CAPÍTULO 1

Hardware y Software

1.1. Introducción

Introducción

1.2. Software de procesamiento de imágenes

Software de procesamiento de imágenes

1.3. Dispositivos móviles

Al trabajar con Apple se cuenta con la ventaja de contar con pocas variantes en cuanto al Hardware utilizado. Básicamente existen tres tipos de dispositivos en los que se pueden desarrollar: iPhone, iPad y iPod Touch. Para cada variante de plataforma existen distintos modelos que hacen que algunas características importantes como capacidad de procesamiento, resolución de cámara o tama?o de la pantalla entre otras puedan verse afectadas. A continuación se relata el surgimiento de cada uno de lo dispositivos al mercado y se describen brevemente las principales características.

1.3.1. iPhone

Sin dudas el iPhone fue uno de los saltos más grandes en el mundo tecnológico en los últimos a?os. Logró llenar el hueco que los PDAs de la década de los 90 no habían sabido completar y comenzó a desplazar al invento que revolucionó el mercado de los contenidos de música, el iPod. Gracias a su pantalla táctil capacitiva de alta sensibilidad logró reunir todas las funcionalidades agregando solamente un gran botón y algunos extra para controlar volumen o desbloquar el dispositivo.
La primera generación del iPhone fue lanzada por Apple en Junio de 2007 en Estados Unidos, luego de una gran inversión de la operadora ATT que exigía exclusividad de venta dentro de dicho país durante los siguientes cuatro a?os. La misma soportaba tecnología GSM cuatribanda y se lanzó en dos variantes de 4GB y 8GB de ROM. El segundo modelo lanzó como novedad el soporte de tecnología 3G cuatribanda y GPS asistido. Luego le siguieron el iPhone 3GS, 4, 4S y el 5, siendo este último, la sexta y última generación disponible al momento de la redacción de este trabajo.
Las dimensiones del iPhone 5 son de 58.6 x 123.8 x 7.6 millimetres, resolución de pantalla de 640 x 1136, tiene una velocidad de reloj en la CPU de 1200MHz, RAM de 1GB y la ROM varía según la variante (16GB, 30GB o 64GB).

1.3.2. iPad

Esta línea de dispositivos es la más potente en lo que respecta a capacidad de procesamiento.

1.3.3. iPod Touch

1.4. Software de Apple Inc

1.4.1. Sistemas Operativos

Para poder desarrollar aplicaciones sobre dispositivos móviles de la firma Apple Inc es necesario contar con computadoras que corran el sistema operativo Mac OS X. Esto puede ser llevado a cabo, ya sea adquiriendo plataformas de desarrollo de la mencionada firma o creando máquinas virtuales que corran dicho sistema operativo. Para la segunda opción (la más económica pero con ciertas dificultades de performance), es necesario que la computadora cuente con virtualización de hardware. Se comenzó trabajando de esta manera hasta el momento de adquirir plataformas de desarrollo que contaran con Mac OS X en forma nativa.
Mac OS X refiere a la versión número 10 (en números romanos) de una serie de sistemas operativos que comenzaron a desarrollarse en la década de los 80 (Mac OS 1 data del a?o 1984). En los últimos 28 a?os se han ido sucediendo nuevas versiones que han ido mejorando características en la estructura de datos con la incorporación de la jerarquía de archivos en Mac OS 3 por ejemplo, en la búsqueda de archivos, con la simultaneidad de tareas, multiplicidad de usuarios o incluso con el énfasis en la interfaz de usuario por mencionar algunas características importantes en la evolución de esta familia de sistemas operativos. Dentro de Mac OS X existen distintas versiones, siendo la más actual la Versión 10.8: Mountain Lion lanzada durante 2012.
Por su parte todas las plataformas móviles de Apple Inc corren otro dispositivo de código cerrado: iOS. Originalmente llamado así por ser el sistema operativo utilizado por la plataforma iPhone, este sistema operativo está también en las plataformas iPad, iPod Touch y Apple TV en todas sus versiones. La versión más reciente de este SO es el iOS 6.1.
Una de las grandes innovaciones de estas plataformas es el hecho de poder desarrollar aplicaciones y correrlas en el propio dispositivo (por supuesto también sucede lo mismo en el mundo Android). Para poder lograr esto, es necesario como se ha dicho, contar con una máquina que corra Mac OS X y contar con el SDK apropiado llamado **xCode**. Este entorno de desarrollo y su lenguaje se explican en la sección 1.4.2.

1.4.2. Objective-C

El lenguaje que fue elegido por Apple Inc para desarrollar sobre plataformas móviles es Objective-C. Este lenguaje fue desarrollado en la década de 1980 como un superconjunto de C orientado a objetos. Es decir que es una extensión del standard ANSI C que incorpora un modelo orientado a objetos basado en **Smalltalk**. Una de las diferencias sustanciales del modelo orientado a objetos de Objective-C respecto a otros lenguajes como Java o C++, es el hecho de la invocación de los métodos (procedimientos) de las instancias de clases. En objective-C esta invocación se da enviando mensajes, algo que se hereda de Smalltalk. Así entonces para invocar un método se procede con el siguiente código por ejemplo:

```
[receiver message];
```

Donde *receiver* es un objeto que recibe un mensaje (acción) *message* a realizar. Esta acción puede tener parámetros asociados, como por ejemplo el siguiente código real:

```
[myRectangle setWidth:20.0];
```

Esta diferencia conceptual de utilizar mensajes se representa en el hecho de que en tiempo de compilación estos mensajes no son más que una etiqueta y no están asociados al bloque de código como es el caso de Java o C++. Entonces es factible que suceda el hecho de que ese mensaje o método no esté implementado por esa clase y recién en tiempo de ejecución es que saltará el error al sustituirse esa etiqueta por un código inexistente, pues un objeto recibe un mensaje para realizar un método que no está dentro de su repertorio. Para esto es que en la documentación de Apple Inc se recomienda utilizar ciertos trucos para garantizar que el objeto que reciba el mensaje sea capaz de responder correctamente, como por ejemplo consultando primero si es capaz de realizar dicha acción y luego en caso de poder realizar dicha acción.
Otro detalle a destacar es que este lenguaje, al igual que Java también soporta la herencia múltiple. Esto es, dado un conjunto de métodos que son comunes a un conjunto de clases pero que no llegan a tener un lazo tan fuerte como para estar jerárquicamente relacionadas con una superclase común, se puede generar una clase abstracta cuyos métodos sean implementados por más de una clase sin necesidad de generar ese vínculo fuerte que es la herencia. Así como en Java existen las interfaces, que hacen esto posible, en Objective-C existen los protocolos. Existen protocolos formales e informales y con métodos obligatorios de implementar y otros opcionales. Una clase que implemente un protocolo dado tiene que tener dentro de su encabezado declarado el nombre del protocolo. Esto es:

```
@interface ClassName : ItsSuperclass < protocol list >
```

Hay otras particularidades del lenguaje pero que no van más allá de la sintaxis como los métodos de clase y los métodos de instancia, como los métodos *get* y *set* para acceder y setear atributos (propiedades) de los objetos, como la notación de *import* en lugar de *include* para quienes están acostumbrados a C y así varias detalles más. Sin embargo más allá de estas y otras diferencias y particularidades resulta un lenguaje relativamente ágil y dentro de todo sencillo de aprender para quien tiene ya un conocimiento de otros lenguajes orientados a objetos.

1.4.3. xCode

Como se dijo anteriormente el entorno de desarrollo de aplicaciones típico es xCode, el cual es gratuito y permite compilar código C, C++, Objective-C, Objective-C++, Java y AppleScript. También viene con *Frameworks* como **Cocoa** y **Cocoa Touch** que proveen de herramientas útiles para desarrollar más fácilmente aplicaciones para Mac OS X e iOS respectivamente.

1.4.4. Librerías

Como muchos lenguajes orientados a objetos, Objective-C cuenta también con una serie de *Frameworks* que se pueden separar en cuatro grandes capas según el nivel de abstracción:

(1) Cocoa Touch Layer

(2) Media Layer

(3) Core services Layer

(4) Core OS Layer

A su vez, dentro de cada capa existen distintos *Frameworks* según la funcionalidad. A continuación se explica un poco más en detalle el rol de cada capa, los distintos *Frameworks* que tiene cada una y para qué sirven.

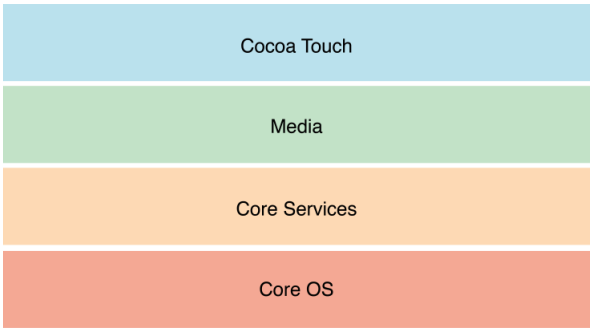

Figura 1.1: Capas de iOS

1.4.4.1. Cocoa Touch Layer

Cocoa Touch es la capa de más alto nivel de iOS y es la encargada de proveer al desarrollador de ciertos *Frameworks* que permitan lograr distintas tecnologías como multitarea, entrada a la aplicación a través de la pantalla táctil, notificaciones y alertas, preservación del estado de la aplicación al salir de la misma, reconocimiento de gestos en la pantalla y otro tipo de funcionalidades de alto nivel. Permiten al desarrollador, sin tener que involucrarse demasiado a bajo nivel, el acceso a determinados servicios que ya están resueltos en forma bastante modular.
Cocoa Touch está basado en la arquitectura Modelo-Vista-Controlador, en el que se separa en tres áreas distintas el modelo de la información, la interfaz de usuario y el conjunto de reglas que negocian la presentación de la información en base a la interacción con el usuario. Así pues, el usuario y una aplicación se podrían considerar dos sistemas que interaccionan. Por su parte el usuario tiene como entrada la vista de la aplicación y como salida tiene su respuesta a esta entrada, generando efectos sobre el control de la aplicación. Por otro lado la aplicación tiene como entrada las órdenes dadas por el usuario que tienen efectos sobre el modelo de la información y este sobre la vista, quien resulta ser la salida de la aplicación. Esta interacción se puede ilustrar con la figura 1.2. Como se

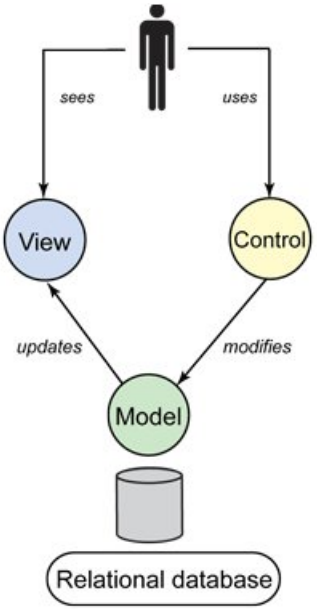

Figura 1.2: Interacción entre las tres partes del MVC

dijo, dentro de Cocoa Touch, existen distintos *Framewokrs* enfocados en permitirle al desarrollador resolver en alto nivel distintos aspectos. Los mismos son los siguientes:

(1) Address Book UI Framework

(2) Event Kit UI Framework

(3) Game Kit Framework

(4) iAd Framework

(5) Map Kit Framework

(6) Message UI Framework

(7) Twitter Framework

(8) UIKit Framework

Quizá sea bueno mencionar que varias de estas API no fueron utilizadas en el presente proyecto dada su función específica y que no fueron necesarias. Sin embargo hay una en particular que tiene bastante importancia y que permite la mayoría de las funcionalidades básicas que toda aplicación tiene. Se trata del **UIKit**, encargado de gestionar la aplicación, su interfaz de usuario y gráficos, encargado soportar eventos frente al toque de la pantalla, de manejar sensores como el acelerómetro y giroscopio, y de tener acceso a la cámara y galería de fotos entre lo más importante a destacar. El soporte de la multitarea y de **Storyboards** también está a cargo de este *Framework*.
Hay funcionalidades que han ido cambiando con las distintas versiones de iOS. Una de ellas y quizá una que ha generado bastantes diferencias respecto a versiones anteriores a iOS 5, es esta última, el Storyboard, una herramienta muy útil de programación gráfica, que permite generar instancias de clases y vínculos entre las mismas en forma visual a la vez de ser una contraparte de interfaz de usuario. Con una biblioteca de objetos disponibles, listos para ser usados, mediante el uso de Storyboard se hace accesible con algunas horas de dedicación implementar aplicaciones sencillas. Esta herramienta vino para sustituir los archivos *.nib* que permitían dise?ar la interfaz pero no tantas funcionalidades programáticas como el Storyboard. En particular éste último permite agregar la funcionalidad de *segues*, encargados de vincular un *ViewController* con otro. Un Storyboard luce como en la figura

FIGURA DE STORYBOARD EJEMPLO

Si bien se podría extender bastante más la explicación sobre los detalles de Cocoa Touch, a los efectos del presente proyecto, no es de tanta relevancia excederse en este punto.

1.4.4.2. Media Layer

Media Layer es la capa encargada de gestionar correctamente todo o relacionado a audio, imagen y video. Todo lo vinculado a multimedia es responsabilidad de esta capa.
Dentro de las tecnologías más destacadas está todo lo vinculado a **gráficos** 2D y 3D, dentro de lo que se puede incluir algunos *Frameworks* bastante utilizados en el presente proyecto, tales como **Core Graphics**, muy utilizado para dibujos 2D, Quartz Core, quien contiene las herramientas necesarias para interactuar con otro *Framework* para animación de vistas, de una capa de más bajo nivel como *Core Animation*, que es comentado más adelante en la sección 1.4.4.3. También es parte de lo vinculado a gráficos, el *Framework* **Core Image**, conteniendo lo vinculado a procesamiento de imágenes a través filtros que utilizan GPU y otros dos *Frameworks* bastante importantes en lo que respecta a *rendering* como **Open GL ES** y **GLKit** (utilizado por el motor de juegos *Isgl3d* entre

otros).
Por otra parte hay otra gran familia de *Frameworks* dentro de Media Layer que apunta a resolver todo lo vinculado al manejo de audio, ya sea de grabación como procesamiento y reproducción de alta calidad. Existen algunos SDK como **iSpeech** o **Dragon Mobile** que resuelven de manera similar al proyecto Siri, el procesamiento de la voz humana reconociendo palabras e interpretando, que utilizan algunos de los *Frameworks* de procesamiento de audio de esta familia.
Video Technologies Core Audio, Media, Animation MediaPlayer
AvFoundation

1.4.4.3. Core Services

Animation

1.4.4.4. Tweeter y mensajería

1.4.4.5. Simulador

Uno de los detalles más importantes del entorno de desarrollo es la capacidad de simular lo que se programa antes de probarlo en un dispositivo. Esto es útil por cuestiones de seguridad e incluso permite programar sin la necesidad a priori de contar con una plataforma. Esto existe para *xcode* y es necesario decir que funciona muy bien, generando una representación bastante fiel de lo que sucede en el dispositivo real. La única crítica que se le podría hacer es el hecho de no contar con cámara y para el caso de aplicaciones de realidad aumentada esto es algo bastante importante. Sin embargo, sin ser eso, el simulador cuenta con conexión a internet, pantalla multitáctil, con información de GPS ingresada por el programador, acceso a la galería de fotos, capacidad de procesamiento y todas las funcionalidades que un dispositivo real tiene.

1.4.4.6. Instruments

Dentro de las herramientas que vienen con el entorno de desarrollo viene *Instruments*, un set de herramientas que permiten analizar la performance de una aplicación para iOS o para Mac OS X desde distintos puntos de vista. Resulta muy útil pues muchas veces sucede que una aplicación compila y se ejecuta correctamente y sin embargo puede el desarrollador puede no estar conforme en cuanto a los tiempos de procesamiento o el uso de memoria consumido.
El el presente proyecto se hizo uso principalmente del **Time Profiler** que permite analizar tiempos y del **Memory Leak** que permite hacer un análisis de la reserva de memoria no liberada.
El **Time Profiler**....
El **Memory Leak** ...

1.5. Herramientas

Herramientas

[**?**].