ISSN 0121–1129

# Desarrollo de aplicaciones móviles bajo la plataforma de Iphone

## Mobile Apps Development on the Iphone's Platform

Fecha de recepción: 20 de agosto de 2011
Fecha de aprobación: 15 de noviembre de 2011

Karen Melissa Rojas Lizarazo*, Jaime Esteban Roa Castañeda**, Andrea Catherine Alarcón Aldana***

## Resumen

El iPhone es un dispositivo móvil que lleva cerca de cinco años en el mercado de celulares, y sus funciones van más allá del uso como tal de un teléfono celular, gracias al *App Store*, que ofrece miles de aplicaciones creadas por desarrolladores alrededor del mundo. La arquitectura y el sistema operativo del iPhone permiten que las aplicaciones creadas por desarrolladores usen tecnologías especializadas para diferentes tipos de usuarios, permitiendo incluir características como gráficos de alta definición o usar accesorios de *hardware*, haciendo uso de los *frameworks* soportados por la arquitectura del dispositivo; todo esto es posible crearlo gracias a las herramientas de desarrollo provistas por Apple. El almacenamiento y la persistencia de datos se ven plasmados en el desarrollo de una aplicación caso de estudio que hace uso de Core Data para el manejo de la información.

**Palabras clave:** Dispositivo Móvil, App Store, iPhone, iOS, Core Data.

## Abstract

The iPhone is a mobile device that has been almost five years in the cell phones market and its functions go beyond the use of it as a phone, thank to the App Store that offers thousands of applications made by developers around the world. The architecture and operation system of the iPhone let appcations made by developers use specialized technologies for different kind of users, allowing them to include features like high-definition graphics or to use hardware accessories by utilizing frameworks supported by the device's architecture. All this is possible thank to the Apple´s development tools. Data and persistent storage are showed in the development of a case-of-study-application that employs Core Data for information management.

**Key words:** Mobile Device, App Store, iPhone, iOS, Core Data.

* Ingeniera de Sistemas y Computación, Universidad Pedagógica y Tecnológica de Colombia, Investigadora del Grupo de Investigación en Software –GIS–. kameroli@gmail.com
** Ingeniero de Sistemas y Computación, Universidad Pedagógica y Tecnológica de Colombia, Investigador del Grupo de Investigación en Software –GIS–. estebanroatoi@gmail.com
*** Ingeniera de Sistemas y Computación, Especialista en Ingeniería de Software, Magíster en Software Libre, Universidad Pedagógica y Tecnológica de Colombia, Docente e investigadora del Grupo de Investigación en Software –GIS–. andrea.alarconaldana@uptc.edu.co

## I. Introducción

Los dispositivos móviles hacen parte de un mercado creciente tanto en el campo comercial como en el de la investigación; esto ha llevado a sus creadores a permanecer en constante competencia, que implica la creación de nuevos y más potentes dispositivos, con capacidades tecnológicas que cada vez se asemejan más a las de un computador personal. El iPhone, desarrollado por Apple, ha sido popular desde su creación en el año 2007, ya que permitió a los usuarios realizar actividades diferentes a las de hacer y recibir llamadas, acciones típicas de un celular. Además de las llamadas, el iPhone permite reproducir música y videos, navegar en internet, capturar imagen y video, y hacer uso del sistema de posicionamiento global, entre otras actividades. Debido al auge de este dispositivo, y a sus capacidades tecnológicas, la empresa Apple permitió a los desarrolladores la creación y distribución de aplicaciones diferentes a las desarrolladas por la misma compañía, dando así la oportunidad de explotar al máximo el dispositivo, y de que se generaran ingresos económicos para los creadores de tales aplicaciones.

En Colombia, el iPhone no es popular debido al costo y al desconocimiento de él; por esta razón, el porcentaje de aplicaciones en el App Store alusivas a Colombia es muy bajo, lo que demuestra el poco interés en la programación para este dispositivo. Debido a esto, en este trabajo se documenta la investigación acerca del uso de la plataforma de desarrollo de aplicaciones para el iPhone, y además se tratan temas como la arquitectura, el sistema operativo, las herramientas para el desarrollo de sus aplicaciones, los métodos de almacenamiento de información y, finalmente, se plantea una aplicación caso de estudio para ser desarrollada a lo largo de la investigación.

## II. Conceptualización

Es de gran importancia conocer la siguiente terminología, asociada con esta investigación:

*Ingeniería de software.* Es la disciplina que comprende los procesos técnicos de desarrollo de software, la gestión de proyectos y el desarrollo de herramientas, entre otros aspectos, que van desde la especificación de requisitos hasta el mantenimiento del software [16].

*Administración de datos.* Es el desarrollo y ejecución de las arquitecturas, políticas, prácticas y procedimientos que gestionan adecuadamente el ciclo de vida completo de las necesidades de datos de una empresa [8].

*Dispositivo móvil.* Es un aparato diseñado para una función específica, aunque puede llevar a cabo otras funciones generales; se caracteriza por ser de pequeño tamaño, con capacidad de procesamiento y memoria limitadas y conexión (permanente o intermitente) a una red; este tipo de dispositivo computarizado depende cada vez más del software, funciona bajo una plataforma que permite a los desarrolladores crear y ejecutar aplicaciones y funciones complejas.

*Desarrollo de software para dispositivos móviles.* Con el auge de los dispositivos móviles el desarrollo de aplicaciones ha avanzado con fines lucrativos, de investigación y de satisfacción de necesidades, entre otros. Las aplicaciones son creadas mediante herramientas y kits de desarrollo específicos para cada plataforma . Por lo general, cada plataforma ofrece un simulador para probar las aplicaciones, sin embargo, la mejor prueba es en el dispositivo real.

*App Store.* Es una tienda virtual creada por Apple Inc., donde se encuentran más de 425.000 aplicaciones creadas por desarrolladores de Apple y externos a la compañía, las cuales están disponibles para ser descargadas de forma gratuita o pagando con tarjeta de crédito.

## III. Iphone

Es un dispositivo móvil sofisticado, catalogado como smartphone (celular inteligente), perteneciente a la compañía Apple Inc. Tiene un teclado virtual, la tecnología de pantalla permite el reconocimiento de gestos multi-touch; capacidad de almacenamiento de hasta 64 Gb, y elementos de hardware, como antena GPS, acelerómetro y cámara de 5 Mp, por nombrar algunos [14].

### *A. Arquitectura*

El iPhone cuenta con arquitectura y tecnologías que permiten ejecutar aplicaciones bajo el sistema operativo iOS 4; dicha arquitectura está compuesta por cuatro capas, como se observa en la Fig. 1.

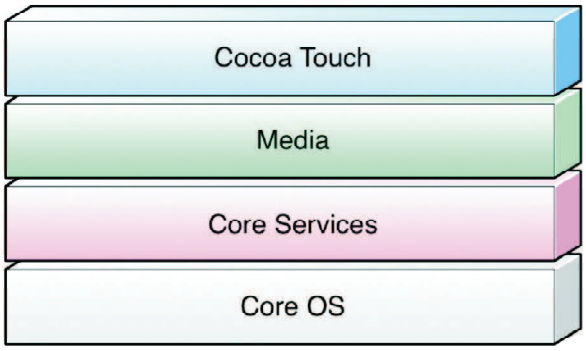



**Fig. 1**. Capas de la arquitectura de iOS. Fuente: [7]

1) ***Core OS.*** Hace referencia al ambiente del kernel, drivers de interfaces básicas del sistema operativo de iPhone, administra memoria virtual, cadenas, sistema de archivos, redes y comunicaciones, entre otros procesos [7].

2) ***Core Services.*** Ofrece acceso a los servicios fundamentales del sistema operativo [7], dentro de los cuales están SQLite Library, XML Libraries, CFNetwork Framework, Core Foundation Framework y Security Framework, entre otros.

*3)* ***Media.*** Contiene audio, video y tecnologías gráficas que son diseñadas para proveer capacidades de animación al dispositivo; esta capa permite agregar gráficos de alta calidad a la aplicación, haciendo uso de tecnologías 2-D y 3-D , entre las cuales están Core Graphics, Quartz, OpenGL ES, Core Animation, Core Audio & Audio ToolBox Frameworks, OpenAL, Core Audio & Audio ToolBox Frameworks, entre otras.

*4)* ***Cocoa Touch.*** Provee las clases primarias para implementar un evento gráfico; cada aplicación en el iPhone usa un framework para implementar interfaces de usuario; también incluye otros frameworks que permiten el acceso a características del dispositivo [7], entre los cuales están UIKit, Address Book, Core Location, Message y Game Kit, por nombrar algunos.

### *B. iOS 4*

El sistema operativo del iPhone era conocido anteriormente como iPhone OS; sin embargo, con el lanzamiento de la versión 4, en el evento "Apple WWDC 2010"[2], se vio la necesidad de cambiarle el nombre al de iOS 4, debido a que no solo está presente en el iPhone, sino además en el iPad y iPod. Algunas características implementadas en dicho sistema operativo son: Multitarea, Carpetas, Bandeja de entrada unificada, Cámara mejorada, Game Center, iBooks, iAds, Mapas y Brújula, FaceTime, entre otras.

### *C. Xcode*

Set de herramientas pertenecientes a la compañía Apple, encontrado bajo el nombre de Xcode Developer Tools (Herramientas de desarrollo de Xcode) [7]; permiten desarrollar aplicaciones basadas en el ambiente orientado a objetos llamado Cocoa, codificar, realizar administración de proyectos, diseñar interfaces de usuario, depurar y

realizar análisis de rendimiento, entre otras actividades . Las herramientas utilizadas para dichas actividades son:

***1) Xcode IDE.*** Es el entorno de desarrollo integrado (IDE) que provee todas las herramientas para codificar, crear y depurar aplicaciones desarrolladas para los sistemas operativos Mac OS X y iOS [7]; está integrado con Cocoa, que es el ambiente orientado a objetos, y sus librerías, llamadas Cocoa Frameworks [4]; además soporta el lenguaje orientado a objetos Objective C. [15]. Debido a que un proyecto administra la información asociada con la aplicación, ella se concentra en una única ventana de proyecto (Project Window), en donde se encuentra Barra de Herramientas, Lista de Grupos y Archivos, Vista de Detalles, Editor de Código y Barra de Estado, entre otros, como se observa en la Fig. 2.

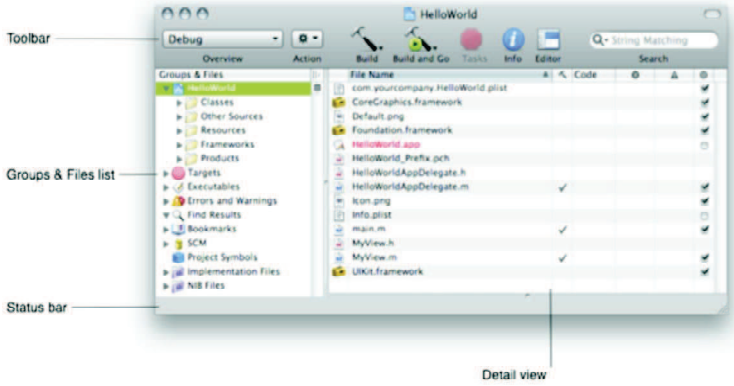



**Fig. 2**. Ventana Principal de Xcode. Fuente [7].

*2)* ***Interface Builder.*** El Constructor de Interfaces es una herramienta que permite y facilita el diseño de interfaces de usuario tanto para Mac OS X como para iOS, sin necesidad de escribir código para diseñarlas. Trabaja en tiempo real con Xcode, de manera que simplemente se diseña la interfaz de usuario e inmediatamente se conectan los controles gráficos agregados con el código escrito en Xcode mediante el uso de apuntadores para que la aplicación funcione completamente [4]. Interface Builder no genera código, crea objetos del lenguaje de programación Objective –C y luego los conecta con el archivo NIB [11]. Un archivo nib tiene una extensión .xib y contiene toda la información de la interfaz gráfica.

El Constructor de Interfaces contiene una serie de ventanas que ayudan a realizar las interfaces de usuario, entre las cuales están Catálogo, Inspección, Documentos y Principal, tal y como se observa en la Fig. 3.

***3) Instruments.*** Es una herramienta que permite analizar el rendimiento de la aplicación, mediante la revisión y examen de los procesos. Es útil para que el software se comporte de manera adecuada, ya que permite rastrear dificultades para descubrir problemas en el código, hacer análisis de rendimiento, actividad de los discos, actividad de la red, uso de memoria, rastrear desbordamientos de memoria, entre otras [5]. Estos análisis se pueden realizar con la ayuda de un gráfico conocido como "Timeline", y de los diferentes paneles de la ventana del documento de rastreo (Trace Document Window), entre los cuales están [5]: Panel de Instrumentos, Panel de Rastreo, Barra de Navegación, Panel de Detalle y Panel de Extendido, como se observa en la Fig. 4.

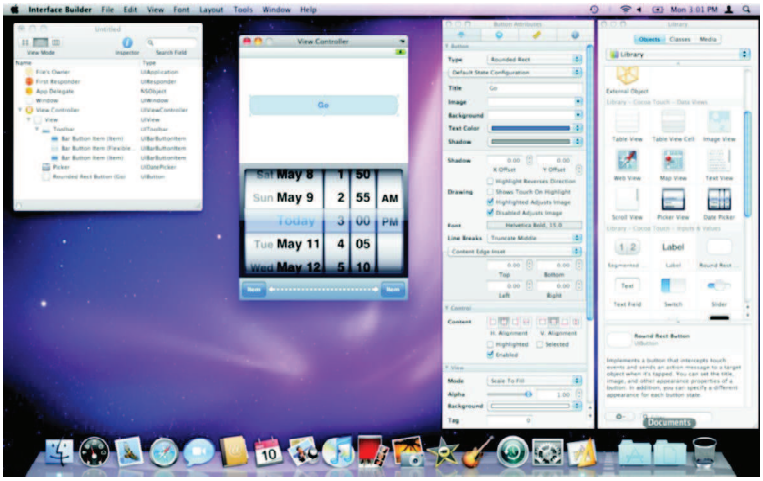


**Fig. 3**. Interface Builder. Fuente [7]

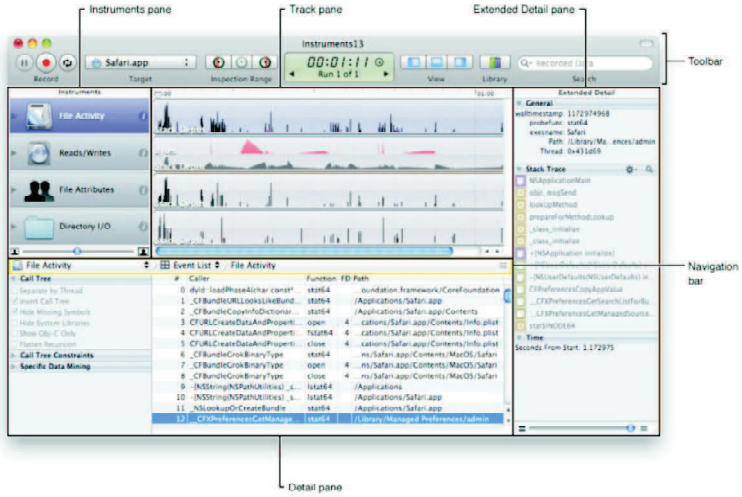



**Fig. 4**. Ventana Instruments. Fuente [5]

***4) iOS SDK.*** Es el kit de desarrollo de aplicaciones para los dispositivos móviles iPhone, iPod Touch y iPad; contiene el código, la información y las herramientas de desarrollo que funcionan de la mano y se acoplan perfectamente con Xcode para crear aplicaciones nativas para dichos dispositivos [11]. Cuando se trabaja con iOS SDK, las mismas herramientas pertenecientes a Xcode se acomodan a las necesidades del dispositivo para el cual se va a programar, utilizando diferentes frameworks, como el llamado Cocoa Touch Framework. También incluye la herramienta iOS Simulator, utilizada para realizar pruebas de funcionamiento.

***5) iOS Simulator.*** El Simulador iOS está incluido en el iOS SDK; es una herramienta que ayuda a los desarrolladores a realizar pruebas de funcionamiento de las aplicaciones creadas; ejecuta la aplicación como si se estuviera haciendo en el dispositivo real y simula los eventos táctiles con la ayuda del mouse [4], la rotación de la pantalla (de landscape a portrait o viceversa) y advertencias de memoria baja, entre otras [12]. Esta herramienta simula el comportamiento del iPhone, sin embargo, tiene limitaciones [12], no realiza llamadas ni envía mensajes, no usa la cámara ni el micrófono, y no puede instalar aplicaciones provenientes del App

Store, entre otras; dichas limitaciones indican que no se puede confiar totalmente en el simulador, porque él mismo depende del rendimiento del computador. Entonces, es importante probar las aplicaciones también en el dispositivo para comprobar el verdadero comportamiento de la aplicación [12].

### *D. Métodos de almacenamiento de datos en el iPhone*

Algunas aplicaciones que se ejecutan en el iPhone crean datos que necesitan ser almacenados para que estén disponibles la próxima vez que un usuario corra la aplicación; por esta razón, existen tres métodos para almacenar los datos generados por las aplicaciones [1]:

***1) Archivos planos.*** Almacenan pequeños datos de texto que no están estructuralmente relacionados. Estos archivos guardan datos que tienen que ver con configuraciones de aplicaciones o programas, y los usuarios de dichas aplicaciones nunca tienen acceso a su contenido [20].

***2) SQLite.*** Es un paquete de dominio público y código libre para cada propósito; provee un Administrador de Bases de Datos Relacionales. SQLite es una base de datos ligera en términos de configuración, sobrecarga de administración y uso de recursos, además es multiplataforma, no requiere de un servidor, ya que la librería de SQLite accesa los archivos de almacenamiento directamente, y soporta la gran mayoría de las características de los lenguajes de consulta pertenecientes al Estándar SQL92, entre otras [9].

Debido a que solo una librería contiene todo el sistema de base de datos, esta librería está incluida en el iOS SDK y es útil para el trabajo de almacenar datos generados por las aplicaciones; utiliza una escritura dinámica, lo cual indica que la columna donde se está escribiendo el valor reconoce automáticamente qué tipo de dato es. Los tipos de datos que puede almacenar SQLite son: NULL, INTEGER, REAL, TEXT y BLOB . Es de destacar que SQLite es una opción para proyectos que no requieren alta concurrencia de usuarios y para aplicaciones que no sean Cliente-Servidor [17].

***3) Core Data.*** Es un framework de persistencia versátil que sirve para crear aplicaciones bajo el patrón Modelo-Vista-Controlador, y es la forma en la cual Cocoa almacena los datos de sus aplicaciones sin importar el tamaño que estas tengan [6]. Trabajar con Core Data es sencillo debido a que ayuda a esconder todas esas complejidades existentes en el almacenamiento de datos, gracias a que el Creador de Interfaces contiene objetos controladores de Core Data para que sea mínimo el esfuerzo de codificar [3]. La administración de datos en las aplicaciones para dispositivos móviles es importante, ya que permite que los datos que las aplicaciones generan persistan para su posterior uso.

Es importante tener en cuenta que Core Data no es un motor de bases de datos y no necesita usar una base de datos relacional para el almacenamiento, aunque permite el uso de entidades, atributos y relaciones [13]. Core Data ofrece varios tipos de almacenamiento, como XML, binario y SQL, que almacenan los datos dependiendo del diseño del modelo. El tipo de almacenamiento SQL está basado en la librería SQLite [18]. Core Data ofrece ventajas como soporte automático para deshacer y rehacer, mantiene las relaciones recíprocas entre objetos, ayuda a conservar memoria en el dispositivo - evitando su desbordamiento, para que las aplicaciones no finalicen inesperadamente-, permite definir el modelo de objetos en un editor GUI y provee una infraestructura para el versionado y la migración de los datos, entre otras [3].

Una vez analizados estos tres métodos de almacenamiento disponibles en el iPhone, se eligió Core Data como el que será utilizado en la aplicación caso de estudio, debido a que ofrece una mejor integración con el resto de los frameworks de desarrollo de Cocoa Touch. Además, la compañía Apple está incitando a todos los desarrolladores de aplicaciones para iPhone a usar Core Data, lo que llevó a que desapareciera la documentación existente en la página del programa Developer acerca del uso de SQLite en las aplicaciones para iPhone.

## IV. Desarrollo de la aplicación caso de estudio

La aplicación caso de estudio se realiza con el fin de poner en práctica los conocimientos acerca de la arquitectura y sistema operativo del iPhone, haciendo uso de las herramientas que Xcode ofrece e implementando el método de almacenamiento de datos Core Data para la persistencia de datos. Para el desarrollo de la aplicación caso de estudio se toma como referencia las fases de la metodología de desarrollo XP (Programación eXtrema), descritas a continuación.

### *A. Planificación del proyecto*

La aplicación caso de estudio va dirigida a médicos que desempeñan su labor en el área de consulta externa y necesitan almacenar datos de pacientes, agregar datos comunes de consulta y llevar un registro del historial de las consultas. Los anteriores requisitos surgen de las historias de usuario definidas por médicos. El plan de entregas se define a un tiempo de realización de dos semanas por cada historia de usuario definida por los médicos, con un total de seis semanas de desarrollo. Para cumplir con el proyecto se realizan una serie de actividades, entre las que se incluyen construcción de los prototipos de la aplicación, diseño del modelo de datos, pruebas para determinar el buen funcionamiento de la aplicación, documentación y consideraciones adicionales a la aplicación caso de estudio MiConsulta, entre otras; el tiempo estimado para el desarrollo total del proyecto es de doce semanas.

### *B. Diseño*

En esta fase se diseña la aplicación de acuerdo con las historias de usuario descritas por los médicos; para ello se diseñan los prototipos en la herramienta Interface.

***1) Módulo Paciente.*** Como se observa en la Fig. 5, este módulo presenta dos vistas: la primera, llamada "Pacientes", presenta una lista ordenada alfabéticamente de todos los pacientes que han sido tratados; la segunda, llamada "Paciente", permite diligenciar los datos del paciente por tratar.

***2) Módulo Consulta.*** La Fig. 6 presenta una vista en la cual se diligencian los datos referentes a la salud del paciente.

***3) Módulo Historia Clínica.*** En este módulo, como lo muestra la Fig. 7, es posible visualizar una lista, organizada por fecha, de todas las consultas realizadas a un paciente.

***4) Modelo de datos.*** El modelo de datos de la aplicación caso de estudio MiConsulta está compuesto por 2 entidades, como se observa en la Fig. 8, cada una con sus correspondientes atributos. Este modelo fue creado con la herramienta modeladora de datos de Xcode.

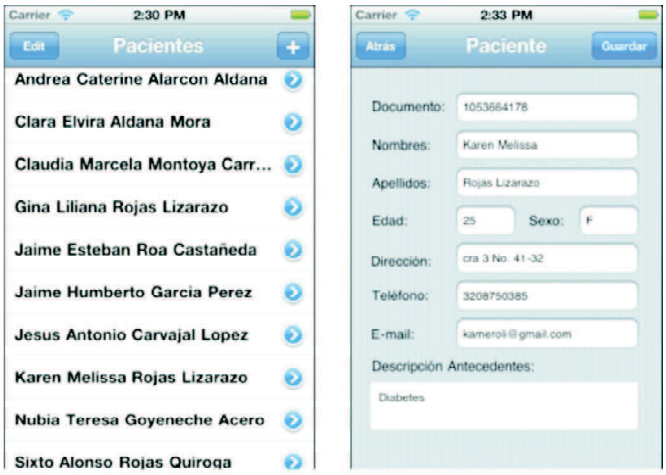



**Fig. 5**. Módulo Paciente.

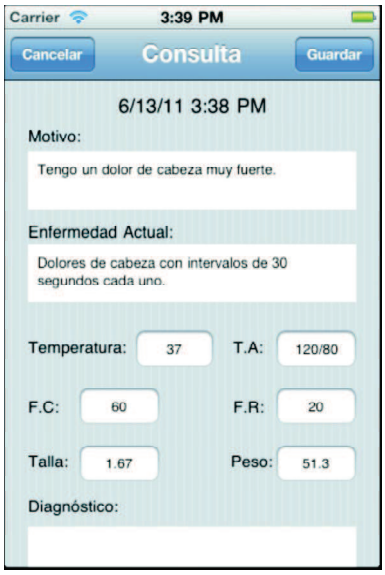



**Fig. 6**. Módulo Consulta.

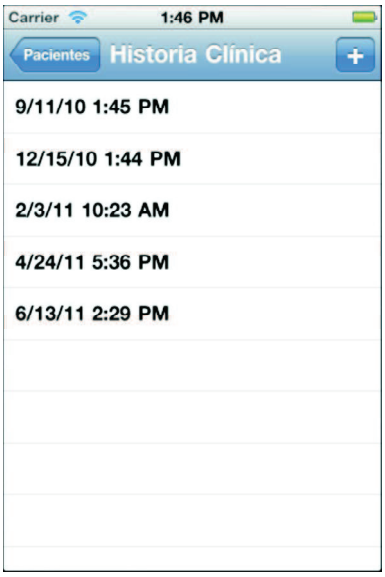



**Fig. 7**. Módulo Historia Clínica.

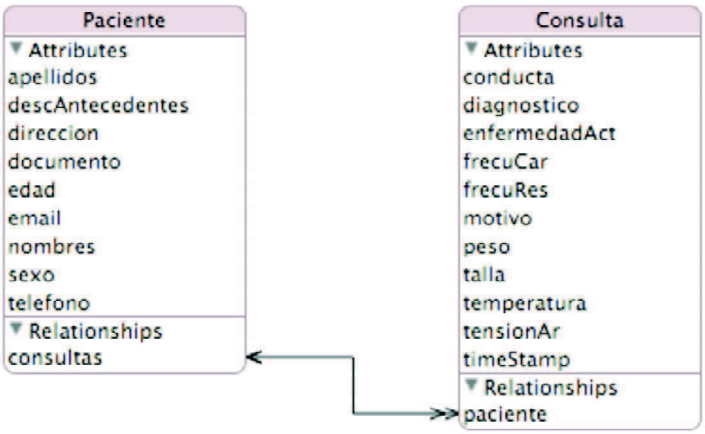



**Fig. 8**. Modelo de datos aplicación MiConsulta

### *C. Codificación*

En esta fase de codificación, la aplicación caso de estudio, llamada MiConsulta, se desarrolló mediante el paradigma de programación por parejas; por esta razón cada uno de los módulos fue realizado en equipo, haciendo uso de las herramientas Xcode y iOS SDK.

Teniendo en cuenta que Xcode trabaja con el lenguaje de programación orientado a objetos Objective-C, la codificación en este lenguaje fue el principal inconveniente, debido a que su sintaxis se basa en el uso del pseudocódigo, lo que lo hace diferente. Algunos de sus conceptos, como el uso de properties y synthesize para la generación de getters y setters, clases delegate, así como archivos con extensión .XIB, generaron confusión al momento de su aplicación en el desarrollo del caso de estudio.

Para el desarrollo de la aplicación fue necesario el uso de la plantilla Navigation-based application, porque provee tablas y botones para facilitar la navegación necesaria y genera una serie de métodos para el manejo de la información en filas. El framework usado para la persistencia de datos fue Core Data, que no necesita de una base de datos relacional para el almacenamiento, y en este caso utiliza el tipo de almacenamiento SQL. Este framework hace uso de sus tres clases principales: NSManagedObjectContext, encargada de manejar el contexto; NSManagedObjectModel, que es el modelo en sí, y NSPersistentStoreCoordinator, encargada de la persistencia de datos; estas clases son implementadas en la clase delegate.

Se crearon cinco clases que se encargan de dar funcionalidad a la aplicación. La primera es MiConsultaAppDelegate, de tipo NSObject, y usa el protocolo <UIApplicationDelegate>; esta clase implementa los métodos que provee Core Data para el almacenamiento de datos. Las dos siguientes clases son de tipo UITableViewController, llamadas RootViewController y PacienteListViewController; son las encargadas de listar la información correspondiente a los pacientes y consultas, además manejan la funcionalidad de la aplicación gracias a sus botones ubicados en las barras de navegación. Las dos últimas clases son PacienteViewController y ConsultaViewController, que son de tipo UIViewController y son las encargadas de guardar la información personal de los pacientes y sus consultas.

Además de las clases, la aplicación tiene los archivos MiConsulta.xcdatamodel, que contiene el modelo de datos, y PacienteViewController.xib y Consulta ViewController.xib, encargados del manejo de los elementos de la interfaz de usuario para las clases que llevan el mismo nombre.

Por último, antes de ejecutar la aplicación es necesario tener cuidado con el concepto de administración de memoria, que es muy importante debido a que la aplicación va a ser ejecutada en un dispositivo móvil de capacidades limitadas; por esta razón es importante liberar la memoria que es creada en el proceso.

### *D. Pruebas*

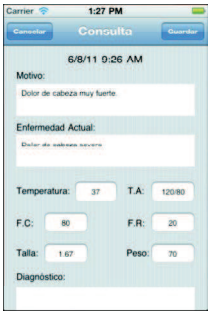
Se realizan tres tipos de pruebas: la primera, haciendo una evaluación del almacenamiento de datos y la persistencia de estos, llamada prueba unitaria; la segunda, ejecutando la aplicación en el dispositivo real y poniéndola a prueba por uno de los médicos que colabora con las historias de usuario, llamada prueba de funcionalidad, y la tercera, ejecutando la aplicación desde el iOS Simulator y evaluándola en la herramienta Instruments, llamada prueba de estrés.

Las pruebas unitarias son llevadas a cabo por el equipo de desarrolladores, y sus resultados se presentan a continuación:

CUADRO 1. Prueba unitaria N.º 1

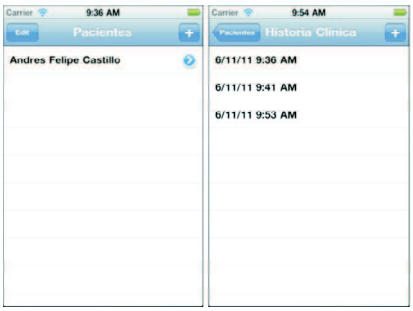
| APLICACIÓN MICONSULTA | |
|---|---|
| **INFORMACIÓN GENERAL** | |
| **Prueba: 01** | |
| • **Fecha:** | 11/06/01 |
| • **Tipo de prueba:** | Unitaria |
| • **Persona que realizó la prueba** | Karen Melissa Rojas<br>Jaime Esteban Roa |
| • **Nombre de la prueba** | Almacenar paciente |
| • **Objetivo** | Almacenar un nuevo paciente |
| • **Área de prueba** | Simulador iOS |
| • **Requisitos** | - No haber almacenado datos antes.<br>- Ingresar en el momento en que la aplicación se encuentre activa en el simulador. |
| **DESARROLLO DE LA PRUEBA**<br>1.Ejecutar la aplicación en el simulador.<br><br>2.Ubicarse en la barra de navegación con nombre del paciente y dar clic en el botón +.<br><br>3.Diligenciar los datos del paciente.<br><br>4.Hacer clic en el botón guardar de la figura presentada en la columna de la derecha. |  |
| • **Resultado esperado** | La aplicación almacena los datos del paciente que fueron diligenciados. |
| • **Resultado obtenidos** | Los datos fueron almacenados correctamente.<br>  |

CUADRO 2. Prueba unitaria N.º 2

| APLICACIÓN MICONSULTA | |
|---|---|
| **INFORMACIÓN GENERAL** | |
| **Prueba: 02** | |
| • **Fecha** | 11/06/01 |
| • **Tipo de prueba** | Unitaria |
| • **Persona que realizó la prueba** | Karen Melissa Rojas<br>Jaime Esteban Roa |
| • **Nombre de la prueba** | Agregar consulta a un paciente existente |
| • **Objetivo** | Agregar consulta |
| • **Área de prueba** | Simulador iOS |
| • **Requisitos** | - Haber almacenado datos de pacientes<br>- Ver el listado de los pacientes existentes |
| **DESARROLLO DE LA PRUEBA**<br>1.Ejecutar la aplicación en el simulador<br><br>2.Ubicarse en la lista de pacientes<br><br>3.Dar clic en la flecha azul de uno de los pacientes<br><br>4.Ubicarse en la barra de navegación Historia Clínica y dar clic en el botón +<br><br>5.Diligenciar los datos de la consulta<br><br>6.Hacer clic en el botón guardar |  |
| • **Resultado esperados** | La aplicación almacena los datos de la consulta perteneciente a un paciente |
| • **Resultado obtenidos** | Los datos fueron almacenados correctamente<br>  |

La prueba de funcionalidad es realizada por el doctor Hassan Matar, médico general egresado de la Universidad Nacional; el resultado se presenta a continuación:

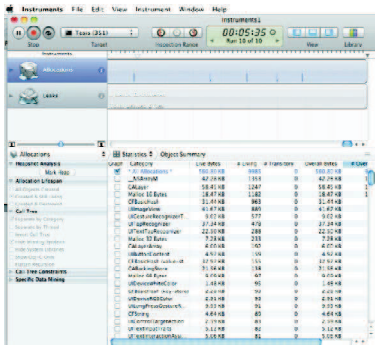
CUADRO 3. Prueba de funcionalidad

| APLICACIÓN MICONSULTA | |
|---|---|
| **INFORMACIÓN GENERAL** | |
| **Prueba: 03** | |
| • **Fecha** | 11/06/01 |
| • **Tipo de prueba** | Funcionalidad |
| • **Persona que realizó la prueba** | Doctor Hasan Matar |
| • **Nombre de la prueba** | Persistencia de datos y funcionalidad de la aplicación |
| • **Objetivo** | Evaluar la persistencia de datos en el dispositivo y la funcionalidad de la aplicación |
| • **Área de prueba** | iPhone |
| • **Requisitos** | - No haber almacenado datos antes |
| **DESARROLLO DE LA PRUEBA**<br>1. Ejecutar la aplicación en el iPhone<br>2. Dar clic en el botón + de la barra de navegación de pacientes<br>3. Diligenciar los datos del paciente y dar clic en el botón guardar<br>4. Dar clic sobre el la flecha azul del paciente creado<br>5. Dar clic en el botón + de la barra Historia Clínica<br>6. Diligenciar los datos de consulta y dar clic en el botón guardar<br>7. Cerrar la aplicación<br>8. Ejecutar nuevamente la aplicación |  |
| • **Resultado esperados** | La aplicación persiste los datos que fueron almacenados |
| • **Resultado obtenidos** | Los datos persistieron correctamente en el dispositivo |

La prueba de estrés se realiza con la herramienta Instruments incluida en Xcode, y su resultado de presenta a continuación:

Las pruebas arrojaron los resultados esperados en cuanto a funcionalidad de la aplicación, persistencia de datos y administración de memoria.

CUADRO 4. Prueba de estrés

| **APLICACIÓN MICONSULTA** | |
|---|---|
| **INFORMACIÓN GENERAL** | |
| **Prueba: 04** | |
| • **Fecha** | 11/06/01 |
| • **Tipo de prueba** | Estrés |
| • **Persona que realizó la prueba** | Karen Melissa Rojas<br>Jaime Esteban Roa |
| • **Nombre de la prueba** | Análisis de memoria |
| • **Objetivo** | Analizar asignación y desbordamiento de memoria |
| • **Área de prueba** | Simulador iOS |
| • **Requisitos** | - Previamente haber almacenado pacientes y consultas |
| **DESARROLLO DE LA PRUEBA**<br>1.Ejecutar la aplicación en el simulador<br>2.Abrir Instruments y agregar los instrumentos Leaks y Allocations<br>3.Elegir el Target que se va a analizar, en este caso MiConsulta<br>4.Dar clic en el botón rojo record<br>5.Interactuar con la aplicación<br>6.Esperar 10 minutos<br>7.Documentar los resultados<br>8.Dar clic en el botón rojo stop |  |
| • **Resultado esperados** | La aplicación no presenta desbordamiento de memoria y hace la asignación de ella |
| • **Resultado obtenido** | No se presenta desbordamientos de memoria y la asignación de ella se hace correctamente |

## V. Conclusiones

Recientemente se dio a conocer, en el evento WWDC 2011 (World Wide Developers Conference) de Apple, que existen más de 425.000 aplicaciones en el App Store, y más de 200 millones de dispositivos vendidos alrededor del mundo, lo que muestra una gran oportunidad de abordar un mercado mundial sin necesidad de salir de la casa; la descarga de 15 billones de aplicaciones del App Store es un incentivo para comenzar a participar de los 2.5 billones de dólares que Apple ha pagado a desarrolladores externos a la compañía.

El aporte colombiano de aplicaciones al App Store es escaso, se visualiza en el 0.02%; por tal motivo es tentativa la oportunidad de ser pioneros en el desarrollo y distribución de aplicaciones creadas por colombianos.

La arquitectura del iPhone está conformada por cuatro capas; permite la administración de memoria virtual en la capa más baja, es decir, Core OS; pasa por el acceso a los servicios fundamentales del sistema operativo en la capa Core Services; muestra los gráficos de alto nivel que la capa Media permite, y finaliza con la implementación de frameworks que permiten el uso de hardware en la capa Cocoa Touch. El sistema operativo del iPhone (iOS 4) evoluciona para brindar a los desarrolladores nuevas características que permiten crear aplicaciones cada vez más complejas y completas, para cumplir o satisfacer en un mayor grado las necesidades de los usuarios finales.

Las herramientas usadas para crear aplicaciones nativas en el iPhone son Xcode y iOS SDK, que proveen una serie de características, paneles y barras para facilitar el desarrollo de aplicaciones.

Los métodos de almacenamiento de datos que son generados por las aplicaciones para iPhone son: archivos planos, SQLite y Core Data; cada uno de estos métodos pueden servir para un fin específico, dependiendo de las necesidades de la aplicación.

Core Data es un framework de persistencia versátil que utiliza Cocoa para el almacenamiento de datos de sus aplicaciones; este framework se integra correctamente con los demás frameworks de desarrollo de Cocoa Touch.

La aplicación caso de estudio permite tener un acercamiento al proceso que se lleva a cabo desde la generación de una idea, pasando por el proceso de desarrollo de la aplicación y llegando a la depuración en un dispositivo real, con el fin de prestar un servicio a una población específica.

Aunque el iOS simulator de Apple es una herramienta completa y sofisticada, no puede igualar las características ni la versatilidad que ofrece un dispositivo real al momento de correr una aplicación para hacer las pruebas y navegar a través de ella; sin embargo, facilita el desarrollo de aplicaciones, gracias a que permite ejecutarlas durante el proceso de creación.

## Referencias

[1] A. Alasdair. *Learning iPhone Programming*. O´Reilly, 2010.

[2] Apple. *Apple WWDC 2010 Keynote Address*. 2010. [Sitio web]. Disponible en: <http://www.apple.com/apple-events/>. [con acceso el 15 de marzo de 2011].

[3] Apple. *Core Data Tutorial for iOS. Data Management*. 2010. [Documento en linea] Disponible en: <http://developer.apple.com/library/ios/navigation/>. [con acceso el 22 de marzo de 2011].

[4] Apple. *Developer Tools. Tools you'll love to use*. 2011. [Sitio web]. Disponible en: <http://developer.apple.com/technologies/tools/>. [con acceso el 23 de febrero de 2011].

[5] Apple. *Instruments User Guide. Tools & Languages: Performance Analysis Tools*. 2010.

[Documento en linea]. Disponible en: <http://developer.apple.com/library/mac/navigation/>. [con acceso el 20 de marzo de 2011].

[6] Apple. *iOS Data Management*. 2011. [Sitio web]. Disponible en: <http://developer.apple.com/technologies/ios/data-management.html>. [con acceso el 4 de febrero de 2011].

[7] Apple. *iOS Technology Overview*. 2010. [Documento en linea]. Disponible en: <http://developer.apple.com/library/ios/navigation/>. [con acceso el 20 de febrero de 2011].

[8] DAMA. *Data Management International* 2011. [Sitio web]. Disponible en: <http://www.dama.org/i4a/pages/index.cfm?pageid=1>. [con acceso el 10 de junio de 2011].

[9] J. A. Kreibich. *Using SQLite*. O'Reilly, 2010.

[10] Sonera Media Lab. *Mobile Java Application Development* 2002. >.

[11] J. LaMarche, Jeff. Beginning iPhone Development: *Exploring the iPhone SDK*. Apress: 2008.

[12] W-M. Lee. *Beginning iOS 4 application development*. Wrox 2010.

[13] M. Privat, R. Warner. *Pro Core Data for iOS*. Apress: 2011.

[14] A. Nowak. *iPhone Description - Part I* 2008. [Sitio web]. Disponible en: <http://www.articlesbase.com/computers-articles/iphone-description-part-i-420010.html>. [con acceso el 15 de febrero de 2011].

[15] I. Pipe. *Learn Xcode Tools for Mac OS X and iPhone development*. Apress, 2009.

[16] I. Somerville. *Ingeniería del Software* Séptima edición ed. Pearson Addison Wesley, 2005.

[17] SQLite. *Appropriate Uses For SQLite*. 2011. [Sitio web]. Disponible en: <http://www.sqlite.org/whentouse.html>. [con acceso el 22 de marzo de 2011].

[18] S. Stevenson. *Cocoa and Objective - C: Up and Running*. O'Reilly, 2010.

[19] Techotopia. *The iPhone OS Media Layer*. 2011. [Sitio web]. Disponible en: <http://www.techotopia.com/index.php/The_iPhone_OS_Media_Layer>. [con acceso el 16 de febrero de 2011].

[20] TopBits. *Flat File*. 2010. [Sitio web]. Disponible en: <http://www.tech-faq.com/flat-file.html>. [con acceso el 22 de marzo de 2011].