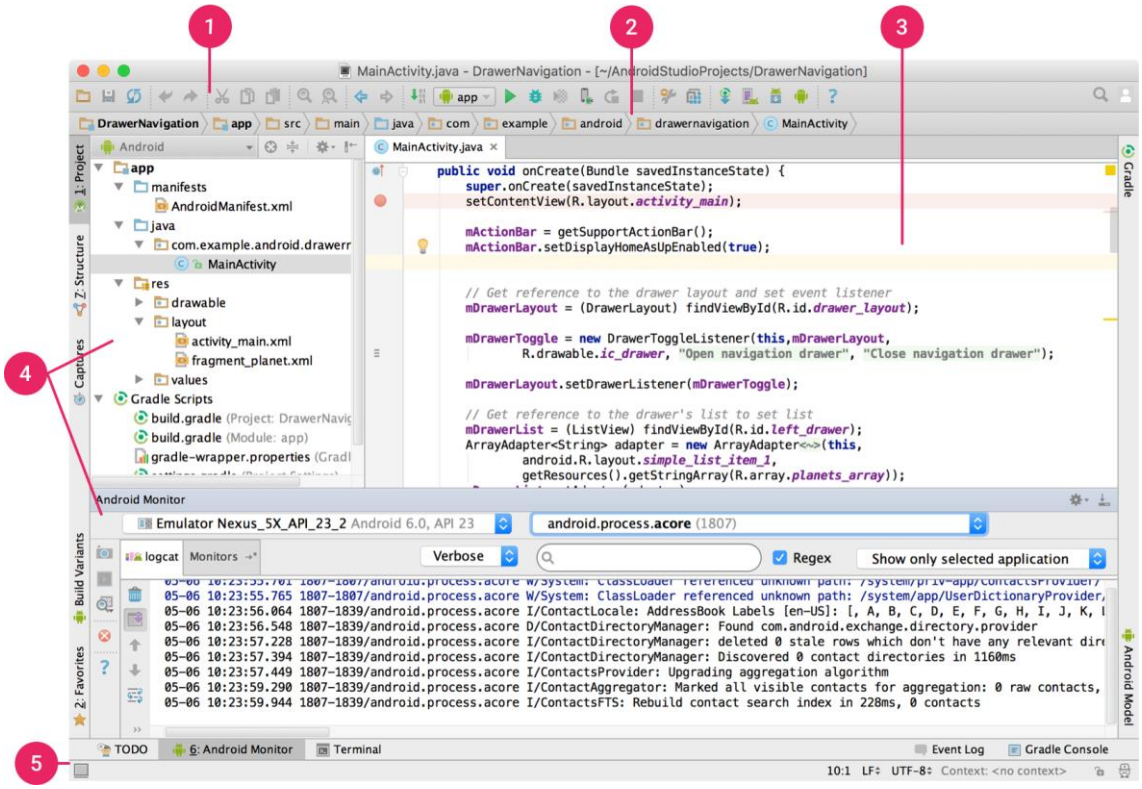

Ilustración 82. *Ventana principal de Android Studio*

1. La barra de herramientas permite realizar una gran variedad de acciones, como la ejecución de la app y el inicio de herramientas de Android.
2. La barra de navegación ayuda a explorar el proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana Project.
3. La ventana del editor es el área en la que se puede crear y modificar código. Según el tipo de archivo actual, el editor puede cambiar. Al visualizar un archivo de diseño, por ejemplo, el editor muestra el Editor de diseño.
4. Las ventanas de herramientas permiten acceder a tareas específicas, como la administración de proyectos, la búsqueda y los controles de versión, entre otras. Se pueden expandir y contraer estas ventanas.
5. En la barra de estado se muestra el estado del proyecto y el IDE, además de advertencias o mensajes.

Se puede organizar la ventana principal para tener más espacio en pantalla ocultando o desplazando barras y ventanas de herramientas. También se pueden usar combinaciones de teclas para acceder a la mayoría de las funciones del IDE.

En cualquier momento, puedes realizar búsquedas en tu código fuente, bases de datos, acciones, elementos de la interfaz de usuario, etc., presionando dos veces la tecla `Shift` o haciendo clic en la lupa que se encuentra en la esquina superior derecha de la ventana de Android Studio. Esto puede ser muy útil, por ejemplo, si intentas localizar una acción específica del IDE que olvidaste cómo activar.

El estilo y formato en Android Studio también se puede personalizar. Mientras editas, Android Studio aplica automáticamente formatos y estilos según lo especificado en la configuración de estilo de código. Se puede personalizar la configuración de estilo de código programando el idioma, que incluye la especificación de convenciones para pestañas y sangrías, espacios, ajuste y llaves, y líneas en blanco. Para personalizar la configuración de estilo del código, hau que hacer clic en `File > Settings > Editor > Code Style`. Si bien el IDE aplica formato automáticamente mientras se trabaja, también se puede llamar explícitamente a la acción `Reformat Code` presionando `Control+Alt+L`, o aplicar sangrías automáticas a todas las líneas presionando `Control+Alt+I`.

Otro aspecto útil que incluye Android Studio es el control de versiones. Android Studio admite diferentes sistemas de control de versión (VCS), incluidos *Git*, *GitHub*, *CVS*, *Mercurial*, *Subversion* y *Google Cloud Source Repositories*.

Después de importar la app a Android Studio, hay que usar las opciones del menú del VCS de Android Studio a fin de habilitar la compatibilidad con VCS para el sistema de control de versión deseado, crear un repositorio, importar los nuevos archivos al control de versión y realizar otras operaciones de control de versión[4]:

1. En el menú VCS de Android Studio, hacer clic en `Enable Version Control Integration`.
2. En el menú desplegable, seleccionar un sistema de control de versión para asociarlo con la raíz del proyecto y luego haz clic en `OK`.

En el menú del VCS se muestran diversas opciones de control de versión según el sistema que se haya seleccionado.

8.1.6 SISTEMA DE COMPILACIÓN DE GRADLE

Android Studio usa *Gradle* como base del sistema de compilación, y proporciona más características específicas de Android a través del Complemento de Android para *Gradle*. Este sistema de compilación se ejecuta en una herramienta integrada desde el menú de Android Studio, y lo hace independientemente de la línea de comandos. Puedes usar las funciones del sistema de compilación para lo siguiente: (Android Developers, 2016)

- Personalizar, configurar y extender el proceso de compilación.
- Crear varios APK para una misma app con diferentes funciones usando el mismo proyecto y los mismos módulos.
- Volver a usar códigos y recursos entre conjuntos de orígenes.

[4] También puedes usar la opción del menú `File > Settings > Version Control` para configurar y modificar los ajustes de control de versión.

Recurriendo a la flexibilidad de *Gradle*, se puede lograr todo esto sin modificar los archivos de origen de la App. Los archivos de compilación de Android Studio se denominan `build.gradle`. Son archivos de texto sin formato que usan sintaxis *Groovy* para configurar la compilación con elementos proporcionados por el complemento de Android para *Gradle*. Cada proyecto tiene un archivo de compilación de nivel superior para todo el proyecto y archivos de compilación de nivel de módulo independientes para cada módulo. Cuando importas un proyecto existente, Android Studio genera automáticamente los archivos de compilación necesarios.

8.1.7 FICHERO ANDROIDMANIFEST.XML

El fichero `AndroidManifest.xml` es un fichero de configuración situado en la raíz de las Apps donde se pueden aplicar las configuraciones básicas de nuestras Apps. En la Ilustración 83 se puede ver un ejemplo de este fichero. Se utiliza notación XML para crear las configuraciones. En este fichero se declaran, por ejemplo, todas las `activity` de las que consta nuestra App utilizando la etiqueta `<activity></activity>`. También se puede crear el estilo de nuestra App, así como el icono de la misma dentro de la etiqueta `<application></application>`.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.raul.fundamay">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/icono"
        android:label="Fundamay"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name=".MainActivity"
            android:label="Fundamay"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".paginaPrincipal"
            android:label="Fundamay"
            android:parentActivityName=".MainActivity"
            android:theme="@style/AppTheme.NoActionBar">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".MainActivity"/>
        </activity>
        <activity
            android:name=".todosSeguimientos"
            android:label="Fundamay"
```

Ilustración 83. *Ejemplo de fichero* `AndroidManifest.xml`

Un tema importante de configuración son los permisos que va a soportar nuestra App como el acceso a internet, acceso a la cámara del teléfono y todos los permisos que se nos puedan ocurrir. Todos estos permisos se declaran también en este fichero. Se declaran utilizando la etiqueta `<uses-permission/>`. En la Ilustración 84 se muestran algunos permisos que se pueden proporcionar a nuestra App como el acceso a internet y la lectura/escritura en una tarjeta SD/microSD (almacenamiento externo).

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Ilustración 84. *Ejemplos de permisos `AndroidManifest.xml`*

8.1.8 IMPORTAR LIBRERÍAS EN ANDROID STUDIO

Para importar librerías en Android Studio hay que seguir los siguientes pasos:

1. Una vez situado en el proyecto Android Studio, crear la carpeta `libs` a la altura de `build` y `src` en caso de que no exista.
2. Una vez creada esta carpeta, copiar todos los ficheros `.jar` que se necesiten.
3. Tras esto hay que añadir las dependencias a `Build.Gradle` (Module:app). En este fichero, en dependencies, se añaden las líneas `compile fileTree (dir: 'libs', include ['*.jar'])` y todas las librerías añadidas con su respectivo nombre con la línea `compile files ('nombreDeLaLibreria.jar')` como se muestra en la Ilustración 85. En este caso se ha añadido la librería `mpandroidchartlibrary-2-2-4.jar`.
4. Tras esto solo hay que hacer la sincronización del proyecto con el `Gradle` haciendo clic en `Sync Project with Gradle Files`.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.2.1'
    compile 'com.android.support:design:23.2.1'
    compile files('libs/mpandroidchartlibrary-2-2-4.jar')
}
```

Ilustración 85. *`Dependencies` del fichero `Gradle.Build`*

8.1.9 CONSIDERACIONES ÚTILES PARA CREAR UNA APP

En esta sección se exponen algunos consejos a la hora de crear una aplicación. Estos consejos sirven para estructurar de una manera más clara la App y que el resultado visual sea más llamativo y mejor estructurado. (Android Developers, 2016)

8.1.9.1 LAYOUT

Un aspecto importante es el directorio `Layout`. Un `layuot` o diseño define la estructura visual para una interfaz de usuario, como la IU (Interfaz de Usuario) para una actividad o widget de una App. Estos diseños se pueden declarar de 2 maneras:

- Declarar elementos de la IU en XML: Android proporciona un vocabulario XML simple que coincide con las clases y subclases de vistas, como las que se usan para widgets y diseños.
- Crear una instancia de elementos del diseño en tiempo de ejecución: la aplicación puede crear objetos `View` y `ViewGroup` (y manipular sus propiedades) programáticamente.

Declarar el diseño de la IU en XML en lugar del código de tiempo de ejecución es útil por diferentes motivos, pero es especialmente importante para que se puedan crear diferentes diseños para distintos tamaños de pantallas. Por ejemplo, se pueden crear dos versiones de un diseño e indicarle al sistema que use uno en pantallas "pequeñas" y el otro en pantallas "grandes".

Android también distingue 2 directorios para almacenar estos diseños, `layout` y `layout-land`. En el directorio `layout` se almacenan los diseños para la orientación vertical del dispositivo y en el directorio `laoyout-land` se almacenan los diseños para la posición horizontal (apaisado) del dispositivo. Esto es realmente útil para que la IU se adapte a todas las posiciones del dispositivo y no se descuadren todos los elementos de la IU. Por eso es interesante crear 2 `layout` por cada actividad de la App.

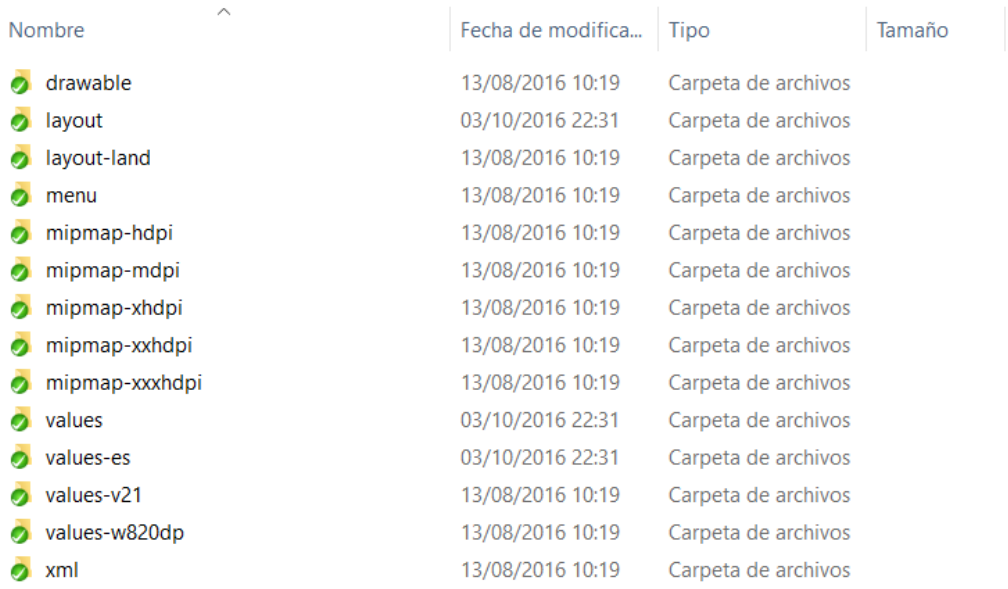

Ilustración 86. *Estructura del directorio `res` de Android*

En la Ilustración 87 se muestra un ejemplo de diseño utilizando un fichero XML para su creación. Por lo general, en los diseños para vistas verticales, se suele utilizar la etiqueta `<LinearLayout></LinearLayout>` tanto con la orientación horizontal como con la vertical (`Android:orientation="horizontal"/"vertical"`). Con este `layout` los elementos se disponen de manera lineal de manera vertical u horizontal dependiendo del parámetro utilizado. Sin embargo, cuando se crea un diseño para la posición apaisada del dispositivo se suele utilizar la etiqueta `<RelativeLayout></RelativeLayout>`. Con este layout los elementos se colocan dependiendo de la posición relativa de los otros elementos de la UI. En la Ilustración 88 se muestra un ejemplo de este tipo de `layout`.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:orientation="horizontal"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
</LinearLayout>
```

Ilustración 87. *Ejemplo de LinearLayout*

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

Ilustración 88. *Ejemplo de RelativeLayout*

Aunque la regla de diseño anterior es la que se suele utilizar a la hora de crear un diseño para una actividad esto no quiere decir que, si en algún momento es preciso, se pueda utilizar un `RelativeLayout` en el diseño de la IU en la posición vertical del dispositivo.

Cuando se compila la aplicación, cada archivo de diseño XML se compila en un recurso View. Se debe cargar el recurso de diseño desde el código de la aplicación, en la implementación de callback `Activity.onCreate()`. Para hacerlo, se llama a `setContentView ()`, pasarle la referencia a el recurso de diseño en forma de: `R.layout.layout_file_name`. Por ejemplo, si el diseño XML se guarda como main_layout.xml, se cargaría para la actividad como se muestra en la Ilustración 89.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

Ilustración 89. *Cargar el diseño XML para la actividad creada*

8.1.9.2 DIRECTORIO VALUES

Siempre es una buena práctica extraer *strings* (cadenas de texto) de IU del código de la App y conservarlas en un archivo externo. Android facilita esta tarea con un directorio de recursos en cada proyecto del sistema operativo.

Android posee el directorio `res/` donde se almacenan los recursos de la App entre otros como se muestra en la Ilustración 86. También hay algunos archivos predeterminados, como `res/values/strings.xml`, que contiene los valores de las *strings*.

Para agregar compatibilidad con más idiomas, solo hay que crear directorios `values` adicionales dentro de `res/` que incluyan un guión y el código de idioma ISO al final del nombre del directorio. Por ejemplo, `values-es/` es el directorio que contiene recursos simples para las configuraciones regionales con el código de idioma “es” (español). Android carga los recursos correspondientes según las configuraciones regionales del dispositivo en tiempo de ejecución.

A continuación, se presentan unos ejemplos de distintos ficheros strings.xml con los distintos idiomas para observar cómo se configuran. Esto también es extrapolable, no solo a *strings*, sino también a *arrays*.

Inglés (configuración regional predeterminada), `/values/strings.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="title">My Application</string>
    <string name="hello_world">Hello World!</string>
</resources>
```

Español, `/values-es/strings.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="title">Mi Aplicación</string>
    <string name="hello_world">Hola Mundo!</string>
</resources>
```

Francés, `/values-fr/strings.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="title">Mon Application</string>
    <string name="hello_world">Bonjour le monde !</string>
</resources>
```

Ilustración 90. *Ejemplo del fichero `strings.xml` en distintos idiomas*

Se puede hacer referencia a los recursos de `string` en el código fuente y en otros archivos XML usando el nombre del recurso definido por el atributo `name` del elemento `<string>`.

En el código fuente, se puede hacer referencia a un recurso de `string` con la sintaxis `R.string.<string_name>`. Existen diferentes métodos disponibles que aceptan un recurso de `string` de esta manera. También se puede utilizar el método `getResources.getString(R.string.string_name)` pasándole como argumento el identificador de *string*. Ambos ejemplos se muestran en la Ilustración 91.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
```

Ilustración 91. *Obtener string en un fichero XML*

```
// Get a string resource from your app's Resources
String hello = getResources().getString(R.string.hello_world);

// Or supply a string resource to a method that requires a string
TextView textView = new TextView(this);
textView.setText(R.string.hello_world);
```

Ilustración 92. *Obtener string en código fuente*

En otros archivos XML, se puede hacer referencia a un recurso de `string` con la sintaxis `@string/<string_name>` siempre que el atributo XML acepte un valor de *string*.

8.1.9.3 COMPATIVILIDAD CON DIFERENTES PANTALLAS

Android se ejecuta en distintos dispositivos que ofrecen diferentes tamaños y densidades de pantallas. Para las aplicaciones, el sistema Android proporciona un entorno de desarrollo uniforme en todos los dispositivos y se encarga de la mayor parte del trabajo para adecuar la interfaz de usuario de cada aplicación a la pantalla en la que se muestra. Al mismo tiempo, el sistema proporciona las API que te permiten controlar la IU de tu aplicación para densidades y tamaños específicos de las pantallas, a fin de optimizar tu diseño de IU para configuraciones de pantalla diferentes.

Android cataloga el tamaño de la pantalla del dispositivo utilizando 2 propiedades generales: tamaño y densidad:

- Hay 4 tamaños generalizados: small, normal, large, xlarge (pequeño, normal, grande, extra largo)
- Hay 6 densidades generalizadas: low (ldpi), medium (mdpi), high (hdpi), extra high (xhdpi), extra extra high (xxhdpi), extra extra extra high (xxxhdpi)
    - ldpi (baja) ~120 dpi
    - mdpi (media) ~160 dpi
    - hdpi (alta) ~240 dpi
    - xhdpi (extraalta) ~320 dpi
    - xxhdpi (extra extraalta) ~480 dpi
    - xxxhdpi (extra extra extraalta) ~640 dpi

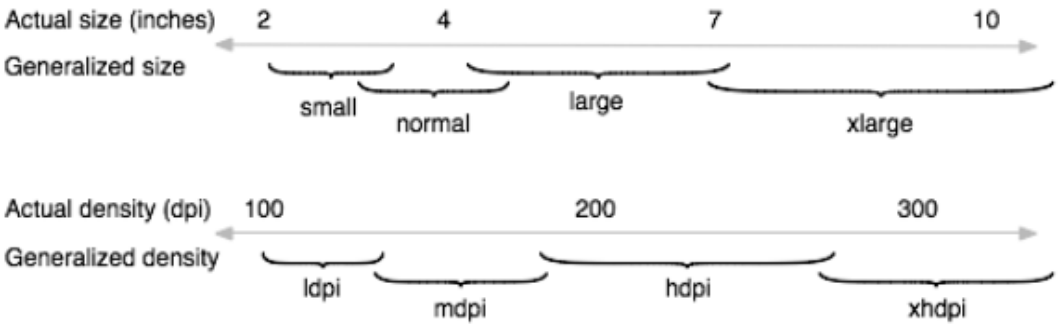

Ilustración 93. *Clasificación de los tamaños de pantalla*

El aspecto básico de la compatibilidad de Android con pantallas múltiples es su capacidad para manejar la representación del diseño y de los elementos de diseño de mapa de bits de una aplicación de manera apropiada según la configuración de la pantalla actual. El sistema se encarga de la mayor parte del trabajo para representar tu aplicación de modo apropiado en la configuración de cada pantalla ajustando diseños para que se adecuen al tamaño y a la densidad de la pantalla, y también los elementos de diseño del mapa de bits según la densidad de pantalla, según corresponda. Sin embargo, para manejar correctamente las diferentes configuraciones también hay que hacer lo siguiente:

- Declarar en el manifiesto de forma explícita los tamaños de pantalla que admite la aplicación: Al hacerlo, podrá asegurarte de que la aplicación solo pueda descargarse mediante dispositivos que cuenten con pantallas compatibles. Para declarar los tamaños de pantalla compatibles con la aplicación, hay que incluir el elemento `<supports-screens>` en el archivo de manifiesto (`AndroidManifest.xml`).

```
<supports-screens android:resizeable=["true"| "false"]
                  android:smallScreens=["true" | "false"]
                  android:normalScreens=["true" | "false"]
                  android:largeScreens=["true" | "false"]
                  android:xlargeScreens=["true" | "false"]
                  android:anyDensity=["true" | "false"]
                  android:requiresSmallestWidthDp="integer"
                  android:compatibleWidthLimitDp="integer"
                  android:largestWidthLimitDp="integer"/>
```

Ilustración 94. *Elemento `<supports-screens>`*

- Proporcionar diseños diferentes para diferentes tamaños de pantalla: De forma predeterminada, Android cambia el tamaño del diseño de la aplicación para ajustarse a la pantalla actual del dispositivo. En la mayoría de los casos, esto funciona bien. En otros, tal vez la IU no se vea tan bien y necesite ajustes para diferentes tamaños de pantalla. Por ejemplo, en una pantalla más grande, se recomienda ajustar la posición y el tamaño de algunos elementos para aprovechar el espacio adicional de pantalla o, en una pantalla más pequeña, hay que ajustar ajustar los tamaños para que todo pueda ajustarse a la pantalla. Para ello hay que realizar diferentes ficheros layout XML, uno para cada tamaño de pantalla que se quiera soportar.
    - Los calificadores de configuración que se pueden usar para proporcionar recursos específicos de tamaño son small, normal, large y xlarge. Por ejemplo, los diseños para una pantalla extragrande deben entrar en `layout-xlarge/`[5]

```
MyProject/
    res/
        layout/             # default (portrait)
            main.xml
        layout-land/        # landscape
            main.xml
        layout-large/       # large (portrait)
            main.xml
        layout-large-land/  # large landscape
            main.xml
```

Ilustración 95. *Directorios para almacenar distintos layout en función del tamaño de la pantalla*

- Proporcionar diferentes elementos de diseño del mapa de bits para diferentes densidades de pantalla: De forma predeterminada, Android ajusta los elementos de diseño de mapa de bits (archivos `.png`, `.jpg` y `.gif`) y los elementos de diseño *nine-patch* (archivos `.9.png`) para que se representen en el tamaño físico correspondiente en cada dispositivo. Por ejemplo, si la aplicación solo proporciona elementos de diseño de mapa de bits para la densidad de pantalla media (mdpi) de referencia, el sistema luego los aumenta en una pantalla de densidad alta y los reduce en pantallas con una densidad baja. Este ajuste puede provocar alteraciones en los mapas de bits. Para asegurar que los mapas de bits se vean de la mejor forma posible, es recomendable incluir versiones alternativas en resoluciones diferentes para diferentes densidades de pantalla.
    - Los calificadores de configuración que se pueden usar para los requisitos específicos de densidad son ldpi (baja), mdpi (media), hdpi (alta), xhdpi (extraalta), xxhdpi (extra extraalta) y xxxhdpi (extra extra extraalta). Por ejemplo, los mapas de bits para pantallas de alta densidad deben ir en `drawable-hdpi/`.

[5] A partir de Android 3.2 (nivel de API 13), los grupos de tamaño anteriores dejaron de estar disponibles y hay usar el calificador de configuración `sw<N>dp` para definir el ancho mínimo disponible que los recursos de diseño requieren. Por ejemplo, `layout-sw600dp/`

```
MyProject/
    res/
        drawable-xhdpi/
            awesomeimage.png
        drawable-hdpi/
            awesomeimage.png
        drawable-mdpi/
            awesomeimage.png
        drawable-ldpi/
            awesomeimage.png
```

Ilustración 96. *Directorios para almacenar distintos mapas de bits en función del tamaño de la pantalla*

8.1.9.4 COMPATIBILIDAD CON DIFERENTES VERSIONES ANDROID

Mientras que la última versión de Android proporciona grandes APIs para la aplicación, hay que preocuparse por dar soporte a versiones anteriores de Android hasta que la mayoría de los dispositivos se actualicen.

Es recomendable dar soporte al 90% de los dispositivos Android activos antes de crear una aplicación que soporte solo la última versión Android disponible. El gráfico para la plataforma de versiones de Android se actualiza regularmente para mostrar la distribución de dispositivos activos de cada versión de Android, basándose en el número de dispositivos que visitan Google Play.

En el fichero AndroidManifest.xml se describen los atributos `minSdkVersion` y `targetSdkVersion` para el elemento `<uses-sdk>` que identifica la API más baja que soporta la aplicación y la API más alta con la que se ha creado la aplicación. En la Ilustración 97 se muestra la declaración de estos atributos en el fichero correspondiente.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" ... >
    <uses-sdk android:minSdkVersion="4" android:targetSdkVersion="15" />
    ...
</manifest>
```

Ilustración 97. *Declaración del elemento `<uses-sdk>`*

A medida que se lanzan nuevas versiones de Android, algunos estilos y comportamientos pueden cambiar. Para permitir que la aplicación aproveche estos cambios y te asegures de que la aplicación se ajuste al estilo del dispositivo de cada usuario, hay que establecer el valor `targetSdkVersion` para que coincida con la última versión de Android disponible.

Android proporciona un código único para cada versión de la plataforma en la clase `Build`. Si se utiliza el código que se muestra en la Ilustración 98, nos aseguramos que el código que depende de niveles de API más altos se ejecute sólo cuando esas APIs están disponibles en el sistema.[6]

```
private void setUpActionBar() {
    // Make sure we're running on Honeycomb or higher to use ActionBar APIs
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
        ActionBar actionBar = getActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);
    }
}
```

Ilustración 98. *Código para ejecutar la API más alta disponible en el sistema*

Otro método para proporcionar compatibilidad con versiones anteriores es la utilización de bibliotecas de compatibilidad. Las bibliotecas de compatibilidad permiten que las Apps se ejecuten en versiones anteriores de la plataforma de Android para admitir funciones disponibles en versiones más nuevas de la plataforma. Por ejemplo, una App que se ejecuta versiones de Android anteriores a la 5.0 (nivel de API 21) y se basa en clases de *framework* no puede mostrar elementos de `material design`, ya que esa versión del *framework* de Android no lo admite. Sin embargo, si la app incorpora la biblioteca `appcompat` de la biblioteca de compatibilidad, tendrá acceso a muchas funciones disponibles en el nivel de API 21, incluida la compatibilidad con `material design`. Como resultado, la App puede brindar una experiencia más uniforme en una amplia variedad de versiones de plataformas.

[6] Al analizar los recursos XML, Android ignora los atributos XML que no son compatibles con el dispositivo actual.