UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

APLICACIÓN iOS DE HEPATOLOGÍA EN UN CONTEXTO DE "*FLIPPED CLASSROOM*"

AUTORA: PAULA CANTORAL GUTIÉRREZ

TUTORA: MÍRIAM ANTÓN RODRÍGUEZ

Valladolid, Julio de 2018.

TÍTULO: Aplicación iOS de hepatología en un contexto de *"Flipped Classroom"*

AUTOR: Paula Cantoral Gutiérrez

TUTORA: Míriam Antón Rodríguez

DEPARTAMENTO: Teoría de la Señal y 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 de 2018

CALIFICACIÓN:

RESUMEN DEL PROYECTO

Actualmente, las aplicaciones móviles forman parte de nuestro día a día. Por ello, este trabajo de fin de grado se centra en el desarrollo de una aplicación móvil para la ayuda en el aprendizaje del carcinoma hepatocelular a un nivel universitario. En un principio, nos centraremos en el análisis de las distintas opciones disponibles para el desarrollo de aplicaciones móviles, tanto a nivel de las distintas tecnologías móviles (aplicaciones web, híbridas o nativas), así como a nivel de los distintos tipos de Sistemas Operativos más dominantes en el mercado (Android, *Windows Phone* y iOS).

Tras el análisis desarrollado, se ha optado por realizar una aplicación nativa para el Sistema Operativo iOS con la finalidad de ayudar a los alumnos de la Universidad de Calgary en el estudio del carcinoma hepatocelular. Esta aplicación está basada en un contexto de *Flipped Classroom*, el cual permite el acceso a una gran cantidad de información y realizar un aprendizaje más dinámico.

PALABRAS CLAVE

Aplicación, calculadora, iOS, *Swift*, hepatología

ABSTRACT

Nowadays, mobile applications are part of our day. Thus, this final thesis focuses on the development of a mobile application that help in learning hepatocellular carcinoma at university level. In the beginning, we will focus on the analysis of the different options available for the development of a mobile application, both at the level of the different mobile technologies (web applications, hybrid and native), as well as the level of the different types of Operating Systems more dominant in the market (Android, Windows Phone and iOS).

After the analysis developed, we have chosen to make a native application for the Operating System iOS, with the purpose of helping the students of the University of Calgary, in the study of hepatocellular carcinoma. This application is based on a *Flipped Classroom* context, which enables access to a wealth of information and make a more dynamic learning.

KEY WORDS

Application, calculator, iOS, *Swift*, hepatocellular.

Índice

Índice de Ilustraciones

Índice de Tablas

Índice de Diagramas

Índice de Ecuaciones

CAPÍTULO 1: INTRODUCCIÓN

Las primeras aplicaciones para dispositivos móviles datan de los años 90. Desde esa fecha, el interés por las nuevas tecnologías ha crecido de forma exponencial en todo el mundo año tras año. Si observamos la siguiente ilustración, se puede apreciar que el número de usuarios conectados a Internet, cada año crece.

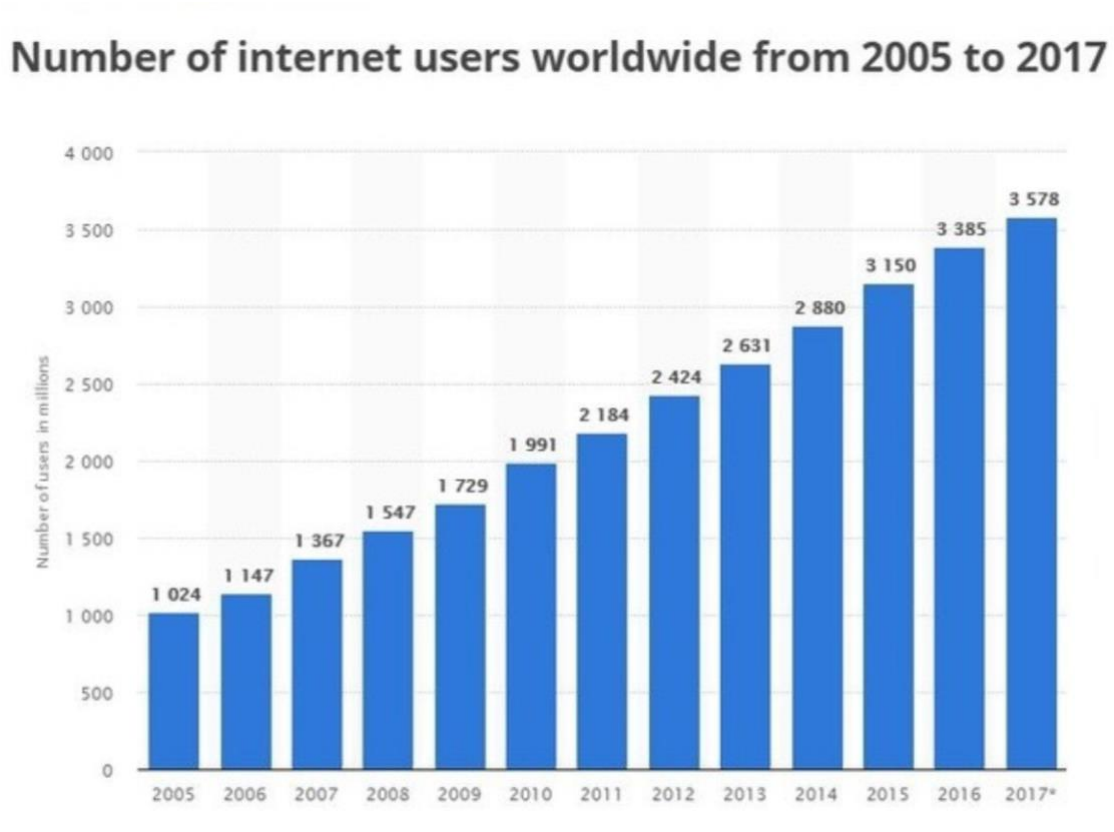

Ilustración 1. Uso de internet en todo el mundo desde 2005 hasta 2017. Fuente statista, 2018

Actualmente, en España el 62% de la población hace uso de Internet a través de un *smarthphone*, frente a un 25,2% de la población lo hace desde una *Tablet*. Se observa un incremento del uso de dispositivos móviles frente al uso de un PC, que pasa de poseer un 69,3% en el año 2016, frente al 55,4% de la actualidad (Prensario Internacional, 2018). Este hecho ha favorecido la aparición de distintas aplicaciones móviles de todo tipo. Debido a ello, el número de descargas aumenta cada año, como podemos observar en la siguiente ilustración.

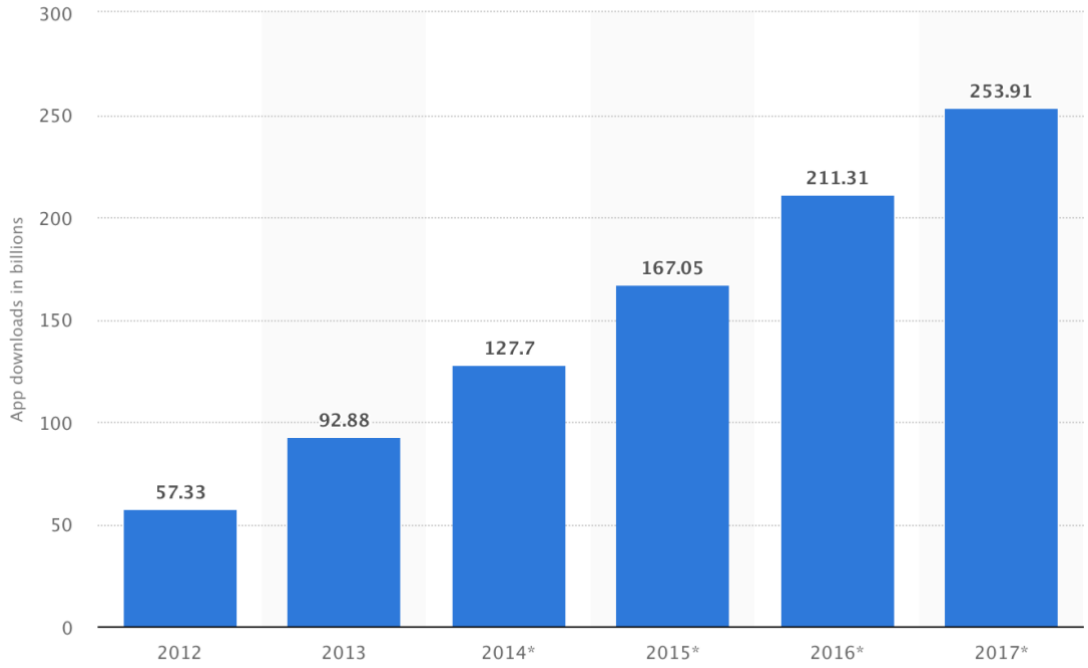

Ilustración 2. Evolución número de descargas de aplicaciones móviles. Fuente statista 2018

Como consecuencia de la evolución de las tecnologías, las personas deben de conocer más sobre el funcionamiento de ellas, ya que si queremos convivir en la era en la que nos encontramos es fundamental.

Las nuevas tecnologías están agilizando, optimizando y perfeccionando algunas de las actividades que realizamos en nuestro día a día. La capacidad de comunicación es fundamental en la actualidad gracias a su gran evolución en los últimos años, siendo ésta cada vez, mucho más rápida. Un ejemplo de comunicación habitual en nuestro día a día, en concreto, gracias a Internet, es el uso de mensajería instantánea, que nos permite mandar mensajes a cualquier persona en cualquier momento del día, compartir fotos, vídeos, audios y archivos al instante. Por lo que los nuevos dispositivos nos facilitan nuestra forma de vivir, ya que nos permiten realizar acciones de una forma rápida y eficaz.

Por otro lado, nos encontramos que, en el ámbito universitario, las nuevas tecnologías han tomado importancia. Las universidades han ido incorporando diferentes aplicaciones para la enseñanza, como son el campus virtual, recursos *online*, laboratorios virtuales, ... En concreto, el *e-learning* se define como un sistema que permite la formación a través de Internet (Jochems, 2005). Las características principales que definen el *e-learning* son que este tipo de enseñanza rompe las barreras geográficas, incorpora flexibilidad en el estudio, gracias a una comunicación asíncrona, permitiendo al alumno la autogestión del tiempo. Con ello se reducen los gastos, ya que no se necesita una ubicación física o materiales físicos para impartir las clases. En el contexto del *e-learning*, apareció lo que se conoce como *Flipped Classroom*, modelo pedagógico que permite que ciertos procesos

lectivos se realicen fuera del aula, mientras que el tiempo de clase se dedique a resolver dudas y practicar el conocimiento adquirido fuera del aula, consiguiendo mejores resultados. Este método de aprendizaje, según *John Dewey* (filósofo visionario del concepto *Flipped Classroom*), conlleva una gran responsabilidad por parte del alumno, ya que los conocimientos adquiridos tras el estudio fuera del aula se deben convertir en conocimiento a través de una aplicación práctica en el aula (Bates & Galloway, 2012).

Tras el análisis de las ventajas obtenidas con el método pedagógico *Flipped Classroom*, se pensó en desarrollar una aplicación móvil para la ayuda del aprendizaje del carcinoma hepatocelular, poniendo en práctica este modelo educativo.

1.1 OBJETIVOS

La finalidad de este Trabajo de Fin de Grado es la creación de una aplicación móvil para un dispositivo iOS en un contexto de *Flipped Classroom*.

Por ello, se ha investigado sobre las distintas necesidades que debe cumplir la aplicación, eligiendo la tecnología más adecuada según el contexto geográfico en el que se enmarca, así como tratar de que la aplicación sea fácil de usar, rápida y dinámica.

1.2 ESTRUCTURA

El presente Trabajo de Fin de Grado se ha estructurado en seis capítulos, cumpliendo los objetivos anteriormente detallados, de la manera más fluida posible.

En el primer capítulo, en el cual nos encontramos, realizamos una pequeña introducción a grandes rasgos del contexto y el contenido de la aplicación a desarrollar.

En el segundo capítulo, nos centramos en las distintas tecnologías móviles disponibles para desarrollar una aplicación, analizando sus ventajas y desventajas. Tras el análisis, se realiza una elección del tipo de aplicación a realizar, describiendo las principales características del Sistema Operativo elegido y su importancia en el mercado.

A continuación, en el tercer capítulo, se realiza un análisis del Sistema Operativo iOS utilizado para el desarrollo de la aplicación.

En el cuarto capítulo se realiza una descripción técnica de las funcionalidades de las que dispone la aplicación.

Seguidamente, en el quinto capítulo, se muestra un manual de usuario, con distintas ilustraciones de las pantallas de las que dispone, para facilitar el manejo de la aplicación.

Para finalizar, en el sexto y último capitulo, se describen las conclusiones obtenidas tras la realización de este Trabajo de Fin de grado, indicando también posibles mejoras en un futuro.

CAPITULO 2: TECNOLOGÍAS MÓVILES

Tras la aparición de Internet, las tecnologías móviles marcan nuestras vidas. De hecho, si observamos la actividad diaria de una persona, podemos observar, que el móvil ha reemplazado al despertador, el reloj, cámaras fotográficas, etc. El triunfo del uso de tecnologías móviles radica en su gran evolución, pudiendo acceder a dicha tecnología todo el mundo (Marcombo, 1998).

El principal hándicap con el que se encuentran los desarrolladores de aplicaciones móviles es la ausencia de interoperabilidad entre los distintos tipos de Sistemas Operativos existentes en el mercado, por ello, surgieron nuevas posibilidades de desarrollo, además del desarrollo nativo, como son las aplicaciones web o híbridas.

2.1 APLICACIONES WEB

La web fue creada en el año 1989 por *Tim Berners*. Su principal objetivo era organizar la información utilizando como medio físico de comunicación, la red de internet en conjunto con la utilización del protocolo HTTP (*HyperText Transference Protocol*). Dicho protocolo permite la transferencia de hipertexto que utilizan los navegadores para realizar peticiones a los servidores web y obtener respuestas de ellos. En conclusión, el protocolo HTTP nos permite visualizar páginas web (Martín, 2014).

Las principales ventajas de la utilización de aplicaciones web es que no se requiere de un *software* añadido para su utilización, así como que todas las prestaciones ofrecidas son accesibles a través de una única aplicación, que es el navegador web. También, dichas aplicaciones hacen un menor uso de los recursos del servidor y son transportables e independientes de la plataforma. Es decir, se puede acceder a ellas desde cualquier navegador de cualquier equipo, incluyendo dispositivos móviles, como son los *smartphones* y las *tablets*.

Por el contrario, las principales desventajas que presentan son la necesidad constante de conexión a internet, ya que deben estar vinculadas a un servidor que proporcione la aplicación de la web y no hay un control directo sobre los datos. También cabe destacar que reducen la velocidad de ejecución, no pueden ser descargadas desde la tienda de aplicaciones del dispositivo y no pueden utilizar todos los elementos del *hardware* del dispositivo, como es la cámara de fotos, el GPS, el uso de notificaciones, … (Miguel, 2015).

2.2 APLICACIONES HÍBRIDAS

El desarrollo de aplicaciones móviles híbridas se enfoca en la programación para dispositivos móviles ofreciendo un aspecto y sensación como si se tratará de una aplicación escrita en un lenguaje nativo. El desarrollo del contenido de la aplicación se hace principalmente en HTML5, JS y CSS.

La principal ventaja que ofrecen las aplicaciones hibridas es que no se necesita reescribir el código de la aplicación para cada uno de los distintos dispositivos móviles, teniendo así una aplicación multiplataforma (Luna et al, 2013). Otra de las ventajas que ofrece desarrollar una aplicación híbrida es que se puede hacer absolutamente todo en ella, agregando la capacidad de actualización, sin tener que distribuirla, reduciendo así los costes y el tiempo de desarrollo (Hereter & Zanini, 2016).

En contraposición a las aplicaciones híbridas, cabe destacar que como funcionan únicamente a través del navegador, esto puede afectar a la velocidad y el rendimiento de la aplicación, siendo esta una característica muy importante para la mayoría de los usuarios de telefonía móvil. Este tipo de aplicaciones están disponibles en las tiendas dedicadas para tal fin, y las aplicaciones deben ser instaladas en el dispositivo.

2.3 APLICACIONES NATIVAS

Las aplicaciones nativas son aquellas aplicaciones desarrolladas con el *software* que ofrece cada uno de los distintos Sistemas Operativos. Dicho *software* generalmente se conoce como *Software Development Kit* o SDK. Por ello, Android, iOS o *Windows Phone* poseen uno diferente. Por ende, las aplicaciones nativas se diseñan y programan específicamente para cada uno de los Sistemas Operativos, con el coste y complejidad que ello conlleva.

Las aplicaciones desarrolladas se distribuyen a través de las distintas tiendas disponibles en cada Sistema Operativo, como son *Play Store*, *App Store* o *Windows Phone Store*. Normalmente, las aplicaciones nativas se actualizan frecuentemente, siendo el usuario el que tiene que descargar cada una de las nuevas actualizaciones para estar a la última versión, que es la que corrige errores de la versión anterior o incluye mejoras (Cuello & Vittone, 2014).

Dado que las aplicaciones nativas dependen del Sistema Operativo al que pertenezcan, permiten hacer un gran uso de los distintos recursos del *hardware* del dispositivo, como puede ser el uso de notificaciones, GPS, cámara, etc. (Hereter & Zanini, 2016). Además, no requieren de Internet para funcionar, por lo que ofrecen una mejor experiencia de usuario, debido a que se encuentran realmente integradas en el dispositivo móvil.

A nivel de diseño, estas aplicaciones tienen una interfaz basada en las distintas guías de cada Sistema Operativo, logrando coherencia y consistencia, favoreciendo a la usabilidad, y beneficiando directamente al usuario, ya que encuentran interfaces similares.

2.3.1 SISTEMA OPERATIVO ANDROID

Android se trata de un Sistema Operativo diseñado para dispositivos móviles como son los *smarthphones* o las *tablets* cuyo núcleo está basado en Linux. *Android* fue desarrollada por la compañía *Handset Alliance* (liderada por *Google*), usando distintos tipos de herramientas de *software* de código abierto.

Este Sistema Operativo permite a los desarrolladores la creación de aplicaciones que aprovechan al máximo las herramientas que pueda ofrecer el dispositivo.

La arquitectura implementada permite que cualquier aplicación pueda tener acceso a las herramientas del dispositivo móviles, tales como, acceder a la cámara, al GPS, poder realizar llamadas o enviar mensajes de texto, permitiendo crear mejores experiencias de usuario.

En cuanto a la arquitectura, *Android* se trata de un Sistema Operativo por capas, como se puede observar en la siguiente ilustración. En ella podemos ver, que cada una de las capas hace uso de los servicios ofrecidos por las capas anteriores, y, ofreciendo a su vez los suyos propios a las capas de niveles superiores. A continuación, se analizarán cada una de las capas (Android Developer, 2018):

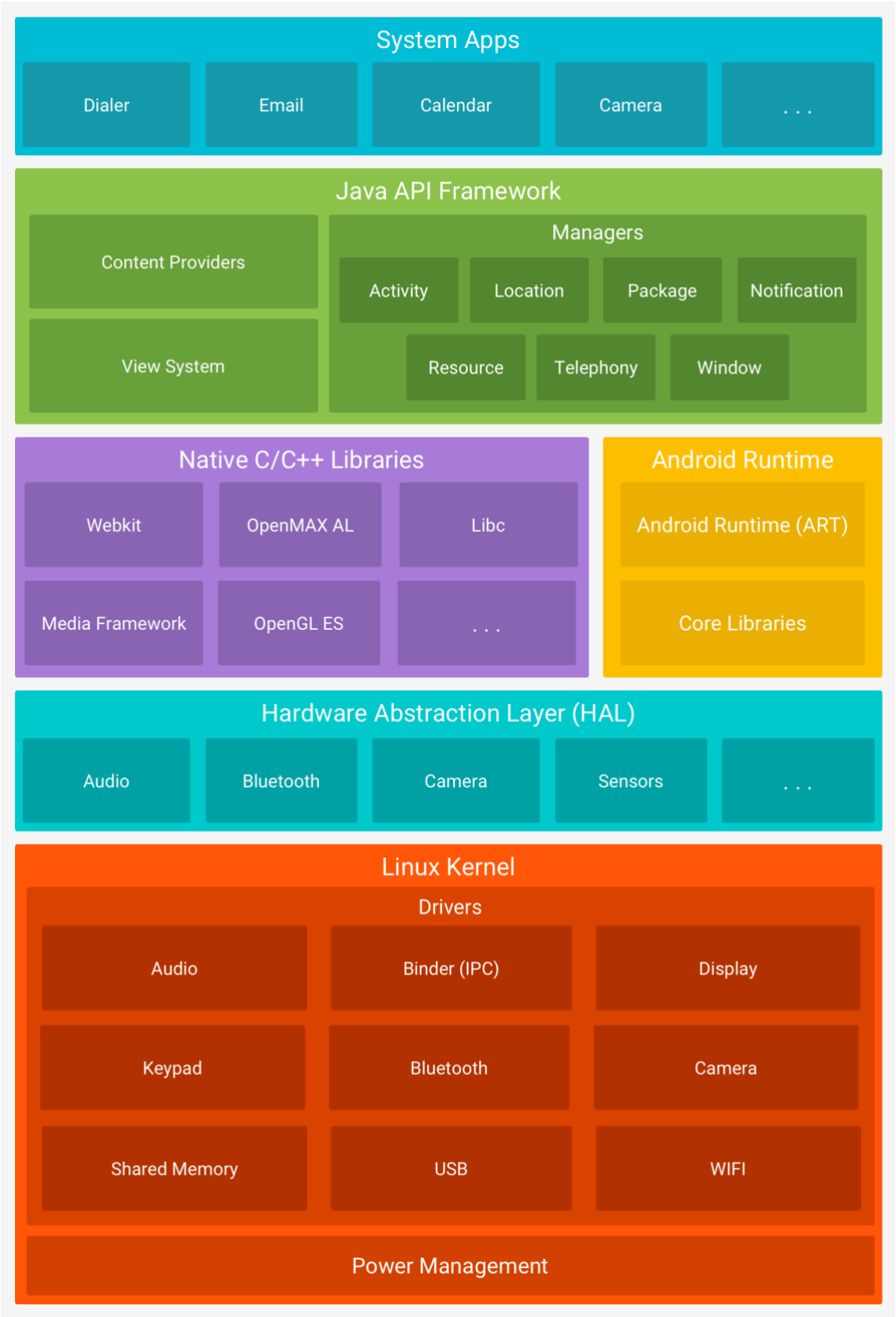

Ilustración 3. Arquitectura Sistema Operativo Android

- ***Systems Apps*:** esta capa contiene tanto las aplicaciones descargadas por el usuario, como las propias del Sistema Operativo. Todas estas aplicaciones hacen

uso de los servicios, las APIs (*Application Programming Interface*) y las librerías de las capas anteriores.

- ***Java API Framework*:** este representa el conjunto de todas las herramientas de desarrollo disponibles para cualquier tipo de aplicación. Todas las aplicaciones desarrolladas para *Android* poseen el mismo conjunto de APIs y el mismo *framework*. Entre las APIs más importantes, destacan las siguientes:

    - *Activity Manager*: conjunto API que gestiona el ciclo de vida de las aplicaciones.
    - *Window Manager*: gestiona las ventanas de las aplicaciones y utiliza una librería *Surface Manager*.
    - *Telephone Manager*: incluye todas las APIs vinculadas a las funcionalidades propias del teléfono, como son las llamadas, mensajes, GPS, …
    - *Content Provider*: permite que cualquier aplicación pueda compartir sus datos con las demás aplicaciones de *Android*, como puede ser los contactos, mensajes, registros de llamadas, …
    - *View System*: ofrece distintos elementos para poder construir interfaces de usuario (GUI).
    - *Location Manager*: permite a las aplicaciones obtener localización y posicionamiento.
    - *Notification Manager*: permite al usuario recibir notificaciones durante la ejecución de distintas aplicaciones móviles con el mismo formato.
    - *XMPP Service*: colección API para utilizar el protocolo de mensajes basado en XML (*eXtensible Markup Language*).

- ***Native C/C++ Libraries*:** en esta capa se encuentran las librerías utilizadas en *Android*. Éstas están escritas en C/C++ y proporcionan la mayor parte de las capacidades de *Android*, que, junto con el *kernel* de Linux, constituyen el corazón del Sistema Operativo. Las librerías más importantes son:

    - Librería libc: incluye las cabeceras y funciones estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.
    - Librería *Surface Manager*: se encarga de componer los elementos de navegación de la pantalla.
    - *OpenGL/SL* y *SGL*: representa las librerías gráficas. *OpenGL/SL* maneja gráficos en 3D, y en caso de que este disponible en el dispositivo, se trata del *hardware* encargado en proporcionar gráficos en 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que es la librería habitualmente utilizada por casi todas las aplicaciones. Cabe destacar, que se pueden desarrollar aplicaciones que combinen tanto gráficos en 2D, como gráficos en 3D.

  - *FreeType*: permite trabajar de forma rápida y sencilla distintos tipos de fuentes
  - Librería SSL: permite la utilización del protocolo SSL (*Secure Sockets Layer*) para proteger las comunicaciones.
  - Librería *SQLite*: creación y gestión de BBDD relacionales.
  - Librería *WebKit*: proporciona un motor para la navegación y es el núcleo principal del navegador incluido por defecto en *Android*.

- **Tiempo de ejecución de Android:** se sitúa al mismo nivel que las librerías *Android*. Constituido por las *Core Libraries*, las cuales poseen clases *Java* y el Android *Runtime*, que se trata de un entorno de ejecución de las distintas aplicaciones. Actualmente, *Android Runtime*, reemplaza a la máquina virtual *Dalvik*, utilizada hasta el momento.

- **Núcleo Linux:** el núcleo utilizado por *Android* es un núcleo Linux. Se trata de una capa de abstracción para el *hardware* disponible en los dispositivos. En esta capa se encuentran los *drivers* necesarios para que cualquier componente *hardware* pueda ser llamado.

El lenguaje nativo utilizado en *Android* es Java. Cualquier aplicación que use directamente el *hardware* y se comunique con el Sistema Operativo, utilizará código Java. Este lenguaje es muy extendido, tanto para aplicaciones a nivel de red como a nivel local (Arias, 2016).

2.3.2 SISTEMA OPERATIVO WINDOWS

Windows Phone se trata de un Sistema Operativo móvil desarrollado por *Microsoft*, como sucesor de *Windows Mobile*. El primer móvil con este Sistema Operativo apareció el 15 de Febrero de 2010 en la *Mobile World Congress* de Barcelona.

Las principales características del *Windows Phone* se describen a continuación:

La interfaz gráfica, se compone de mosaicos dinámicos, que no son más que accesos directos a otras aplicaciones. Se denominan dinámicos, ya que, en función de los últimos movimientos del usuario, el orden de las aplicaciones varía, siendo la primera en aparecer, la última utilizada por el usuario.

El asistente virtual de *Windows Phone* es *Cortana*, que permite realizar búsquedas tanto dentro del dispositivo como en la web. También permite realizar búsquedas en la web a través de códigos QR.

El fuerte de estos dispositivos es un *hub* con Xbox, que integra funcionalidades del *Xbox Live*, a través de las cuales el usuario puede retar a amigos, personalizar su propio *avatar* y compartir los logros conseguidos (Jangla, 2012).

El 8 de Octubre de 2017, *Joe Belfiore*, anunció que *Microsoft* no desarrollaría nuevas funciones o *hardware* para teléfonos debido a su baja tasa de mercado. Por ello, actualmente se encuentra disponible la versión *Windows 10 Mobile*, sucesor de *Windows Phone 8.1*. Esta versión se centrará únicamente en nuevas versiones y parches de mantenimiento.

La gran novedad de *Windows 10* fue la presentación de la Plataforma Universal de *Windows* (UWP), que presenta una plataforma común de aplicaciones para todos los dispositivos que dispongan de *Windows 10*, debido a que se encuentra en el *Core*. Es decir, si una aplicación utiliza las principales APIs de UWP, éstas son las mismas en todos los dispositivos *Windows* 10, con independencia de si se trata de un ordenador, auriculares, *Xbox*, etc.

Una aplicación para UWP está escrita en C++/*WinRT* o C++/*CX*, ya que éstas tienen acceso a la API de Win32 que forman parte de UWP (Microsoft, 2018).

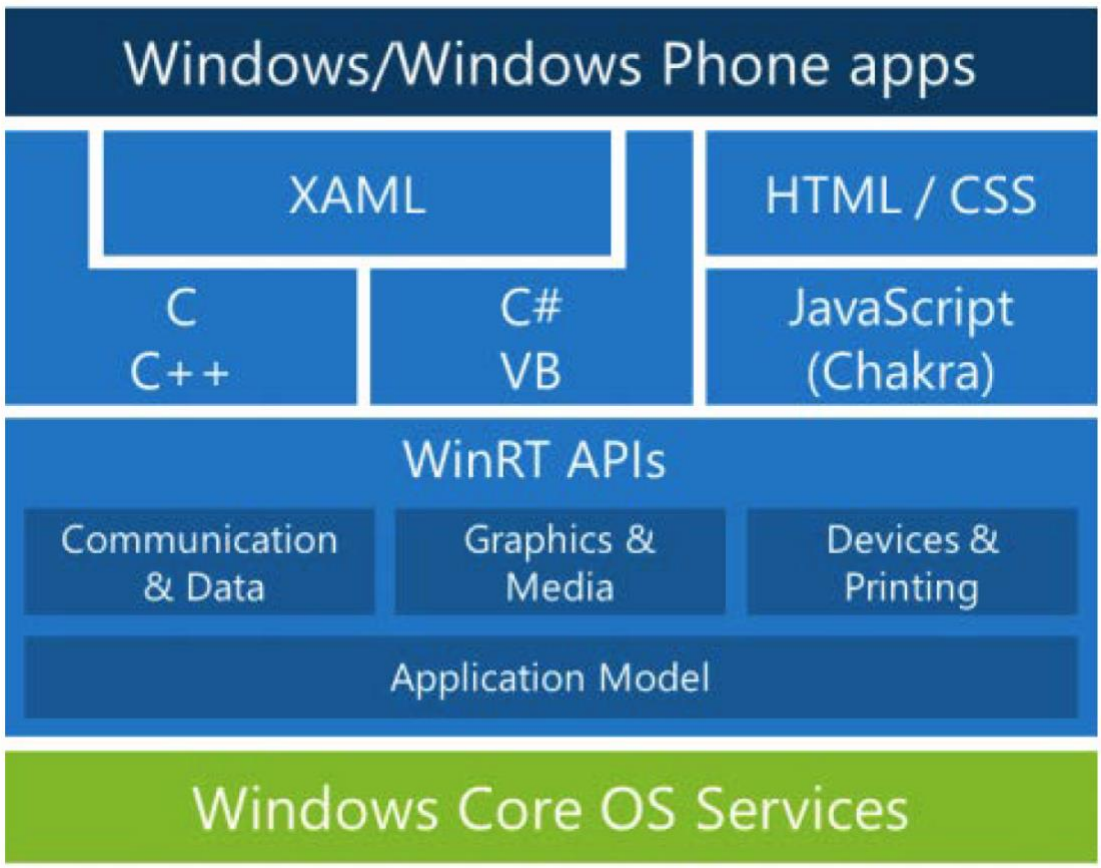

Ilustración 4. Arquitectura Windows Phone

En cuanto a los lenguajes de programación soportados por *Windows Phone*, son C y *Visual Basic .NET*. Esto se debe a que ambos son entendidos por el *framework* .NET. En cuanto al diseño, se utiliza un lenguaje *Silverlight*, también conocido como XAML (Microsoft, 2017).

2.3.3 SISTEMA OPERATIVO iOS

iOS se trata del Sistema Operativo creado por Apple para sus dispositivos móviles. Este Sistema Operativo nació el 19 de Junio de 2007 con el lanzamiento del primer iPhone. Actualmente se utiliza en distintos dispositivos de la empresa como son el iPad, iPod Touch y Apple TV.

El lanzamiento del primer iPhone supuso una gran revolución en el sector de la telefonía, haciendo que los gigantes de la telefonía móvil de aquella época, como lo eran Nokia, BlackBerry o Sony, se quedaran obsoletos de la noche a la mañana, sin tener tiempo de reacción para competir con Apple.

Desde su primer lanzamiento, cada año salen nuevas versiones del Sistema Operativo para aportar nuevas funcionalidades y mejoras (Fernández Pérez, 2013).

La interfaz gráfica de iOS se basa en gestos multitáctiles, que facilitan la interacción del usuario con el dispositivo de una forma más natural e intuitiva.

En cuanto la arquitectura, se estructura en distintas capas, las cuales integran funcionalidades para soportar el manejo de red y el soporte a múltiples sistemas de archivos. En esta arquitectura, las capas superiores son las encargadas de contener los servicios y tecnologías más importantes para el desarrollo de distintas aplicaciones, mientras que, en las capas más bajas, se controlan los servicios más básicos (Maskrey, 2016).

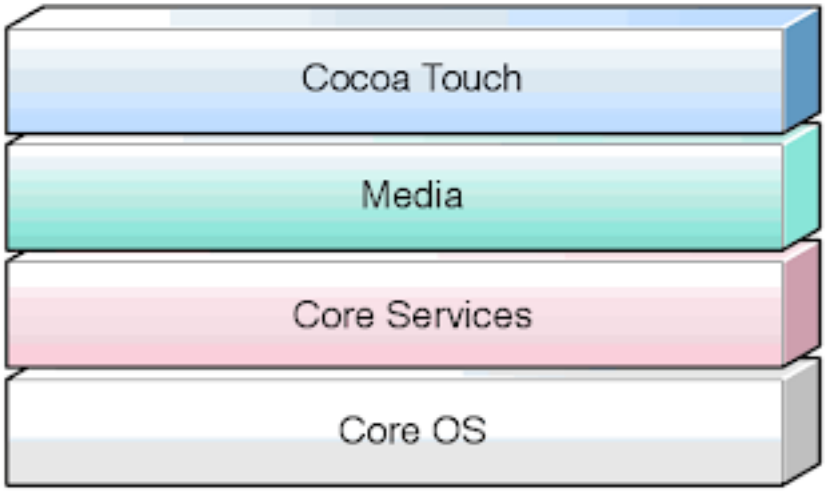

Ilustración 5. Estructura básica del Sistema Operativo iOS

A continuación, se analiza cada capa por separado:

- ***Cocoa Touch*:** se trata de la capa superior y una de las más importantes para el desarrollo de aplicaciones. Dicha capa posee un conjunto de *frameworks* que proporciona el API de *Cocoa* para desarrollar aplicaciones. Los dos *frameworks* principales de esta capa son (Apple, 2018b):

    - *UIKIT*: proporciona la infraestructura necesaria para aplicaciones tanto para iOS como TVOS. Contiene las clases necesarias para el desarrollo de una interfaz de usuario, con el manejo de distintos eventos de *Multi-Touch* y otros tipos de entrada. Otras características ofrecidas, son el soporte de animación, soporte de documentos y soporte de dibujos e impresión, administración y visualización de texto, así como soportes de búsqueda y accesibilidad a distintos recursos

    - *Foundation Framework*: define las clases más básicas para el manejo y acceso de objetos y servicios del Sistema Operativo.

- ***Media*:** provee tanto los servicios gráficos como de multimedia de la capa inmediatamente superior.

- ***Core Services*:** contiene los servicios fundamentes del sistema, de los cuales, las distintas aplicaciones hacen uso.

- ***Core OS*:** contiene lo que se conoce como características de bajo nivel. Estas características son el sistema de ficheros, el manejo de la memoria, seguridad, *drivers*…

iOS utiliza *Swift* como lenguaje de programación. Fue desarrollado por *Apple* para el desarrollo de aplicaciones iOS y macOS. Fue presentado en la *Worldwide Developers Conference* en el 2014. *Swift* está diseñado para integrarse con *frameworks* de *COCOA* y puede utilizar cualquier biblioteca programada en *Objective-C* y realizar llamadas a funciones en C (Apple, 2018a).

2.4 ELECCIÓN DE LA TECNOLOGÍA

Lo primero a tener en cuenta es que la aplicación está desarrollada anteriormente para dispositivos tanto de Sistema Operativo *Android* como de Sistema Operativo iOS. En el caso de este último, cabe destacar que se desarrolló de forma no nativa a través de la herramienta *PencilCase*, haciendo que la aplicación presente problemas de rendimiento y de tamaño. Con el fin de evitar estos inconvenientes, se ha optado por desarrollar la aplicación de forma nativa.

Otro de los factores que se han tenido en cuenta a la hora de elegir esta tecnología, es que, en el lugar de destino de esta aplicación, Canadá, el Sistema Operativo mayoritario utilizado en el sector de la sanidad es iOS. Como se puede observar en la siguiente ilustración, Apple acapara más del 48% de todos los dispositivos del país.

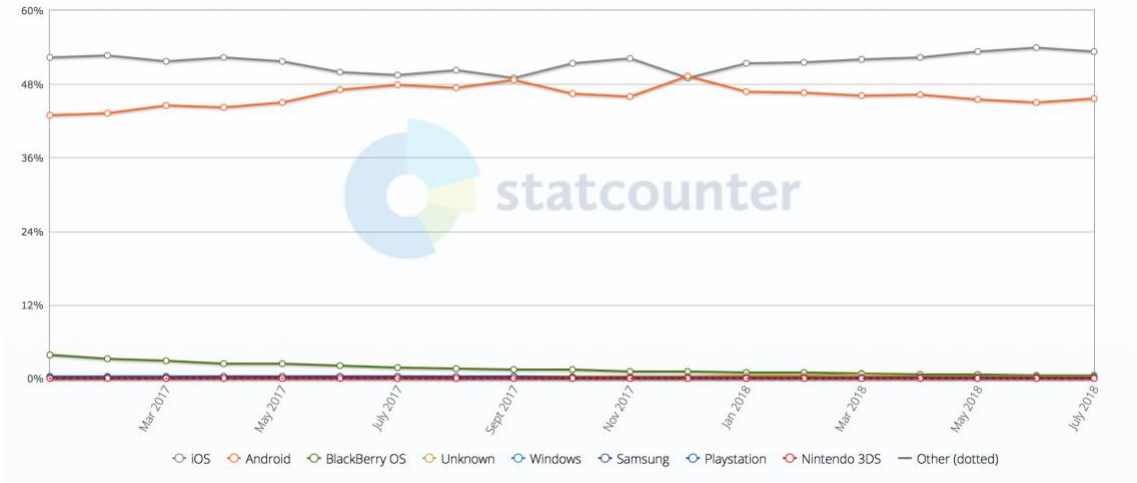

Ilustración 6. Sistemas Operativos más utilizados en Canadá. Fuente statcounter

La elección de desarrollar la aplicación para el Sistema Operativo iOS implica que se está desarrollando para el mayor vendedor de teléfonos móviles. Así mismo, adaptar la aplicación a la última versión del Sistema Operativo implica que se puede obtener el máximo rendimiento del dispositivo, así como las nuevas características que este posea.

Otro factor a tener en cuenta es que las aplicaciones para iOS poseen una mayor experiencia de usuario y se obtiene mayor rendimiento.

CAPITULO 3: TECNOLOGIA UTILIZADA: iOS

iOS se trata del Sistema Operativo de la multinacional *Apple Inc* para sus dispositivos móviles. En un principio se desarrolló para el primer *iPhone* (iPhone OS), pero posteriormente se ha utilizado en distintos dispositivos móviles de la empresa. Una de las particularidades de este Sistema Operativo, es que no permite ser instalado en *hardware* de terceros. Dado que iOS es propiedad de *Apple*, como se ha indicado anteriormente, éste deriva de macOS, por lo que se trata de un Sistema Operativo de tipo *Unix*, basado en *Darwing BSD*.

A nivel mundial, se trata del segundo Sistema Operativo móvil más utilizado en el mundo, detrás *Android*, con una cuota de mercado que ronda el 15%, siendo predominante su uso en los EE.UU.

La última versión del Sistema Operativo fue lanzada en Junio de 2018, se trata de iOS 12, el cual sustituye a iOS 11, con el fin último de mejorar la experiencia del usuario (García Domínguez, 2018).

Los principales elementos de control, a través de los cuales el usuario interacciona con el Sistema Operativo, consisten en deslizadores, interruptores, botones y gestos (tales como agrandar o disminuir una imagen, cerrar aplicaciones, …). Dado que posee una interfaz muy fluida, la respuesta a las órdenes del usuario son inmediatas.

3.1 VERSIÓN UTILIZADA

La versión utilizada para nuestra aplicación del Sistema Operativo iOS es la 10.3, la cual fue lanzada en Marzo de 2017. Se trata de una mejora de la versión de iOS 10. Dicha versión se anunció en la *WorldWide Developers Conference* (WWDC) el 13 de Junio de 2016, aunque no se publicó hasta el 13 de Septiembre de 2016, día en el que se publicó la primera versión beta del sistema (Aplicaciones para móviles, 2018).

Los principales cambios ofrecidos se observaron en el diseño de las notificaciones, las cuales adquieren nuevas funciones, permitiendo interactuar con el 3D *Touch*, aún estando el dispositivo bloqueado. También se desarrollaron ciertos gestos más naturales e intuitivos para los usuarios. De entre ellos destaca que ya no se tenía que deslizar para desbloquear, sustituyéndose esta acción por la de pulsar el botón de inicio.

Se incluyó por primera vez la opción *Raise to Wake*, que permite activar la pantalla simplemente con girar el dispositivo hacia el usuario. Esta nueva mejora se debió a las quejas de los distintos usuarios por la rapidez en la que actúa el reconocimiento táctil, que no les permitía activar la pantalla sin desbloquearla, cuando solo pretendían visualizar las notificaciones.

El mayor cambio se observó en la aplicación de Música, ya que la interfaz se rediseñó por completo, manteniendo todas las funciones previas, pero haciéndolas más fáciles de usar, además de nuevas funciones como nuevos modelos de ordenación inteligente o poder visualizar las letras de las canciones.

El asistente virtual de iOS, *Siri*, permitió se que se incorporaran aplicaciones de terceros, adquiriendo así nuevas funcionalidades.

En relación con Siri, *QuickType* comenzó a mostrar sugerencias a la hora de escribir en función de la pregunta realizada por el interlocutor realizada en la aplicación *iMessage*. Estas sugerencias están relacionadas con preguntas sobre tu posible ubicación, la cual te sugiere enviar tu información o incluso, puede sugerir crear un evento en el calendario partiendo de la información que haya obtenido sobre una conversación. Además, detecta automáticamente el lenguaje en el que se está escribiendo, por lo que no es necesario cambiarlo manualmente. La aplicación de mensajería instantánea, *iMessage*, permitió la inclusión de emoticonos como sugerencias de escritura, o incluso se pueden hacer los emoticonos tres veces más grandes de lo normal. En cuanto a los mensajes, se permitió destacar ciertos mensajes en una conversación, haciéndolo más o menos grande, o poder ocultarlos para que el destinatario tenga que buscarlos. Se añadieron nuevos efectos en la pantalla, como son los cambios de fondo de pantalla al enviar un mensaje. Estos cambios se denominan *stickers* y se tratan de emoticonos que poseen vida. En cuanto a la seguridad de los mensajes, se incluyo el cifrado extremo a extremo.

En lo referente a la galería, permite visualizar las fotografías sobre un mapa, mostrando los distintos lugares en las que fueran creadas. Se incluyó el reconocimiento facial, pudiendo clasificar las fotos en función de la persona que aparezca en ellas, según los objetos o incluso según las escenas

La aplicación de Mapas mejoró mostrando una interfaz mucho más limpia, incluyendo nuevos gestos relacionados con el motor de búsqueda. Dicho motor de búsqueda ofrece sugerencias tales como lugar de trabajo, ya que, debido a la experiencia, la aplicación puede obtener la hora aproximada a la que el usuario acude al trabajo. En cuanto a la navegación, obtiene el estado del tráfico en la ruta del usuario, además de permitir realizar nuevas búsquedas de ubicación con la navegación activada.

Cabe destacar, según fuentes oficiales de Apple, que el día de la presentación de su sucesor, iOS 11, el 86% de los dispositivos compatibles con iOS 10 estaban actualizados a dicha versión. Con lo cual, implica que estamos ofreciendo la mejor experiencia de usuario al llegar a la mayoría de los usuarios que utilizan iOS (Torné, 2018).

3.2 ARQUITECTURA

En un principio, iOS fue diseñado para ser un Sistema Operativo embebido y pequeño. Por ello, iOS comparado con otros Sistemas Operativos, tiene una arquitectura simple, estando diseñada para ocupar poca memoria dentro del dispositivo (Bollapragada et al., 2000).

La arquitectura de iOS se trata de una arquitectura por capas. En la capa superior, iOS trabaja como el intermediario entre el *hardware* y las aplicaciones. Es decir, las aplicaciones no se comunican directamente con el *hardware*.

Las aplicaciones se comunican con el *hardware* a través de una colección de interfaces del sistema definidas. Estas interfaces facilitan la escritura de una aplicación que hace uso constantemente de recursos del *hardware*.

Las capas inferiores ofrecen todos los servicios básicos de los que consta toda aplicación, mientras que la capa de nivel superior proporciona gráficos sofisticados y servicios relacionados con la interfaz.

Apple proporciona la mayoría de sus interfaces de sistemas en paquetes especiales denominadas *frameworks*. Los *frameworks* no son más que un directorio que contiene una biblioteca compartida dinámica, es decir, imágenes, texto enriquecido, etc. Cada capa tiene un conjunto de *framework* que el desarrollador utiliza para construir las aplicaciones.

Las distintas capas se caracterizan por (Sol Llaven, 2015):

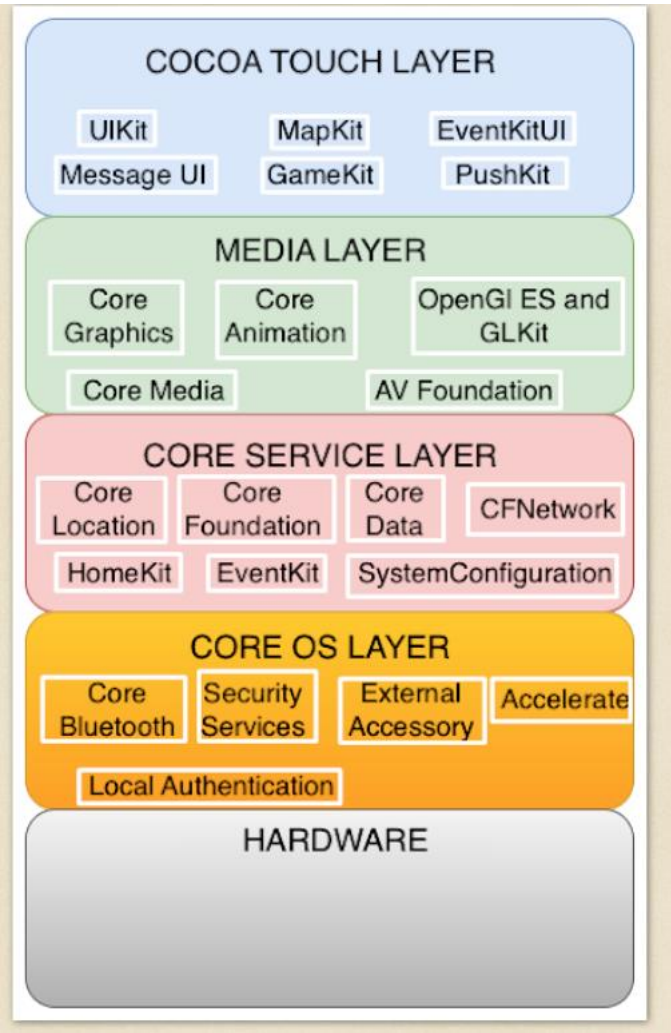

Ilustración 7. Estructura del Sistema Operativo iOS basada en el kernel Darwin

- *Core OS*: capa principal del Sistema Operativo, la cual contiene características de bajo nivel sobre la que se basan la mayoría de las aplicaciones como son el *Bluetooth*, autenticación local y servicios de seguridad entre otros.

- *Core Servides*: algunos de los *frameworks* más importantes que están disponibles para esta capa son los siguientes:

    - *Addres book framework*: ofrece acceso a la base de datos de contactos del usuario.
    - *Cloud kit framework*: proporciona un medio para poder mover datos desde una aplicación hasta iCloud.
    - *Core data Framework:* administra un modelo de datos para una aplicación del tipo *Model View Controller* (MVC). Este modelo por un lado define los componentes para la representación de la información, y por otro lado los componentes para la interacción del usuario

  - *Core Foundation framework*: interfaz que ofrece características fundamentales para la administración de datos y servicios de las aplicaciones.
  - *Core Location framework*: ofrece información de ubicación.
  - *Core Motion framework*: éste *framework* tiene acceso a los datos relacionados a movimientos disponibles en un dispositivo, como puede ser el acelerómetro.
  - *Fundation framework*: es un *framework* de interfaz de usuario *responsive*. proporciona una cuadrícula *responsive* e incluye componentes de interfaz de usuario HTML y CSS, plantillas, y fragmentos de código, incluyendo tipografía, formularios, botones, barras de navegación y otros componentes de interfaz usuario, así como extensiones de *Javascript* opcionales.
  - *Healthkit framework*: *framework* para el manejo de información relacionada con la salud del usuario.
  - *Homekit framework*: nuevo *framework* que permite controlar los dispositivos conectados en la casa del usuario.
  - *Social framework*: ofrece una interfaz para el acceso a las redes sociales.
  - *StoreKit framework*: brinda soporte en la compra de contenido y servicios dentro de las aplicaciones. Esto se conoce como *asIn-App Purchase.*

- Media: provee de tecnología para gráficos, audio y vídeo. Para ello dispone de distintos *frameworks*.

  - *Framework* para gráficos:
    - *UIKit Graphics*: soporte a alto nivel para el diseño de imágenes y se utiliza para animar el contenido de las vistas de una aplicación.
    - *Core Graphics framework*: es el motor de dibujo nativo para las aplicaciones de iOS y brinda soporte para el renderizado 2D basado en vectores e imágenes.
    - *Core Animation*: se trata de una tecnología que optimiza la de animación de las aplicaciones.
    - *Core Images:* ofrece soporte avanzado para controlar vídeo e imágenes inmóviles de una manera no destructiva.
    - *OpenGl ES* y *GLKit*: gestiona la representación avanzada en 2D y 3D mediante interfaces aceleradas (RAD) por hardware.
    - *Metal*: permite un rendimiento muy elevado en la renderización de gráficos sofisticados. Ofrece muy bajo consumo de acceso a la GPU del dispositivo.

  - *Frameworks* para audio:
    - *Media Player Framework*: es un *framework* de alto nivel que utiliza la biblioteca de iTunes del usuario y es compatible con la reproducción de listas de reproducción.

        - *AV Foundation*: es una interfaz *Objective-C* para manejar la grabación y reproducción de audio y vídeo.
        - *OpenAL*: es una tecnología estándar de la industria para proporcionar audio.

    - *Frameworks* para vídeo:
        - *AV Kit: framework* que ofrece una colección de interfaces fáciles de usar para presentar vídeo.
        - *AV Foundation:* brinda capacidad avanzada de reproducción y grabación de vídeo.
        - *Core Media*: se trata de un *framework* que describe las interfaces de bajo nivel y los tipos de datos para medios operativos.

- *Cocoa*: está compuesta de los siguientes *frameworks*:
    - *EventKit framework:* permite a los controladores de vista mostrar las interfaces del sistema para ver y modificar eventos relacionados con el calendario
    - *GameKit Framework*: da soporte a *Game Center* que permite a los usuarios compartir información relacionada con juegos en línea
    - *iAd Framework*: permite publicar publicidades basadas en *banners* desde su aplicación.
    - *MapKit Framework*: proporciona un mapa desplazable que puede incluir en la interfaz de usuario de la aplicación.
    - *PushKitFramework:* proporciona soporte de registro para aplicaciones VoIP.
    - *Twitter Framework:* admite una interfaz de usuario para generar *tweets* y soporte para crear URL que permiten acceder al servicio de *Twitter*.
    - *UIKit Framework*: proporciona una infraestructura vital para aplicaciones gráficas controladas por eventos en iOS. Algunas de las funciones importantes del marco del Kit de UI son:
        - Soporte multitarea.
        - Gestión e infraestructura de aplicaciones básicas.
        - Gestión de la interfaz de usuario
        - Soporte para eventos táctiles y de movimiento.
        - Cortar, copiar y pegar soporte y muchos más.

3.3 ENTORNO DE DESARROLLO

Xcode se trata de un entorno de desarrollo integrado (IDE – *Integrated Development Environment*) para los Sistemas Operativos de Apple. Dicho entorno contiene las herramientas creadas por Apple para el desarrollo de aplicaciones para los distintos

dispositivos de la compañía Apple, es decir, para macOS, iOS, watchOS y tvOS (Apple Developer, 2018).

La primera versión de Xcode fue el 24 de Octubre de 2003, junto a la versión 10.3 del Sistema Operativo Mac OS X. Fue desarrollado a partir del anterior entorno de desarrollo, denominado *Project Builder*, al que sustituyó.

La siguiente gran aparición de Xcode fue con la versión 2.1 en Junio de 2005. El boom de esta versión fue que permitió a los desarrolladores el conjunto de herramientas necesarias para poder crear binarios universales, dichos binarios permiten que el *software* creado para macOS pueda ser ejecutado tanto en la arquitectura *PowerPC* (nombre original de una arquitectura de tipo RISC, desarrollada por IBM, Motorola y Apple), como en las arquitecturas basadas en Intel. Esta versión integró, además, nuevas herramientas y marcos de trabajo del tipo *WebObjects*, los cuales permitían construir aplicaciones y servicios web de *Java*. Con anterioridad a esta versión, las herramientas *WebObjects* se vendían como un producto separado (Daniel, 2011).

Con el lanzamiento de Mac OS X con la versión 10.5, se lanzó también Xcode 3.0, cuyas principales novedades fue la inclusión de *Objective-C 2.0*, con un nuevo *Interface Builder*, además de poder refactorizar proyectos y *screenshots* del proyecto.

La última gran versión fue Xcode 4.0, lanzada a principios de 2011, simplemente incluía una nueva interfaz y nuevas compatibilidades.

Como se indicó al principio del capítulo, Xcode se trata de un entorno de desarrollo integrado (IDE), en consecuencia ofrece las herramientas y características necesarias para la creación de una aplicación. A través de ella podemos escribir, ejecutar y *debuggear* nuestro código, a parte de poder probar la interfaz de usuario (Ohland & Varma, 2016):

Un proyecto de Xcode consta de archivos fuentes (como son archivos en *Swift*, *nibs* de construcción de la interfaz y modelos de objetos), recursos (como pueden ser imágenes o archivos de texto enriquecido) y, por último, un archivo del proyecto Xcode, en el que podemos encontrar la configuración y las reglas de construcción. Un proyecto en Xcode se puede entender como una colección de fuentes y recursos, junto con un archivo del proyecto, unidos todos juntos.

Xcode permite combinar distintos proyectos en un único *workspace*, el cual se denomina *workspace window*. El *workspace window* está dividido en múltiples áreas, como podemos observar en la siguiente ilustración.

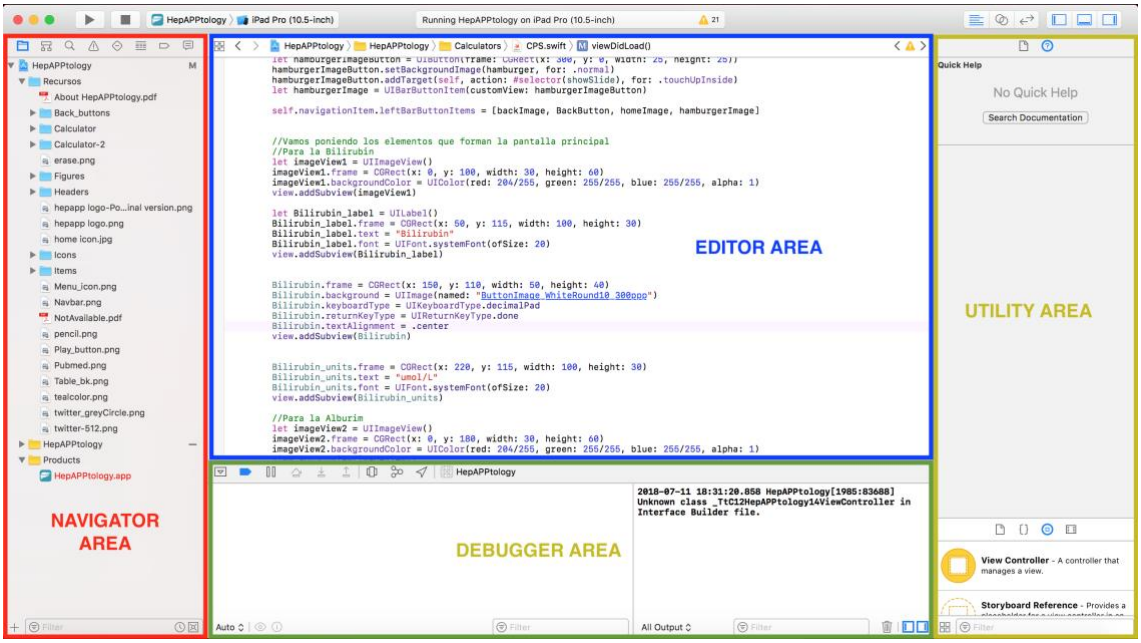

Ilustración 8. Entorno desarrollo Xcode

- ***Navigator area*:** el área de navegación consiste en un conjunto de paneles. Se encuentra en la parte izquierda de la ventana del *workspace*, siendo la interfaz principal para organizar y explorar los distintos paneles de navegación, como son: el navegador del proyecto, el navegador de símbolos, el navegador de problemas de construcción, el navegador de *logs* de ejecución, el navegador de puntos de ruptura, el navegador de hilos y pilas, y el navegador de búsquedas en el proyecto.

- ***Editor area*:** el área del editor cambia dinámicamente entre los distintos editores en función del archivo seleccionado (ya sea a través del navegador del proyecto o a través del *bar jump*). Esto significa que, para los archivos de código, se observará tanto editor fuente (archivos con extensión *.swift*) como el editor de la interfaz (archivos con extensión .h). En el contexto del entorno de desarrollo, incluye código autocompletado y texto destacado para código escrito tanto en *Swift* como en *Objetive-C*. Otros lenguajes que también están soportados son: *JavaScript*, *Ruby*, *C*, *AppleScript*, *XML* y *JSON*.

- ***Utility area*:** el área de utilidad ofrece información complementaria y controles para el actual editor. Esencialmente, cualquier cosa que se necesite para el desarrollo, se encuentra en esta área.

- ***Debug area*:** esta área aparece por defecto cuando se ejecuta la aplicación. Se trata de la interfaz primaria del *debug*, basado en LLDB (*LLDB Debugger*). Ésta incluye tanto una barra de control, como una consola y una vista de las variables cuando el programa en ejecución está parado.

3.4 LENGUAJE DE PROGRAMACIÓN UTILIZADO: SWIFT

Swift se comenzó a desarrollar en Julio de 2010 por *Chris Latter*, con la colaboración de programadores de Apple. *Swift* se basa en varios lenguajes de programación, entre los que destacan *Objective-C, Rust, Haskell, Ruby, Python, C, CLU* y muchos más. El 2 de Junio de 2014, Apple presentó la primera aplicación escrita con *Swift* en la *Worldwide Developers Coference.* La primera versión beta del lenguaje se ofreció únicamente a los desarrolladores registrados, aunque la compañía no se comprometió con que la versión final del lenguaje fuera compatible con la versión beta. Por ello, Apple planteó realizar convertidores de código para el primer lanzamiento completo.

La primera versión se presentó el 9 de Septiembre de 2014, con la aparición de la versión 6 de Xcode. En el año 2015, *Swift* fue premiada con el primer premio al "*Most Loved Programming Language"* de *Stack Overflow*. Y en 2015, obtuvo el segundo premio.

En 2015, para facilitar el trabajo de los desarrolladores, IBM creó una web que permitió a los desarrolladores escribir código en un panel y mostrar su salida en otro.

Durante la *Worldwide Developers Coference* del 2016, Apple anunció una aplicación exclusiva para iPad, denomina *Swift Playgrounds*, con la que se pretendía enseñar a las personas a programar con *Swift*. Se presentó a través de un vídeo en 3D, que proporciona retroalimentación cuando alguna línea de código posee algún error.

Swift se trata de una alternativa al lenguaje *Objective-C*, proporcionando nuevos conceptos de lenguajes de programación y presentando una sintaxis más simplificada.

En contraste con *Objective-C*, *Swift* no posee punteros. Además, se realizan llamadas a métodos que no contienen estilo de notación, por lo que es más familiar para los programadores de lenguajes orientados a objetos, como lo son Java o C. *Swift* introduce verdaderos parámetros nombrados y contiene conceptos claves de *Objective-C*, como son los protocolos, cierres y categorías, utilizando una sintaxis más limpia.

CAPITULO 4: DESCRIPCIÓN TÉCNICA

HepAPPtology se trata de una aplicación móvil ideada por el Dr. Kelly Burak, docente en la Universidad de Calgary, quien imparte una asignatura relacionada con el carcinoma hepatocelular. Esta aplicación está ya desarrollada para dispositivos Android y en caso de dispositivos iOS, a través de la aplicación *PencilCase*, de forma no nativa. Al no tratarse de una aplicación nativa, presenta problemas de rendimiento y de espacio, ya que es mucho más pesada. Para solventar los problemas presentados con *PencilCase*, se pensó en desarrollar la aplicación de forma nativa.

4.1 INTERFAZ DE USUARIO

La aplicación consta de distintos tipos de interfaces, para poder ofrecer una mayor sensación de dinamismo.

Como podemos observar en la siguiente ilustración, *AppDelegate.swift* se trata de la clase principal, en la que se indicamos el controlador de vista con el que se inicia la aplicación (*loginController.swift)*.

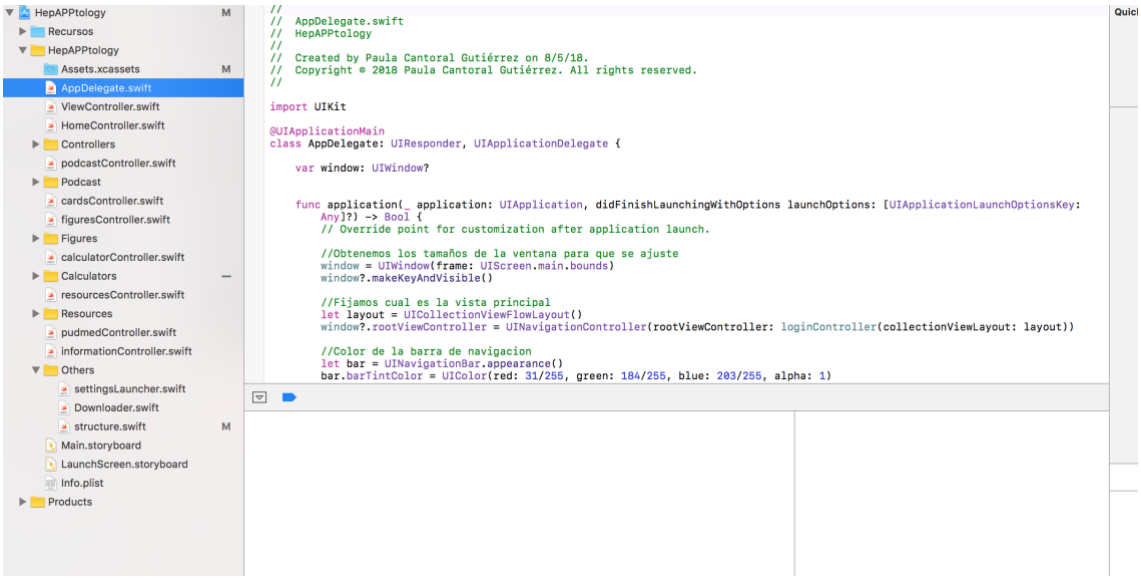

Ilustración 9. Vista principal del proyecto HepAPPtology

El principal objetivo de la clase *AppDelage.swift* es asegurarse de que la aplicación interactúa correctamente con el sistema y con otras aplicaciones. Concretamente, las clases que delegan de esta conservan las principales características definidas para el proyecto. Estas características son el *segue* y el *navigationBar*, el cual consta de distintos *barButtonItems* (botones *back*, *Home*, *Menu* y *Screenshots*) y del título de cada interfaz, que varía en función de la vista en la que nos encontremos como podemos observar en la Ilustración.

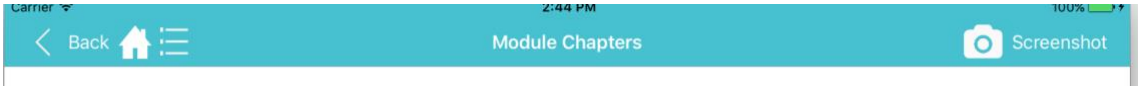

Ilustración 10. NavigationBar del proyecto HepAPPtology

4.2 NAVIGATIONBAR

Como hemos mencionado anteriormente, el *navigationBar* consta de los *barButtonItems* y del título de la interfaz de usuario en la que nos encontremos.

El *navigationBar* está compuesto de:

- **Botón de retroceso:** está formado por la imagen de retroceso, así como el texto *back*. Con este botón conseguimos ir a la vista inmediatamente anterior.

- **Botón *Home*:** con este botón vamos a la vista principal, en la que podemos observar los distintos módulos de los que consta la aplicación.

- ***Screenshot*:** está formado por la imagen de una cámara de fotos así como de un texto indicativo. Con esto realizamos una captura de pantalla de la interfaz en la que nos encontramos, para poder compartirla.

El título depende de la vista en la que nos encontremos.

4.3 ESTRUCTURA DE FICHEROS

El proyecto desarrollado, presenta la siguiente estructura de ficheros:

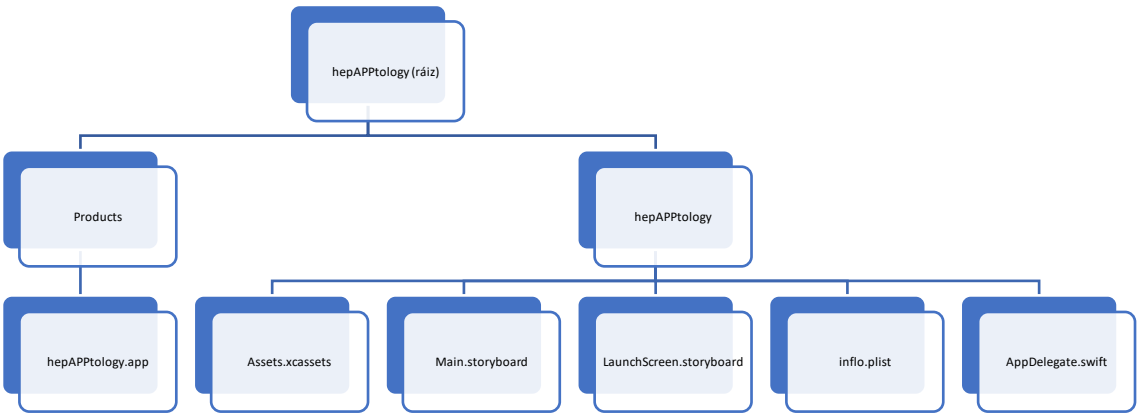

Ilustración 11. Estructura de ficheros

Como se puede observar, en el nivel superior tenemos la ruta principal (hepAPPtology (raíz)), de la cual cuelga la carpeta *Products*, en la que tenemos la aplicación, y la carpeta hepAPPtology. En esta última, nos encontramos con:

- ***Assets.xcassets*:** se utiliza para manejar los distintos iconos e imágenes utilizados en la aplicación.

- ***LaunchScreen.storyboard*:** permite crear interfaces de la aplicación, en las que se puede ver los flujos de navegación y en la que no se utiliza apenas código, ya que es una aplicación multivista.

- ***Info.plist*:** en ella se definen las propiedades del proyecto, basándose en un diccionario del tipo clave-valor que se puede almacena en nuestro sistema de archivos. En el caso de nuestro proyecto, las propiedades definidas más destacadas son: se permite utilizar cualquier protocolo de navegación (TCP/IP, UDP, …), permitir el acceso a la galería para poder almacenar las capturas de pantalla realizas, así como a la ruta de documentos para poder almacenar los distintos capítulos de la aplicación y la orientación del dispositivo.

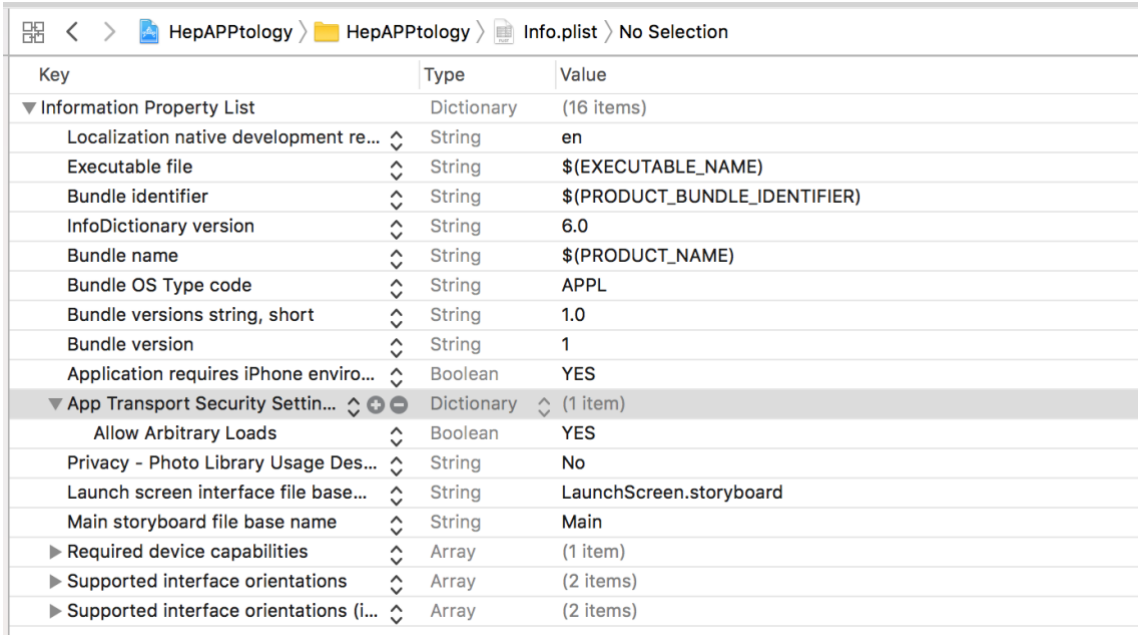

Ilustración 12. Propiedades del proyecto

- ***AppDelegate.swift*:** se trata del punto inicial del que parte nuestra aplicación. En el caso de nuestra aplicación, se definen los parámetros comunes a todas las pantallas de la aplicación y, como hemos mencionado antes, se define la interfaz que aparece al iniciar la aplicación. En nuestro caso, ésta se trata de una pantalla de *login*. Una vez el usuario se haya *logueado*, se muestra la pantalla principal, en la que el usuario puede acceder a las distintas funcionalidades disponibles

4.4 FUNCIONALIDADES

La aplicación consta de distintas funcionalidades, en un principio el usuario debe autenticarse en la aplicación:

Diagrama 1. Diagrama UML autenticación usuario

Una vez que el usuario se ha autenticado en la aplicación, dispone de las siguientes funcionalidades:

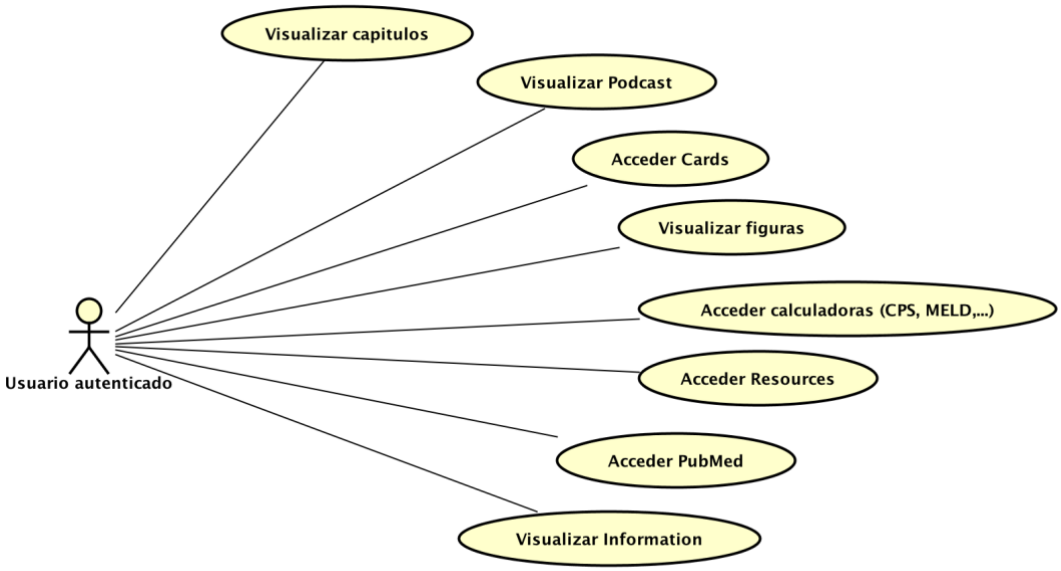

Diagrama 2. Diagrama UML funcionalidades usuario autenticado

A continuación, se describen los casos de uso de cada una de las funcionalidades.

4.4.1 VISUALIZAR CAPITULOS

IDENTIFICADOR	Visualizar Capítulos	
VERSIÓN	1	
FECHA ÚLTIMA REVISIÓN	30/06/2018	
AUTORES	Paula Cantoral Gutiérrez	
DESCRIPCIÓN	Visualización de los diferentes capítulos.	
ACTORES	Usuario autenticado	
PRECONDICIÓN	El usuario debe estar autenticado y disponer de conexión a internet	
ESCENARIO SECUNCIAL DE ÉXITO O SECUENCIA O FLUJO NORMAL	**PASO**	**ACCIÓN**
	1	El usuario debe elegir la opción *Chapters* del módulo principal.
	2	El sistema presentará los distintos módulos disponibles.
	3	El usuario elegirá un módulo.
	4	El sistema presentará los distintos capítulos disponibles para el módulo seleccionado por el usuario.
	5	El usuario seleccionará un capítulo.
	6	El sistema presentará el capítulo seleccionado
	7a	El usuario finaliza el capítulo
	7b	El usuario elige la opción descargar el capítulo
POSTCONDICIÓN	Se ha accedido correctamente a la funcionalidad deseada.	
ALTERNATIVAS O FLUJO ALTERNATIVO	**PASO**	**ACCIÓN**
	1-6	El usuario pulsa el botón *back*, la secuencia retrocede una secuencia de uso.
	1-6	Si el usuario pulsa el botón *Home*, se cambia a la

		pantalla principal y el caso de uso finaliza sin lograr la postcondición
	1-6	El usuario pulsa el botón de *Menu* pudiendo cambiar de capítulo o pasar a otro caso de uso.
REQUISITOS ESPECIALES	Ninguno.	
FRECUENCIA ESPERADA	Media.	
IMPORTANCIA	Media.	
URGENCIA	PD	
COMENTARIOS	No hay	

Tabla 1. Caso de uso Visualizar Capítulos

4.4.2 VISUALIZAR PODCASTS

IDENTIFICADOR	**Visualizar *Podcasts***	
VERSIÓN	1	
FECHA ÚLTIMA REVISIÓN	30/06/2018	
AUTORES	Paula Cantoral Gutiérrez	
DESCRIPCIÓN	Visualización de los diferentes *Podcasts*	
ACTORES	Usuario autenticado	
PRECONDICIÓN	El usuario debe estar autenticado y disponer de conexión a internet	
ESCENARIO SECUNCIAL DE ÉXITO O SECUENCIA O FLUJO NORMAL	**PASO**	**ACCIÓN**
	1	El usuario debe elegir la opción *Podcasts* del módulo principal.
	2	El sistema presentará los distintos módulos disponibles.
	3	El usuario elegirá un módulo.
	4	El sistema presentará los distintos *podcasts* disponibles para el módulo seleccionado por el usuario.
	5	El usuario seleccionará un *podcast*.

	6	El sistema presentará el *podcast* seleccionado.
	7a	El usuario finaliza el *podcast*.
	7b	El usuario elige la opción ver capítulos del módulo al que pertenece el *podcast*.
	7c	El usuario elige ir a un *podcast* continuo
POSTCONDICIÓN	Se ha accedido correctamente a la funcionalidad deseada.	
ALTERNATIVAS O FLUJO ALTERNATIVO	**PASO**	**ACCIÓN**
	1-6	El usuario pulsa el botón *back*, la secuencia retrocede una secuencia de uso.
	1-6	Si el usuario pulsa el botón *Home*, se cambia a la pantalla principal y el caso de uso finaliza sin lograr la postcondición
	1-6	El usuario pulsa el botón de *Menu* pudiendo cambiar de capítulo o pasar a otro caso de uso.
REQUISITOS ESPECIALES	Ninguno.	
FRECUENCIA ESPERADA	Media.	
IMPORTANCIA	Media.	
URGENCIA	PD	
COMENTARIOS	No hay	

Tabla 2. Caso de uso Visualizar Podcasts

4.4.3 ACCEDER CARDS

IDENTIFICADOR	**Visualizar *Cards***
VERSIÓN	1
FECHA ÚLTIMA REVISIÓN	30/06/2018
AUTORES	Paula Cantoral Gutiérrez
DESCRIPCIÓN	El usuario accederá a la página web *Cards*.
ACTORES	Usuario autenticado

PRECONDICIÓN	El usuario debe estar autenticado y disponer de conexión a internet	
ESCENARIO SECUNCIAL DE ÉXITO O SECUENCIA O FLUJO NORMAL	**PASO**	**ACCIÓN**
	1	El usuario debe elegir la opción *Cards* del módulo principal.
	2	El sistema presentará la página web *Cards.*
POSTCONDICIÓN	Se ha accedido correctamente a la funcionalidad deseada.	
ALTERNATIVAS O FLUJO ALTERNATIVO	**PASO**	**ACCIÓN**
	1-2	El usuario pulsa el botón *back*, la secuencia retrocede una secuencia de uso.
	1-2	Si el usuario pulsa el botón *Home*, se cambia a la pantalla principal y el caso de uso finaliza sin lograr la postcondición
	1-2	El usuario pulsa el botón de *Menu* pudiendo cambiar de capítulo o pasar a otro caso de uso.
REQUISITOS ESPECIALES	Ninguno.	
FRECUENCIA ESPERADA	Media.	
IMPORTANCIA	Media.	
URGENCIA	PD	
COMENTARIOS	No hay	

Tabla 3. Caso de uso Acceder Cards

4.4.4 VISUALIZAR FIGURAS

IDENTIFICADOR	**Visualizar Figuras**
VERSIÓN	1
FECHA ÚLTIMA REVISIÓN	30/06/2018
AUTORES	Paula Cantoral Gutiérrez
DESCRIPCIÓN	Visualización de las diferentes figuras.
ACTORES	Usuario autenticado
PRECONDICIÓN	El usuario debe estar autenticado.

	PASO	ACCIÓN
ESCENARIO SECUNCIAL DE ÉXITO O SECUENCIA O FLUJO NORMAL	1	El usuario debe elegir la opción *Figures* del módulo principal.
	2	El sistema presentará los distintos módulos disponibles.
	3a	El usuario elige *Table of Contents.*
	4a	El sistema presentará los distintos *Table of Content* disponibles
	5a	El usuario elegirá un *Table of Content.*
	6a	El sistema presentará el *Table of Content* seleccionado por el usuario
	3b	El usuario elige *Schemes.*
	4b	El sistema presentará los distintos *Schemes* disponibles
	5b	El usuario elegirá un *Scheme.*
	6b	El sistema presentará el *Table of Content* seleccionado por el usuario
	3c	El usuario elige *Interactive Figures.*
	4c	El sistema presentará los distintos *Interactive Figures* disponibles
	5c	El usuario elegirá un *Interactive Figures.*
	6c	El sistema presentará el *Interactive Figures* seleccionado por el usuario.
	7c	El usuario puede utilizar la *slider* para visualizar distintas imágenes
	3d	El usuario elige *Drawing.*

	4d	El sistema presentará los distintos *Drawing* disponibles
	5d	El usuario elegirá un *Drawing*.
	6d	El sistema presentará el *Drawing* seleccionado por el usuario
	7d	El usuario puede dibujar sobre la figura seleccionada
POSTCONDICIÓN	Se ha accedido correctamente a la funcionalidad deseada.	
ALTERNATIVAS O FLUJO ALTERNATIVO	**PASO**	**ACCIÓN**
	1-6	El usuario pulsa el botón *back*, la secuencia retrocede una secuencia de uso.
	1-6	Si el usuario pulsa el botón *Home*, se cambia a la pantalla principal y el caso de uso finaliza sin lograr la postcondición
	1-6	El usuario pulsa el botón de *Menu* pudiendo cambiar de capítulo o pasar a otro caso de uso.
	7d	El usuario puede borrar el dibujo realizado
REQUISITOS ESPECIALES	Ninguno.	
FRECUENCIA ESPERADA	Media.	
IMPORTANCIA	Media.	
URGENCIA	PD	
COMENTARIOS	No hay	

Tabla 4. Caso de uso Visualizar Figuras

4.3.5 ACCEDER CALCULADORAS

IDENTIFICADOR	**Acceder Calculadoras**
VERSIÓN	1
FECHA ÚLTIMA REVISIÓN	30/06/2018
AUTORES	Paula Cantoral Gutiérrez

DESCRIPCIÓN	Visualización de las diferentes calculadoras	
ACTORES	Usuario autenticado	
PRECONDICIÓN	El usuario debe estar autenticado.	
ESCENARIO SECUNCIAL DE ÉXITO O SECUENCIA O FLUJO NORMAL	**PASO**	**ACCIÓN**
	1	El usuario debe elegir la opción *Calculators* del módulo principal.
	2	El sistema presentará las distintas calculadoras disponibles.
	3a	El usuario elige *Child Pugh Score.*
	4a	El sistema presentará la calculadora *Chil Pugh Score.*
	5a	El usuario deberá introducir los datos necesarios para la calculadora *Child Pugh Score.*
	6a	El sistema presentará el resultado de la calculadora *Child Pugh Score.*
	3b	El usuario elige MELD.
	4b	El sistema presentará la calculadora MELD.
	5b	El usuario deberá introducir los datos necesarios para la calculadora MELD.
	6b	El sistema presentará el resultado de la calculadora MELD.
	3c	El usuario elige *Okuda.*
	4c	El sistema presentará la calculadora *Okuda*.
	5c	El usuario deberá introducir los datos necesarios para la calculadora *Okuda.*

	6c	El sistema presentará el resultado de la calculadora *Okuda*.
	3d	El usuario elige *Clip*.
	4d	El sistema presentará la calculadora *Clip*.
	5d	El usuario deberá introducir los datos necesarios para la calculadora *Clip*.
	6d	El sistema presentará el resultado de la calculadora *Clip*.
	3e	El usuario elige *All Calculator*.
	4e	El sistema presentará la calculadora *All Calculator*.
	5e	El usuario deberá introducir los datos necesarios para la calculadora *All Calculator*.
	6e	El sistema presentará el resultado de la calculadora *All Calculator*.
POSTCONDICIÓN	Se ha accedido correctamente a la funcionalidad deseada.	
ALTERNATIVAS O FLUJO ALTERNATIVO	**PASO**	**ACCIÓN**
	1-6	El usuario pulsa el botón *back*, la secuencia retrocede una secuencia de uso.
	1-6	Si el usuario pulsa el botón *Home*, se cambia a la pantalla principal y el caso de uso finaliza sin lograr la postcondición
	1-6	El usuario pulsa el botón de *Menu* pudiendo cambiar de capítulo o pasar a otro caso de uso.
	7a,b,e	El usuario elige la opción *More Information*

	7e	El usuario pulsa sobre el botón ajustes.
	7e	El usuario elige la opción *Alberta HCC Algorithm*
REQUISITOS ESPECIALES	Ninguno.	
FRECUENCIA ESPERADA	Media.	
IMPORTANCIA	Media.	
URGENCIA	PD	
COMENTARIOS	No hay	

Tabla 5. Caso de uso Acceder Calculadoras

4.3.6 ACCEDER RESOURCES

IDENTIFICADOR	**Acceder *Resources***	
VERSIÓN	1	
FECHA ÚLTIMA REVISIÓN	30/06/2018	
AUTORES	Paula Cantoral Gutiérrez	
DESCRIPCIÓN	El usuario accederá a la página web *Resources*.	
ACTORES	Usuario autenticado	
PRECONDICIÓN	El usuario debe estar autenticado y disponer de conexión a internet	
ESCENARIO SECUNCIAL DE ÉXITO O SECUENCIA O FLUJO NORMAL	**PASO**	**ACCIÓN**
	1	El usuario debe elegir la opción *Resources* del módulo principal.
	2	El sistema presentará la página web *Resources.*
POSTCONDICIÓN	Se ha accedido correctamente a la funcionalidad deseada.	
ALTERNATIVAS O FLUJO ALTERNATIVO	**PASO**	**ACCIÓN**
	1-2	El usuario pulsa el botón *back*, la secuencia retrocede una secuencia de uso.
	1-2	Si el usuario pulsa el botón *Home*, se cambia a la pantalla principal y el caso de uso finaliza sin lograr la postcondición
	1-2	El usuario pulsa el botón de *Menu* pudiendo cambiar

		de capítulo o pasar a otro caso de uso.
REQUISITOS ESPECIALES	Ninguno.	
FRECUENCIA ESPERADA	Media.	
IMPORTANCIA	Media.	
URGENCIA	PD	
COMENTARIOS	No hay	

Tabla 6. Caso de uso Acceder Resources

4.3.7 ACCEDER PUBMED

IDENTIFICADOR	**Acceder *PubMed***	
VERSIÓN	1	
FECHA ÚLTIMA REVISIÓN	30/06/2018	
AUTORES	Paula Cantoral Gutiérrez	
DESCRIPCIÓN	El usuario accederá a la página web *PubMed*.	
ACTORES	Usuario autenticado	
PRECONDICIÓN	El usuario debe estar autenticado y disponer de conexión a internet	
ESCENARIO SECUNCIAL DE ÉXITO O SECUENCIA O FLUJO NORMAL	**PASO**	**ACCIÓN**
	1	El usuario debe elegir la opción *PubMed* del módulo principal.
	2	El sistema presentará la página web *PubMed.*
POSTCONDICIÓN	Se ha accedido correctamente a la funcionalidad deseada.	
ALTERNATIVAS O FLUJO ALTERNATIVO	**PASO**	**ACCIÓN**
	1-2	El usuario pulsa el botón *back*, la secuencia retrocede una secuencia de uso.
	1-2	Si el usuario pulsa el botón *Home*, se cambia a la pantalla principal y el caso de uso finaliza sin lograr la postcondición
	1-2	El usuario pulsa el botón de *Menu* pudiendo cambiar de capítulo o pasar a otro caso de uso.

REQUISITOS ESPECIALES	Ninguno.
FRECUENCIA ESPERADA	Media.
IMPORTANCIA	Media.
URGENCIA	PD
COMENTARIOS	No hay

Tabla 7. Caso de uso Acceder PudMed

4.4.8 VISUALIZAR INFORMATION

IDENTIFICADOR	**Visualizar *Information***	
VERSIÓN	1	
FECHA ÚLTIMA REVISIÓN	30/06/2018	
AUTORES	Paula Cantoral Gutiérrez	
DESCRIPCIÓN	El usuario accederá a la página web *Cards*.	
ACTORES	Usuario autenticado	
PRECONDICIÓN	El usuario debe estar autenticado y disponer de conexión a internet	
ESCENARIO SECUNCIAL DE ÉXITO O SECUENCIA O FLUJO NORMAL	**PASO**	**ACCIÓN**
	1	El usuario debe elegir la opción *Information* del módulo principal.
	2	El sistema presentará la información de la página principal.
POSTCONDICIÓN	Se ha accedido correctamente a la funcionalidad deseada.	
ALTERNATIVAS O FLUJO ALTERNATIVO	**PASO**	**ACCIÓN**
	1-2	El usuario pulsa el botón *back*, la secuencia retrocede una secuencia de uso.
	1-2	Si el usuario pulsa el botón *Home*, se cambia a la pantalla principal y el caso de uso finaliza sin lograr la postcondición
	1-2	El usuario pulsa el botón de *Menu* pudiendo cambiar de capítulo o pasar a otro caso de uso.
REQUISITOS ESPECIALES	Ninguno.	
FRECUENCIA ESPERADA	Media.	

IMPORTANCIA	Media.
URGENCIA	PD
COMENTARIOS	No hay

Tabla 8. Caso de uso Visualizar Information

4.5.2 CALCULADORA MELD

La calculadora MELD (*Model for End-Stage Liver Disease*) se basa en el modelo del mismo nombre, y surge del índice anteriormente analizado (*Child Pugh Score*). Por ello, se trata de un sistema también de puntuación que permite medir la gravedad de distintas enfermedades hepáticas crónicas, aunque en un principio se desarrolló para predecir la muerte de pacientes de cirugía (Kamath and Wiesner, 2001).

La principal ventaja de este algoritmo es que permite medir la mortalidad de los pacientes en un plazo de tres con un porcentaje de fiabilidad del 80%. Para su cálculo, se hace uso de la siguiente ecuación:

$$MELD = 3.8 \times ln(Bilirubin) + 112 \times ln(INR) + ln(Creatine) + 6.43$$

Ecuación 1. Ecuación calculadora MELD

De este modelo, surgieron dos variantes más. Éstas son:

- MELDNa: se ha sugerido que la adición del *Sodium* al MELD mejora la capacidad de predicción de supervivencia del paciente.

$$MELDNa = MELD - Sodium - \ (0.025 \times MELD \times (140 - Sodium)) + 140$$

Ecuación 2. Ecuación calculadora MELDNa

- 5vMELD: este añade, aparte del *Sodium*, la *Albumin*.

$$5vMELD = MELDNa + \ \left(5.275 \times (4 - Albumin)\right) - \left(0.163 \times MELD \times (4 - Albumin)\right)$$

Ecuación 3. Ecuación calculadora 5vMELD

4.5.3 CALCULADORA OKUDA

La calculadora *Okuda*, se basa en un sistema de estatificación. En la última década, se ha utilizado ampliamente en pacientes que poseen un carcinoma hepatocelular. Los principales parámetros están relacionados con el estado del hígado y la etapa en la que se encuentra el tumor, es decir, si está extendido sobre un 50% de la superficie del hígado (Okuda, Ohtsuki and Obajta, 1985).

El cálculo de este algoritmo se basa en un sistema de puntuación dependiendo del valor de los parámetros requeridos.

	0	1
Albumin	>=30	<30
Bilirubin	<51	>=51
Tumour Extend	NO	YES
Ascites	NO	YES

Tabla 9. Valores calculadora OKUDA

En función de la puntuación obtenida, el algoritmo se clasifica en:

PUNTOS	CLASE	ETAPA
0	I	Inicial
1 - 2	II	Media
3 - 4	III	Terminal

Tabla 10. Interpretación calculadora OKUDA.

4.4.4 CALCULADORA CLIP

La calculadora *CLIP* se basa en un sistema de puntuación italiano de cuatro variables, que proporciona un sistema de clasificación de siete etapas. En comparación con otros algoritmos como pueden ser *Okuda* o TNM, obtiene el mayor poder discriminatorio, aunque su validación ha quedado en entre dicho, debido a que distintitos médicos han reportado tasas de supervivencia mucho mayores a las originales (Ncbi, 2005)

Dado que se trata de un sistema de puntuación, éste varía en función de los resultados obtenidos de las cuatro variables intervinientes (*AFP, Number of Tumours, Child Pugh Score, Tumour Extend* y *PVT -Main Portal Vein Thrombosis*).

	0	1	2
CPS	A	B	C
Tumour Extend	<=50%	<=50%	>=50%
#Tumours	1	>1	>1
PVI	NO	YES	

AFP	<400	>=400	

Tabla 11. Valores calculadora CLIP

Como se ha visto en calculadoras anteriores, en función del valor de los parámetros de entrada, éstos contribuyen con un valor u otro. La suma de estos valores es el resultado total de la calculadora.

PUNTOS	ETAPA
0	*Early Stage*
1 - 3	*Intermediate Stage*
4 - 6	*Advanced Stage*

Tabla 12. Interpretación calculadora CLIP

4.5.5 CALCULADORA GETCH

Se trata de una clasificación francesa para evaluar el pronóstico de un paciente con carcinoma hepatocelular (Chevret, Trinchet & Mathieu, 1999).

Como en el resto de los casos vistos anteriormente, se basa en un sistema de puntuación, en función de los parámetros de entrada:

- ECOG: si es igual a cero o uno, se suma un punto, en el resto de los casos se suma cero puntos
- *Bilirubin*: si presenta cantidades superiores a 50 umol/L, suma un punto, en caso contrario, cero puntos.
- $\frac{ALP}{ALP\ Limit}$: Si es mayor que dos, se suman dos puntos, en caso contrario, ninguno.
- AFP: si la cantidad presentadas son iguales o superiores a 35 ug/L, se suman dos puntos, en caso contrario, ninguno.

4.4.5 CALCULADORA TNM

Esta calculadora solo contiene variables relacionadas con la etapa del tumor. Ha sido probada mayoritariamente en el entorno quirúrgico, pero presenta una escasa probabilidad de predicción en pacientes con carcinoma hepatocelular con intención de someterse a un trasplante (Lei, Chau & Lui, 2006).

La clasificación de los pacientes se hace en función de los parámetros de entrada:

ETAPA	#TUMOURS	PVI	NODES	METASTASOS
I	1	NO	NO	NO
II	1 >1(<5CM)	YES NO	NO	NO
IIIA	>1	NO	NO	NO
IIIB	>=1	YES	NO	*NO*
IVA	>=1		YES	NO
IVB	>=1			YES

Tabla 13. Interpretación calculadora TNM

4.4.6 CALCULADORA CUPI

El algoritmo Cupi, se trata de un sistema de estatificación de análisis realizado por investigadores japoneses. Para su predicción, se basaron en seis variables distintas, dividiendo a los pacientes en tres etapas. Algunos científicos, consideran que este algoritmo prevé una mayor supervivencia del paciente que en el caso del algoritmo CLIP y *Okuda* (Leungh, Tang & Zee, 2002).

Al igual que las calculadoras anteriormente comentadas, también se basa en un sistema de puntuación, en función del valor de los parámetros de entrada.

	0	1	2	3	4	-4
Bilirubin	<34			34 - 52	>=52	
Ascites				NO		
ALP				>=200		
AFP			>=500			
TNM	IVA o IVB	IIIA o IIIB		I o II		
ECOG						0

Tabla 14. Valores calculadora Cupi

Como se trata de un sistema de puntuación, como hemos indicado, su clasificación se hará en función de la suma obtenida del valor de las variables de entrada. La clasificación quedaría de la siguiente manera:

PUNTOS	ETAPA
0-1	*Low*
2 - 7	*Intermediate*
>=8	*High*

Tabla 15. Interpretación calculadora Cupi

4.4.6 CALCULADORA BCLC

Este sistema de estatificación del carcinoma hepatocelular se desarrolló en el Hospital Clinic de Barcelona. A diferencia de los algoritmos anteriormente vistos, no se trata de un sistema de puntuación, sino que se conforma por una clasificación por estados en función de la etapa del tumor, el estado del hígado, el estado físico del paciente, así como los distintos síntomas relacionados con el cáncer. Une las cuatro etapas descritas anteriormente con un algoritmo de tratamiento.

Se ha sugerido que se trata de la clasificación más adecuada para determinar un tratamiento, y en particular, para indicar los pacientes que se encuentran en la fase más temprana de la enfermedad, que podrían beneficiarse de terapias curativas (Llovet, Brú & Bruix, 1999).

La clasificación de los pacientes se realiza de la siguiente manera:

	CPS	ECOG	#Tumours	SIZE	OTHERS
Very Early Stage	A	0	1	<2 cm	
Early Stage	A o B	0	1		
			3	<3 cm	
Intermediate Stage	A o B	0	<3		
Adavanced Stage	A o B	1 ó 2	>=1		*No PVI* *No Nodes* *No Metastasis*
End Stage	C	>=2	*>=1*		

Tabla 16. Interpretación Calculadora BCLC

4.4.7 CALCULADORA ALBERTA

El algoritmo Alberta HCC se conforma a partir del siguiente esquema, que ha de ser interpretado desde arriba hacia abajo, llegando al tratamiento más adecuado para el paciente, según los datos introducidos, mostrando un pronóstico de la esperanza de vida (Cuccheti et al., 2016).

Este diagrama de bloques se traduce en el siguiente esquema. A través de él, se puede observar de forma más clara los posibles escenarios.

Si observamos detalladamente el esquema, observamos en él, que, en la parte superior, aparece la etapa del algoritmo BCLC en la que se encuentra en el paciente.

Siguiendo el camino del algoritmo, podemos ver que éste se encuentra dividido en secciones. En una primera sección, se hace referencia al número de tumores y el tamaño de éstos.

La siguiente sección depende únicamente del resultado obtenido a través del algoritmo *Child Pugh Score*.

En la siguiente sección, se tratan las cuestiones clínicas. A medida que el paciente se encuentre en un estado peor de salud, se van evaluando diferentes datos. Éstos pueden ser, la evaluación de *Portal Hypertension* o *Bilirubin*.

En las dos últimas secciones, se indica el tratamiento más adecuado del paciente y un pronóstico de vida.

En el caso de que sea necesario aplicar un trasplante al paciente, éste se realiza en base al criterio seleccionado. Los criterios disponibles son:

- **Criterio de Milán**: para que un paciente, sea candidato de trasplante a través de este criterio, se deben cumplir las siguientes premisas (Aeeh, 2011):

    - El paciente posee un único tumor de tamaño menor o igual a 5 centímetros de diámetro.
    - El paciente puede tener hasta tres tumores, pero ninguno de ellos puede llegar a tener un tamaño de 3 centímetros de diámetro.

- **Criterio UCSF:** este criterio se basa en que el paciente tenga un tumor de hasta 6,5 centímetros, o incluso 2 ó 3 tumores con un tamaño de hasta 4.5 centímetros, siendo la suma de todos ellos, menor a 8 centímetros (Yusuf, Necdet, & Onur, 2015).

- **Criterio *UpToSeven*:** para que el paciente cumpla este criterio, deben darse las siguientes premisas (Mazzaferro, et al., 2009):

    - El paciente tiene un tumor con un tamaño menor a 6 centímetros.
    - El paciente tiene dos tumores, uno con tamaño menor a 6 centímetros de diámetro, y el otro, con tamaño menor a 5 centímetros.
    - El paciente tiene tres tumores, uno de ellos de tamaño menor a 6 centímetros de diámetro, otro menor a 5 centímetros y otro menor a 4 centímetros.

  - El paciente tiene cuatro tumores, uno menor a 6 centímetros, otro menor a 5 centímetros, otro menor a 4 centímetros y otro menor a 3 centímetros.
  - El paciente tiene cinco tumores, con tamaños menores a 6, 5, 6, 3 y 2 centímetros respectivamente.
  - El paciente tiene seis tumores, con tamaños menores a 6, 5, 4, 3, 2 y 1 centímetro respectivamente.

- **TTV + AFP:** este criterio se basa en el volumen total de los tumores y el AFP. Por ello, si el paciente tiene un volumen menor o igual 115, y que las cantidades presentes en el paciente de AFP sean menores a 400, se trata de un candidato apto para trasplante, en caso contrario, no lo será.

CAPITULO 5: MANUAL DE USUARIO

En el presente capitulo vamos a centrarnos en el funcionamiento de la aplicación implementada, mediante capturas tanto de las pantallas en el simulador de Xcode como de la aplicación en un dispositivo real. Con ello, pretendemos facilitar el uso de la aplicación para los distintos usuarios.

5.1 ICONO APLICACIÓN Y PANTALLA DE LOGIN

Al descargar e instalar la aplicación, aparecerá en el escritorio del dispositivo el icono de la aplicación.

Pulsando sobre el icono, se mostrará un *LaunchScreen* en la que aparece un mensaje de bienvenida para la aplicación, así como el icono de la aplicación como podemos observar en las siguientes ilustraciones:

A continuación, aparece una pantalla de *login* en la que podemos observar el icono de la aplicación, así como dos campos de entrada. Dichos campos de entrada se corresponden con usuario – contraseña (*Nickname – Password*). Actualmente, la aplicación no usa datos desde una base de datos, por ello, el usuario y contraseña, se trata de un definido directamente en la aplicación. Una vez introducidos, pulsando en el botón LOGIN, podemos acceder en la página principal de la aplicación.

Una vez se han introducido los campos usuario y contraseña, que como podemos observar en la ilustración anterior, la contraseña no se muestra en texto claro, y que estos datos de entrada sean correctos, aparece la pantalla principal.

La pantalla principal consta de distintos botones, a través de los cuales podemos acceder a las distintas funcionalidades de la aplicación, quedando definidas por el nombre de cada uno de los botones.

También podemos observar en la pantalla principal, un botón de *logout* que se encuentra en la parte superior izquierda. Si pulsamos sobre dicho botón, cerramos la sesión en la aplicación y volveremos a la pantalla de *login*. En la parte superior derecha tenemos un botón denominado *Screenshot*, el cual realiza una captura de pantalla, de la pantalla en la que nos encontremos, para poder compartirla.

5.2 PANTALLA CHAPTERS

Al pulsar sobre el botón *Chapters* de la pantalla principal, aparecerá una nueva pantalla que muestra los distintos módulos de capítulos de los que está compuesta la aplicación como botones.

En esta pantalla, podemos observar una cabecera que consta de un botón *back*, un botón *Home*, un botón *Menu*, el título de la pantalla en la que nos encontramos, y nuevamente un botón *Screenshot.*

Como podemos observar en la ilustración anterior, la pantalla está compuesta de tres botones que nos permiten acceder a los diferentes módulos. Estos botones se presentan con el icono predeterminado para acceder a los capítulos, el número y el título de cada modulo.

Si pulsamos sobre uno de los módulos, accedemos a los capítulos correspondientes del módulo seleccionado. En las siguientes ilustraciones, podemos observar los distintos módulos.

Las ilustraciones anteriores, muestran distintos botones que se corresponden con los distintos capítulos que pertenecen a cada módulo. También se puede observar, que, en algunos casos, puede aparecer un capítulo que aún no está disponible y aparece con el mensaje "*Not available*" y el botón no se encuentra accesible para el usuario.

Los distintos botones de capitulo, están formados por el icono predeterminado para los capítulos, el número del capítulo y el título del capítulo. Si pulsamos sobre un capítulo, aparece una pantalla que muestra la información relacionada con dicho capítulo. Hasta que se muestre el capítulo, aparece un icono de carga de este.

En la cabecera, podemos observar los botones descritos anteriormente. Así mismo podemos observar que aparece un nuevo botón *Download*, que nos permite descargar el *pdf* que estamos visualizando. Cuando se descarga el documento aparece el siguiente mensaje:

En el caso de que el documento ya se encuentre almacenado en el dispositivo, aparece un mensaje indicativo.

5.3 PODCAST

Si pulsamos el botón *Podcast* de la pantalla principal, se muestra la cabecera descrita en el apartado anterior y tres botones relacionados con los *Podcast* de cada módulo.

Al pulsar sobre uno de los botones de un módulo *podcasts*, aparece una nueva pantalla con distintos botones que se corresponden al podcast de cada capítulo. Cada uno de los botones consta del icono predeterminado para los *podcasts*, el número de *podcast* y el título. A continuación, se muestran las pantallas relacionadas con cada módulo.

Una vez seleccionado el *Podcast* deseado, se abre una nueva pantalla en la que podemos observar el *Podcast* seleccionado, que se trata de un vídeo. Además, aparece un botón

que nos redirige a la pantalla correspondiente al módulo de capítulo que nos encontramos. También aparecen unos botones, que nos permiten pasar de un *podcast* a otro, sin necesidad de ir a la pantalla principal del podcast de cada módulo.

5.4 PANTALLA CARDS

A través de la pantalla principal, podemos acceder a la pantalla *Cards* pulsando el botón *Cards*. En esta pantalla, se muestra el sitio web que pertenece a la Universidad de *Calgary*, que solo es accesible para usuarios registrados. En la siguiente Ilustración, se observa como aparece en la aplicación.

5.5 PANTALLA FIGURES

Desde la pantalla principal, podemos acceder a la pantalla *Figures*, a través del botón correspondiente. En dicha pantalla, podemos observar distintos botones relacionados con tablas de contenidos, esquemas, figuras interactivas y figuras en las que podemos dibujar sobre ellas.
Si pulsamos sobre el botón *Table of Contents*, podemos observar cuatro nuevos botones, que previsualizan la tabla de contenidos y un título de la tabla de contenidos a la que pertenecen.

Si pulsamos uno de los botones de la pantalla *Table of Contents*, podemos visualizar la tabla de contenidos completa y unos botones que nos permiten navegar entre las distintas tablas de contenidos sin tener que ir a la pantalla *Table of Contents.*

Si nos posicionamos de nuevo en la pantalla *Figures* y accedemos al botón *Schemes*, se muestra una nueva pantalla que consta de ocho botones. Cada botón previsualiza una imagen del esquema correspondiente y un título del contenido.

Si pulsamos sobre uno de los botones la pantalla *Schemes*, podemos visualizar el esquema correspondiente a pantalla completa y unos botones que nos permiten navegar entre los distintos esquemas sin tener que volver a la pantalla *Schemes.*
De nuevo, si nos posicionamos sobre la pantalla *Figures*, y pulsamos el botón *Interactive Figures*. En ella podemos observar distintos botones que previsualizan la figura interactiva disponible con su descripción.

Si seleccionamos uno de los botones, podemos observar la figura interactiva a pantalla completa. Así mismo, se puede observar una *slider*, que nos permite superponer imágenes que están relacionadas entre sí. Es decir, tenemos una imagen con texto, que si movemos el *slider*, visualizamos la misma imagen, pero con texto descriptivo de las partes más relevantes.

Por último, tras pulsar el botón *Drawing* de la pantalla *Figures*, aparece una nueva pantalla. Dicha pantalla consta de ocho botones en los que se previsualiza una imagen y una pequeña descripción de cada imagen.

Al seleccionar uno de los botones de la pantalla *Drawins*, aparece una figura sobre la que se puede dibujar. Si observamos la siguiente Ilustración, vemos que aparece el texto "*Select a color to draw*" y una serie de colores. Seleccionando un color, podemos dibujar sobre la imagen. A continuación, aparecen tres botones. El botón *Clear*, borra por completo lo dibujado, el botón *Undo* borra lo último dibujado y, por último, el botón *Save* guarda la imagen con el dibujo realizado sobre ella.

5.6 PANTALLA CALCULATORS

Desde la pantalla principal, pulsando sobre el botón *Calculators*, accedemos a la siguiente pantalla:

Como podemos observar, la pantalla *Calculators* consta de seis botones, donde cada botón se corresponde con una calculadora distinta.

- ***Child Pugh Score***:

Pulsando sobre el botón *Child Pugh Score*, aparece la siguiente pantalla. Dicha pantalla consta de las siguientes variables: *Bilirubin*, *Albumin*, INR, *Encephalopathy*, *Ascites* e *International Units*. Estas variables son necesarias para calcular el valor de la calculadora.

Además, consta de un botón *Reset Values*, el cual resetea los datos introducidos a cero, y un botón de *More Information*, que proporciona información de la calculadora.

Por último, el botón principal es *Calculate CP Score*, el cual calcula el resultado de la calculadora en función de los valores de entrada. Si se da a calcular *CP Score* sin algún dato de entrada, sale un error indicando el parámetro que no contiene datos:

- **MELD:**

Si seleccionamos la calculadora MELD, se muestra la siguiente pantalla:

Para realizar el cálculo, es necesario introducir en los datos requeridos a través de la pantalla. Estos son: INR, *Bilirubin*, *Creatinine*, *Sodium*, *Albumin* y *Dialysis*.

En esta pantalla se puede observar una pequeña descripción de los limites de los datos de entrada en función de si estamos en unidades internacionales o no. Además, consta de tres botones, uno resetea los valores de entrada a cero, otra muestra más información sobre la calculadora, y por último, el botón *Calculate MELD*, que calcula los valores de la calculadora MELD, MELDNa y 5vMELD.

- **Okuda:**

Si seleccionamos la calculadora *Okuda*, aparece la siguiente pantalla. En ella, podemos observar que posee los siguientes argumentos de entrada: *Bilirubin*, *Albumin*, *Ascites*, *Tumour Extend* e *International Units*, necesarios para el cálculo de Okuda.

Como podemos ver, consta de dos botones, uno los valores de entrada, y el botón *Calculate Okuda* realiza los cálculos necesarios.

- **Clip:**

Si seleccionamos la calculadora Clip, aparecerá la siguiente pantalla. En ella, podemos observar que posee los siguientes argumentos de entrada: AFP, *Number of Tumours*, CPS, *Tumour Extend*, PVT (*Main Portal Vein Thrombosis*) e *International Units*, necesarios para el calculo de Clip.

Como podemos ver, consta de dos botones, uno los valores de entrada, y el botón *Calculate CLIP* realiza los cálculos necesarios.

- **BCLC:**

Si seleccionamos la calculadora BCLC desde la pantalla en la que se encuentran todas las calculadoras, accedemos a la siguiente pantalla. En ella podemos observar los siguientes argumentos de entrada: *Numer Of Tumours*, *Tumour Size (DIAM. Cm)*, *Child Pugh Score*, *ECOG*, *PVI (Portal Vein Invasion)*, *Nodes* y *Metastasis*.

Como podemos ver, consta de dos botones, uno los valores de entrada, y el botón *Calculate BCLC* realiza los cálculos necesarios.

- ***All Algorithms*:**

Desde la pantalla en la que se encuentran todas las calculadoras, podemos acceder a la más importante de ellas. Ésta es la calculadora *All Algorithms*. En ella se engloban todas las calculadoras vistas con anterioridad y alguna a mayores.

Esta calculadora consta de tres pantallas:

- ***Diagnostic Imaging:*** tiene como argumentos de entrada *Numer Of Tumours*, *Tumour Size (DIAM. Cm)*, *Tumour Extend*, *PVI (Portal Vein Invasion)*, *Nodes*, *Metastasis*, *Portal Hupertension (varices and/or splenomeglacy)* y *PVT (Main Portal Vein Thrombosis)*. Como se puede observar en la Ilustración siguiente, posee un botón con la etiqueta NEXT. Dicho botón nos lleva la siguiente pantalla de la calculadora.

  Como se puede observar, también hay un botón denominado ENTER. Si pulsamos sobre él, en función del número de tumores introducidos, aparecen tantas cajas como tumores haya. En las cajas se introduce el diámetro de un tumor. Como mucho, se pueden meter hasta 6 tumores.

- ***Laboratory Values:*** consta de los siguientes argumentos de entrada: *INR (International Normalized Ratio)*, *Bilirubin*, *Creatine*, *Sodium*, *Albumin*, *Platelets*, AFP, *AFP*, *AST*, *AST (Upper Limit of Normal)*, *ALT* y *ALT (Upper Limit of Normal)*. Como indicamos en el punto anterior, con el botón NEXT pasamos a la siguiente pantalla de la calculadora.

- ***Clinical Questions:*** consta de los siguientes argumentos de entrada: *Encephalopathy*, *Ascites*, *Cirrhosis*, *Varices* y *ECOG*. Como hemos indicado anteriormente, con el botón NEXT vamos a la última de las pantallas de la calculadora.

- ***Results:*** ésta es la última pantalla de la calculadora. En ella podemos observar los resultados de distintas calculadoras en función de los datos introducidos en las pantallas anteriores. En el caso de que no se hayan introducido los datos necesarios para el cálculo de un calculadora, en el resultado se mostrará un "-".

Podemos observar que, en la cabecera de la ilustración anterior, aparece el icono de un engranaje. Dicho icono representa los ajustes de la calculadora.

Por defecto, el criterio aplicado es *UpToSeven*. En el caso de que se elija otro criterio, se recalculan todos los valores de la calculadora.

Como podemos observar en las ilustraciones anteriores, se disponen de dos botones: *More Information* y *Alberta HCC Algorith*. El primer botón muestra un resumen de todos los datos introducidos relacionado con el tipo de tratamiento recomendado. El segundo de los botones muestra un diagrama de flujo del tratamiento. En el caso de que no se dispongan de datos, y, por ende, no haya un tratamiento disponible, si se pulsa sobre uno de los botones, salta el siguiente mensaje:

También podemos observar el diagrama de flujo de Alberta, que, en función de los resultados obtenidos, tenemos un diagramo u otro. Sobre el caso que hemos analizado en las capturas anteriores, dicho diagrama se correspondería con el mostrado en la siguiente ilustración.

En el caso de que pulsemos el botón *Summary*, se visualiza un resumen de todos los datos introducidos en la calculadora.

5.7 PANTALLA RESOURCES

A través de la pantalla principal, podemos acceder a la pantalla *Resources*, con el botón correspondiente.
Dicha pantalla consta de ocho botones. Cada uno de los botones, muestra una página web distinta, como podemos observar en las siguientes ilustraciones.

- *CAST Guidelines*
- *AASLD Guidelines*
- *EASL Guidelines*
- *ACG Guidelines*
- *AGA Guidelines*
- *ILCA Guidelines*
- *Lindsay Atlas*
- *References*

5.8 PUBMED

Desde la pantalla principal, podemos acceder a la pantalla *PubMed*, a través del correspondiente botón. En esta pantalla visualizamos la siguiente página web:

5.9 PANTALLA INFORMATION

A través de la pantalla principal, si pulsamos el botón *Information*, se abre el siguiente pdf, que muestra información sobre la aplicación, para que en caso de que se necesiten realizar cambios, bastaría con modificar dicho pdf.

5.10 FUNCIONAMIENTO DEL MENÚ

Si observamos todas las ilustraciones hasta el momento, podemos ver que, en la cabecera de éstas, en la parte superior izquierda aparece un icono con tres rayas horizontales. Este icono es el Menú. Si pulsamos sobre él, podemos visualizar de un vistazo los principales contenidos de la aplicación y navegar entre ellos.

6. CONCLUSIONES

Comencé a desarrollar está aplicación por la curiosidad de aprender un nuevo lenguaje que no es muy común a lo largo del grado y así poder realizar una aplicación para un dispositivo móvil. Me ha permitido comenzar a tener pequeños conocimientos de esta tecnología y de como enfocar un proyecto de movilidad, superando varios retos que me he encontrado a lo largo del desarrollo de ese Trabajo de Fin de Grado, como, por ejemplo, poder guardar los capítulos en un dispositivo o guardar la información de las distintas pantallas de la calculadora *All Algorithms*.

Para conseguir los objetivos descritos en el primer capítulo, he ido siguiendo los pasos que se indican a continuación.

En primer lugar, se ha realizado un estudio de los distintos Sistemas Operativos disponibles en el mercado, así como las diferentes opciones de desarrollo de aplicaciones móviles. Tras un análisis de las ventajas y desventajas de cada una de ellas, he optado por desarrollar una aplicación nativa para el Sistema Operativo iOS, debido a que iOS es el Sistema Operativo predominante en el campo y la localización geográfica para los que se destina esta aplicación, así como el desarrollo en un lenguaje nativo, nos permite mejorar los tiempos de latencia y poder utilizar todas las herramientas del dispositivo.

Tras tener clara la tecnología a utilizar, se procedió a elaborar un análisis funcional de la aplicación, indicando los requisitos necesarios para un buen funcionamiento de ésta.

A continuación, se realiza un manual de usuario, comprobando que la aplicación cumple los objetivos descritos anteriormente. Además, con el desarrollo de este apartado, se ha podido comprobar que la aplicación funciona correctamente.

Bajo mi punto de vista, los principales *skills* que he podido obtener con la realización de esta aplicación, son: aprender a ser más autodidacta, constante y a buscar información. Este Trabajo de Fin de Grado me ha permitido, sobre todo, comprender en mayor profundidad el concepto de *Flipped Classroom*. También cabe destacar, que, a lo largo del desarrollo de ésta, he encontrado dificultades a la hora de encontrar información del lenguaje *Swift*, ya que éste no es muy común. El peor problema ha sido intentar optimizar la aplicación para no obtener excepciones por fallo de rendimiento de ésta.

Cabe destacar, que gracias al entorno de desarrollo integrado Xcode que proporciona Apple para desarrolladores, es fácil la elección del Sistema Operativo, ya que por defecto se sitúa en la última versión disponible. Así mismo, el simulador de Xcode es muy real y adaptativo a las pantallas, por lo que el proyecto es fácilmente escalable.

Por último, gracias a los conocimientos adquiridos a lo largo de todos años de universidad, he conseguido aplicarlos en el desarrollo de este Trabajo de Fin de Grado.

6.1 LINEAS FUTURAS

Tras el desarrollo de la aplicación han ido surgiendo ideas de posibles mejoras futuras de la misma. Pudiendo introducir nuevas funcionalidades a la aplicación e intentar que sea más dinámica. Estas líneas futuras son:

Vincular la aplicación a una Base de Datos, para que los cambios que se realicen en los contenidos de la aplicación, no sea necesario realizar un nuevo despliegue de ésta. Así mismo, que la aplicación sólo la puedan ejecutar usuarios dados de alta en el sistema.

Incluir alguna calculadora individual más, para que no sea necesario utilizar la calculadora *All Algorithms*, disminuyendo así el tiempo de ejecución, ahorrando memoria y mejorando la experiencia de usuario.

Bajo mi punto de vista, estos cambios supondrían una mejora en la aplicación. Así como que los alumnos puedan mejorar su capacidad de aprendizaje en un contexto de *Flipped Classroom*.

ANEXO: CÓMO EMPEZAR A TRABAJAR CON IOS

Para poder empezar a trabajar en el desarrollo aplicaciones iOS es necesario disponer de un Mac, debido a que las herramientas de desarrollo solo están disponibles para Mac OS.

Posteriormente, debemos de descargarnos el entorno de desarrollo de Apple para sus dispositivos, denominado Xcode, que lo podemos encontrar en la tienda de descargas

(*App Store*) de nuestro Mac. Dicho entorno de desarrollo nos proporciona el iOS SDK en el que disponemos de todas las herramientas, compiladores y *frameworks* necesarios.

Una vez tenemos estos dos prerrequisitos, podemos comenzar a desarrollar nuestra aplicación. Para ello, cuando abramos Xcode, aparece la siguiente pantalla. En ella podemos elegir entre crear *playgrounds* (que se utilizan para crear aplicaciones a nivel principiante), crear un nuevo proyecto o elegir un proyecto ya existente. En el caso de que ya tengamos algún proyecto, lo podremos elegir de la parte lateral derecha.

Ilustración 13. Pantalla principal Xcode

En este caso, elegiremos crear un nuevo proyecto. Aparece un nuevo cuadro de diálogo en el que elegimos para que tipo de dispositivo Apple queremos crear la aplicación (iOS, watchOS, tvOS, macOS o Cross-platform). En nuestro caso elegimos una aplicación iOS del tipo *Single View Application*.

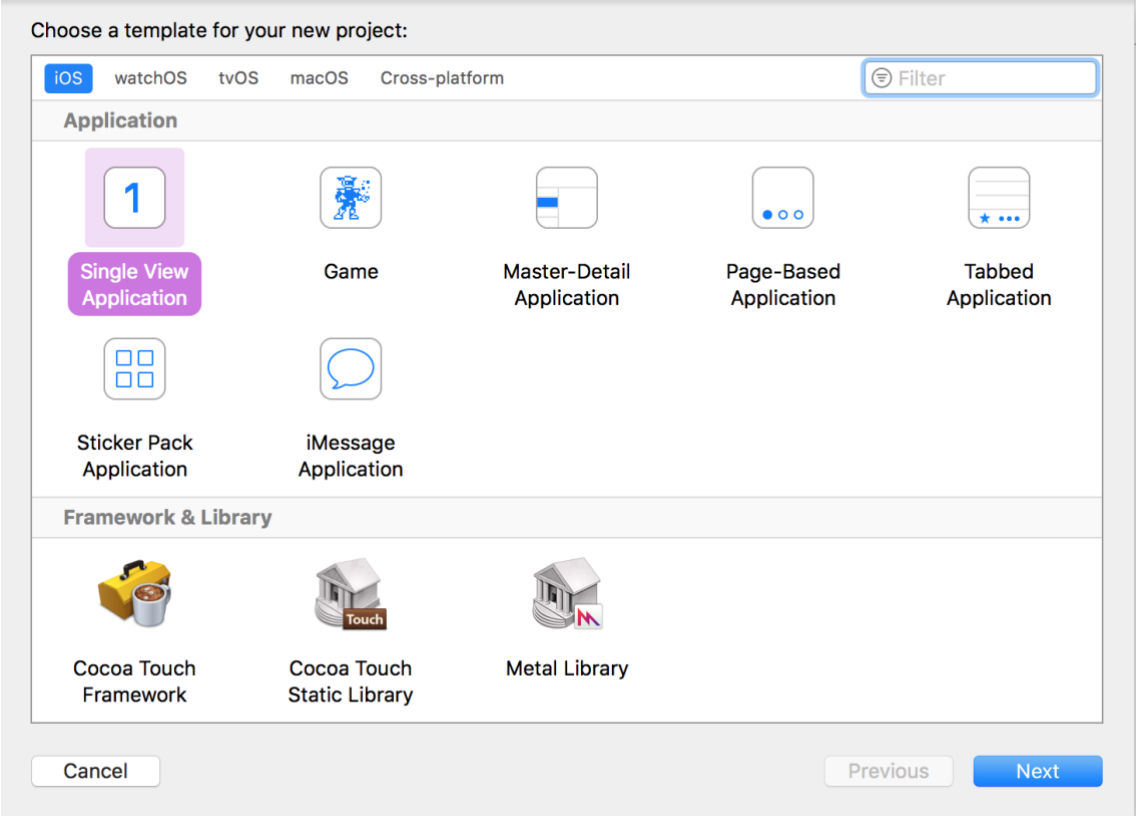

Ilustración 14. Selección tipo aplicación Xcode

Tras haber creado la aplicación, nos encontramos en la pantalla principal de Xcode. En ella podemos ver la clase *AppDelegate.swift*, en la que se definen las características principales de la aplicación. También tenemos la clase *ViewController.swift*, que está relacionada con el *playground* principal.

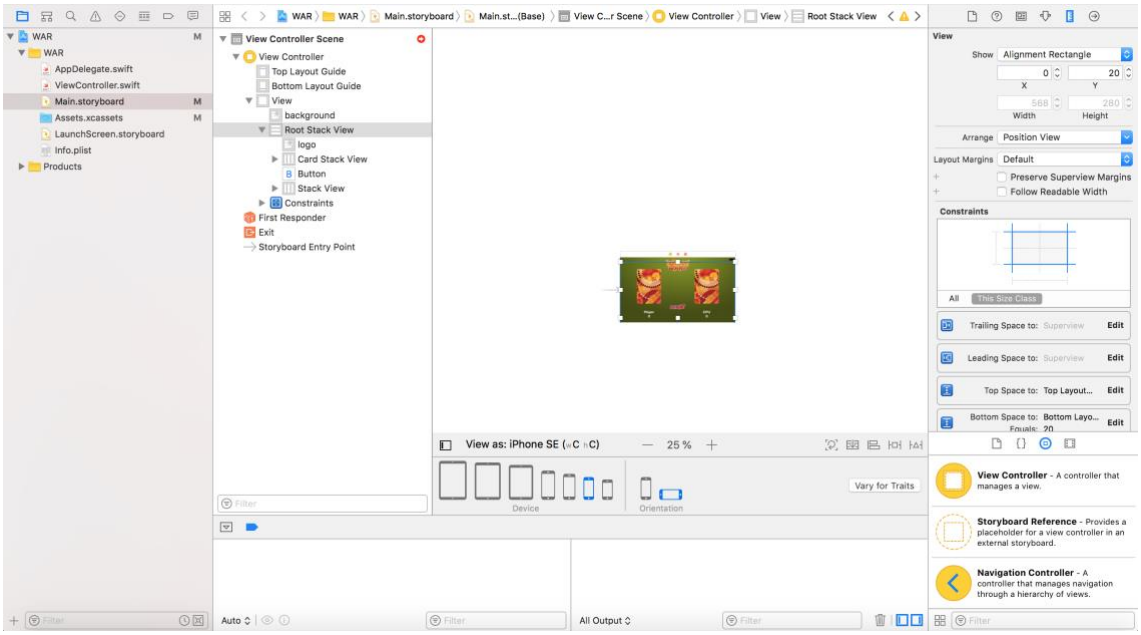

Ilustración 15. Pantalla proyecto Xcode

Como estamos realizando una pequeña introducción, nos centraremos en el desarrollo de la aplicación a través de un *playground*. Seleccionamos *Main.playground* y aparece una vista de un dispositivo iOS. Podemos elegir si se trata de un iPad o un iPhone y el tipo de vista (vertical u horizontal). Una vez elegido esto, podemos empezar a introducir botones, imágenes, etiquetas, objetos, etc.

Ilustración 16. Herramientas disponibles playground Xcode

Basta con arrastrar el tipo de herramienta al *playground* y posicionarlo en una parte de la pantalla.

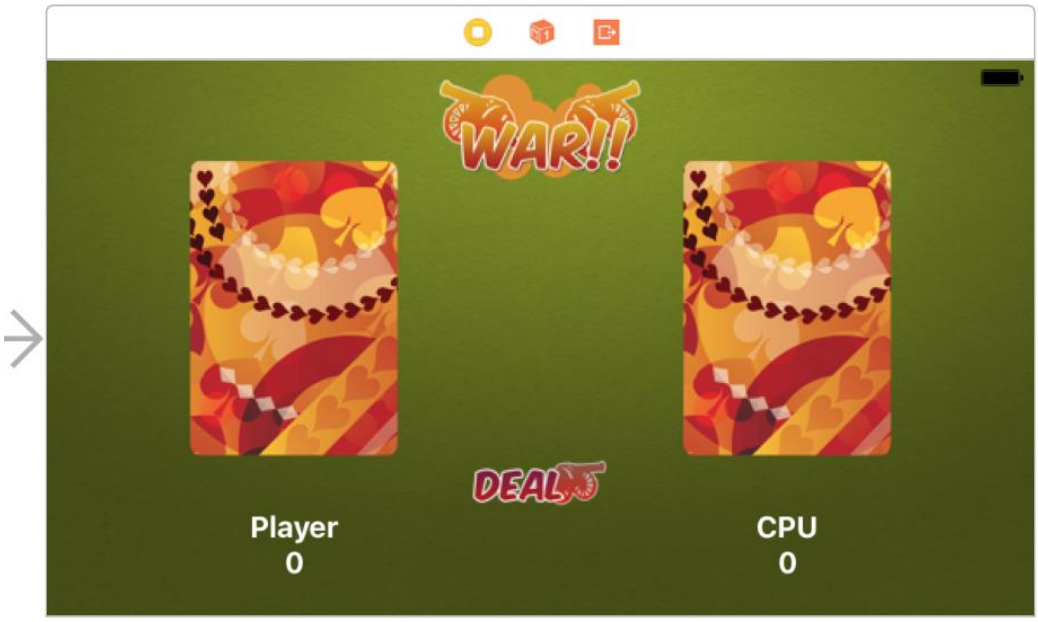

Ilustración 17. Ejemplo playground

Con ello tendríamos creada nuestra pequeña aplicación. Ésta se puede probar en el simulador que nos ofrece Xcode o bien exportarla a un dispositivo que tengamos conectado a nuestro Mac.

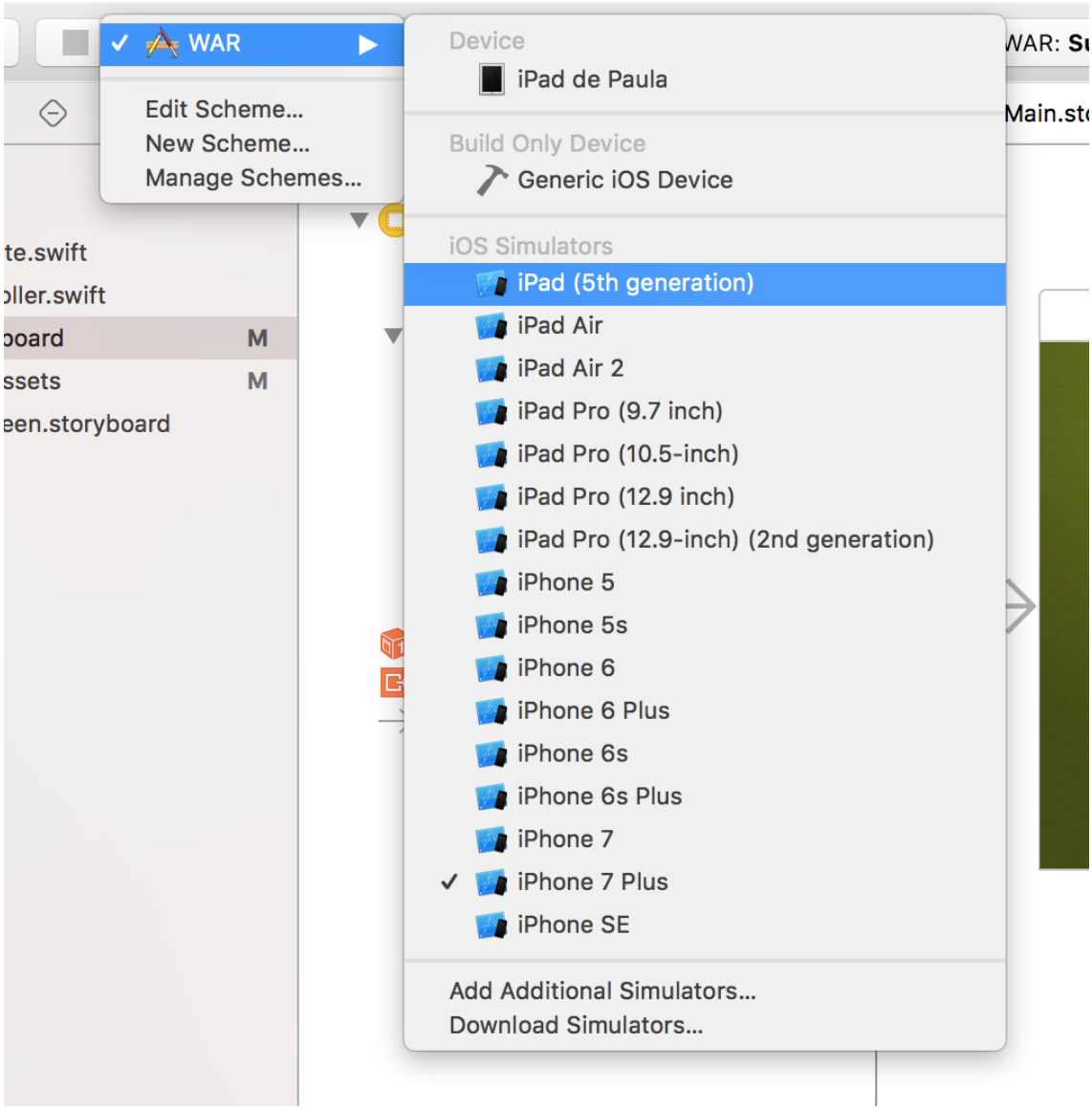

Ilustración 18. Simuladores Xcode

BIBLIOGRAFÍA

Aeeh. (Junio de 2017). Trasplante hepático y tumores. Obtenido el 7 de Julio de 2018 de aeeh.es/wp-content/uploads/2011/12/v10n3a675pdf001.pdf

Android Developer(2018). Arquitectura de la plataforma. Obtenido el 11 de Julio de 2018 de https://developer.android.com/guide/platform/

Aplicaciones móviles (2018). iOS 10.3: principales características de la actualización. Obtenido el 1 de Julio de 2018 de https://www.aplicacionesparamoviles.com/lista-novedades-actualizacion-ios-103-apple-iphone-2017/.

Apple (2018a). Xcode - Apple Developer. Obtenido el 2 de Julio de 2018 de https://developer.apple.com/xcode/

Apple (2018b). UIKit | Apple Developer Documentation. Obtenido el 1 de Julio de 2018 de https://developer.apple.com/documentation/UIKit

Apple (2018c). Swift. Obtenido el 4 de Julio de 2018 de https://swift.org

Apple (2014). The Swift Programming Language (Swift 4.1 Edition). Apple Inc.

Apple (n.d.). Using Swift with Cocoa and Objective-C (Swift 4 Edition). Apple Inc.

Arias, A. (2016). Curso de Programación de Apps. Android y iPhone. 2ª Edición. Obtenido el 3 de Julio de 2018 de https://books.google.es/books?id=uA_8CwAAQBAJ&printsec=frontcover&dq=Curso+de+Programación+de+Apps.+Android+y+iPhone.+2ª+Edición.&hl=es&sa=X&ved=0ahUKEwjty_SHxZncAhUJVhQKHSDdDIgQ6AEIKDAA#v=onepage&q=Curso%20de%20Programación%20de%20Apps.%20Android%20y%20iPhone.%202ª%20Edición.&f=false

Bates, S., Galloway, R. (2012). The inverted classroom in a large enrolment introductory

physics course: a case study. STEM Conference, London

Bollapragada, V., Murphy, C. and White, R. (2000). Inside Cisco IOS software architecture. Indianapolis: Cisco Press

Chevret , S., Trinchet, J., & Mathieu, D. (1999). A new prognostic classification for predicting survival in patients with hepatocellular carcinoma: Groupe d'Etude et de Traitement du Carcinome Hépatocellulaire.

Child, C. G., & Turcotte, G. (1964). Surgery and portal hypertension. In: The liver and portal hypertension. (C. Child, Ed.) 50-64. Obtenido de https://www.ncbi.nlm.nih.gov/pubmed/4541913

Cuello, J. and Vittone, J. (2014). Diseñando apps para móviles.

Daniel, S. F. (2011). Xcode 4 iOS Development Beginner's Guide. Birmingham: Packt Pub.

Fernández Pérez, G. (2018). iOS, Todo lo que siempre has querido saber sobre tu iPhone

García Domínguez, R. (2018). Todas las novedades de iOS 12, el nuevo sistema operativo para iPhone. AS.com. Obtenido el 2 de Julio de 2018 de https://as.com/betech/2018/06/04/portada/1528132311_239830.html.

Jangla, K. (2015). Windows 10 Revealed: The Universal Windows Operating System for PC, Tablets, and Windows Phone

y iPad.

Luna, F., Millahual, C. P., Lacono, M. (2013). Potenciar la faceta full stack. Programador Web Full Stack, 21. Obtenido el 22 de Junio de 2018, de https://books.google.es/books?id=9pdGDwAAQBAJ&dq=Potenciar+la+faceta+full+stack&hl=es&source=gbs_navlinks_s

Hereter, L. and Zanini, V. (2016). jQuery Mobile: Diseño y desarrollo de aplicaciones para smartphones y tablets.

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

Lei, H., Chau, G., & Lui, W. (2006). Prognostic value and clinical relevance of the 6th Edition 2002 American Joint Committee on Cancer staging system in patients with resectable hepatocellular carcinoma.

Leung, T., Tang, A., & Zee, B. (2002). Construction of the Chinese University Prognostic Index for hepatocellular carcinoma and comparison with the TNM staging system, the Okuda staging system, and the Cancer of the Liver Italian Program staging system: a study based on 926 patients. Obtenido de https://www.ncbi.nlm.nih.gov/pubmed/11920539

Llovet, J., Brú, C., & Bruix, J. (1999). Prognosis of hepatocellular carcinoma: the BCLC staging classification.

Marcombo, S.A. (1998). Telecomunicaciones móviles (2nd ed.). Barcelona: Marcombo. Obtenido el 20 de Junio de 2018, de https://books.google.es/books?id=ztTpTayFeSUC&dq=tecnologias+moviles+marcombo&lr=&hl=es&source=gbs_navlinks_s

Martín, A. R. (2014). Aplicaciones Web. Madrid: Paraninfo.

Maskrey, M. (2016). *App Development Recipes for iOS and watchOS. Mazzaferro, V., Llovet , J., Miceli, R., Bhoori, S., Schiavo, M., & Mariani, L. (2009). Predicting survival after liver transplantation in patients with hepatocellular carcinoma beyond the Milan criteria: a retrospective, exploratory analysis.* Obtenido el 11 de Julio de 2018 de https://books.google.es/books?id=idl6DAAAQBAJ&pg=PA555&dq=App+Development+Recipes+for+iOS+and+watchOS&hl=es&sa=X&ved=0ahUKEwi5rJ3jupncAhXCTcAKHS7mDHAQ6AEIJzAA#v=onepage&q

=App%20Development%20Recipes%20for%20iOS%20and%20watchOS&f=false

Miguel, J. T. (2015). Implantación de aplicaciones web en entornos internet, intranet y extranet. Madrid: Paraninfo.

Microsoft (2017). Elección de un lenguaje de programación - UWP app developer. obtenido el 3 de Julio de 2018 de https://docs.microsoft.com/es-es/windows/uwp/porting/getting-started-choosing-a-programming-language.

Microsoft (2018). ¿Qué es una aplicación para Plataforma universal de Windows (UWP)?. Obtenido el 11 de Julio de 2018 de https://docs.microsoft.com/es-es/windows/uwp/get-started/universal-application-platform-guide

Ohland, B. and Varma, J. (2016). Xcode 7 Essentials - Second Edition. Birmingham: Packt Publishing, Limited.

Okuda, K., Ohtsuki, T., & Obata , H. (1985). Natural history of hepatocellular carcinoma and prognosis in relation to treatment. Study of 850 patiens.

Prensario Internacional. (2018). España: el consumo de televisión en Internet se duplicó en los últimos | Prensario Internacional. Obtenido el 28 de Junio de 2018 de https://www.prensario.tv/novedades/1360-espana-el-consumo-de-television-en-internet-se-duplico-en-los-ultimos-tres-anos.

Cucchetti,A., Cescon, M., Golfieri, R., Piscaglia, F., Renzulli, M., Neri, F., Cappelli, A., Mazzotti, F., Mosconi, C., Colecchia, A., Ercolani, G., Pinna, A. D. (2016). *Hepatic venous pressure gradient in the preoperative assessment of patients with resectable hepatocellular carcinoma.* Obtenido el 28 de Julio de 2018 de https://www.sciencedirect.com/science/article/pii/S0168827815005978

Sol Llaven, D. (2015). Sistemas operativos. Distrito Federal: Larousse - Grupo Editorial Patria. Obtenido el 1 de Julio de 2018 de https://books.google.es/books?id=qdFUCwAAQBAJ&dq=Sistemas+operativos&hl=es&source=gbs_navlinks_s

Statcounter. (Julio de 2018). Obtenido el 11 de Julio de 2018 de http://gs.statcounter.com/os-market-share/mobile/canada/#monthly-201701-201807

Statista. (Mayo de 2017). Obtenido el 4 de Julio de 2018 de https://es.statista.com/

Torné, K. (2018). Llegó iOS 10 y estas son sus novedades. Obteneido el 26 de Junio de 2018 de https://applesencia.com/2016/06/llego-ios-10

Torre, F. J. (213). Desarrollo de aplicaciones para Windows Phone. Obtenido el 2 de Julio de 2018 de https://earchivo.uc3m.es/bitstream/handle/10016/17873/Memoria%20PFC,%20F.J.Castellanos.pdf?sequence=1

Yusuf , G., Necdet, G., & Onur, Y. (2015). Living Donor Liver Transplantation Outcomes for Hepatocellular Carcinoma Beyond Milan or UCSF Criteria.