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 APRENDIZAJE DE HEPATOLOGÍA EN UN CONTEXTO DE "FLIPPED CLASSROOM"

AUTORA:
DÑA. ESTHER MARTÍN GONZÁLEZ

TUTORA:
DRA. DÑA. MÍRIAM ANTÓN RODRÍGUEZ

Valladolid, 11 de julio de 2017

TÍTULO: APLICACIÓN ANDROID PARA EL APRENDIZAJE DE HEPATOLOGÍA EN UN CONTEXTO DE "FLIPPED CLASSROOM"

AUTOR: ESTHER MARTÍN GONZÁLEZ

TUTORES: MÍRIAM ANTÓN RODRÍGUEZ

DEPARTAMENTO: TEORÍA DE LA SEÑAL, COMUNICACIONES E INGENIERÍA TELEMÁTICA

Miembros del Tribunal

PRESIDENTE: MÍRIAM ANTÓN RODRÍGUEZ

SECRETARIO: DAVID GONZÁLEZ ORTEGA

VOCAL: MARIO MARTÍNEZ ZARZUELA

SUPLENTE 1: FRANCISCO JAVIER DÍAZ PERNAS

SUPLENTE 2: M.ª ÁNGELES PÉREZ JUÁREZ

FECHA DE LECTURA: Julio 2017

CALIFICACIÓN:

RESUMEN DEL PROYECTO

Hoy en día las aplicaciones móviles nos facilitan muchos aspectos de nuestra vida. Este trabajo de fin de grado busca el desarrollo de una aplicación que ayude en el aprendizaje de conocimientos a nivel universitario. Para ello se analizan las diferentes opciones que a día de hoy nos encontramos para el desarrollo de Apps, tanto diferentes tipos de desarrollo (web, híbrido y nativo) como los sistemas operativos más importantes (Android, iOS y Windows Phone).

Después de la elección, se desarrolla una aplicación móvil nativa para el sistema operativo *Android* con la finalidad de dar solución al aprendizaje de hepatología en un contexto *Flipped Classroom*. Las aplicaciones basadas en *Flipped Classroom* se encuentran en continuo auge en todos los niveles educativos, ya que facilitan el acceso a contenidos de gran valor para el alumnado, permitiendo un aprendizaje más dinámico en las aulas.

PALABRAS CLAVE

Flipped Classroom, Android, hepatología, aplicación, java, calculadora.

ABSTRACT

Nowadays mobile applications facilitate aspects of our life. This report seeks the development of an application that helps in the learning of knowledge at university level. For this we analyze the different options that we have nowadays for the development of Apps, both different types of development (web, hybrid and native) and the most important operating systems (Android, iOS and Windows Phone).

After the election, a native mobile application is developed for the Android operating system in order to give solution to hepatology learning in a Flipped Classroom context. Applications based on Flipped Classroom are constantly booming in all educational levels, since they facilitate the access to contents of great value for the students, allowing a more dynamic learning in the classrooms.

KEY WORDS

Flipped Classroom, Android, hepatology, application, java, calculator.

Índice de contenido

Índice de figuras

Índice de ecuaciones

Índice de tablas

Capítulo 1: Introducción

La tecnología se ha convertido en parte de nuestro día a día, resultando difícil desconectarse de ella. Uno de los dispositivos más utilizados actualmente es el Smartphone, lo que ha con llevado una demanda cada vez mayor de software específico.

Tanto los Smartphone como las Tablet permiten el acceso a un gran número de contenidos y facilitan la difusión y el uso compartido de información. Esta característica ha impulsado el desarrollo de aplicaciones para el uso académico.

Las TIC están provocando un cambio en el enfoque pedagógico, siento una tendencia mayoritaria su integración en diferentes niveles educativos. Esto se debe a la importancia del aprendizaje activo que facilita la tecnología. De este cambio surge el término de e-learning, el cual se refiere a la capacitación, formación y educación a través de internet. Con el fin de facilitar la experiencia del aprendizaje se crea un espacio virtual destinado a la enseñanza. Es un concepto educativo reciente que revoluciona la forma de enseñar, de la que han surgido diferentes modelos de aprendizaje que han cambiado la enseñanza tradicional (Menárguez, 2016).

A nivel universitario, las instituciones han venido realizando cambios para convertirse en un entorno cada vez más tecnológico mediante la incorporación de diferentes aplicaciones para el aprendizaje como son las wikis, aulas virtuales, foros, blogs…También es fácil encontrar repositorios virtuales, transmisiones en directo de conferencias, acceso online a programas especializados y plataformas educativas.

Se denomina E-learning a la capacitación, formación y educación a través de internet. A grandes rasgos, es un espacio virtual destinado a la enseñanza creado con el fin de facilitar la experiencia del aprendizaje. Es un concepto educativo reciente que revoluciona la forma de enseñar, brinda una oportunidad facilitada en varios aspectos para lograr una mejor capacitación. En la actualidad existen otros términos que pueden utilizarse como sinónimos, como por ejemplo ¨educación on-line¨ o ¨enseñanza virtual¨ (e-ABC, 2011).

Uno de los mayores defensores de la integración de las TIC en la educación es Jon Bregmann. Reconocido por la casa blanca en 2002 como el mejor profesor de estados unidos de matemáticas y ciencias. Desarrollo en llamado Flipped Classroom rompiendo con la metodología tradicional de enseñanza. Esta metodología consiste en que el alumno realice las tareas menos llamativas en casa y dejar las clases para resolver dudas, trabajar en equipo o investigar. En la actualidad más de 20 universidades investigan las ventajas de este modelo de aprendizaje. Las conclusiones más recurrentes de algunos de estos estudios son el aumento de alumnos activos en clase y la motivación de los profesores (Menárguez, 2016).

Debido a las ventajas y el crecimiento del Flipped Clasroom, se pensó en una aplicación que ayudara en el aprendizaje de la Hepatología, poniendo en práctica este modelo educativo.

1.1 Objetivos

La finalidad de este trabajo de fin de grado es diseñar una aplicación para el autoaprendizaje de una asignatura universitaria.

Para conseguirlo se han seguido una serie de pasos:

- Investigar sobre los contenidos y necesidades que debe cubrir la aplicación.
- Elegir la tecnología móvil más adecuada.
- Conseguir que la aplicación sea fácil de usar, escalable y dinámica.

1.2 Estructura

Para poder abarcar estos objetivos y seguir se desarrollo de manera fluida, este informe se ha dividido en 6 capítulos.
El primer capítulo, en el cual nos encontramos, se realiza una introducción a los contenidos del trabajo y se contextualiza la finalidad y los objetivos.
En el capítulo dos, se definen diferentes opciones a la hora de desarrollar una aplicación móvil, indicando sus ventajas y desventajas. Una vez elegido el tipo de aplicación, se explican las características de los principales sistemas operativos móviles y su importancia en el mercado actual.
A continuación, se detalla con más profundidad Android Kitkat ya que es el sistema operativo y la versión en la que se desarrolla la aplicación HepApp.
En el capítulo 4 se realiza una descripción técnica de las funcionalidades de la aplicación. Mediante las tablas de casos de uso, diagramas de flujo y las ecuaciones en las que se basan las calculadoras.
Seguidamente se hace una descripción a nivel de usuario, explicando a través de ilustraciones cómo funciona la aplicación y que aspecto tiene. En este capítulo se pretende enseñar y facilitar al usuario el manejo de la app.
Para finalizar este trabajo, se describen las conclusiones obtenidas tras la realización de este trabajo de fin de grado. Además, se indica un estudio económico con el fin de conocer los gastos que supone el desarrollo de una aplicación como HepApp y las ideas futuras que se podrían llevar a cabo para mejorarla.

Capítulo 2: Tecnologías móviles

En los últimos años la presencia de dispositivos móviles en nuestra vida cotidiana ha ido en aumento. Hasta tal punto que actualmente hay más dispositivos Android que equipos con el sistema operativo Windows (Statista, 2017).
El crecimiento de las diferentes plataformas móviles supone un reto a la hora de desarrollar aplicaciones que proporcionen una solución para todas ellas sin que el coste sea muy elevado.
El desarrollo de aplicaciones Web, hibridas y nativas, proporcionan diferentes soluciones para este problema.

2.1 Aplicaciones Web

La principal ventaja de esta tecnología (Delía, Galdamez, Thomas, & Pesado, s.f) es que no necesita la instalación de ningún componente en particular, ni la aprobación para que las aplicaciones sean publicadas. Debido a que los cambios se aplican en el servidor, las actualizaciones son visualizadas directamente en el dispositivo. Esto hace que sean rápidas de actualizar y de poner en marcha. Esta tecnología es independiente de la plataforma por lo que no son necesaria las modificaciones para adecuarse a los distintos sistemas operativos.

Por lo contrario, disminuyen la velocidad de ejecución y son menos atractivas que las aplicaciones nativas. Además, podrían tener bajo rendimiento por problemas de conectividad. Otra desventaja importante surge debido a que este tipo de aplicaciones no pueden utilizar todos los elementos de hardware, como por ejemplo, cámara, GPS, entre otros.

2.2 Aplicaciones Híbridas

Las aplicaciones híbridas utilizan tecnologías multiplataforma como HTML, Javascript y CSS, pero se puede acceder a buena parte de las capacidades específicas de los dispositivos. Son desarrolladas utilizando tecnología web y son ejecutadas dentro de un contenedor web sobre el dispositivo móvil.
Entre las principales ventajas nos encontramos con la posibilidad de distribución de la aplicación a través de las *App Stores*, la reutilización de código para múltiples plataformas y la posibilidad de utilizar las características de *hardware* del dispositivo.

Una de las principales desventajas frente a las aplicaciones nativas está en la apariencia, ya que al utilizar la misma interfaz para todas las plataformas, esta no será igual de intuitiva. Este tipo de aplicaciones ocupan un mayor espacio de memoria en el dispositivo y el tiempo de ejecución es mayor que en una aplicación nativa.

2.3 Aplicaciones Nativas

Las aplicaciones nativas (Delía et al., s.f) son aquellas que se desarrollan para ejecutarse en una plataforma específica, es decir, se debe considerar el tipo de dispositivo, el sistema operativo a utilizar y su versión. Al igual que en las tradicionales aplicaciones de escritorio, el código fuente se compila para obtener código ejecutable. Cuando la aplicación está lista para ser distribuida debe ser transferida a las App stores (tiendas de aplicaciones) específicas de cada sistema operativo. Estas tienen un proceso de auditoría para evaluar si la aplicación se adecúa a los requerimientos de la plataforma a operar. Cumplido este paso, la aplicación se pone a disposición de los usuarios.

La principal ventaja de este tipo de aplicaciones es la posibilidad de interactuar con todos los elementos *hadware* del dispositivo (cámara, GPS, acelerómetro, agenda, entre otras). Además no es estrictamente necesario poseer acceso a internet. Su ejecución es rápida, puede ejecutarse en modo *background* y notificar al usuario cuando ocurra un evento que necesite su atención. Claramente estas ventajas tienen como contrapartida un mayor tiempo de desarrollo debido a que utiliza un lenguaje de programación diferente según la plataforma. Además, si se desea cubrir varias plataformas, se deberá generar una aplicación para cada una de ellas. Esto conlleva a mayores costos de actualización y distribución de nuevas versiones. Esta desventaja se ve reducida gracias a la plataforma Xamarin (Xamarin, 2017), ya que permite a los desarrolladores programar aplicaciones nativas para Android, iOS y Windows median el lenguaje C#. Tambien permite compartir código a través de múltiples platformas como macOS y Windows.
A pesar de esta desventaja, la experiencia de usuario es mejor que en los otros tipos de aplicaciones debido a su mayor rendimiento y a su menor tamaño.
El enfoque nativo destaca por su rendimiento y acceso de los dispositivos, pero conlleva mayor coste y requiere actualizaciones. El enfoque Web es mucho más simple, menos costoso y más fácil de actualizar, pero actualmente su funcionalidad es limitada y no puede alcanzar un alto nivel de experiencia del usuario como el de las llamadas API nativas.

2.3.1 Sistema Operativo Windows Phone

Windows Phone (Microsoft, 2017) es un sistema operativo móvil desarrollado por Microsoft, como sucesor de la plataforma Windows Mobile. A diferencia de su predecesor, está enfocado en el mercado de consumo generalista en lugar del mercado empresarial por lo que carece de muchas funcionalidades que proporcionaba la versión anterior. Microsoft ha decidido no hacer compatible Windows Phone con Windows Mobile por lo que las aplicaciones existentes no funcionan en Windows Phone haciendo necesario desarrollar nuevas aplicaciones. Con Windows Phone, Microsoft ofrece una nueva interfaz de usuario que integra varios servicios en el sistema operativo. Microsoft planeaba un estricto control del hardware que implementaría el sistema operativo, para evitar la fragmentación con la evolución del sistema, pero han reducido los requisitos de hardware de tal forma que puede que eso no sea posible. Debido a la evidente

fragmentación de sus sistemas operativos, Microsoft dio de baja a Windows Phone, para enfocarse en un único sistema más versátil denominado Windows 10 Mobile, disponible para todo tipo de plataformas.
La razón principal de esto, es pretender que los usuario entiendan sus apps de manera universal, sin importar que tipo de dispositivo se esté usando (El comercio, 2015).

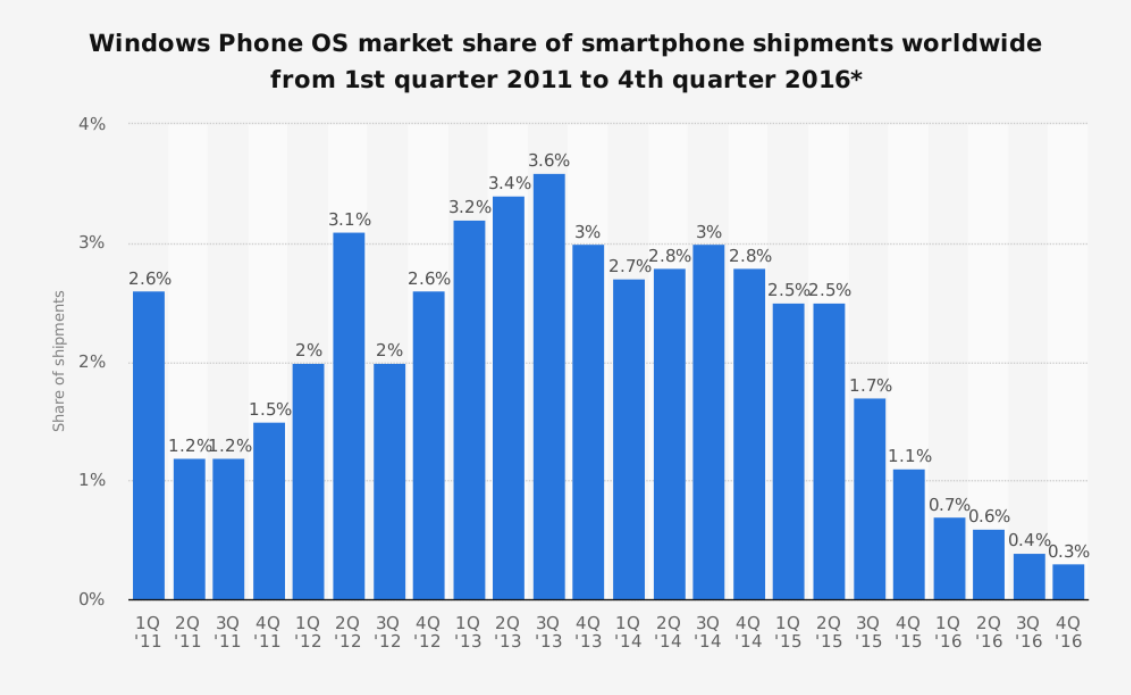


Figura 1: Porcentaje de mercado de Smartphone con Windows Phone (Statista, 2017).

La arquitectura del sistema operativo Windows Phone tiene 4 capas definidas (Torre, 2013):

- *Hardware*: Es la capa que representa cualquier hardware de dispositivo móvil existente en el que se encuentre instalada el sistema operativo.

- *Kernel*: En la capa del núcleo se encuentran los drivers, el sistema de archivos, las redes, el sistema de reinderizado, los gráficos, el sistema de actualizaciones entre otros.

- *Modelo*: 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.

- *Application Runtime:* En esta capa de la aplicación se ejecutan todas las aplicaciones del sistema operativo.

La gran mayoría de las críticas hacia este sistema operativo están enfocadas en la excesiva fragmentación y las incompatibilidades entre sus diversas versiones.

La primera versión de este sistema operativo fue *Windows Phone 7* (Torre, 2013), la cual reemplazó a *Windows Mobile*. Tiene una interfaz *Modern UI*, que luego fue implementada en las versiones posteriores. Las aplicaciones para esta versión no son compatibles con *Windows Mobile 6* debido a que la extension es *.xap*.

Windows Phone 8 (Branscombe, 2014) es la segunda generación, fue lanzada el 14 de septiembre de 2012. Esta versión sustituye la arquitectura previamente basada en Windows CE por una basada en el kernel de Windows NT. Windows 8 soporta tamaños de pantalla mayores al igual que procesadores multi-núcleo. Incorpora más funciones destinadas al mercado de la empresa, como son la administración de dispositivos, encriptación BitLocker y la capacidad de distribución de aplicaciones de manera privada entre los empleados.

Windows Phone 8 no está disponible como actualización de Window Phone 7, por lo que Microsoft lanzó Windows Phone 7.8, que tiene una interfaz similar a esta versión.

2.3.2 Sistema Operativo Windows 10 Mobile

Windows 10 *mobile* (Callaham, 2014) es parte de las ediciones de Windows 10 y sucesor de Windows Phone 8.1. Su finalidad principal es llevar la integración y unificación con el sistema operativo de PC Windows 10, proporcionando una plataforma para smartphons y tablets de pequeño tamaño (8 pulgadas). Las especificaciones mínimas para Windows 10 son similares a Windows Phone 8, con una resolución de pantalla mínima de 800x480 y 1 GB de RAM.

Este sistema operativo añadió la sincronización de notificaciones entre dispositivos, la compatibilidad, se mejoró la usabilidad de la pantalla de inicio y del teclado de tal manera que se pudiera usar una mano en dispositivos de patalla grande. A pesar de esto, la mejora principal fue *Continuum*, que nos permite usar el teléfono como si fuera un ordenador con tan solo conectarle una pantalla. Al conectar el móvil a una pantalla externa se ejecuta la aplicación de siempre, pero vemos la interfaz de escritorio en lugar de la del móvil.

Windows 10 (Dudley, 2014) introduce la “Plataforma Windows Universal” (UWP) que incluye el modelo de Windows Runtime y lo lleva a Core unificado de Windows 10. Como parte del core, el UWP proporciona la disponibilidad de plataforma de aplicaciones comunes para cada dispositivo que ejecute Windows 10. Esto significa que se puede crear una sola aplicación que puede ser instalada en un rango muy grande de dispositivos.

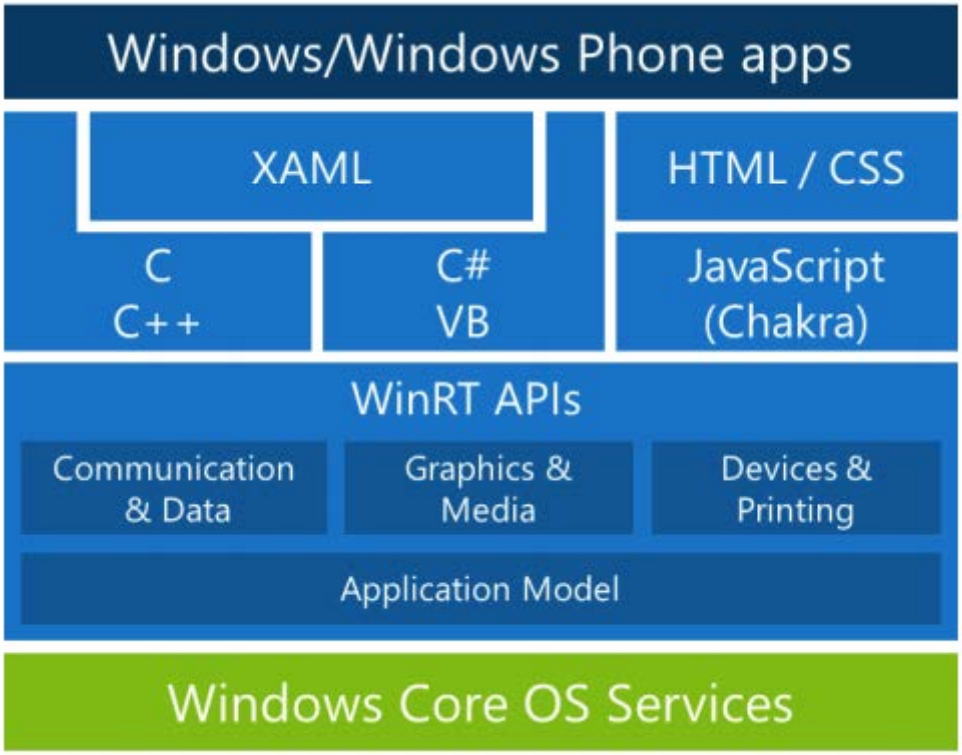


Figura 2: Arquitectura Windows 10

2.3.3 Sistema Operativo iOS

iOS (Apple, 2014) es un sistema operativo que se desarrolló inicialmente para el iPhone, propiedad de Apple Inc. Actualmente se usa también en dispositivos como el iPod touch y el iPap.

El sistema operativo fue presentado en enero de 2007 en la Macworld Conference & Expo, aunque no tuvo nombre oficial hasta la primera versión beta del iPhone SDK en el 2008, desde ese momento se llamaría iPhone OS. No se llamó iOS hasta el lanzamiento del iPad en 2010.

La tienda de aplicaciones de Apple contenía alrededor más de 2 millones de aplicaciones en enero de 2017 (Figura 3). En septiembre de 2016, Apple anunció que había alcanzado la cifra de 140 mil millones de descargas. El número de descargas en 2016 aumentó en mayor medida que en los años anteriores como podemos ver en la Figura 4 (Statista, 2017).

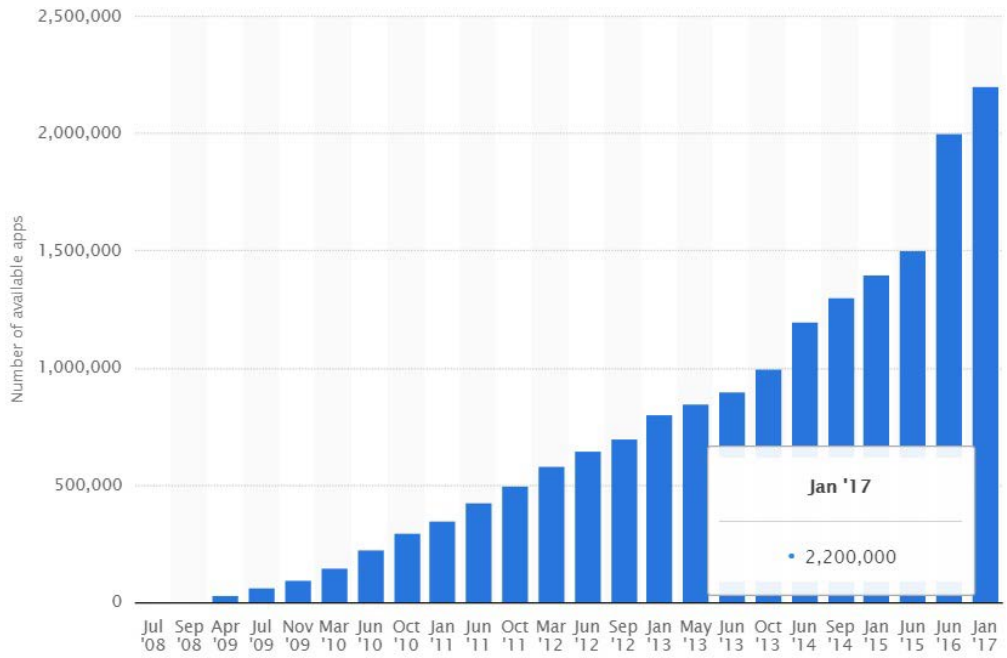


Figura 3: Número de aplicaciones disponibles en la App Store desde Julio de 2008 hasta Enero de 2017 (Statista, 2017).

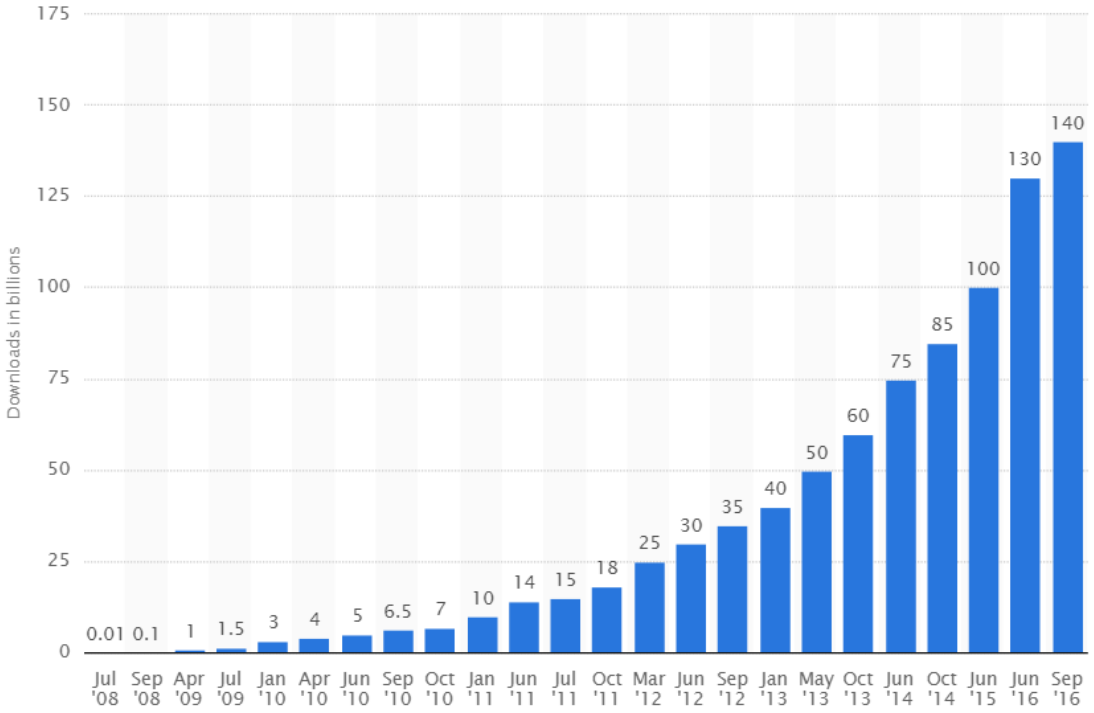


Figura 4: Número de descargas de aplicaciones de la App Store (Statista, 2017).

Este sistema operativo, quiso ser desde un primer momento un derivado del sistema para ordenadores de la compañía, *Mac OS X*. De hecho, está basado en *Darwin BSD*, derivado de *Unix*. Está basado en una arquitectura de cuatro capas, en las que las capas inferiores ofrecen servicios a bajo nivel, mientras que las superiores contienen servicios y tecnologías mucho más sofisticadas y complejas.

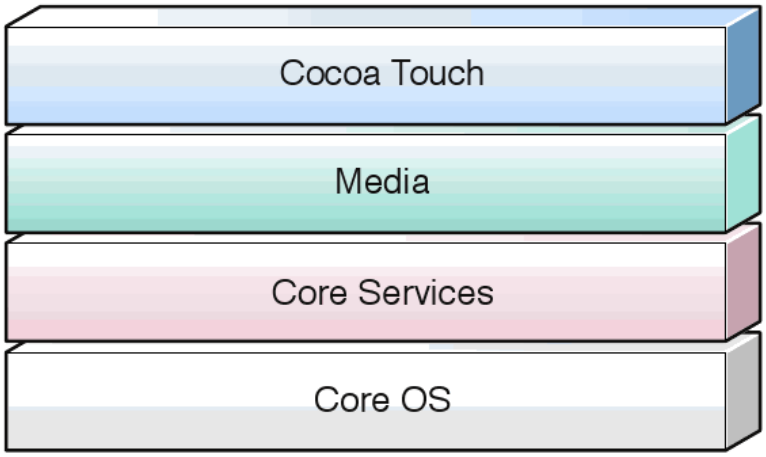


Figura 5: Arquitectura iOS

- *Cocoa Touch*: Cocoa Touch en la capa más importante para el desarrollo de aplicaciones iOS. Posee un conjunto de *Frameworks* que proporcionan el API de Cocoa para desarrollar aplicaciones.
  Hasta 2014 esta librería solo se podía programar en *Objective-C*, actualmente se puede usar el lenguaje *Swift* versión 4.
  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

- *Media*: los servicios de gráficos y multimedia a la capa superior. Implementa servicios multimedia como *OpenAI*, reproducción de video, formatos de archivos de imágenes…

- *Core Services*: Contiene los servicios fundamentales del sistema que usan todas las aplicaciones, como por ejemplo *Networking*, base de datos *SQLite*, soporte XML…

- *Core OS*: Contiene las características de bajo nivel: ficheros del sistema, manejo de memoria, seguridad, drivers del dispositivo. Actúa como núcleo del sistema operativo.

La versión actual de iOS es la 10.0, dedica alrededor de 1.8 Gb de memoria de los dispositivos al propio sistema. Esta versión funciona en dispositivos superiores al iPhone 5, iPad 4, iPad Pro, iPad Mini 2 y iPod Touch 6G.

iOS presenta más ventajas a parte de la facilidad de actualización, como la usabilidad del framework Cocoa Touch. Presenta una gran comunidad de programadores, los que facilita la disponibilidad de información para nuevos programadores. Esto hace que resulte asequible iniciarse en la programación de aplicaciones nativas para iOS lo que a su vez conlleva el continuo aumento de esta comunidad.

Las aplicaciones presentes en la App Store aseguran un mínimo de calidad y de seguridad debido a que Apple restringe y supervisa todas las aplicaciones antes de subirlas. Además estas aplicaciones están diseñadas para una arquitectura particular ya que el número de dispositivos de Apple está limitado. El resultado es que las apps suelesn ser de mayor calidad que en Android (Apple, 2014).

Por otro lado, Objective-C es un lenguaje de programación que no es tan amigable como el empleado en Android, aunque esta desventaja se reduce con la aparición de Swift. La libertad de desarrollo es menor ya que Apple restringe las modificaciones de los componentes de su *framework*. Probablemente la mayor desventaja de iOS es la necesidad de utilización de una serie de terminales especiales. Es decir, se necesita un MAC (existe la alternativa de *Hackintosh* o una máquina virtual), y tener el sistema operativo *Mac OS X* para poder desarrollar aplicaciones nativas en iOS. A estas desventajas se le suma el hecho de tener que pagar una cuota anual de 99$ para poder poder subir las aplicaciones a la *App Store* (Apple, 2014).

2.3.4 Sistema Operativo Android

Android es una plataforma libre creada inicialmente por *Android Inc*, empresa que *Google* respaldó económicamente y la cual compró en 2005. Es un sistema operativo basado en el núcleo de *Linux*. Fue presentado en 2007 junto a la fundación del *Open Handset Alliance*, un consorcio de compañías hardware, sotfware y telecomunicaciones, formado por *Google*, *Intel*, *Texas Instruments*, *T-Molile*, entre otros (Gironés, 2016).
A la fecha, se ha llegado ya a la cifra 2.800.000 (Figura 6) aplicaciones disponibles en *Google Play* (tienda oficial de aplicaciones para android) sin tener en cuenta aplicaciones de otras tiendas no oficiales para Android como la tienda de aplicaciones *Samsung Apps* de *Samsung*, *Slideme* de Java y *Amazon Appstore*.
Google Play es la tienda de aplicaciones en línea administrada por Google, aunque existe la posibilidad de obtener software externamente. La tienda *F-Droid* es completamente de código abierto así como sus aplicaciones. Los programas están escritos en el lenguaje de

programación Java. Sin embargo, no es un sistema operativo libre de *malware*, aunque la mayoría de ello es descargado de sitios de terceros (Gironés, 2016).

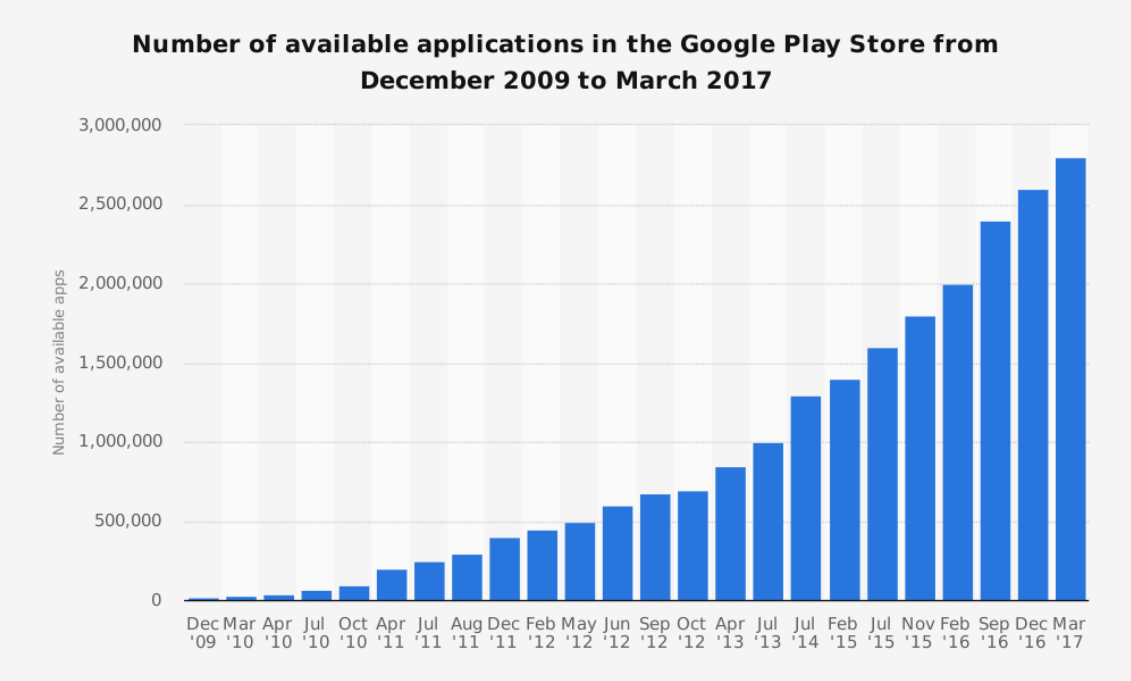


Figura 6: Número de aplicaciones disponibles en Google Play desde Diciembre de 2009 hasta Marzo de 2017 (Statista, 2017).

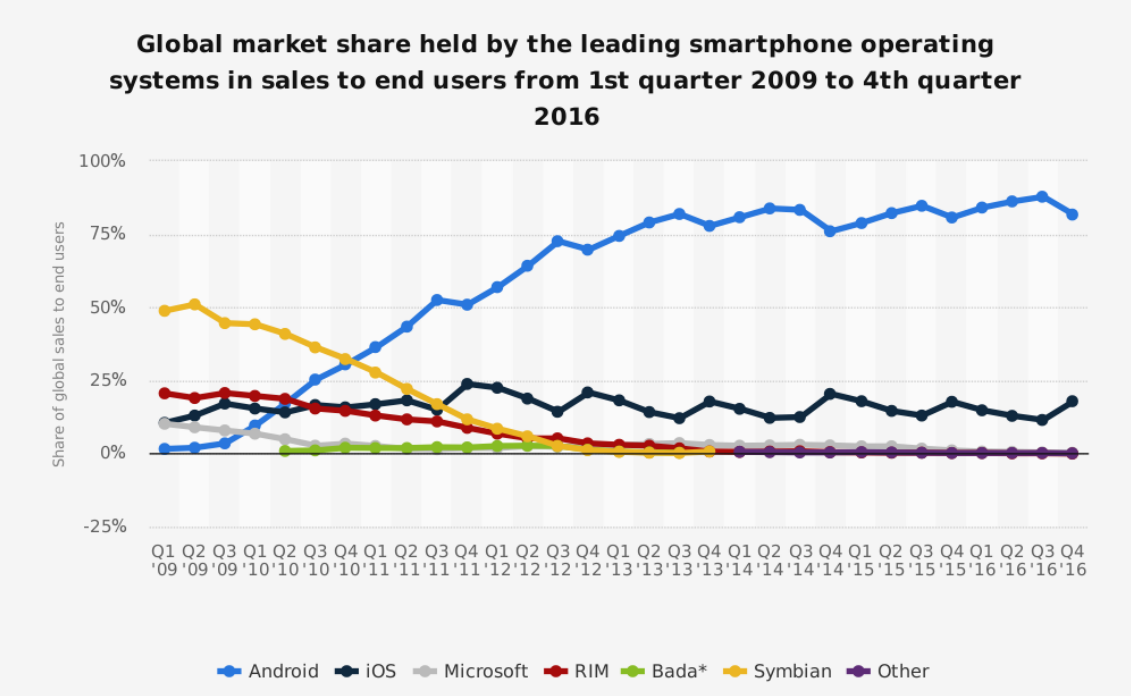


Figura 7: Reparto del mercado por sistema operativo de Smartphone (Statista, 2017).

En la Figura 7, podemos observar como Android es el líder del mercado desde el 2011. Actualmente más del 81% de smartphones son Android. Pese a esto, el número de

descargas de aplicaciones de *Google Play* en mayo de 2016 es de 65 mil millones (Figura 8), cifra bastante menor que la de la tienda oficial de Apple (ver Figura 3).

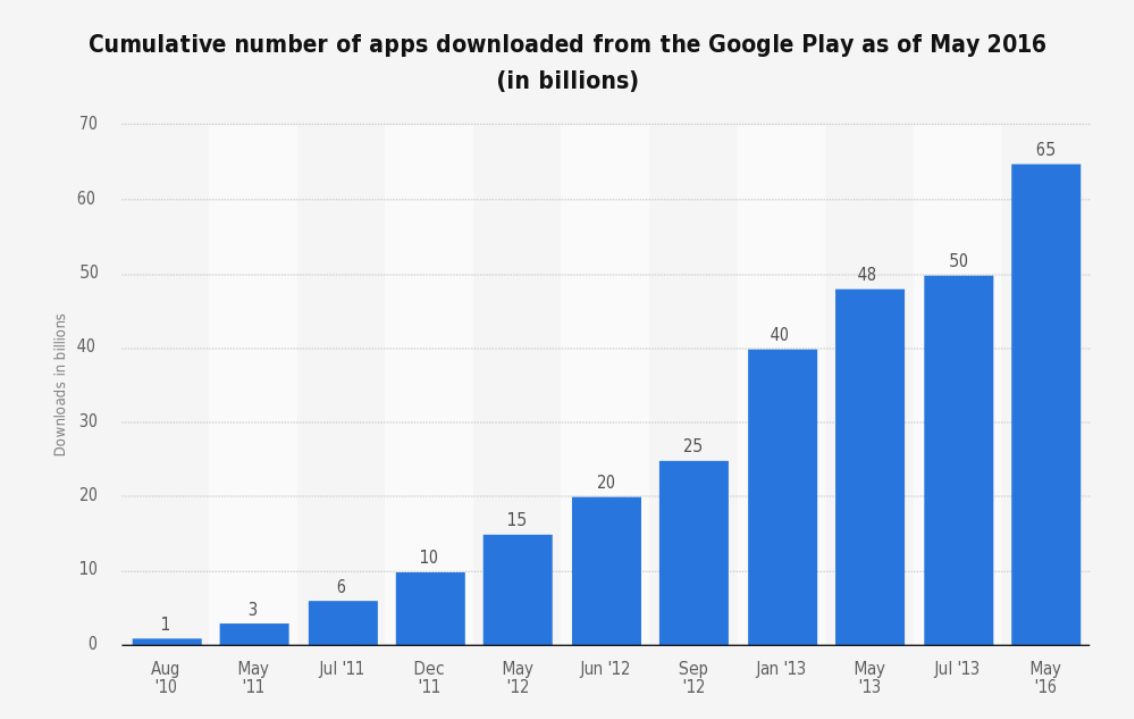


Figura 8: Número de descargas de aplicaciones de Google Play (Statista, 2017).

Las aplicaciones son desarrolladas generalmente en Java empleando la herramienta *Android Software Development Kit* (Android SDK), la cual se puede obtener de manera gratuita. Android SDK el cual está formado por un conjunto de herramientas necesarias para el desarrollo de aplicaciones nativas y las APIs necesarias para basar estas herramientas en Java. Las aplicaciones Android de pueden desarrollar en cualquier sistema operativo.El hecho de que use en lenguaje de programación Java es una ventaja ya que se encuentra ampliamente extendido.

A partir de la versión 3 de Android Studio, se incluye el lenguaje de programación Kotlin (Kotlin, 2017) ,el cual corre en la máquina virtual de Java. . Este lenguaje ayuda a mejorar los tiempos de ejecución y reducir el numero de lineas de código de las aplicacones. Además, se puede introducir de forma gradual en los proyectos ya desarrollados.

Para la publicación de una aplicación solo es necesario registrarse y subir el APK. Tras un par de horas la aplicación se podrá descargar de la tienda de Google. Poder subir aplicaciones cuesta un pago único de 25$. En cambio, esto mismo para iOS nos costaría más dinero y tiempo.

Android es un sistema más abierto y “libre” para el desarrollador, permitiéndole crear apps sin tantas trabas, pudiendo tener en una app un control mucho más amplio del terminal que en iOS. Android incluye todas las máquinas virtuales de todas las versiones

anteriores para que no se altere el funcionamiento de una app ya subida por un cambio en algún componente de una versión nueva.

Por otro lado, la principal desventaja de Android se debe al gran número de fabricantes y por lo tanto de diferentes dispositivos. Esta necesidad ya se tuvo en cuenta desde los inicios de desarrollo del sistema operativo, por lo que se adapta mejor a los diferentes dispositivos que los demás sistemas operativos. Sumado a esto, implementa una multitarea real. Si una aplicación pasa a segundo plano, ésta se sigue ejecutando en el procesador del terminal. En un principio esto no sería un inconveniente, pero si el usuario tiene muchas tareas abiertas, el dispositivo se ralentizará (Rodríguez, 2016).

2.4 Elección de la tecnología

Lo primero a tener en cuenta es que la aplicación ya está desarrollada para iOS aunque no de forma nativa, por lo que presenta problemas de rendimiento. Para evitar que la nueva aplicación tenga estos mismos problemas, la mejor opción es el desarrollo nativo.
Como se ha podido ver, la mayor parte del mercado son dispositivos Android por lo que elegir esta tecnología haría que la aplicación estuviera presente en los dos tipos de sistemas operativos con mayor tasa de mercado.
En cuanto a las diferentes versiones de Android, cuanto menor sea la versión mayor cupo de mercado abarcaremos, pero habrá disponibles menores funcionalidades. Al igual que si la versión en muy alta el nicho de mercado se reduce.

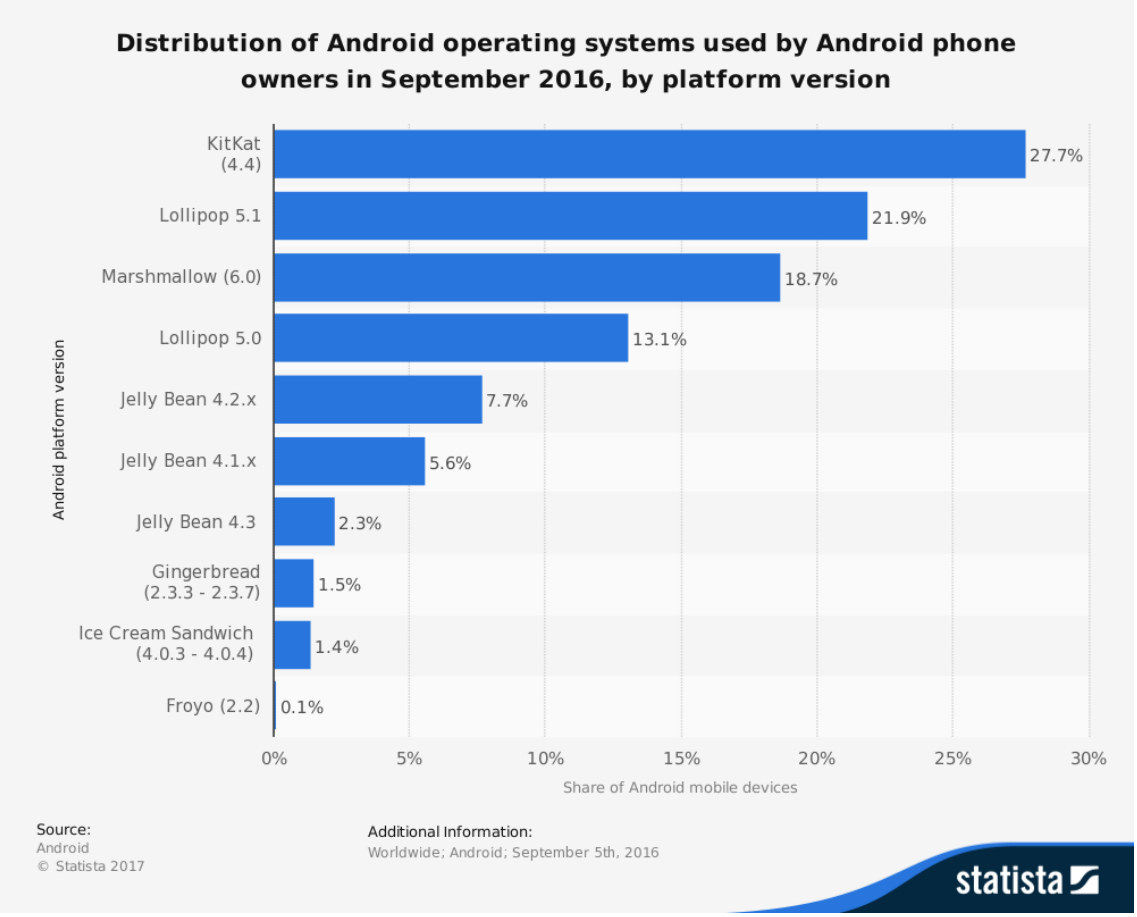


Figura 9: Distribución de las versiones de Android según su uso en 2016 (Statista, 2017).

Como se puede observar en la Figura 9, la mayoría de los dispositivos *Android* tienen instalada la versión *KitKat* (4.4). Al elegir esta versión se cubre el 81.4% de los dispositivos.

Capítulo 3: Tecnología utilizada. Android KitKat

Android es un sistema operativo creado por *Android Inc*, empresa que en Agosto de 2005 fue comprada por *Google*. Hasta el 5 de Noviembre de 2007 no fue presentado oficialmente a los medios de comunicación (Gironés, 2016).
Android es una plataforma para dispositivos móviles que contiene una pila de software formado por un sistema operativo, un entorno de ejecución de aplicaciones basado en java, un conjunto de bibliotecas de bajo y medio nivel y un conjunto inicial de aplicaciones para el usuario final (Gironés, 2016).
La primera versión fue *Apple Pie*, lanzado el 22 de Octubre de 2008. *HTC Dream* también conocido como *Google Phone* fue el primer dispositivo en incorporar este sistema operativo. Incluyó la primera versión de la *Android Market*, un navegador web, soporte para mensajes de texto *SMS* y *MMS*, y una aplicación para tomar fotos sin los ajustes de blancos y resolución.

Además se incluyeron algunas de las aplicaciones más populares de *Google* como *Google Maps*, *Google Sync* para sinronizar *Gmail*, Contactos y Calendario, *Google Search*, *Google Talk* y *YouTube*.

También se incluyó una aplicación capaz de acceder a los servidores de correo de terceros capaz de sincronizarse con aplicación de *Gmail*, *Google* y *Google Calendar*, con soporte para los estándares *POP3*, *IMAP4*, y *SMTP.14*.
La ventana de notificación desplegable, fue la gran novedad que introdujo Android, apostando desde el principio por un sistema de notificación con el que tener toda la información a la vista. Android ofreció desde un principio el soporte para *WiFi* y *Bluetooth*, además de la posibilidad de configurar alertas con LED, *ringtone* o vibración.

3.1 Versión 4.4 (KitKat)

Fue lanzado oficialmente el 31 de Octubre de 2013 junto con el *LG Nexus 5*. *Android* 4.4 KitKat introdujo una disminución del tamaño del sistema operativo junto con algunas pequeñas variaciones estética pero manteniendo la interfaz *Holo* (Android, s.f.).
Los cambios más visuales, fueron el incremento del tamaño de los íconos y la condensación del texto para una visión más clara y simple. Además, la zona de notificaciones que antes era negra, pasaba a ser transparente fusionando el resto de la pantalla. Sumado a esto, tanto la barra de navegación como la de notificaciones desaparecerían en ciertas aplicaciones para dejarnos ver en pantalla completa.

En cuanto al rendimiento, Android 4.4 KitKat trajo la implementación de zRAM, para optimizar el rendimiento en dispositivos con memoria RAM de 512MB y de esta manera incluir los Smartphone de gama baja de los mercados emergentes. Cabe mencionar que en 2013 la versión más utilizada de Android era *GingerBread* (Android 2.3) y esto se traducía en serias complicaciones para Google que trataba de impulsar a los fabricantes a incorporar sus nuevas funcionalidades y servicios. Además de una experiencia de usuario

más fluida que sólo se podía disponer en dispositivos con versiones más actuales y de especificaciones más elevadas.

Sumado a esto, Android 4.4 KitKat sustituyó algunos elementos de la interfaz anterior de azul a blanco para darle un aspecto más limpio, las horas del reloj ya no se mostrarían como números en negrita sino con una tipografía más fina tanto en las horas como en los minutos y un nuevo marco de transiciones y efectos visuales dentro de la interfaz *Holo*. Dentro de los cambios estéticos se creó un *widget* para la reproducción de música que se podía controlar desde la pantalla de bloqueo sin la necesidad de ingresar a la interfaz principal (Gironés, 2016).

En esta versión se implementó la funcionalidad para que el servicio de mensajería SMS oficial de Google fuera *Hangouts*. Además, se incorporó un editor de fotos dentro de la galería que nos permitiría realizar modificaciones en nuestras fotos sin la necesidad de aplicaciones de terceros.

En esta versión se desactivó el acceso de aplicaciones de terceros a las estadísticas de la batería y se movieron los monitores de actividad de red y señal al menú de ajustes rápidos.

El modo de inmersión en pantalla completa oculta todas las interfaces del sistema (barras de navegación y de estado) de tal manera que una aplicación puede aprovechar el tamaño de la pantalla completa. *WebViews* (componentes de la interfaz de usuario para mostrar las páginas Web) puede mostrar contenido basado en *HTM5* ya que pasa a basarse en el software de *Chrome* de Google y por lo tanto se mejora la conectividad con soporte de *NFC* para emular tarjetas de pago tipo *HCE*, varios protocolos sobre *Bluetooth* y soporte para mandos infrarrojos. Sumado a esto, se mejoran los sensores para disminuir su consumo y se incorpora un sensor contador de pasos (Android, s.f.).
Se facilita el acceso de las aplicaciones a la nube con un nuevo marco de almacenamiento. Este marco incorpora nuevas formas para abrir y crear documentos y una ventana de diálogo que permite al usuario seleccionar ficheros. Se incorpora un administrador de impresión para enviar documentos a través de *WiFi* a una impresora. Además,se añade un *content provider* para gestionar los SMS (Android, s.f.).

3.2 Arquitectura

La arquitectura de Android está formada por una serie de capas las cuales podemos observar en la Figura 10 (Madrid, 2010).

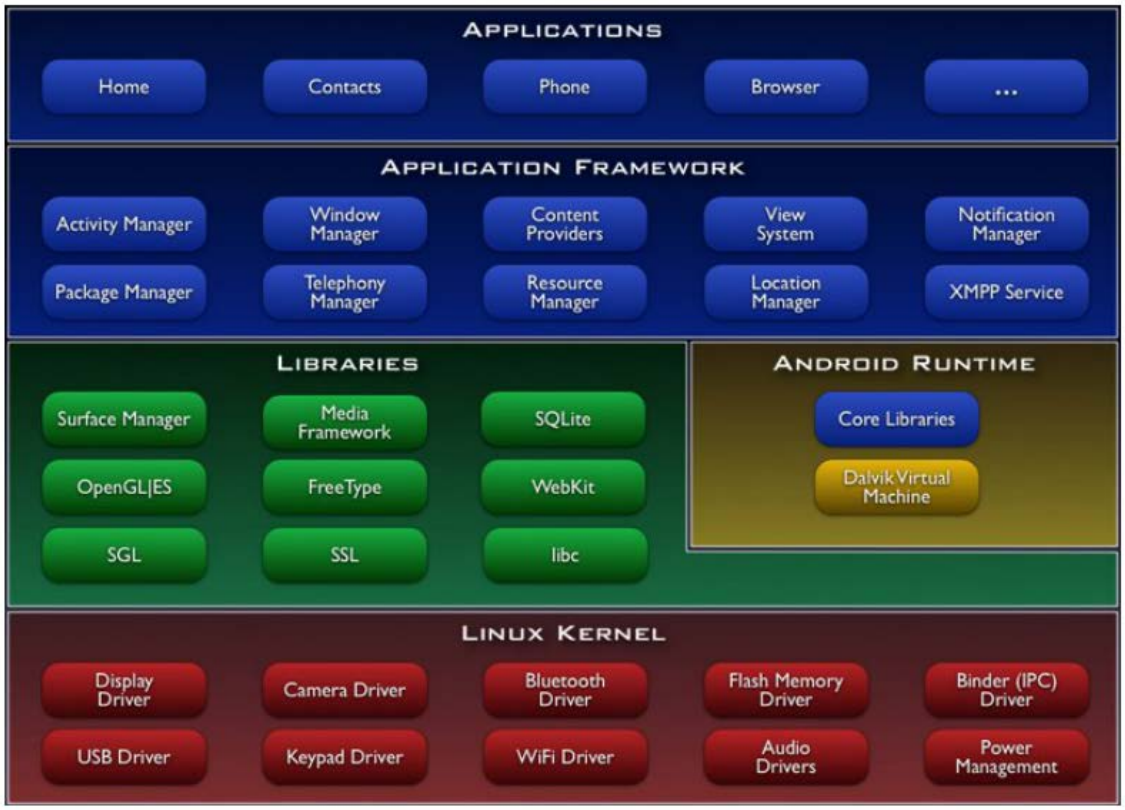


Figura 10: Arquitectura de Android

- Aplicaciones: Conjunto de aplicaciones instaladas en el sistema Android, tanto las incluidas por defecto como aquellas que el usuario añada posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores. Normalmente son aplicaciones escritas en java, aunque también hay la posibilidad de desarrollo en C/C++.
- *Framework* de Aplicaciones: Conjunto de herramientas para el desarrollo de aplicaciones. Todas las aplicaciones que se desarrolle para Android utilizan el mismo conjunto de API y el mismo "framework", representado por este nivel. Esta capa fue con el objetivo de reutilizar componentes. Permite a los usuarios reemplazar componentes de otras aplicaciones.
  Los servicios más importantes que incluye son:
  - *Activity Manager:* Conjunto de API que gestiona el ciclo de vida de las aplicaciones y proporciona un sistema de navegación entre ellas.
  - *Window Manager:* Gestiona las ventanas de las aplicaciones.
  - *Telephone Manager:* API vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).

- *Content Provider:* Permite a cualquier aplicación compartir sus datos con las demás aplicaciones. Por ejemplo, la información de contactos, agenda, mensajes, etc. será accesible para otras aplicaciones.
- *View System:* Incluye 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.
- *Location Manager:* Gestiona la obtención de información de localización y posicionamiento.
- *Notification Manager:* Comunican al usuario eventos que ocurran en durante su ejecución: una llamada entrante, un mensaje recibido, conexión Wi-Fi disponible, ubicación en un punto determinado, etc. Siguen un formato común.
- *XMPP Service:* Colección de API para utilizar este protocolo de intercambio de mensajes basado en XML.

- Librerías: Conjunto de librerías escritas utilizando C/C++ que proporcionan a Android la mayor parte de sus capacidades. Están compiladas en código nativo del procesador. Muchas librerías utilizan proyectos de código abierto.
  Entre las librerías más importantes se pueden encontrar las siguientes:
  - *Librería libc:* Incluye todas las cabeceras y funciones estándar del lenguaje C.
  - *Librería Surface Manager:* Compone 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. OpenGL/SL maneja gráficos en 3D y en 2D, por lo que será la librería más utilizada por la mayoría de las aplicaciones. Es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.
  - *Librería Media Libraries:* Proporciona todos los códecs necesarios para el contenido multimedia (vídeo, audio, imágenes estáticas y animadas, etc.)
  - *FreeType:* Maneja de forma rápida y sencilla distintos tipos de fuentes.
  - *Librería SSL:* Posibilita el establecimiento de comunicaciones seguras.
  - *Librería SQLite:* Creación y gestión de bases de datos.
  - *Librería WebKit:* soporte para las aplicaciones de tipo navegador. Se trata de la misma librería que utiliza Google Chrome y Safari de Apple.

- Tiempo de ejecución de *Android*: Está basado en el concepto de máquina virtual. Éste lo constituyen las *Core Libraries*, que son librerías con multitud de clases Java y la máquina vistual *Dalvik*. Cada aplicación se ejecuta como un proceso

Linux con su propia instancia de la máquina virtual. Está basada en registros y optimizada para ahorrar memoria.

- Núcleo *Linux*: *Android* utiliza el núcleo de *Linux* 3.4 (*Android* 4.4.2) como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para la utilización de cualquier componente. Proporciona servicios como la seguridad, el multiproceso, el manejo de memoria y la pila de protocolos.

3.3 Entorno de desarrollo

Android Studio es el entorno de desarrollo integrado oficial 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 (Gironés, 2016).
Está basado en el software *IntelliJ IDEA* de *JetBrains*. Está disponible de manera gratuita para las plataformas *Microsoft Windows*, *Mac OS X* y *GNU/Linux*.
Android studio ofrece funciones como:

- Sistema de compilación basado en *Gradle* flexible.
- Rápida emulación.
- Entorno unificado para todos los dispositivos.
- Compatibilidad con C++ y *NDK*.
- Soporte para *Google Cloud Platform*.

Cada proyecto en Android Studio contiene uno o más módulos con archivos de código fuente y archivos de recursos.

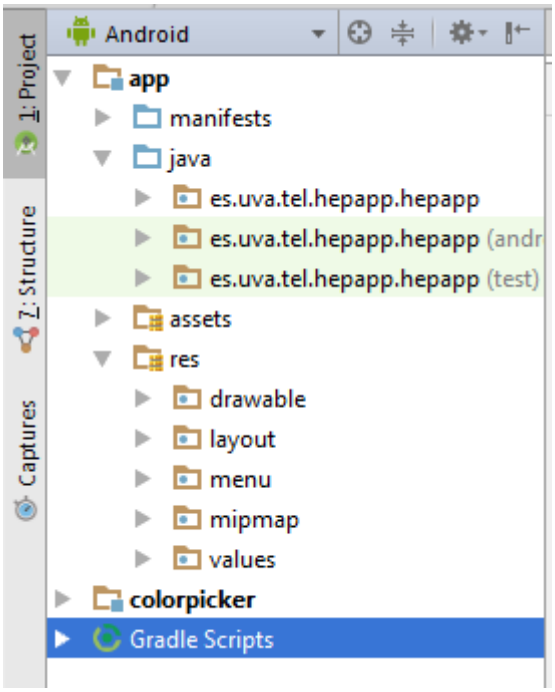


Figura 11: Archivos del proyecto

Como podemos observar en la Figura 11, cada proyecto contiene uno o más módulos con ficheros de código fuente. Siguen una estructura de directorio (Gironés, 2016).

- *AndroidManifest.xml*: Define las características fundamentales de la aplicación y sus componentes.
- *Java*: Contiene los archivos de código fuente, incluyendo el código de prueba *JUnit*.
- *Res*: Contiene todos los recursos de la aplicación.
    - *Drawable-resolution/:* directorio para imágenes y descriptores de imágenes. Para cada uno de los cuatro tipos básicos de resolución hay un directorio.
    - *Layout/:* almacena los ficheros que definen la interfaz de usuario.
    - *Menu/:* directorio con los menús de cada actividad.
    - *Values/:* contiene colecciones de recursos, como por ejemplo, cadenas de caracteres, definiciones de colores…

Estos módulos facilitan la organización de cada parte de la aplicación. En las aplicaciones además, se pueden diferenciar diferentes coponentes (Madrid, 2010):

- Actividad: Cada aplicación está formada por un conjunto de elementos básicos de visualización. Una actividad representa una pantalla de la aplicación. Para crear una interfaz de usuario completa se dispone de varias actividades. Toda actividad debe pertenecer a una clase descendiente de Activity.
- Servicios: Procesos que se ejecutan como tareas de fondo, sin interactuar con el usuario. Pueden ser de tipo local, utilizados por aplicaciones del mismo terminal, o remotos, se puedes utilizar desde otros terminales.

- Receptor de anuncios: Recibe y reacciona a anuncios de tipo difusión. Son anuncios lanzados por el sistema o generados por la aplicación, no tienen interfaz de usuario.
- Proveedor de contenido: Permite compartir datos sin necesidad de comprometer la seguridad del sistema de ficheros. Con este componente podemos acceder a datos de otras aplicaciones o proporcionar datos a otras aplicaciones.

Capítulo 4: Descripción técnica

La aplicación HepApp fue ideada por Dr. Kelly Burak, para la docencia que imparte de hepatogía en la Universidad de Calgary. Debido a su previa implementación para iOS, el diseño se ha realizado para que se asemejara lo máximo posible a la aplicación ya desarrollada.

4.1 Interfaz de usuario

Para conseguir una mayor sensación de dinamismo, la aplicación consta de diferentes tipos de interfaces. Las actividades principales están formadas por *Relative Layout* , a las cuales se le añaden funcionalidades con interfaces de tipo *Navigation Drawer*, *Toolbar e* incluso una interfaz personalizada.

4.1.1 NavigationDrawer

La plataforma *Android Studio* nos proporciona un proyecto básico en el cual podemos ver el desarrollo de este tipo de interfaz (Google, Developers, s.f.).

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```

Figura 12: Archivo XML del Layout básico NavigationDrawer

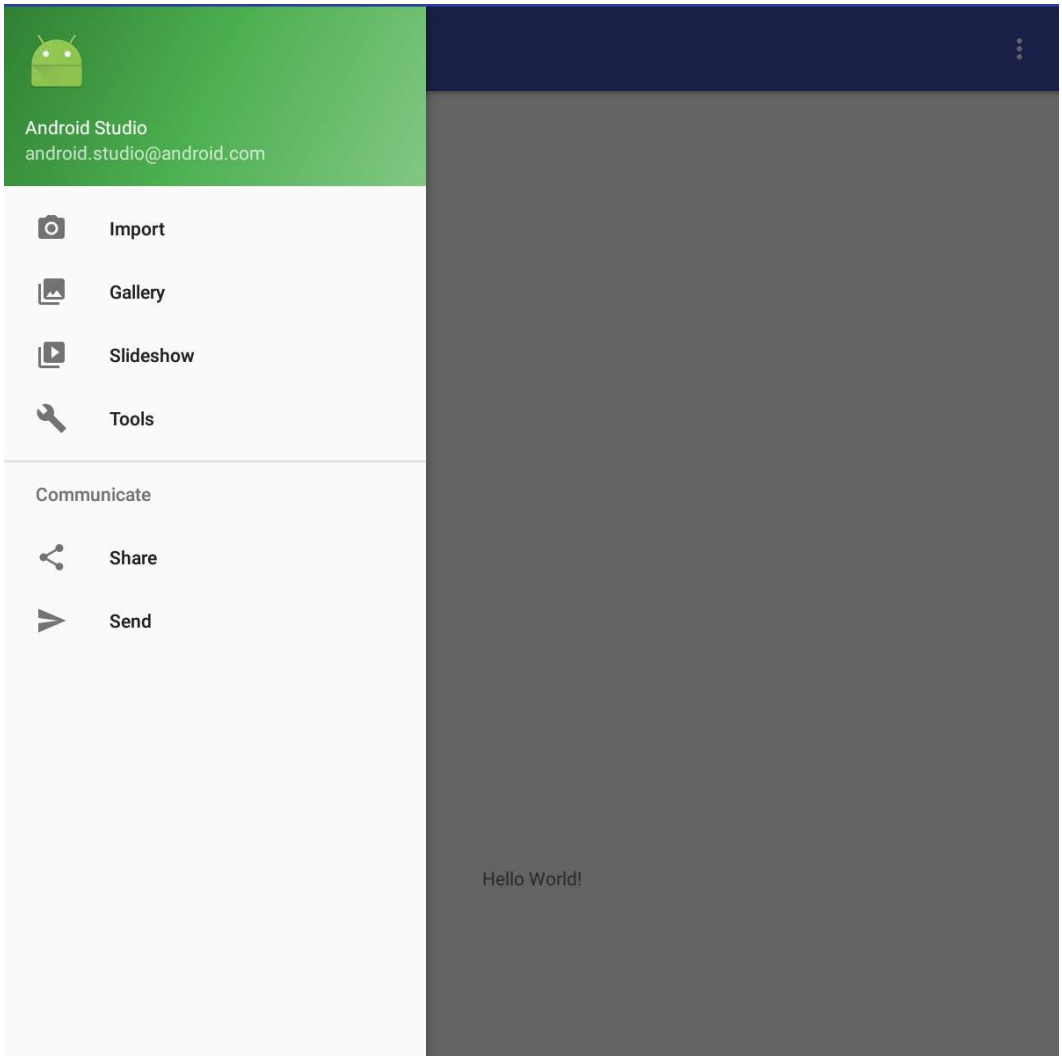


Figura 13: NavigationDrawer básico

Como se puede observar en la Figura 12, el objeto principal del *layout* es *DrawerLayout*. Dentro del elemento principal se incluye la vista de la actividad y se añade otra vista que contiene el *Navigation Drawer*. Dentro de esta vista podemos incluir diferentes elementos que se mostrarán en el *Navigation*, un ejemplo de esto es la Figura 13. En el caso de esta aplicación, se ha incluido una lista con las diferentes secciones de la aplicación para mejorar la navegación.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:openDrawer="start"
    tools:context="es.uva.tel.hepapp.hepapp.ChaptersActivity">
    <android.support.design.widget.CoordinatorLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <RelativeLayout...>

    </android.support.design.widget.CoordinatorLayout>

    <ListView android:id="@+id/left_drawer"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:choiceMode="singleChoice"
        android:dividerHeight="1dp"
        style="@style/navigationDrawer"
        android:layout_toStartOf="@+id/toolbar"
        android:layout_marginTop="56dp" />

</android.support.v4.widget.DrawerLayout>
```

Figura 14: Archivo XML de la aplicación HepApp.

La lista tienen que estar definida dentro de *DrawerLayout* pero fuera del objeto *CordinatorLayout* o *NavigationDrawer*. En las Figuras 14 y 15, se muestra la implementación en la aplicación HepApp.

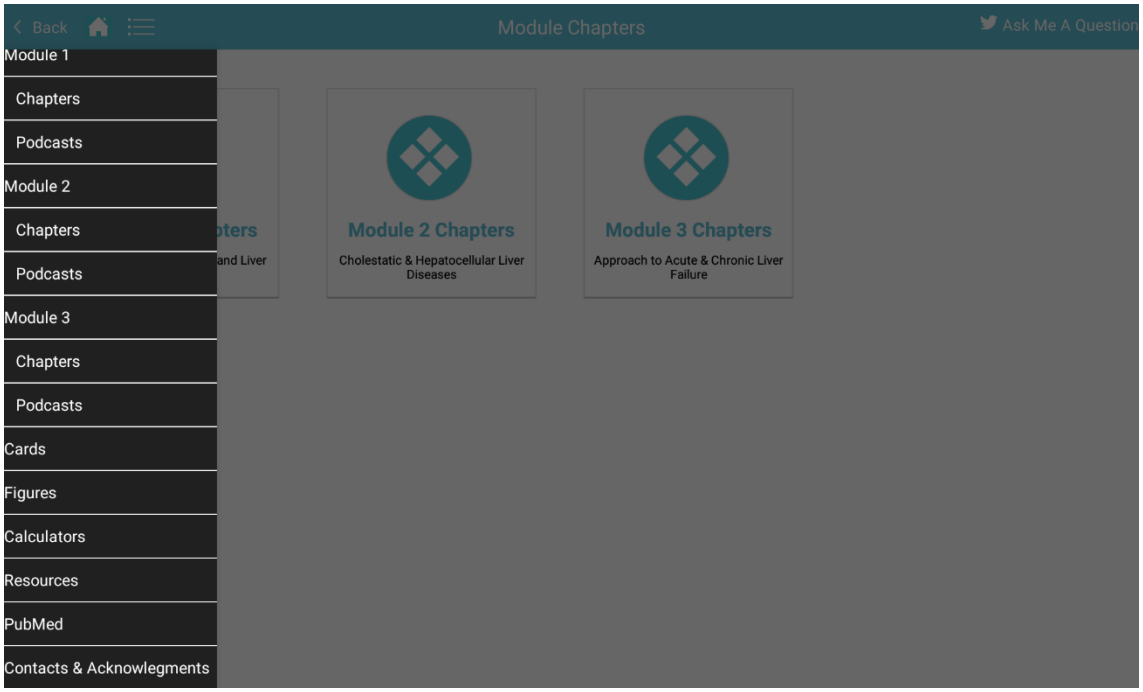


Figura 15: NavigationDrawer de la aplicación HepApp

4.1.2 Toolbar

Una *Toolbar* es una generalización de *ActionBar* para su uso en los diseños de aplicaciones. Mientras que una *ActionBar* es tradicionalmente parte de la decoración opaca de la ventana de una actividad controlada por el marco, una *Toolbar* puede colocarse en cualquier nivel arbitrario de anidamiento dentro de una jerarquía de vistas (Google, Developers, s.f.).

La *Toolbar* admite un conjunto de funciones más centrado que *ActionBar*. Una *Toolbar* puede contener una combinación de los siguientes elementos opcionales (Google, Developers, s.f.):

- Un botón de navegación. Esto puede ser una flecha hacia arriba, el menú de *NavigationDrawer*, cerrar, colapsar, hecho u otro elemento de la elección de la aplicación. Este botón siempre debe usarse para acceder a otros destinos de navegación dentro del contenedor de la *Toolbar* y su contenido ha de ser significativo. El botón de navegación está alineado verticalmente dentro de la altura mínima de la *Toolbar*.
- Un logotipo. Puede extenderse a la altura de la barra y puede ser arbitrariamente ancho.
- Título y subtítulo. El título debe ser una señal para la posición actual de la *Toolbar* y el contenido en la jerarquía de navegación. El subtítulo, si está presente, debe indicar cualquier información extendida sobre el contenido actual. Si una aplicación utiliza una imagen de logotipo, debe considerar la omisión de un título y un subtítulo.

- Una o varias vistas personalizadas. La aplicación puede agregar vistas secundarias arbitrarias a la *Toolbar*. Si *Toolbar.LayoutParams* de una vista *Toolbar*. LayoutParams indica un valor de *Gravity* de *CENTER_HORIZONTAL* la vista intentará centrarse dentro del espacio disponible restante en la *Toolbar* después de que se hayan incluido todos los demás elementos.
- Un *ActionMenu* . El menú de acciones pasará al final de la *Toolbar* ofreciendo algunas acciones frecuentes, importantes o típicas junto con un menú de desbordamiento opcional para acciones adicionales. Los botones de acción están alineados verticalmente dentro de la altura mínima de la *Toolbar*.

En las interfaces de usuario modernas de Android, Google aconseja a los desarrolladores apoyarse más en un esquema de colores visualmente distinto para las barras de herramientas que en el icono de la aplicación. El uso del icono de la aplicación más el título como un diseño estándar se desaconseja en los dispositivos API 21 y más posteriores (Google, Developers, s.f.).

```
15      <RelativeLayout
16      android:orientation="horizontal"
17      android:layout_width="match_parent"
18      android:layout_height="match_parent"
19      android:gravity="left"
20      android:paddingBottom="16dp">
21
22          <android.support.v7.widget.Toolbar
23              android:layout_height="56dp"
24              android:layout_width="match_parent"
25              android:minHeight="0dp"
26              android:background="@color/colorPrimary"
27              android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
28              android:id="@+id/toolbar"
29              android:layout_alignParentTop="true"
30              android:layout_alignParentStart="true">
31
32              <RelativeLayout
33                  android:layout_width="wrap_content"
34                  android:layout_height="wrap_content"
35                  android:orientation="horizontal"
36                  android:layout_alignTop="@+id/toolbar"
37                  android:layout_centerHorizontal="true">
38
39              </RelativeLayout>
40
41          </android.support.v7.widget.Toolbar>
```

Figura 16: Archivo XML de la Toolbar de HeppApp

En el caso de la aplicación HepApp, la *Toolbar* no es más que una banda de color a la que se le añaden elementos como el título de la sección, el menú *NavigationDrawer*, botón *Home*, botón *Back* y el *plugin* de *Twitter*. En la Figura 16, se muestra la implementación de la *Toolbar*.

4.1.3 Vista personalizada

Una de las funcionalidades que se detallan en el siguiente apartado, requiere poder dibujar en la pantalla. Para poder conseguir esto he incluido una vista personalizada que cubre toda la zona sobre la que se necesita dibujar. En la Figura 17 podemos ver como se a incluido *SimpleDrawingView* en el *layout* correspondiente.

```
196
197         <es.uva.tel.hepapp.hepapp.SimpleDrawingView
198             android:id="@+id/simpleDrawingView1"
199             android:layout_width="wrap_content"
200             android:layout_height="wrap_content"
201             android:background="@android:color/transparent"
202             android:layout_alignParentLeft="true"
203             android:layout_alignParentRight="true"
204             android:layout_below="@+id/toolbar"
205             android:layout_above="@+id/button_clear" />
206
```

Figura 17: Vista personalizada incluida en HepApp

Para poder dibujar necesitamos definir un objeto *Paint* que controle el estilo y el color que se dibuja.

```
public class SimpleDrawingView extends View {
  // setup initial color
  private final int paintColor = Color.BLACK;
  // defines paint and canvas
  private Paint drawPaint;

  public SimpleDrawingView(Context context, AttributeSet attrs) {
    super(context, attrs);
    setFocusable(true);
    setFocusableInTouchMode(true);
    setupPaint();
  }

  // Setup paint with color and stroke styles
  private void setupPaint() {
    drawPaint = new Paint();
    drawPaint.setColor(paintColor);
    drawPaint.setAntiAlias(true);
    drawPaint.setStrokeWidth(5);
    drawPaint.setStyle(Paint.Style.STROKE);
    drawPaint.setStrokeJoin(Paint.Join.ROUND);
    drawPaint.setStrokeCap(Paint.Cap.ROUND);
  }
}
```

Ahora que tenemos la configuración de *Paint* para tener un color negro y un estilo de trazo en particular, vamos a tratar de dibujar algunos círculos con diferentes colores. Todo el dibujo que sucede en una *view* debe tener lugar dentro del método *onDraw* que se llama automáticamente cuando se visualiza

```
public class SimpleDrawingView extends View {
    // ...variables and setting up paint...
    // Let's draw three circles
    @Override
    protected void onDraw(Canvas canvas) {
      canvas.drawCircle(50, 50, 20, drawPaint);
      drawPaint.setColor(Color.GREEN);
      canvas.drawCircle(50, 150, 20, drawPaint);
      drawPaint.setColor(Color.BLUE);
      canvas.drawCircle(50, 250, 20, drawPaint);
    }
}
```

A continuación necesitamos dibujar un círculo cada vez que el usuario toque la pantalla. Esto nos obligaría a realizar un seguimiento de una serie de puntos para nuestros círculos y, a continuación, añadir un punto para cada evento *onTouch* activado.

```
public SimpleDrawingView(Context context, AttributeSet attrs) {
  super(context, attrs);
  setupPaint(); // same as before
  circlePoints = new ArrayList<Point>();
}

// Draw each circle onto the view
@Override
protected void onDraw(Canvas canvas) {
  for (Point p : circlePoints) {
    canvas.drawCircle(p.x, p.y, 5, drawPaint);
  }
}

// Append new circle each time user presses on screen
@Override
public boolean onTouchEvent(MotionEvent event) {
  float touchX = event.getX();
  float touchY = event.getY();
  circlePoints.add(new Point(Math.round(touchX), Math.round(touchY)));
  // indicate view should be redrawn
  postInvalidate();
  return true;
}

private void setupPaint() {
  // same as before
  drawPaint.setStyle(Paint.Style.FILL); // change to fill
```

Cuando el usuario presiona hacia abajo, vamos a iniciar un camino y luego cuando arrastre vamos a conectar los puntos. Para ello, necesitamos modificar el *onTouchEvent* para añadir estos puntos a nuestro objeto *Path*.

```
public class SimpleDrawingView extends View {
    private Path path = new Path();

    // Get x and y and append them to the path
    public boolean onTouchEvent(MotionEvent event) {
        float pointX = event.getX();
        float pointY = event.getY();
        // Checks for the event that occurs
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            // Starts a new line in the path
            path.moveTo(pointX, pointY);
            break;
        case MotionEvent.ACTION_MOVE:
            // Draws line between last point and this point
            path.lineTo(pointX, pointY);
            break;
        default:
            return false;
       }

       postInvalidate(); // Indicate view should be redrawn
       return true; // Indicate we've consumed the touch
    }

   // ...
}
```

Por ultimo modificamos onDraw para en vez de círculos representemos las líneas que hemos creado.

```
public class SimpleDrawingView extends View {
  // ... onTouchEvent ...

  // Draws the path created during the touch events
  @Override
  protected void onDraw(Canvas canvas) {
      canvas.drawPath(path, drawPaint);
  }

  private void setupPaint() {
    // same as before
    drawPaint.setStyle(Paint.Style.STROKE); // change back to stroke
    // ...
  }
}
```

De esta manera conseguimos poder dibujar en la aplicación (Esquenazi, 2015).

4.2 Funcionalidades

La aplicación consta de diferentes funcionalidades, las cuales se muestran en la Figura 18.

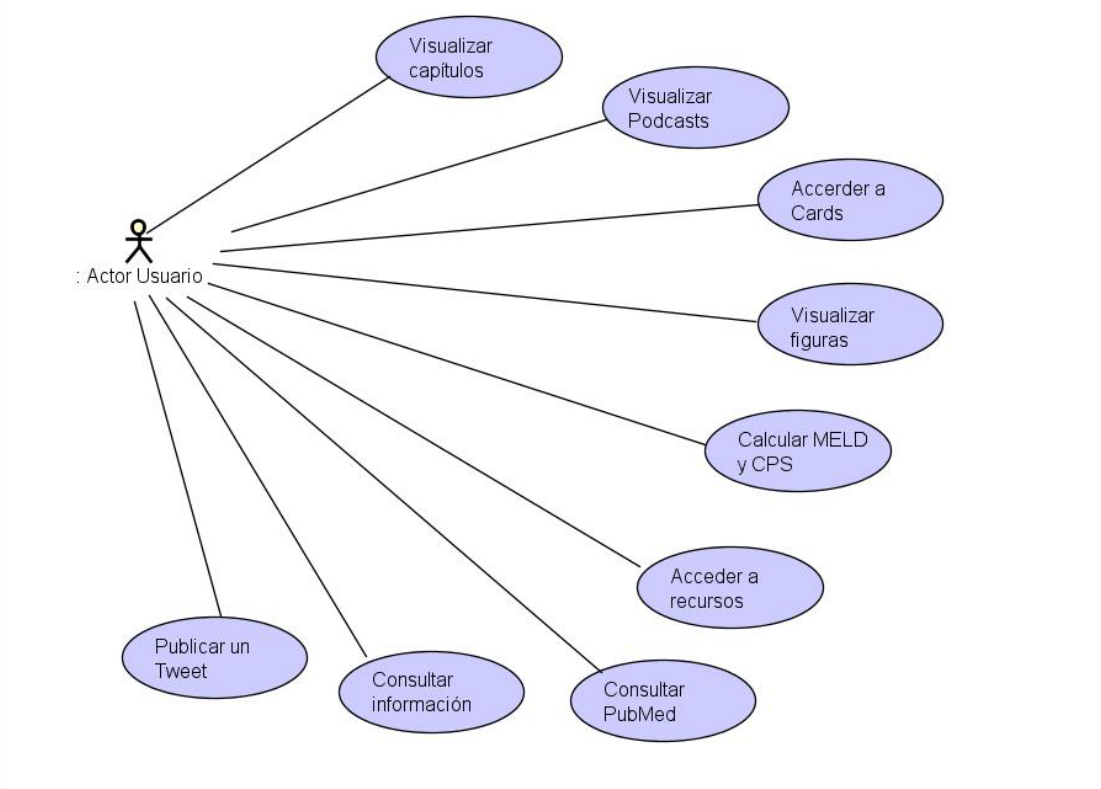


Figura 18: Diagrama UML con las funcionalidades de la aplicación

A continuación se describen los diferentes casos de uso a través de tablas y en algunos casos, diagramas de flujo con la finalidad de facilitar la compresión.

4.2.1 Visualizar capítulos

Identificador	**Visualizar capítulos**	
Versión	1	
Fecha de última revisión	08-05-2017	
Autores	Esther Martín González	
Descripción	El usuario podrá visualizar los diferentes capítulos.	
Actores o personal involucrado	Usuario	
Precondición	El sistema debe tener conexión a internet.	
Escenario secuencial de éxito o secuencia o flujo normal	**Paso**	**Acción**
	1	El usuario elige la opción Chapters.
	2	El sistema presenta los diferentes módulos.
	3	El usuario elige un módulo.

	4	El sistema muestra los capítulos correspondientes al módulo seleccionado.
	5	El usuario selecciona un capítulo.
	6	El sistema muestra el capítulo a pantalla completa, la opción de descargar e ir a los *Podcasts* correspondientes a ese módulo.
	7a	El usuario finaliza.
	7b	El usuario elige la opción de descargar
	8b	El sistema abre el capítulo usando una de las aplicaciones presentes en el dispositivo.
	7c	El usuario elige la opción visualizar *Podcasts*.
	8c	El sistema abre el módulo de *Podcasts* correspondiente al capítulo en el que se encontraba.
Postcondición	Se ha accedido correctamente a la funcionalidad deseada	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-5	Si el usuario pulsa el botón Back, el caso de uso retrocede una secuencia de uso.
	1-5	Si el usuario pulsa el botón del menú lateral, el sistema puede cambiar de caso de uso o de capítulo.
	1-5	Si el usuario pulsa el botón Home, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	
Importancia	Media	
Urgencia	PD	
Comentarios	No hay	

Tabla 1: Caso de uso visualizar capítulos.

4.2.2 Visualizar *Podcasts*

Identificador	Visualizar *Podcasts*	
Versión	1	
Fecha de última revisión	08-05-2017	
Autores	Esther Martín González	
Descripción	El usuario podrá visualizar los diferentes *Podcasts*.	
Actores o personal involucrado	Usuario	
Precondición	El sistema debe tener conexión a internet.	
Escenario secuencial de éxito o secuencia o flujo normal	**Paso**	**Acción**
	1	El usuario elige la opción *Podcasts*.
	2	El sistema presenta los diferentes módulos.
	3	El usuario elige un módulo.
	4	El sistema muestra los *Podcasts* correspondientes al módulo seleccionado.
	5	El usuario selecciona un *Podcast*.
	6	El sistema muestra el *Podcasts* a pantalla completa y la opción ir a los capítulos correspondientes a ese módulo.
	7a	El usuario finaliza.
	7b	El usuario elige la opción visualizar capítulo.
	8c	El sistema abre el módulo de capítulos correspondiente al *Podcasts* en el que se encontraba.
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-5	Si el usuario pulsa el botón Back, el caso de uso retrocede una secuencia de uso.
	1-5	Si el usuario pulsa el botón del menú lateral, el sistema puede cambiar de caso de uso o de *Podcasts*.
	1-5	Si el usuario pulsa el botón Home, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	
Importancia	Media	
Urgencia	PD	

Comentarios	No hay

Tabla 2: Caso de uso visualizar Podcasts.

4.2.3 Acceder a *Cards*

Identificador	**Acceder a *Cards***	
Versión	1	
Fecha de última revisión	08-05-2017	
Autores	Esther Martín González	
Descripción	El usuario podrá acceder a la aplicación *Cards*.	
Actores o personal involucrado	Usuario	
Precondición	El sistema debe tener conexión a internet.	
Escenario secuencial de éxito o secuencia o flujo normal	**Paso**	**Acción**
	1	El usuario elige la opción *Cards*.
	2	El sistema abre la aplicación web *Cards*.
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-2	Si el usuario pulsa el botón Back, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
	1-2	Si el usuario pulsa el botón del menú lateral, el sistema cambia de caso de uso.
	1-2	Si el usuario pulsa el botón Home, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	
Importancia	Media	
Urgencia	PD	
Comentarios	No hay	

Tabla 3: Caso de usa acceder a Cards.

4.2.4 Visualizar figuras

Identificador	Visualizar figuras	
Versión	1	
Fecha de última revisión	09-05-2017	
Autores	Esther Martín González	
Descripción	El usuario podrá visualizar e interactuar con diferentes figuras.	
Actores o personal involucrado	Usuario	
Precondición	El sistema debe tener conexión a internet.	
Escenario secuencial de éxito o secuencia o flujo normal	**Paso**	**Acción**
	1	El usuario elige la opción *Figures*.
	2	El sistema presenta los diferentes tipos de figuras.
	3a	El usuario selecciona la opción *Table of Contens*.
	4a	El sistema muestra las tablas de contenidos de los diferentes módulos.
	5a	El usuario selecciona una tabla.
	6a	El sistema muestra la tabla a pantalla completa.
	3b	El usuario selecciona la opción *Schemes*.
	4b	El sistema muestra esquemas del contenido didáctico.
	5b	El usuario selecciona un esquema.
	6b	El sistema muestra el esquema pantalla completa.
	3c	El usuario selecciona la opción *Interactive Figures*.
	4c	El sistema muestra las figuras en un tamaño reducido
	5c	El usuario selecciona una figura.
	6c	El sistema muestra la figura a tamaño completo y dos *radio buttons*.
	7c	El usuario presiona el *radio button* no preseleccionado.
	8c	El sistema cambia la opacidad de una figura, superponiéndola a la ya mostrada.
	3d	El usuario selecciona la opción *Chapter Figure*.
	4d	El sistema muestra los diferentes módulos.
	5d	El usuario selecciona un módulo.

	6d	El sistema muestra las figuras correspondientes al módulo.
	7d	El usuario selecciona una figura.
	8d	El sistema muestra la figura a pantalla completa.
	3e	El usuario selecciona la opción *Drawing*.
	4e	El sistema muestra las diferentes figuras.
	5e	El usuario selecciona una figura.
	6e	El sistema muestra la figura a pantalla completa.
	7e	El usuario puede dibujar sobre la figura.
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-7	Si el usuario pulsa el botón Back, el caso de uso retrocede una secuencia de uso.
	1-7	Si el usuario pulsa el botón del menú lateral, el sistema cambia de caso de uso.
	1-7	Si el usuario pulsa el botón Home, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
	7e	El usuario selección la opción *Clear*, se elimina el dibujo realizado.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	
Importancia	Media	
Urgencia	PD	
Comentarios	No hay	

Tabla 4: Caso de uso visualizar figuras.

4.2.5 Calculadora Child-Pugh

El índice de Child-Pugh (Child CG, Turcotte JG,1964) data de la década de 1960 y hasta ahora ha sido el índice para evaluar el pronóstico de una enfermedad hepática crónica, principalmente la cirrosis, más utilizado. Este índice consta de cinco variables, de las que tres se refieren a la función hepática (albúmina, bilirrubina y tiempo de protrombina) y las otras dos a complicaciones de la enfermedad (ascitis y encefalopatía).
Cada variable se puntúa entre 1 y 3 puntos según el grado de afectación (Tabla 5), por lo que la puntuación mínima es de 5 puntos y la máxima, de 15.

	1 punto	2 puntos	3 puntos
Abumina	<28	28-35	>35
Bilirrubina	<34	34-50	>50
INR	<1.7	1.7-2.2	>2.2
Ascitis	No	Tratable	Refractaria
Encefalopatía	No	Grado I-II	Grado III-IV

Tabla 5: Índice Child-Pugh

Dependiendo de la puntuación que indique el índice, se indica una clase (A, B o C), la cual determina unas probabilidades de supervivencia (Tabla 6).

Puntos	Clase	Supervivencia tras 1 año	Supervivencia tras 2 años
5-6	A	100%	85%
7-9	B	81%	57%
10-15	C	45%	35%

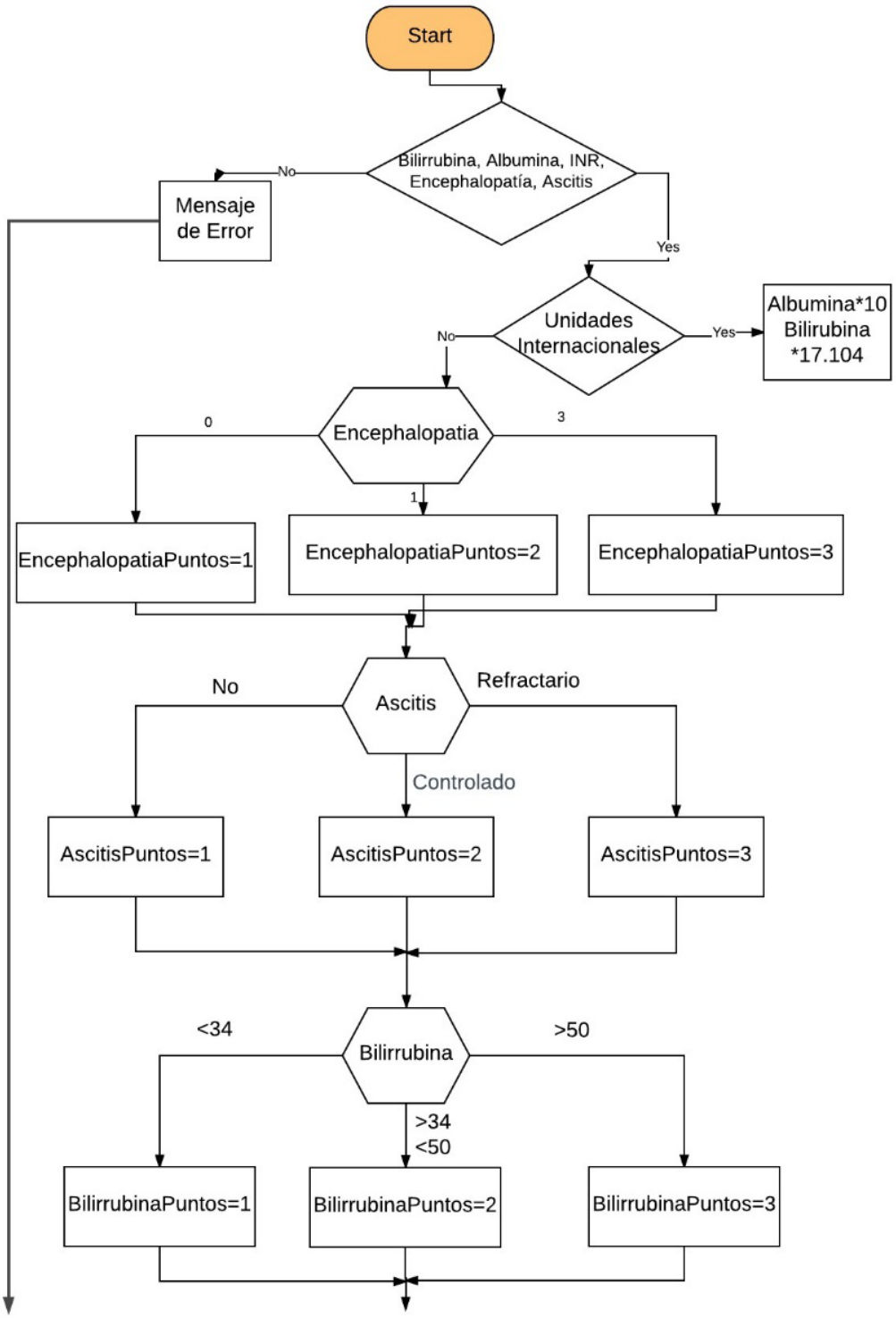
Tabla 6: Interpretación Child-Pugh

A continuación se muestra el caso de uso de dicha calculadora.

Identificador	Calculadora *Child-Pugh*	
Versión	1	
Fecha de última revisión	10-05-2017	
Autores	Esther Martín González	
Descripción	El usuario podrá calcular el parámetro *Child-Pugh*.	
Actores o personal involucrado	Usuario	
Precondición	Ninguna.	
Escenario secuencial de éxito o secuencia o flujo normal	**Paso**	**Acción**
	1	El usuario elige la opción *Calculators*.
	2	El sistema muestra los dos tipos de calculadoras.
	3	El usuario selecciona *Child-Pugh*.

	4	El sistema abre la calculadora *Child-Pugh* y muestra las opciones de *Reset Values, Previous Values* y *More Information*.
	5a	El usuario elige la opción *Reset Values.*
	6a	El sistema borra los datos introducidos en la calculadora.
	5b	El usuario elige la opción *Previous Values.*
	6b	El sistema introduce los últimos valores que se habían asignado a cada variable.
	5c	El usuario elige la opción *More Information*
	6c	El sistema muestra una imagen informativa. La superpone a la calculadora.
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-6	Si el usuario pulsa el botón Back, el caso de uso retrocede una secuencia de uso.
	1-6	Si el usuario pulsa el botón del menú lateral, el sistema cambia de caso de uso.
	1-6	Si el usuario pulsa el botón Home, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	
Importancia	Media	
Urgencia	PD	
Comentarios	No es posible introducir caracteres erróneos, ya que la aplicación abre el teclado numérico.	

Tabla 7: Caso de uso calculadora Child-Pugh.


Start
Bilirrubina, Albumina, INR, Encephalopatía, Ascitis
No
Mensaje de Error
Yes
Unidades Internacionales
No
Yes
Albumina*10 Bilirubina *17.104
Encephalopatia
0
1
3
EncephalopatiaPuntos=1
EncephalopatiaPuntos=2
EncephalopatiaPuntos=3
Ascitis
No
Refractario
Controlado
AscitisPuntos=1
AscitisPuntos=2
AscitisPuntos=3
Bilirrubina
<34
>34
<50
>50
BilirrubinaPuntos=1
BilirrubinaPuntos=2
BilirrubinaPuntos=3

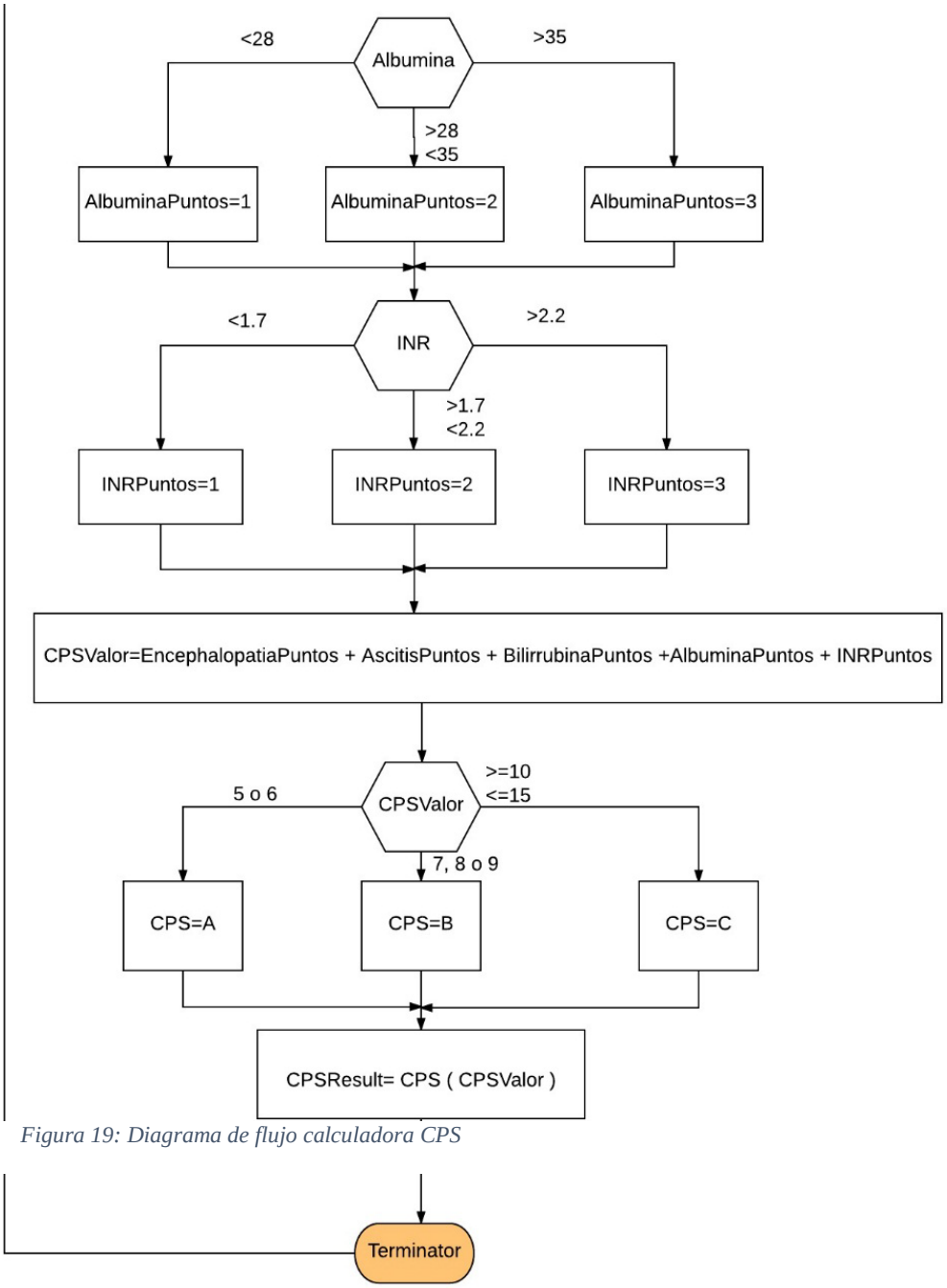


Figura 19: Diagrama de flujo calculadora CPS

4.2.6 Calculadora MELD,MELDNa,MELD5v

El modelo MELD (Model for End-stage Liver Disease) se publicó por primera vez en el año 2000 y tiene un origen similar al índice de Child-Pugh. MELD (Kamath PS, Wiesner RH, 2001) es un sistema de puntuación para medir la gravedad de enfermedades hepáticas crónicas, inicialmente fue desarrollado para predecir la muerte en pacientes de cirugía sometidos a TIPS.

La mejor virtud del MELD es que es muy útil para determinar la mortalidad a tres meses con una C estadística de 0,80 (o sea, que acierta en el 80% de los casos). El cálculo del MELD es complicado, ya que incluye logaritmos neperianos, por lo que se necesita el uso de calculadora. La fórmula para calcular este modelo es (Wiesner R, 2003):

$$MELD = 3.8 * \ln bilirubina + 11.2 * \ln INR + 9.57 * \ln creatinina + 6.43$$

Ecuación 1: Cálculo del modelo MELD

Este modelo presenta diferentes variantes, en la aplicación HepApp se calculan también los modelos *MELDNa* y *MELD5v* (P.Myers, 2013):

$$MELDNa = MELD - Na - [0.025 * MELD * (140 - Na)] + 140$$

Ecuación 2: Cálculo del modelo MELDNa

$$MELD5v = MELDNa + [5.275 * (4 - albumina)] - [0.163 * MELD * (4 - albumina)]$$

Ecuación 3: Cálculo del modelo MELD5v

Identificador	**Calculadora *MELD***	
Versión	1	
Fecha de última revisión	10-05-2017	
Autores	Esther Martín González	
Descripción	El usuario podrá calcular el parámetro *MELD*.	
Actores o personal involucrado	Usuario	
Precondición	Ninguna.	
Escenario secuencial de éxito o secuencia o flujo normal	**Paso**	**Acción**
	1	El usuario elige la opción *Calculators*.
	2	El sistema muestra los dos tipos de calculadoras.
	3	El usuario selecciona *MELD*.
	4	El sistema abre la calculadora *Child-Pugh* y muestra las opciones de *Reset Values*, *Previous Values* y *More Information*.
	5a	El usuario elige la opción *Reset Values*.
	6a	El sistema borra los datos introducidos en la calculadora.
	5b	El usuario elige la opción *Previous Values*.
	6b	El sistema introduce los últimos valores que se habían asignado a cada variable.
	5c	El usuario elige la opción *More Information*

	6c	El sistema muestra dos imágenes informativas. Las superpone a la calculadora.
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-6	Si el usuario pulsa el botón Back, el caso de uso retrocede una secuencia de uso.
	1-6	Si el usuario pulsa el botón del menú lateral, el sistema cambia de caso de uso.
	1-6	Si el usuario pulsa el botón Home, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	
Importancia	Media	
Urgencia	PD	
Comentarios	No es posible introducir caracteres erróneos, ya que la aplicación abre el teclado numérico.	

Tabla 8: Caso de uso calculadora MELD.

El funcionamiento de la calculadora *MELD* sigue el diagrama de flujo mostrado a continuación.

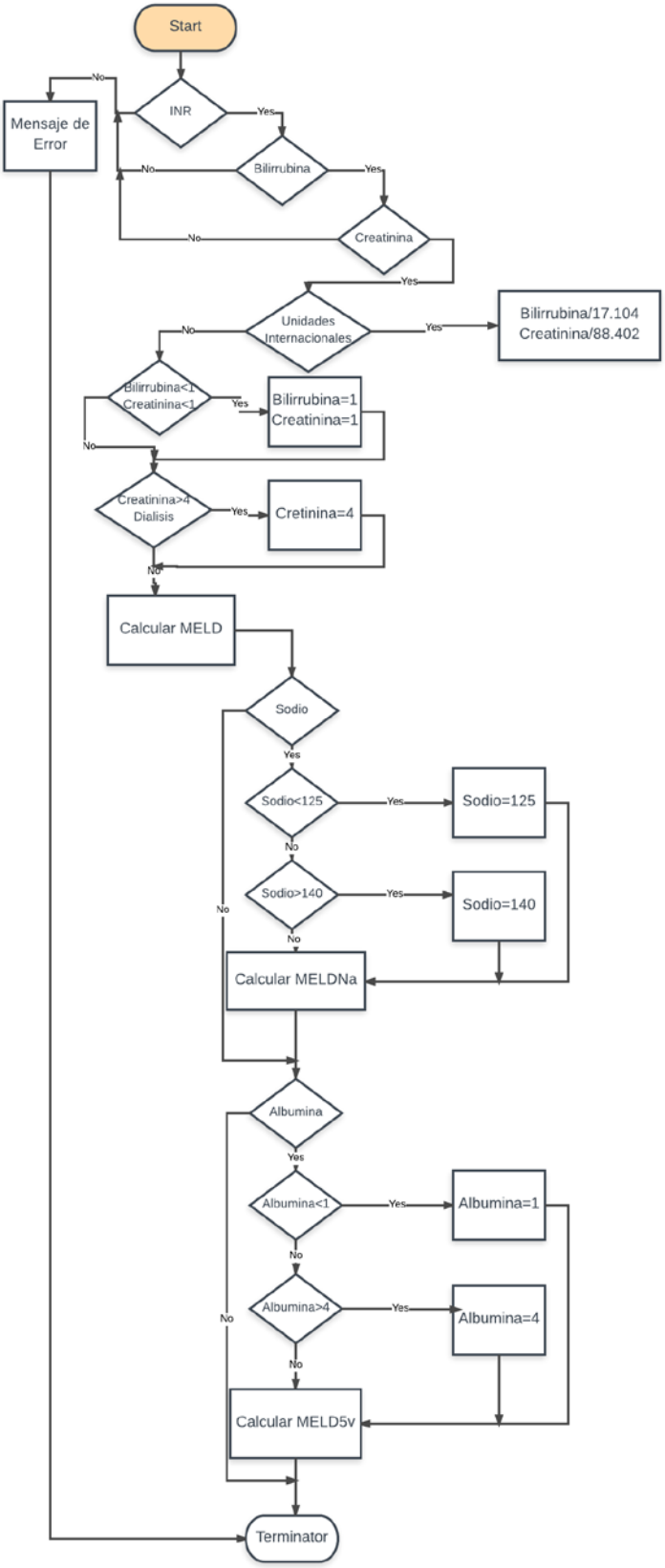


Figura 20: Diagrama de flujo de la calculadora MELD.

4.2.7 Acceder a recursos

Identificador	Acceder a recursos	
Versión	1	
Fecha de última revisión	09-05-2017	
Autores	Esther Martín González	
Descripción	El usuario podrá acceder a los recursos.	
Actores o personal involucrado	Usuario	
Precondición	El sistema debe tener conexión a internet.	
Escenario secuencial de éxito o secuencia o flujo normal	**Paso**	**Acción**
	1	El usuario elige la opción *Resources*.
	2	El sistema muestra los diferentes recursos.
	3	El usuario selecciona un recurso.
	4	El sistema abre la página web correspondiente al recurso seleccionado.
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-4	Si el usuario pulsa el botón Back, el caso de uso retrocede una secuencia de uso.
	1-4	Si el usuario pulsa el botón del menú lateral, el sistema cambia de caso de uso.
	1-4	Si el usuario pulsa el botón Home, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	
Importancia	Media	
Urgencia	PD	
Comentarios	No hay	

Tabla 9: Caso de uso acceder a recursos.

4.2.8 Visualizar *PubMed*

Identificador	Visualizar *PubMed*	
Versión	1	
Fecha de última revisión	09-05-2017	
Autores	Esther Martín González	
Descripción	El usuario podrá acceder a *PubMed*.	
Actores o personal involucrado	Usuario	
Precondición	El sistema debe tener conexión a internet.	
Escenario secuencial de éxito o secuencia o flujo normal	**Paso**	**Acción**
	1	El usuario elige la opción *PubMed*.
	2	El sistema abre la aplicación web *PubMed*.
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-2	Si el usuario pulsa el botón Back, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
	1-2	Si el usuario pulsa el botón del menú lateral, el sistema cambia de caso de uso.
	1-2	Si el usuario pulsa el botón Home, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	
Importancia	Media	
Urgencia	PD	
Comentarios	No hay	

Tabla 10: Caso de uso visualizar PubMed

4.2.9 Visualizar información

Identificador	Visualizar información	
Versión	1	
Fecha de última revisión	09-05-2017	
Autores	Esther Martín González	
Descripción	El usuario podrá acceder a la información.	
Actores o personal involucrado	Usuario	
Precondición	Ninguna.	
Escenario secuencial de éxito o secuencia o flujo normal	**Paso**	**Acción**
	1	El usuario elige la opción *Information*.
	2	El sistema abre un pdf con la información acerca de la aplicación.
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-2	Si el usuario pulsa el botón Back, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
	1-2	Si el usuario pulsa el botón del menú lateral, el sistema cambia de caso de uso.
	1-2	Si el usuario pulsa el botón Home, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	
Importancia	Media	
Urgencia	PD	
Comentarios	No hay	

Tabla 11: Caso de uso visualizar información.

4.2.10 Publicar un *Tweet*

Identificador	Publicar un *Tweet*	
Versión	1	
Fecha de última revisión	09-05-2017	
Autores	Esther Martín González	
Descripción	El usuario podrá publicar un *Tweet*.	
Actores o personal involucrado	Usuario	
Precondición	Ninguna.	
Escenario secuencial de éxito o secuencia o flujo normal	**Paso**	**Acción**
	1	El usuario elige la opción *Ask Me A Question*.
	2	El sistema abre un *plugin* de *Twitter* y adjunta una captura de pantalla.
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-2	Si el usuario pulsa la X, se vuelve a la pantalla de inicio y el caso de uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	
Importancia	Media	
Urgencia	PD	
Comentarios	Este caso de uso está presente en todas las pantallas de la aplicación.	

Tabla 12: Caso de uso publicar Tweet.

Capítulo 5: Manual de usuario

En este capítulo se pretende facilitar el uso de la aplicación de aprendizaje implementada. Mediante capturas y comentarios de los pasos a seguir, se dará a conocer HepApp de manera que cualquier usuario puede utilizar la aplicación cómodamente. A lo largo del capítulo se mostrarán capturas de las diferentes pantallas de la aplicación.

Para poder iniciar la aplicación no es necesario realizar ningún tipo de registro. Todas las funcionalidades están disponibles para cualquier usuario. El acceso dentro de algunas de las páginas web incluidas en la aplicación están restringidas a los alumnos de la universidad de Calgary.

5.1 Icono y pantalla de inicio

Al descargar e instalar la aplicación, aparecerá en el escritorio el icono mostrado a continuación.

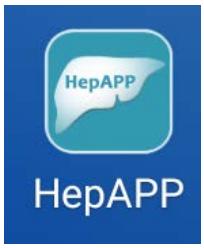


Figura 21: Icono mostrado en el escritorio

Una vez pulsemos sobre el icono, se mostrará la pantalla inicial en la cual se puede ver el icono nuevamente. Si pulsamos sobre *Ask Me A Question*, situado arriba a la derecha, accederemos a la funcionalidad publicar un tweet. La pantalla mostrada con esta funcionalidad se describirá más adelante.

Con los diferentes botones podremos acceder a todas las funcionalidades de la aplicación. Los siguientes apartados quedarán definidos por el nombre de dichos botones.

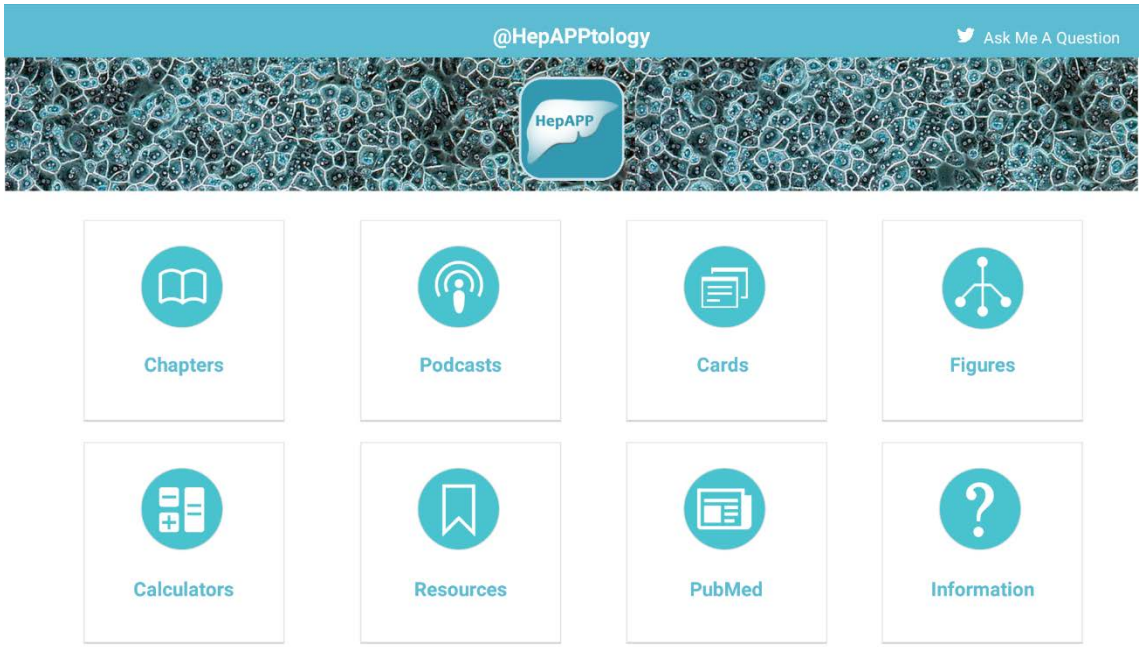


Figura 22: Pantalla principal de la aplicación

5.2 Pantalla *Chapters*

Al pulsar sobre el botón *Chapters* en la pantalla inicial, se mostrará una cabecera en la cual se encuentra el botón back, una casa, un icono formado por tres rayas horizontales, el título del módulo en el que nos encontramos y nuevamente la opción Ask Me A Question.

En esta pantalla también de muestran tres botones desde los cuales podremos acceder a cada uno de los diferentes módulo. Estos botones están formados por el icono asignado al acceso a módulos, el número y tipo de módulo y una breve descripción.

Al pulsar sobre uno de los módulos, se mostrarán los botones de acceso a los capítulos correspondientes al módulo seleccionado y la misma Toolbar.

Las pantallas de cada módulo muestran un botón para cada uno de los capítulos que componen el módulo en el que nos encontramos. Están formados por el icono asignado a los capítulos, el número y título de capítulo que abre. Al pulsar uno de estos botones se abrirá el capítulo correspondiente mostrandolo a pantalla completa. En la cabecera podemos ver el capítulo en el que estamos y el módulo al que pertenece.

Para descargar el pdf, solo tenemos que pulsar el botón *Download*. Con el resto de botones podremos navegar de un capitulo a otro o ver los *Podcast* del módulo en el que nos encontramos.

5.3 Pantalla Podcasts

Al pulsar sobre el botón *Podcasts* en la pantalla inicial, se mostrará la misma cabecera que en el caso anterior y tres botones con los cuales podremos acceder a los *Podcasts* de cada módulo. Estos botones poseen una breve descripción de cada uno de los módulos

Al pulsar sobre uno de los módulos, se mostrarán los botones de acceso a los *Podcasts* correspondientes al módulo seleccionado y la misma *Toolbar*.

En el caso del módulo 1 se muestran 7 botones. Para el caso del módulo dos se muestran 8 botones y en la parte inferior derecha una flecha con la que pasamos a otra pantalla en la que aparecen 6 botones para poder acceder al resto de *Podcasts*. En el módulo 3 nos encontramos con el miso caso que en el módulo 2.

Una vez seleccionemos el *Podcasts* que queremos, se abrirá una pantalla en la cual aparece el *Podcasts*, botones para poder movernos de un *Podcasts* a otro y un botón para poder acceder a los capítulos correspondientes al módulo en el que estamos.

5.4 Pantalla Cards

Desde la pantalla de inicio podemos acceder a *Cards* pulsando el botón *Cards*. Se mostrará la pantalla principal de la aplicación *Cards* de la universidad de Calgary. A partir de esta pantalla solo podremos continuar si estamos registrados.

5.5 Pantalla Figures

Desde la pantalla de incio pulsamos el botón *Figures*. Se abrirá una pantalla con los botones de acceso a tablas de contenidos, esquemas, figuras interactivas, figuras de los capítulos y figuras sobre las que podremos dibujar.

Si pulsamos el botón *Table of Contents*, se mostrarán 4 botones con la tabla de contenidos en vista pequeña y una descripción del contenido de las tablas. Si pulsamos uno de estos botones, se mostrará la tabla a pantalla completa y en la parte inferior derecha unas flechas que nos permitirán cambiar de tabla de contenidos.

Si volvemos a la pantalla *Figures* y pulsamos el botón *Schemes*, se mostrará una pantalla con 8 botones. Estos botones están formados por una imagen del esquema y una descripción del contenido. Si pulsamos en uno de estos botones, se mostrará el esquema a pantalla completa y unas flechas que nos permiten movernos de un esquema a otro.

Al pulsar la opción *Interactive Figures* en la pantalla *Figures* se mostrará una interfaz con 5 botones en los que aparece las figuras a tamaño reducido. Desde esta pantalla podemos seleccionar la figura interactiva que queremos ver. Esta se abrirá a pantalla completa, mostrando dos radio buttons en la parte inferior. Pulsando estos botones, podemos superponer una imagen sobre otra.

Al igual que los capítulos y los *Podcasts*, las figuras también están divididas en módulos. Desde la pantalla *Chapter Figures* podemos acceder a los tres módulos. Cada módulo

consta de múltiples figuras que se van mostrando de 8 en 8. Con las flechas inferiores podemos ir navegando para visualizar todas las figuras y movernos de un módulo a otro.

Si seleccionamos una de las figuras, se mostrará a pantalla completa junto con el título de la figura.

Por último, podemos entrar en *Drawing*, esta opción nos permite dibujar sobre 8 figuras. Al seleccionar una de las figuras esta se mostrará a pantalla completa.

Podemos dibujar sobre la pantalla, este dibujo aparecerá por defecto en negro. Si queremos cambiar el color, tenemos que pulsar el botón *Color* abriéndose el siguiente menú de colores.

Figura 23: Menú de selección de colores

Al pulsar sobre el color deseado pararemos a dibujar con ese color. Además, con el botón *Clear* podemos eliminar los dibujos realizados y así poder volver a empezar.

5.6 Pantalla Calculators

Desde la pantalla *Calculators* podemos seleccionar dos calculadoras diferentes seleccionando los respectivos botones.

- ***Child Pugh Score***

Si seleccionamos *Child Pugh Score,* se mostrará una pantalla con los botones descritos a continuación y los datos necesarios para realizar los cálculos.

Antes de pulsar el botón *Calculate CP Score*, debemos introducir todos los datos que nos solicita en la parte izquierda de la pantalla. Los datos de *Bilirubin*, *Albumine* e *INR* se introducen mediante el teclado que aparece al pulsar el rectángulo situado a la derecha del nombre del dato solicitado. Los datos de *Encephalopathy* y *Ascites* se introducen pulsando sobre los botones situados a la derecha del nombre del dato. Solo será posible seleccionar uno de los botones de cada dato.

Una vez introducidos todos los datos pulsamos en *Calculate CP Score*, en el recuadro de la derecha se sustituirá la palabra *CPS* por el resultado de la calculadora.

Los datos de Bilirubin y Albumin también se pueden introducir en el sistema imperial, cambiándose a mg/dL y g/dL respectivamente. Para ello debemos desactivar las unidades internacionales pulsando en al botón OFF que aparece en la parte superior derecha.

Además, en la parte inferior de la pantalla nos encontramos con una serie de botones. *Reset Values* nos permite eliminar los datos introducidos, *Previous Values* recupera los datos que hemos eliminado pulsando *Reset Values.* Por último, al pulsar el botón *More Information* se abrirá un gráfico con información útil del parámetro que estamos calculando.

- ***MELD***

Si seleccionamos *MELD,* se mostrará la una pantalla similar a la anterior calculadora.

Antes de pulsar el botón *Calculate MELD* debemos introducir al menos los tres primeros datos que nos solicita en la parte izquierda de la pantalla. Todos los datos se introducen mediante el teclado que aparece al pulsar el rectángulo situado a la derecha del nombre del dato solicitado. Podemos indicar diálisis o desactivar las unidades internacionales mediante los radio botones situados debajo de los respectivos textos.

Una vez introducidos todos los datos necesarios pulsamos en *Calculate MELD,* en el recuadro de la derecha se sustituirá la palabra *MELD* por el resultado de la calculadora. En caso de haber introducido también el dato de *Sodium* aparecerá un resultado en *MELDNa* y si también introducimos *Albumin* aparecerá el resultado de *5vMELD*.

Debajo del botón *Calculate MELD,* podemos leer unas indicaciones sobre los datos que se solicitan. Además, en la parte inferior de la pantalla nos encontramos con una serie de botones. *Reset Values* nos permite eliminar los datos introducidos, *Previous Values* recupera los datos que hemos eliminado pulsando *Reset Values.* Por último, al pulsar el botón *More Information* se abrirá un gráfico con información útil del parámetro que estamos calculando.

5.7 Pantalla Resources

En esta pantalla nos encontramos con los botones de acceso a los diferentes recursos (guías CASL, AASLD, EASL, ACG, AGA e ILCA, Lindsay Atlas y referencias). Pulsando en uno de ellos se mostrará una nueva pantalla, la cual muestra la aplicación web correspondiente.

5.8 Pantalla PubMed

Al pulsar el botón *PubMed* de la pantalla principal, se abrirá la pantalla principal de la aplicación web *PubMed*. La *Toolbar* se mantendrá mientras navegamos en el recurso web, pudiendo enviar un *tweet* con su correspondiente captura en cualquier momento.

5.9 Pantalla Information

Pulsando el último botón de la pantalla principal, se abrirá un pdf con informacion sobre la aplicación.

5.10 Funcionamiento Toolbar

Durante todas las pantalla explicadas en este capítulo aparece la misma *toolbar* a excepción de la pantalla principal. Esta *toolbar* esta formada por 4 diferentes botones y el título del lugar donde nos encontramos.

El primer botón, situado a la izquierda, es *Back.* Este botón nos permite volver a la acción anterior. En el caso de la Figura 69, como nos encontraríamos en la pantalla de *Information* tal y como nos indica el título, volveríamos a la pantalla principal al pulsarlo.

El siguiente botón es *Home,* el cual se representa con el dibujo de una casa y se encuentra a la derecha del botón *back*. Al pulsar este botón volvemos a la pantalla principal independientemente de donde nos encontrásemos.

A continuación encontramos el botón que despliega el menú *NavigationDrawer*. Este se abre en el lateral izquierdo y en él se puede ver una lista de los diferentes apartados de la aplicación. Pulsando sobre cualquiera de ellos abriremos la pantalla correspondiente.

Por último, a la derecha se encuentra el botón *Ask Me A Question.* En el momento en el que pulsamos este botón, se realiza una captura de pantalla la cual se adjunta al *plugin* de *Twitter* para poder publicar un *tweet* con la duda que nos haya surgido.

Una vez redactemos el *tweet* bastará con pulsar el botón *Twittear*. Es posible que para que la aplicación pueda realizar la captura de pantalla, tengas que darle permisos de acceso a las imágenes, el contenido multimedia y los archivos del dispositivo. Para ello pulsa permitir en el dialogo que abre la aplicación.

Capítulo 6: Conclusiones y líneas futuras.

En este capítulo se van a indicar las conclusiones a las que he llegado tras la realización de este trabajo de fin de grado. Indicando si se han logrado los objetivos finales propuestos al inicio de esta memoria y las líneas futuras que podría seguir este proyecto. Además, se incluye un apartado en el cual se realiza un estudio económico de los gastos que hubiera supuesto el desarrollo de la aplicación en el mundo empresarial.

6.1 Estudio económico

A pesar de que el mundo de las aplicaciones móviles está creciendo cada día más, el precio de desarrollo sigue siendo una incógnita. Esto se debe sobre todo al hecho de que el tiempo dedicado depende mucho del tipo de aplicación. Las aplicaciones que podemos encontrar son muy diversas por lo que existe un amplio abanico de precios.

La aplicación HepApp no es una aplicación especialmente cara de desarrollar, ya que solo se ha utilizado Sotfware libre.

Estimación de gastos		
Salario programador junior	11€/h * 180h	1980€
Cuota autónomo	275 €/mes	550€
Coste material necesario	Ordenador 500€ Dispositivos Moviles Android 300€	Valor de 2 meses 70€
Sotfware	Licencia Libre	0€
Costes de oficina	Luz+Internet	168€
		2768€

Tabla 13: Estimación de gastos

La duración del desarrollo de la aplicación ha durado cerca de los dos meses. Se ha necesitado dedicarle 180h de desarrollo, lo que equivaldría a 23 días laborables. Esto supondría la necesidad de pagar dos meses de cuota de autónomo, luz e internet. Además, el material necesario para el desarrollo tiene en total un valor aproximado de 800€, aunque este precio podría ser mucho mayor dependiendo de las características del software que se utilizara.

El hecho de que sea una aplicación nativa incrementa el precio, ya que si quisiéramos en un futuro su integración en otro sistema operativo habría que volver a desarrollarla desde cero. En cambio, esto no ocurre en el caso de las aplicaciones híbridas como se ha explicado en capítulos anteriores.

Por otro lado es una aplicación con funcionalidades sencillas, que no necesitan de la integración con plataformas o el desarrollo con software que requiera licencia. Esto hace que el presupuesto solo conste de gastos esenciales para el desarrollo de una App.

6.2 Conclusiones

Esta aplicación la comencé a desarrollar tras cursar la asignatura optativa de desarrollo de aplicaciones móviles. Me ha permitido ampliar los conocimientos que empecé con esta asignatura y a afianzarlos. La realización de este trabajo me ha enseñado a buscar información útil y a ser más autodidacta. Superando los múltiples retos que este Trabajo de Fin de Grado me he ido encontrando, como por ejemplo incluir el plugin de Twitter o el proyecto ColorPicker.

Para conseguir los objetivos descritos en el capítulo 1, he seguido los pasos que se indican a continuación.

En primer lugar he realizado un estudio de las diferentes opciones de desarrollo de aplicaciones móviles que nos encontramos actualmente. Tras comprender las ventajas y desventajas de cada una de ellas he optado por el desarrollo nativo. Esta opción conlleva la elección de un sistema operativo, por lo que he tenido que compararlos tanto desde una visión técnica como desde la visión de mercado. Esto me ha ayudado a aprender a mirar más allá de las ventajas técnicas, teniendo en cuenta también a cuanta gente podría ayudar la aplicación si la desarrollaba para un entorno o para otro. Tras este análisis la elección ha sido Android ya que es el sistema operativo más presente en el mercado. Seguidamente he analizado con más detenimiento esta tecnología en concreto la versión KitKat debido a que con ella se consigue un equilibrio entre mercado y funcionalidad.

Tras tener clara la tecnología, se elabora un análisis funcional. Gracias a esto, se comprueba que la aplicación sigue los requisitos iniciales de escalabilidad y dinamismo.

A continuación, se realiza un manual de usuario con el cual he podido comprobar que la aplicación cumple con el objetivo de facilidad de uso. Además, gracias a este apartado he podido comprobar el correcto funcionamiento de la aplicación y recoger los diferentes escenarios que se pueden presentar.

Por lo tanto, he conseguido ampliar mis conocimiento de java, xml y del desarrollo en Android, mejorando mis conocimiento a la hora de usar las opciones que nos proporciona Android Studio, y sobre todo, a comprender e implementar una aplicación en al contexto del Flipped Classroom.

Con el desarrollo de esta aplicación me he encontrado con problemas técnicos como la limitación de memoria RAM lo que impide incluir todas las figuras que inicialmente estaban pensadas, sin que la aplicación se detenga. Además, el hecho de incluir un proyecto ya desarrollado hace que no esté completamente optimizado ya que se añaden características que no son necesarias para esta aplicación en concreto.

En el desarrollo de este Trabajo de Fin de Grado he aplicado conocimiento adquiridos tras estos años en la universidad de Valladolid. Conocimientos técnicos como java y xml, pero sobretodo habilidades y hábitos adquiridos gracias a los trabajos realizados durante la carrera.

6.3 Líneas Futuras

Tras el desarrollo de la aplicación han surgido ideas y mejoras para futuras versiones de la misma. Pudiéndose desarrollar nuevas funcionalidades que mejoraran la utilidad de HepApp. Estas funcionalidades serían las siguientes:

Integración de todas las calculadoras disponible. Esto permitiría a los alumnos comprobar sus cálculos y aprender de forma más dinámica los posibles resultados.

Debido a que en la universidad de Calgary hay alumnos con diferentes idiomas nativos, sería interesante traducir la aplicación a diferentes idiomas para que sea más amigable y fácil de usar para la mayoría.

En mi opinión, estos cambios supondrían una mejora en la aplicación. Siguiendo el camino del Flipped Classroom se consigue mejorar la capacidad de los alumnos para aprender y ser autodidactas. Estos cambios mejorarían el objetivo de facilitar el aprendizaje que sigue la aplicación HepApp.

Bibliografía

Apple. (Septiembre de 2014). *iOs Technology Overview*. Obtenido el 10 de Julio de 2017, de https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html

Branscombe, M. (25 de Abril de 2014). *Windows phone 8.1: A bittersweet update that appeals to the mainstream*. Obtenido el 10 de Julio de 2017, de http://www.zdnet.com/article/windows-phone-8-1-a-bittersweet-update-that-appeals-to-the-mainstream/

Callaham, J. (30 de Septiembre de 2014). *Yes, Windows 10 is the next version of Windows Phone*. Obtenido el 10 de Julio de 2017, de https://www.windowscentral.com/yes-windows-10-next-version-windows-phone

Child CG, Turcotte JG. Surgery and portal hypertension. In: The liver and portal hypertension. Edited by CG Child. Philadelphia: Saunders 1964k50-64.

Delía, L., Galdamez, N., Thomas, P., & Pesado, P. (s.f.). *Universidad Nacional de la Plata*. Obtenido el 10 de Julio de 2017, de http://sedici.unlp.edu.ar/bitstream/handle/10915/32397/Documento_completo.pdf?sequence=1

Dudley, B. (29 de Septiembre de 2014). *Microsoft reveals Windows 10, with hybrid Start menu*. Obtenido el 10 de Julio de 2017, de http://blogs.seattletimes.com/brierdudley/2014/09/29/microsoft-previews-windows-9/

El comercio. (22 de Enero de 2015). Recuperado el 13 de Abril de 2017, de http://elcomercio.pe/paginas/smartphones-tablets/adios-windows-phone-microsoft-solo-hablara-windows-10-noticia-1786330

Esquenazi, N. (3 de Noviembre de 2015). *github*. Obtenido el 10 de Julio de 2017, de Basic painting with views: https://github.com/codepath/android_guides/wiki/Basic-Painting-with-Views

Gironés, J. T. (2016). *El gran libro de Android.* Barcelona: S.A. Marcombo.

Google. (s.f.). Developers. Obtenido el 10 de Julio de 2017, de https://developer.android.com

Kamath PS, Wiesner RH, Malinchoc M, Kremers W, Therneau TM, Kosberg CL, D'Amico G, Dickson ER, Kim WR. A model to predict survival in patients with end-stage liver disease. Hepatology. 2001 Feb;33(2):464-70.

Madrid, U. C. (13 de Abril de 2010). *Programación en dispositivos móviles*. Obtenido el 10 de Julio de 2017, de https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android

Martorell, J. (38 de diciembre de 2007). *Societat Catalana de transplantament.* Obtenido el 4 de mayo de 2017, de http://www.sctransplant.org/doc/boletin/boletin_38_cast.pdf

Menárguez, A. T. (7 de Noviembre de 2016). Aprender al revés en más efectivo. *El País*. Obtenido el 10 de Julio de 2017, de El País.

Molinero, S. (10 de febrero de 2016). *materialdoc*. Obtenido el 10 de Julio de 2017,d e http://www.materialdoc.es/color-picker/

P.Myers, B. (2013). Revision of MELD to Include Serum Albumin Improves Prediction of Mortality on the Liver Transplant Waiting List. *US National Library of Medicine , 8*(1).

Page, C. (14 de Abril de 2014). *Windows Phone 8.1 is no available for developers*. Obtenido el 10 de Julio de 2017, de https://www.theinquirer.net/inquirer/news/2339769/windows-phone-81-is-now-available-for-developers

Statista. (Mayo de 2017). Obtenido el 7 de Julio de 2017, de https://es.statista.com/

Torre, F. J. (Julio de 2013). *Desarrollo de aplicaciones para Windows Phone*. Obtenido el 4 de Julio de 2017, de https://e-archivo.uc3m.es/bitstream/handle/10016/17873/Memoria%20PFC,%20F.J.Castellanos.pdf?sequence=1

Wiesner R, United Network for Organ Sharing Liver Disease Severity Score Committee, et al. Model for end-stage liver disease (MELD) and allocation of donor livers.Gastroenterology. 2003 Jan;124(1):91-6.

W. Ray Kim, M. S. (4 de Septiembre de 2008). *The new England journal of medicine.* Obtenido el 4 de Julio de 2017,de http://www.nejm.org/doi/full/10.1056/NEJMoa0801209#t=article

Zuriarrain, J. M. (4 de Abril de 2017). *El país*. Obtenido el 12 de Abril de 2017, de http://tecnologia.elpais.com/tecnologia/2017/04/04/actualidad/1491296467_396232.html

Anexo: *Colorpicker*

Para poder añadir al proyecto un selector de colores, he clonado el proyecto *ColorPicker* del repositorio de Google (Google, Developers, s.f.).

Una vez clonado, desde Android Studio seleccionamos la opción *New* del menú y después *import module*, debemos indicar la ruta donde hemos clonado el proyecto. Además debemos indicar en las dependencias que compile el proyecto *colorPicker* añadiendo las líneas de código que se muestran a continuación.

```
dependencies {
    compile project(':colorpicker')
}
```

En el archivo de recursos colors.xml, añadimos los colores que queremos que se puedan seleccionar. Seguidamente, inicializamos el *colorPicker* dentro de la clase en la que queramos que se muestra. Para ello necesitamos indicar el título, el array de colores que hemos definido anteriormente, el número de columnas en las que queremos que se reparta y el total de colores (Molinero, 2016).

```
ColorPickerDialog colorPickerDialog = new ColorPickerDialog();

colorPickerDialog.initialize(

    R.string.title, colors, selectedColor, numColumns,
colors.length);

colorPickerDialog.show(getFragmentManager(), tag);
```