Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería en Sistemas y Telemática

Aplicación móvil para personas con Alzheimer en etapa inicial.

Tesis previa a la obtención del título de
Ingeniero en Sistemas

Autor:
Juan Sebastián Landy Ríos

Director:
Prof. Dr. Francisco Salgado

Cuenca – Ecuador
2015

Dedicatoria.

Para mis padres, quienes con su ejemplo y su gran esfuerzo por triunfar en la vida me enseñaron que el éxito depende de uno mismo y con perseverancia y sacrificio se consiguen todas las metas y sueños que me proponga en esta vida.

Agradecimientos.

A mi director de tesis, Prof. Dr. Francisco Salgado, un verdadero ejemplo a seguir para mí y para todos los estudiantes de la Universidad del Azuay.

Zaida Piedra, mi abuela materna, por su invalorable apoyo en toda mi etapa universitaria.

A todos mis familiares, por las innumerables ocasiones que me ayudaron en las buenas y malas épocas de mi vida, gracias por todo su apoyo.

A mis amigos y compañeros universitarios, gracias por su paciencia y por compartir conmigo los mejores años de mi vida.

Al Dr. Fabián Guapisaca, Geriatra del Hospital Universitario del Río por su profesionalismo y su ayuda incondicional en las pruebas finales de esta tesis.

A mi gran amigo y futuro doctor David Álvarez, gracias por su paciencia y su ayuda en las pruebas finales de la aplicación.

Por su calidad académica y humana, a todos los profesores de la Universidad del Azuay que vivieron junto a mi esta inolvidable y maravillosa etapa.

Índice de Contenidos.

Índice de Figuras.

Índice de Tablas

Resumen.

En este trabajo hemos buscado solventar los problemas que sufren las personas con alzheimer en etapa inicial a través de un asistente móvil, el cuál les permitirá realizar sus actividades diarias de manera autónoma sin la necesidad de un cuidador que brinde su apoyo de manera constante para desempeñarse de manera autónoma en actividades básicas que una persona que sufre dicha enfermedad puede realizarlas día a día. A través del diseño y desarrollo de la aplicación móvil para dispositivos iOS integramos funcionalidades como el uso del gps – sistema de posicionamiento global - para ubicar al paciente en caso de que éste exceda un perímetro establecido por su cuidador, un botón de ayuda remota a través del cual el paciente puede informar a su cuidador de que requiere atención, además de brindarle recordatorios programados con actividades o medicinas que deba tomar con opción de que todos los recordatorios también los reciba el cuidador en su dispositivo móvil permitiendo, tanto al cuidador como al paciente, tener una ayuda digital de gran precisión.

Gracias a la etnografía enfocada que se realiza en pacientes reales que sufren alzheimer en etapa inicial podemos observar el comportamiento de los mismos con la aplicación ejecutándose en dispositivos reales, lo que nos permite diferenciar entre lo que nosotros como actores externos a la enfermedad pensamos que es útil para ellos frente al comportamiento de los pacientes necesitan y expresan satisfacción al poder utilizarla, ya que no solamente son ellos los que se benefician de este software sino sus cuidadores y familiares, los cuales también se incluyen dentro de la etnografía.

En este trabajo podemos concluir que la aplicación móvil es de gran ayuda para todos los personajes involucrados en esta etapa inicial del alzheimer, ya que facilita el tratamiento de la enfermedad al brindarles una ayuda de fácil manejo y con la satisfacción de saber que todas las características principales de la aplicación se ejecutan de manera instantánea, brindando tranquilidad para quienes cuidan a los pacientes ya que reciben todo tipo de alerta o notificación programada dentro de la misma de manera inmediata.

ABSTRACT

This work is aimed at solving the problems faced by people in the early stages of Alzheimer's by means of a mobile assistant, which will enable them to perform their daily activities independently without the need for a caregiver. Through the design and development of this mobile application for iOS devices, we integrate functions by which the patient can let know the caregiver that they need their help. Additionally, this app will remind them of scheduled activities or medicines that need to be taken. There is an option that all reminders are also sent to the caregiver's mobile device, allowing both the caregiver and the patient to have a highly accurate digital support.

UNIVERSIDAD DEL
AZUAY
Dpto. Idiomas

Translated by,
Lic. Lourdes Crespo

Introducción.

Una de las razones principales que motivó el desarrollo de esta tesis es poder compartir los conocimientos obtenidos a lo largo de la carrera universitaria en un campo en el que la tecnología y la medicina se juntan, denominado e-health – Medicina informática – en la cuál, con el uso de la tecnología se facilita el tratamiento de enfermedades de todo tipo, facilitando procesos tanto a pacientes como a profesionales de la salud, manejando información de la manera más precisa y reduciendo tiempos de espera; es por esta razón que hemos visto necesario el diseño y desarrollo de una aplicación para dispositivos móviles que permita a todos los actores involucrados en la enfermedad del alzheimer, facilitar el proceso evolutivo de esta enfermedad degenerativa que no tiene cura hasta el momento.

Gracias a la metodología de desarrollo de software basado en prototipos se diseña y desarrolla la aplicación móvil, obteniendo versiones de la misma que nos permite comprobar el funcionamiento de sus características principales con cada iteración que desarrollamos, sin necesariamente incluir toda la lógica o características del modelo terminado. Esta metodología nos permite evaluar de forma temprana la aplicación además que nos permite que los usuarios finales de la misma compartan su experiencia de uso a través de la etnografía enfocada que se realiza al final de esta tesis, en base a la técnica de observación participante, mediante la cual podremos descubrir cómo los pacientes utilizan la aplicación en su vida cotidiana. (Salgado)

CAPITULO 1.

El alzheimer.

Un trastorno neurológico que crece cada vez más en el siglo XXI, a la cuál se atribuyen millones de muertes anuales a nivel mundial, sin una cura definitiva descubierta hasta el momento; es el tema principal de esta tesis, la cuál busca ayudar a las personas involucradas en el circulo familiar, tanto del paciente como de sus cuidadores, brindando una ayuda parcial para quienes sufren de esta enfermedad en su etapa inicial a través de una aplicación móvil que permita, por un lado, a lo pacientes realizar sus actividades diarias de una forma normal; y por otro lado, a su grupo familiar y cuidadores contar con un medio a través del cuál podrán ayudar en cierta forma, de manera parcial a quien padece alzheimer.

Siendo una enfermedad que no tiene cura hasta el momento, lo que trataremos en este capítulo es explicar el desarrollo de la enfermedad, sus principales síntomas y las etapas por la que atraviesa quien la padece, además de los problemas que se presentan en cada etapa, que dependiendo del paciente pueden presentarse con mayor o menor intensidad y duración de tiempo; permitiendo conocer más a fondo los problemas a los que se enfrentan tanto pacientes, cuidadores y su grupo familiar.

1.1 Desarrollo de la enfermedad.

La demencia, se puede definirse como el desgaste de las capacidades cognitivas, causantes del entorpecimiento de la realización de las actividades diarias que las realizaría con normalidad una persona sana. Un déficit de memoria pronunciado, es

uno de los síntomas del alzheimer, provocando una demencia progresiva en el paciente, desencadenando en problemas emocionales, de lenguaje y perceptivos a medida que la enfermedad avanza.

Esta enfermedad crece de manera alarmante tanto en países desarrollados como del tercer mundo, convirtiéndose en un problema de carácter social grave para muchas familias a nivel mundial. Uno de los mayores factores que preocupa acerca de esta enfermedad es su naturaleza irreversible y la no existencia de un tratamiento médico curativo, además de representar un estrés y un carga emocional para los familiares del afectado.

Dentro de la población anciana, el alzheimer es el tipo de demencia más conocida. Con la edad de cada individuo, las posibilidades de padecer alzheimer se incrementan. La causa de la enfermedad aún es desconocida, según la edad del paciente a la que se empiezan a visualizar síntomas se puede clasificar la enfermedad de la siguiente manera:

Inicio precoz: antes de los 65 años.

Inicio tardío: después de los 65 años.

Cada una de estas categorías puede tener dos subcategorías:

Familiar: si existen parientes con la enfermedad.

Eventual: si no tiene antecedentes de familiares con la enfermedad.

Se calcula que en el mundo hay 22 millones de personas que la sufren y que en tres décadas habrá el doble. Según la Asociación de Alzheimer Internacional, la enfermedad puede comenzar a una edad tan temprana como los 50 años, no tiene cura conocida (Eisendrath S).

1.2 Etapas de la enfermedad.

El alzheimer se divide principalmente en 3 etapas:

Inicial: Se caracteriza por una sintomatología leve que permite al paciente desempeñarse en las tareas cotidianas normales, requiriendo asistencia solamente para actividades complejas. El paciente puede experimentar pérdida de la memoria en mínima cantidad además de cambios de humor, siendo lento para aprender y reaccionar. En ocasiones evitan a la gente nueva y prefieren el ambiente familiar.

Intermedia: En esta etapa el paciente requiere de un cuidador para realizar actividades cotidianas, convirtiéndose claramente en discapacitada. Los problemas de memoria le impiden comprender la situación actual, fecha y hora. En el ambiente familiar, tienen problemas para reconocer a sus seres cercanos. La capacidad de habla, lectura y escritura se ven limitados. Claramente, un paciente en esta etapa no puede estar solo. Su nivel de depresión puede incrementarse convirtiéndolo en depresivo, irritable y aislado.

Avanzada: Esta es la fase final, el paciente ha perdido la capacidad de alimentarse por su cuenta, reconocer a familiares, lugar de residencia y controlar sus funciones corporales, en definitiva es completamente dependiente de un cuidador para realizar

cualquier tipo de tarea. En esta etapa, el paciente puede sufrir de otro tipo de enfermedades y problemas respiratorios.

1.3 Conclusiones.

Uno de los mayores retos para la medicina moderna es determinar las posibles causas del Alzheimer, así como encontrar una cura para esta enfermedad degenerativa que es una de las que mas afecta a millones de personas a nivel mundial.

La tecnología juega un papel muy importante en este aspecto, ya que según las etapas del alzheimer que vimos anteriormente, para pacientes en etapa inicial, es claramente factible el apoyo mediante un asistente digital, que le ayudará en tareas básicas como recordatorios de actividades por realizar a determinadas horas, alertas de medicamentos de debe tomar y que dosis debe hacerlo, localización en tiempo real y ubicación tiempo/espacio de forma inmediata, siendo de gran ayuda tanto para él como para su cuidador y grupo familiar.

CAPITULO 2.

Sistema Operativo Móvil iOS.

El sistema operativo para dispositivos móviles propiedad de Apple Inc. se denomina iOS, lanzado en el año 2007 con la primera generación del iPhone, aunque su nombre oficial se dio a conocer en el 2008 con el lanzamiento del SDK para desarrolladores.

iOS es un sistema operativo derivado de Mac OS X, el sistema operativo para computadores de Apple Inc, basado en Darwind BSD, siendo su núcleo UNIX. Es un sistema de código cerrado, es decir, que su código fuente no esta disponible al público en general. iOS se ejecuta en variedad de dispositivos de la marca Apple, entre ellos se encuentran: iPhone, iPad, iPod, Apple TV y recientemente Apple Watch.

En este capítulo cubriremos temas como las principales características del desarrollo de aplicaciones para iOS, entre las cuales se encuentran: lenguaje de programación, herramientas para desarrolladores y una breve descripción de su SDK.

2.1 Acerca del desarrollo iOS

El desarrollo de aplicaciones iOS se remonta al año 2008, fecha en la que Apple Inc.

liberó sus herramientas de desarrollo para programadores a nivel mundial. Su lenguaje de programación se llama Objective-C, un lenguaje orientado a objetos basado en C que básicamente funciona enviando mensajes a instancias de objetos.

La interfaz gráfica iOS se diseño para aprovechar la pantalla táctil de los dispositivos, la cual permite gestos multitouch, utilizando controles como: sliders, botones, interruptores, cuadros de texto, etc; con una respuesta fluida y rápida al tacto de los usuarios. Además de estas características, se puede hacer uso de acelerómetros, gps, cámara de fotos y grabación de vídeo en alta resolución.

Las tiendas de aplicaciones móviles, en especial el App Store, la tienda oficial de Apple ha crecido exponencialmente desde su fecha de lanzamiento en 2008, hasta llegar a tener un aproximado de más de 500 millones de aplicaciones disponibles para todos los dispositivos soportados por Apple Inc. como son iPhone, iPad, iPod, Apple TV y Apple Watch. Este crecimiento en cuanto al desarrollo de aplicaciones a nivel mundial también ha servido para desarrollar aplicaciones en la categoría e-health, que buscan ayudar a sus usuarios con algún problema de salud, a superar sus problemas con el uso de dispositivos móviles.

Centros de salud a nivel mundial han visto en las aplicaciones una oportunidad para brindar un mejor servicio tanto a sus pacientes como a sus empleados, con aplicaciones de diversos tipos como: Lectores de rayos X, fichas médicas digitales, diagnóstico a distancia a través de fotografías, video llamadas, incluso con el envío de signos vitales a través de accesorios adicionales que se pueden conectar a los denominados iDevices, que son todos los productos móviles que ofrece Apple Inc.

Gracias a todas estas facilidades que ofrecen actualmente los dispositivos móviles, ha permitido incursionar cada vez más en el área médica con aplicaciones móviles que serán de gran ayuda tanto para quienes sufren alguna enfermedad como para los profesionales de la salud, que con la ayuda de internet, pueden ofrecer sus servicios de una manera rápida y efectiva con la movilidad que dichos dispositivos permiten.

Figura 1. Aplicación para medir la frecuencia cardiaca, ejecutándose en iPhone y iPad. Fuente: AirStrip Technologies, LLC

2.2 Interfaz de desarrollo Xcode.

El entorno de desarrollo integrado IDE se denomina Xcode en la cual se pueden diseñar gráficamente las pantallas de las aplicaciones móviles utilizando todos los controles disponibles para los diferentes dispositivos soportados por iOS. Xcode se introdujo en el 2003 conjuntamente con la versión 10.3 del sistema operativo de escritorio Mac OS X denominada Panther, a través del cual se desarrollaba interfaces para software dedicado solamente a computadores de escritorio y portátiles.

Figura 2: Icono de la aplicación Xcode en su versión 6.2. Fuente: propia.

Con la primera versión del SDK iOS para desarrolladores se ajustó Xcode para el desarrollo de aplicaciones móviles. Puede compilar una variedad de código de lenguajes como C, C++, Objective C, Apple Script, Cocoa, Carbón y Java. A partir de la versión 4.2 de Xcode, se incluyó el uso de Storyboards, una herramienta de diseño de interfaces gráficas en las que el concepto de diseño de basa en un tablero

de dibujo, donde el programador arrastra los componentes a las ventanas de la aplicación, representado de forma real le diseño final de la aplicación.

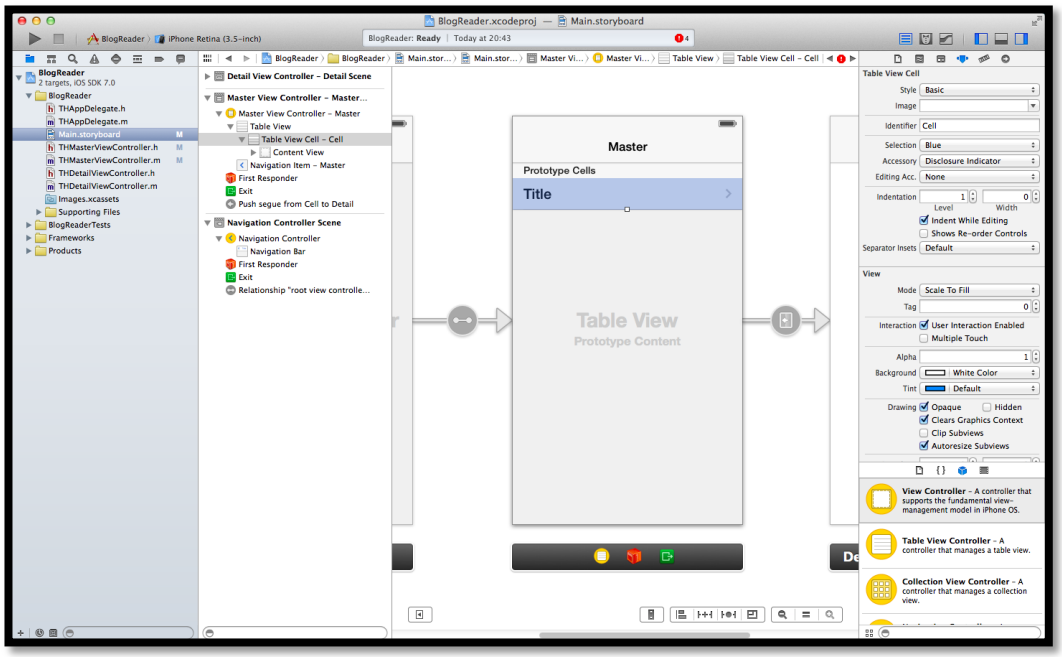


Figura 3. Desarrollo de una aplicación iOS en Xcode usando Storyboards. Fuente: propia.

Una de las características atractivas de Xcode frente a otros IDE para móviles es justamente la capacidad de utilizar Storyboards para diseñar las interfaces gráficas, siendo una de los puntos más fuertes de la aplicación, además del manejo automático de memoria y, desde la versión 6, permite la creación de Storyboards Universales que se ejecutan en todos los dispositivos soportados por iOS, permitiendo a los programadores la creación de un solo proyecto que se ejecutará en varias pantallas sin necesidad de re-escribir código fuente.

Uno de los requisitos principales para utilizar Xcode y desarrollar aplicaciones iOS, al momento de escribir esta tesis, es contar con un equipo Mac, ya que dicho

software esta disponible únicamente para equipos que ejecuten Mac OS X. Existen alternativas para poder instalar Mac OS X en equipos con diferentes sistemas operativos, a través de máquinas virtuales, pero en las pruebas personales que hemos efectuado, el rendimiento de Xcode no es estable, generando varios problemas a nivel de rendimiento del equipo como aparición de errores de ejecución del software; razón por la que recomendamos ejecutar Xcode en un equipo Mac.

2.3 Conclusiones.

Desde el lanzamiento de iOS en 2008, han sido miles las aplicaciones que se han propagado a nivel mundial, incursionando cada vez más en el ámbito de la salud, con aplicaciones de diversos tipos que facilitan y agilizan el accionar del cuerpo médico tanto en tratamiento como diagnóstico de enfermedades de todo tipo, siendo iOS uno de los sistemas operativos para móviles pioneros en este aspecto, incorporando en cada nueva versión más herramientas de desarrollo para solventar cada vez más los requerimientos en el campo de la salud.

El término e-health, también referido como e-Salud, es un término nuevo, que se utiliza desde hace pocos años en la medicina moderna; hace referencia a la práctica médica apoyada en el uso de tecnologías de la información y comunicación TIC, comprendiendo servicios entre la medicina y los servicios sanitarios asistidos tecnológicamente. Dentro de este concepto de salud tecnológica es donde se ve de manera muy útil el uso de iOS para la generación de aplicaciones que sirven como herramientas entre los distintos actores que intervienen en la enfermedad del alzheimer, con el objetivo de mejorar la calidad de vida de quienes padecen de la enfermedad como de quienes velan por el cuidado de los mismos.

CAPITULO 3

Memoria Técnica.

Una de las partes mas importantes del proyecto, se considera la creación de una memoria técnica, la cuál nos permitirá dar una descripción de la ejecución técnica del mismo, explicando de manera detallada, que hemos desarrollado y como lo hemos hecho, permitiendo dar a conocer desde el planteamiento del problema inicial, hasta la consecución, prueba y evaluación de la aplicación en un entorno real con pacientes que sufren de Alzheimer en etapa inicial.

Como parte de esta tesis, recopilaremos toda la creación de la aplicación móvil usando el lenguaje de programación Objective C, conjuntamente con Xcode, en el cuál crearemos la interfaz gráfica de la misma y desarrollaremos la lógica de funcionamiento que nos permita localizar al paciente con el uso del gps del dispositivo, crear recordatorios sobre actividades o medicinas que debe tomar el paciente, además de notificar al cuidador cuando exceda el perímetro establecido entre otras características.

3.1 Creación del proyecto en Xcode

Para este proyecto, utilizamos Xcode 6.2, siendo la última versión disponible de manera gratuita en el Mac App Store. (figura 4)

Figura 4. Versión 6.2 de Xcode. Fuente: propia.

Al iniciar Xcode, visualizaremos la pantalla de bienvenida, en la que podremos observar en la parte izquierda la versión del software, junto con las siguientes opciones: Get started with a playground, Create a new Xcode Project, Check out an existing Project; seleccionamos la opción Create a new Xcode Project. (figura 5)

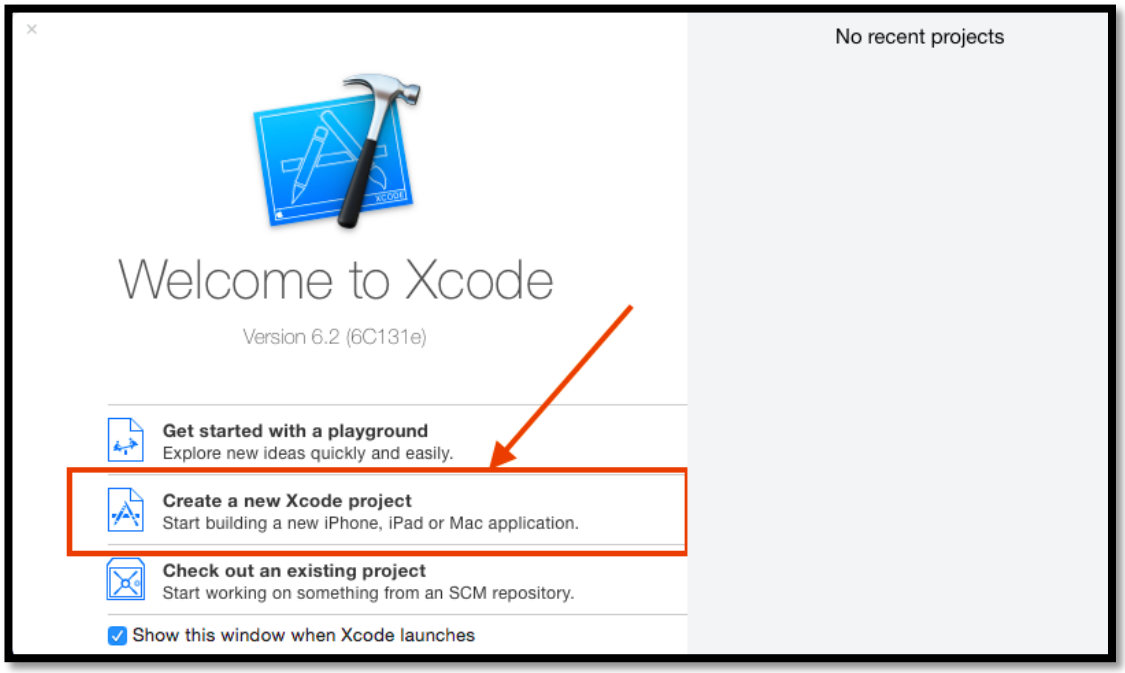


Figura 5. Pantalla de Bienvenida en Xcode. Opción "Create a new Xcode projet". Fuente: propia.

La siguiente ventana que visualizaremos nos permite seleccionar el tipo de proyecto que deseamos crear, tanto para iOS como OS X. Cada una de estas categorías tiene diferentes tipos de proyectos como son: Application, Framework & Library, System Plug-in, Others. La categoría que nos interesa para este proyecto es "Application", dentro de esta podemos visualizar cinco tipos de aplicaciones disponibles: Master-Detail Application, Paged-Bassed Application, Single-View Application, Tabbed Application y Games. Seleccionamos "Single-View Application", la cual Xcode brinda la siguiente descripción "This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a storyboard or nib file that contains the view", y damos click en el botón Next. (figura 6)

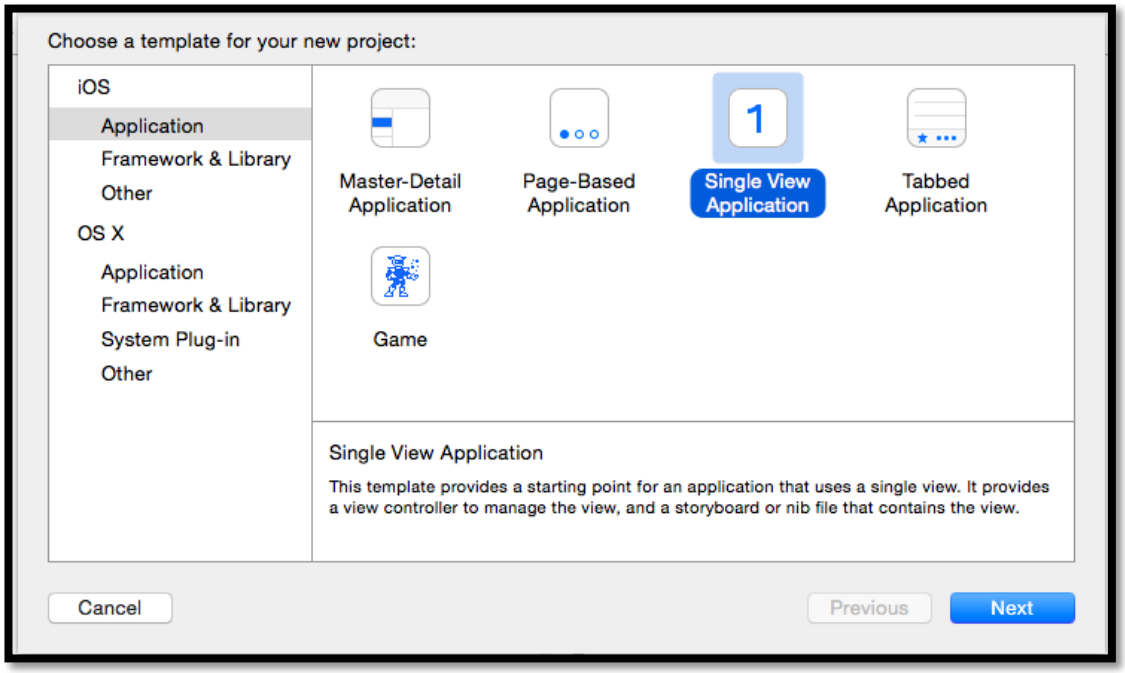


Figura 6. Tipos de aplicaciones disponibles para iOS. Fuente: propia.

La siguiente ventana que visualizamos es las de opciones del proyecto, en la cual podremos ingresar la siguiente información: Product Name, Organization Name, Organization Identifier, Bundle identifier, Language, Devices, User Core Data. (figura 7).

Producto name, hace referencia al nombre del proyecto básicamente, mediante el cual podremos identificarlo posteriormente.

Organization name, es un campo opcional, hace referencia al nombre de la institución que crea el proyecto.

Organization identifier, esta definido en notación de dominio inverso, sirve para identificar el proyecto posteriormente en el App Store y en la creación de certificados de desarrollo y producción en el App Store.

Bundle identifier, se conforma de campo Organization identifier más Product Name, sirve para tener un identificador único en el App Store.

Language, el lenguaje de programación a utilizar para el desarrollo de la aplicación.

Devices, el tipo de dispositivo para el que se va a desarrollar la aplicación, puede ser iPhone, iPad ó Universal. Una app universal puede ser ejecutada en los dos dispositivos, es decir, iPhone y iPad.

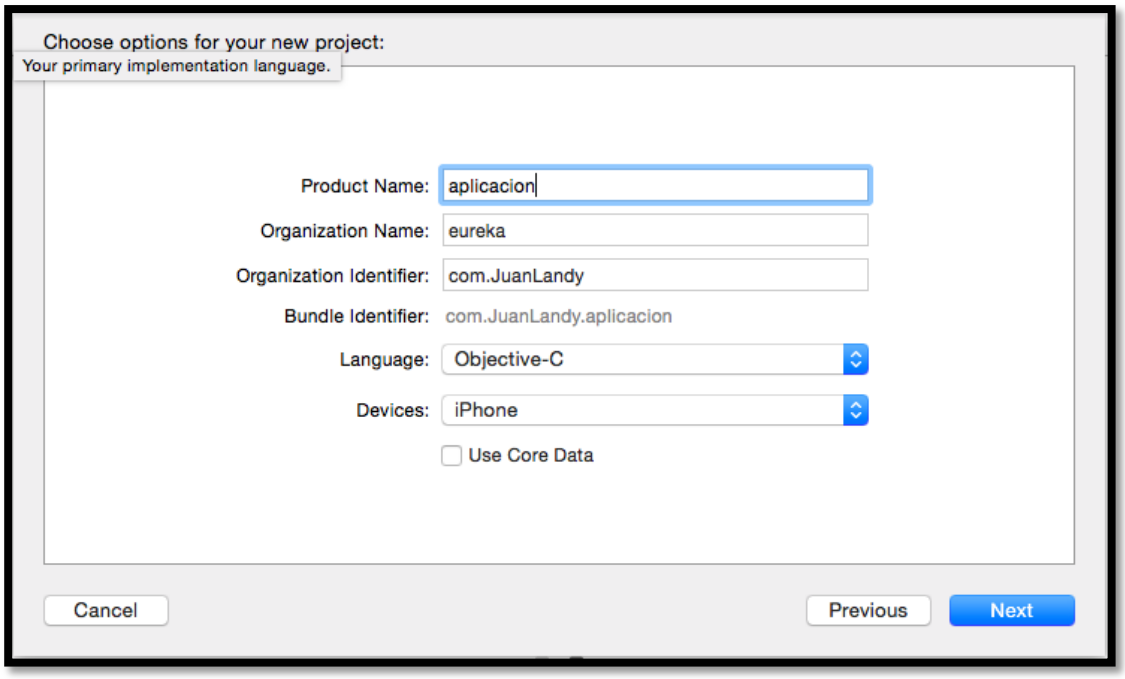


Figura 7. Ventana de opciones del proyecto en Xcode. Fuente: propia

Luego de ingresar todas las opciones del proyecto, damos clic en el botón Next y accedemos a la ventana en la cuál seleccionamos la ubicación en la que deseamos guardar el proyecto. Luego de guardar el proyecto, visualizaremos la interfaz de Xcode.

En la sección izquierda de Xcode, denominada Project Navigator, podemos visualizar todos los archivos que contiene nuestro proyecto como son: las clases que contendrán todo el código de la aplicación, frameworks, imágenes, storyboard, archivos de configuración, etc. Al seleccionar cualquier archivo de esta sección se visualizará su contenido en la parte central de Xcode. Al crear un nuevo proyecto, por defecto la cabecera del Project Navigator estará seleccionada y podremos visualizar información general del proyecto como el Bundle identifier, versión de la app, Deployment target, Devices, Device orientation, etc (figura 8)

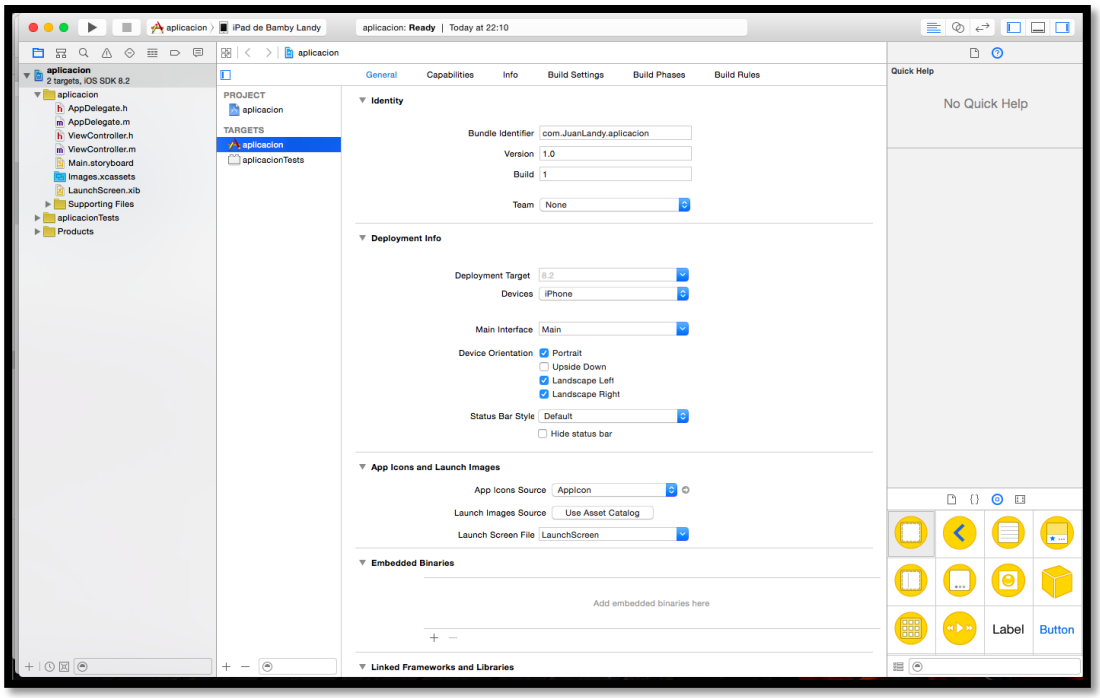


Figura 8: Pantalla de Xcode, con sus configuraciones iniciales al crear un proyecto nuevo. Fuente: propia.

3.2 Uso de storyboards

Mediante el uso de storyboards podemos diseñar gráficamente la interfaz de las aplicaciones iOS a través de vistas, transiciones entre vistas y controles usados para interactuar entre las diferentes vistas. Una vista representa una pantalla que se visualizará en el dispositivo, generalmente en iPhone se visualiza una sola vista a la vez, pero en iPad y OS X se pueden visualizar varias a la vez ya que la interfaz en este tipo de dispositivos puede contener más de una vista. Una transición representa el paso de una vista a otra, la figura 9 muestra un storyboard genérico al crear un proyecto nuevo en Xcode del tipo Master-Detail Application.

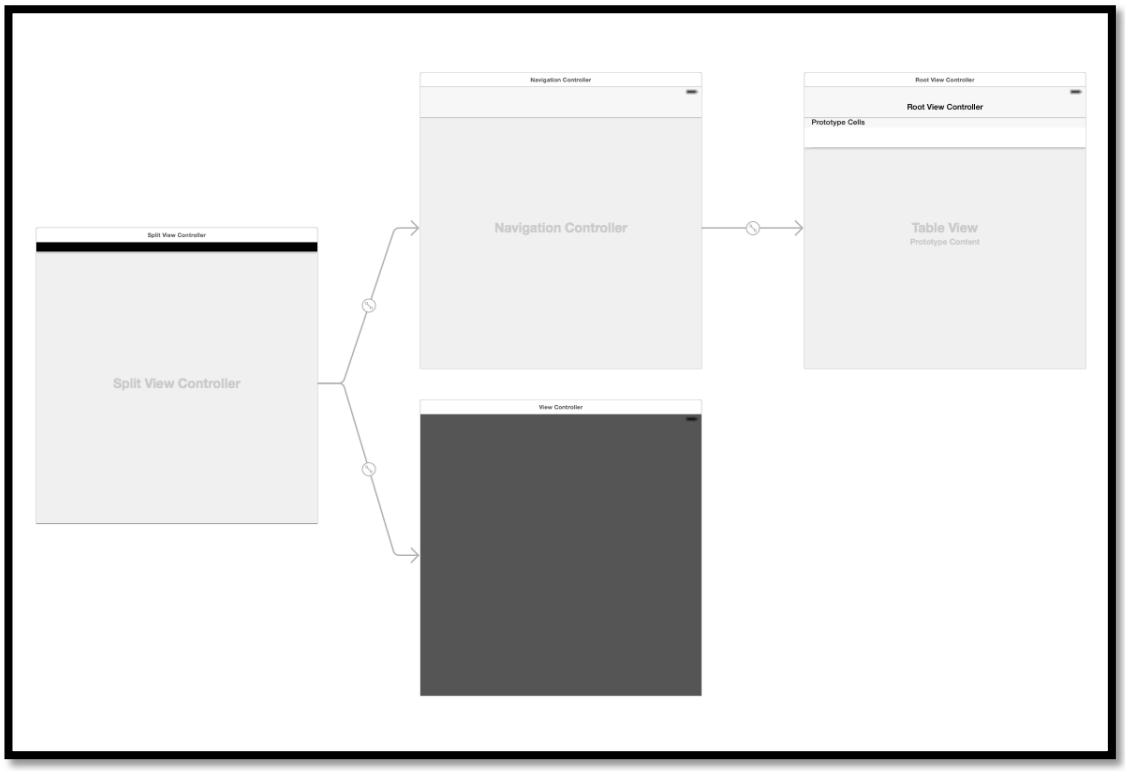


Figura 9. Storyboard con proyecto del tipo Master-Detail Application. Fuente: propia

El storyboard de la figura 9 contiene 4 vistas y 3 transiciones, la primera vista desde la izquierda se denomina Split View Controller, es la encargada de contener a las vistas que se encuentran en la parte derecha del storyboard.

Para trabajar con storyboards es necesario agregar mas vistas y controles que interactúen entre las diferentes vistas, los mismos que se pueden agregar desde la librería de objetos situada en la parte derecha de Xcode. Dentro de esta librería podemos filtrar los objetos que necesitamos por nombre o simplemente los arrastramos al storyboard, agregándolos automáticamente. (figura 10)

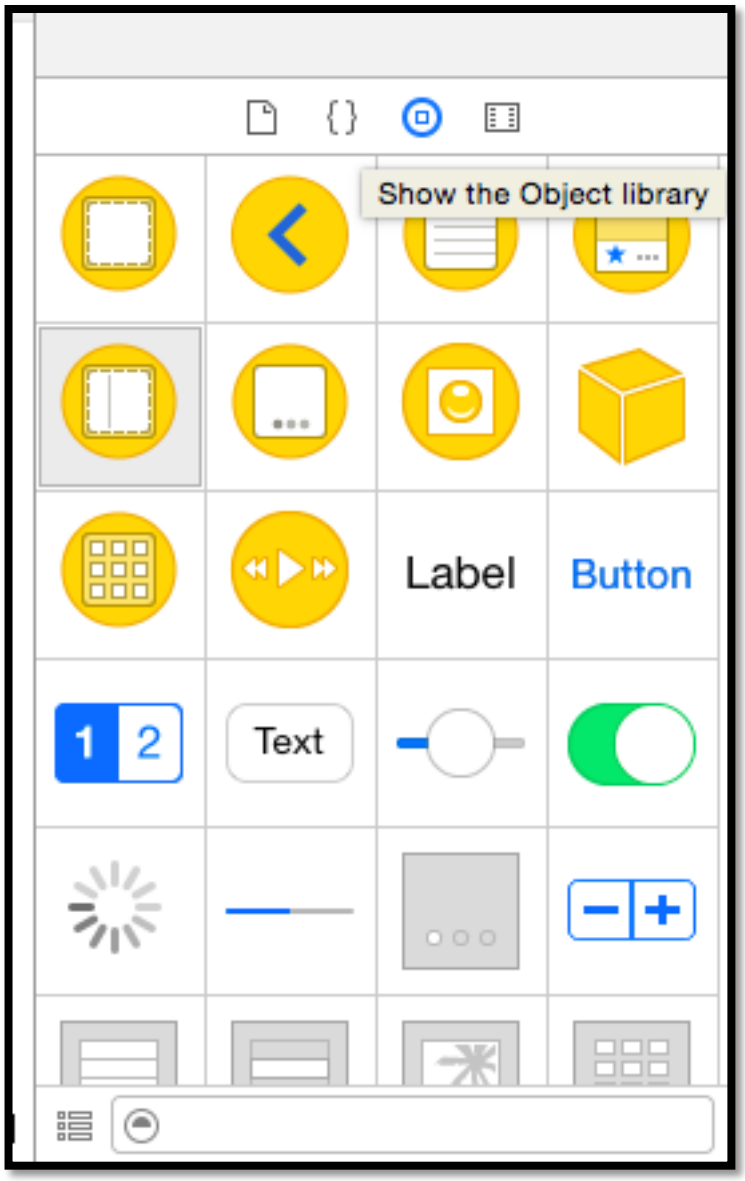


Figura 10. Librería de Objetos de Xcode. Fuente: propia

Como creamos el proyecto en el capítulo 3.1, nuestro storyboard está basado en un vista de tipo Single View Application, sobre la cual podemos empezar a agregar más vistas, transiciones entre vistas y controles. Básicamente la aplicación que vamos a desarrollar constará de las siguientes pantallas: Login, Menú Principal, Cuidador, Paciente, Definir perímetro, Recordatorios y Rastreo del dispositivo.

La vista inicial que se creó con el proyecto en Xcode es nuestra vista inicial, la cuál por defecto tiene una flecha en color gris apuntando hacia ella (figura 10.1), esta flecha nos indica que al iniciar la aplicación dicha vista será la primera en visualizarse en el dispositivo, razón por la cuál esta vista inicial será nuestro menú principal desde la que se podrá acceder a las diferentes ventanas de la aplicación.

A continuación detallaremos la creación de cada una de ellas en Xcode a través de storyboards.

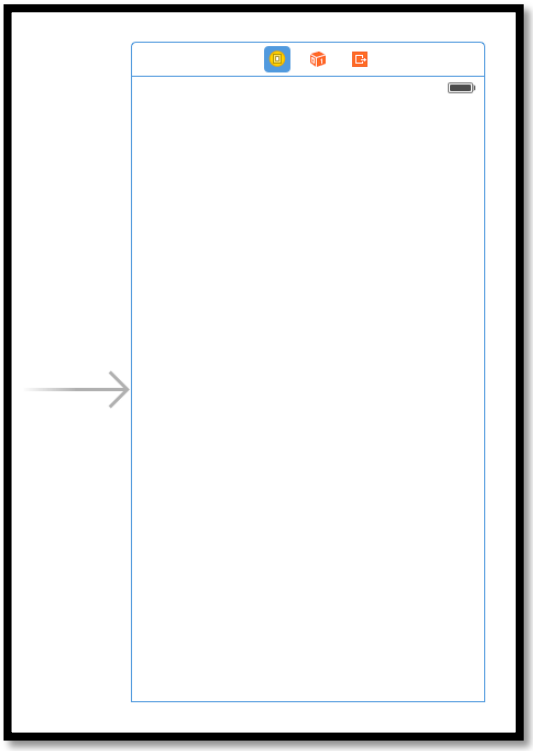


Figura 10.1 Vista Inicial en Xcode. Fuente: propia

3.2.1 Login

La pantalla de login debe permitir al cuidador registrarse con su correo electrónico y una contraseña de mínimo 5 caracteres; el correo electrónico con el que se registre servirá para enviar las notificaciones desde la aplicación. Para diseñar esta pantalla, agregamos una nueva vista dentro de nuestro Storyboard, para esto, vamos a la librería de objetos ubicada en la derecha del Xcode y buscamos el objeto denominado UIViewController (figura 11), lo seleccionamos y arrastramos hasta el storyboard, al integrarlo podemos visualizar una nueva vista vacía sin ningún contenido (figura 12) sobre la cuál vamos a agregar los siguientes objetos, de la misma manera, desde la librería de objetos: dos UITextField, un UILabel y un UIButton.

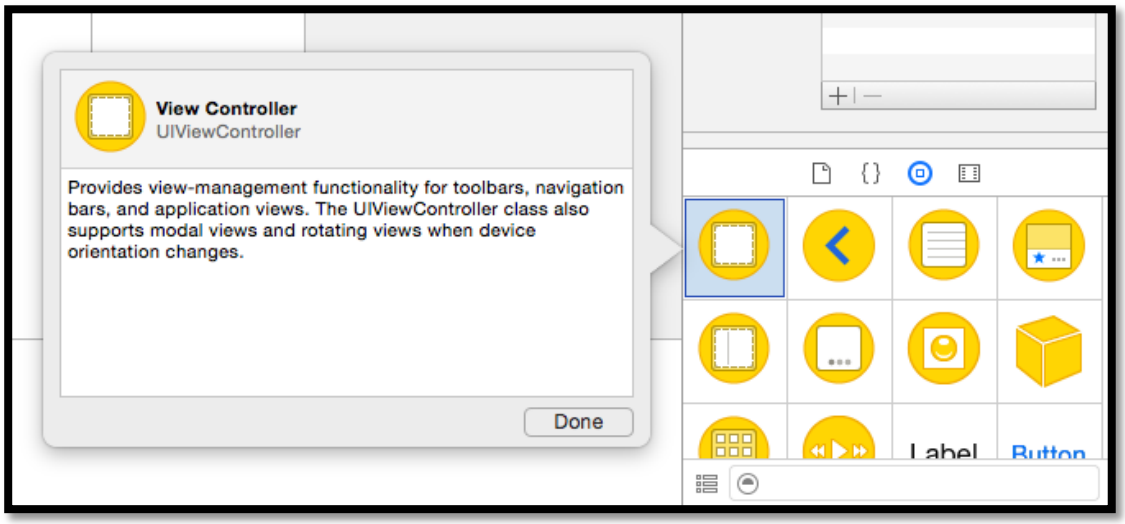


Figura 11. UIViewControler, Lista de Objetos de Xcode. Fuente: propia.

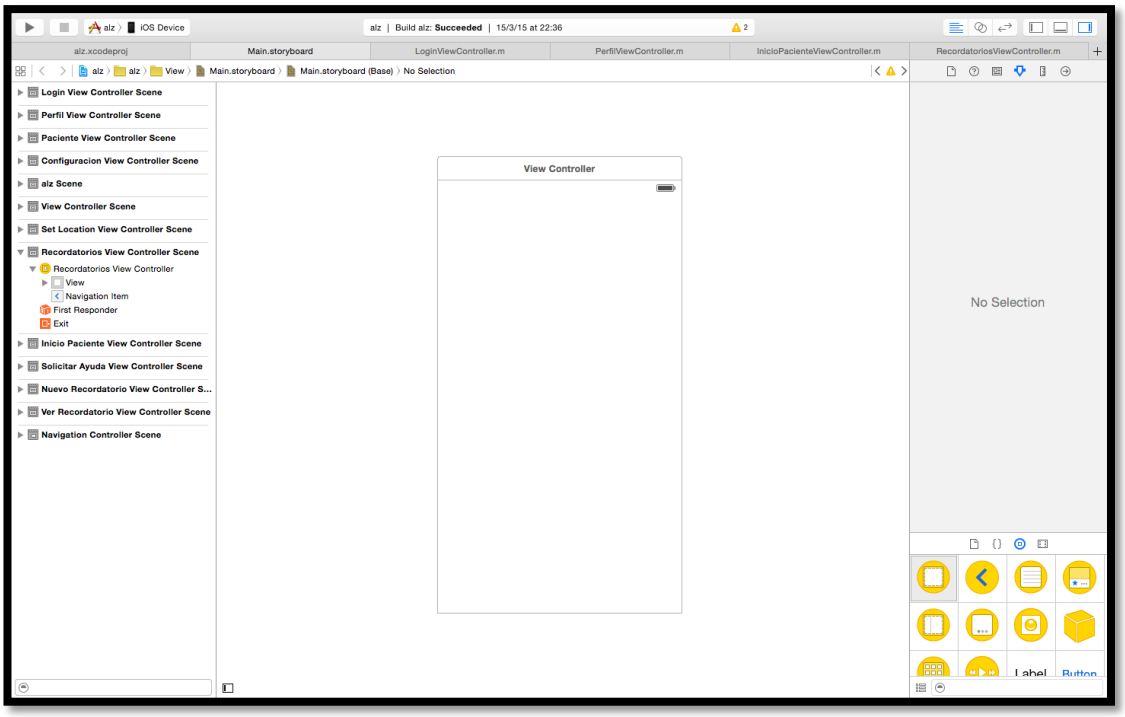


Figura 12. UIViewController nuevo en Xcode. Fuente: propia

Los UITextField son campos de texto que permitirán al usuario ingresar su correo electrónico y su contraseña. El UILabel, es una etiqueta de texto, en la cual ingresaremos un mensaje de bienvenida a la pantalla de Login. A través del UIButton, un botón rectangular, el usuario podrá ingresar a la aplicación presionándolo. En la figura 13 podemos visualizar la pantalla de Login una vez agregados todos los objetos que necesitamos. En la Tabla 1 tenemos un listado de todos los controles utilizados en esta pantalla y las propiedades que hemos utilizado de cada uno de sus objetos.

Figura 13. Pantalla Login en Xcode. Fuente: propia

Control/Propiedad	Title	Type	Placeholder
UILabel Bienvenido	Bienvenido		
UITextField Correo Electrónico		E-mail Address	Correo electrónico
UITextField contraseña		Security Type	Ingrese al menos 5 caracteres
UIButton Ingresar	Ingresar	Custom	

Tabla 1. Lista de componentes utilizados en la pantalla Login. Fuente: propia

Todos los controles que necesitan interactuar con el usuario deben conectarse con Xcode mediante outlets, los cuales nos permiten acceder a sus propiedades y métodos vía código en el lenguaje Objective C, para esto debemos agregar una clase al proyecto de Xcode la cual enlazaremos a la vista creada anteriormente.

Para agregar una nueva clase seleccionamos las siguientes opciones en el menú de Xcode: File, New File, a continuación visualizamos la ventana de Xcode donde podemos seleccionar el tipo de archivo que deseamos agregar al proyecto, en la parte izquierda seleccionamos el tipo Source dentro de iOS y podemos visualizar en la parte derecha algunas variedades de archivos de tipo Source disponibles, de las cuales vamos a escoger Cocoa Touch Class (figura 13.1)

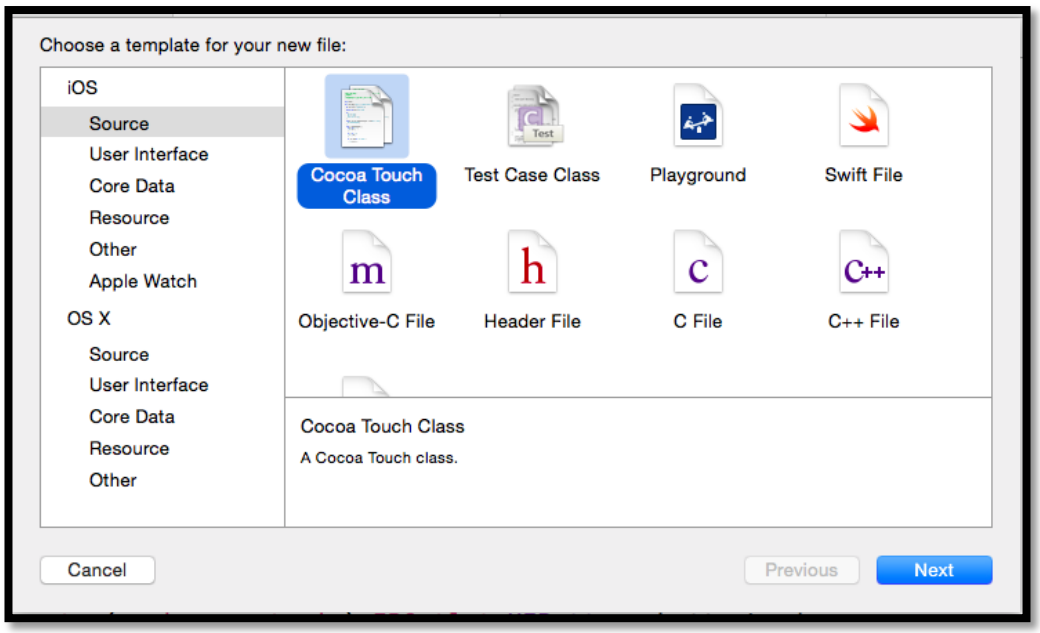


Figura 13.1 Selección de clase tipo Cocoa Touch en Xcode. Fuente: propia

Una vez seleccionado el tipo de archivo a agregar damos click en Next donde debemos ingresar el nombre del archivo conjuntamente con el tipo de clase que deseamos crear, para esta clase ingresamos el nombre LoginViewController de subclase UIViewController perteneciente al lenguaje Objective C y damos click en Next (figura 13.2) donde visualizaremos la ubicación donde almacenar el archivo, por defecto Xcode nos muestra la carpeta donde se encuentra el proyecto sobre el cual estamos trabajando, damos click en guardar.

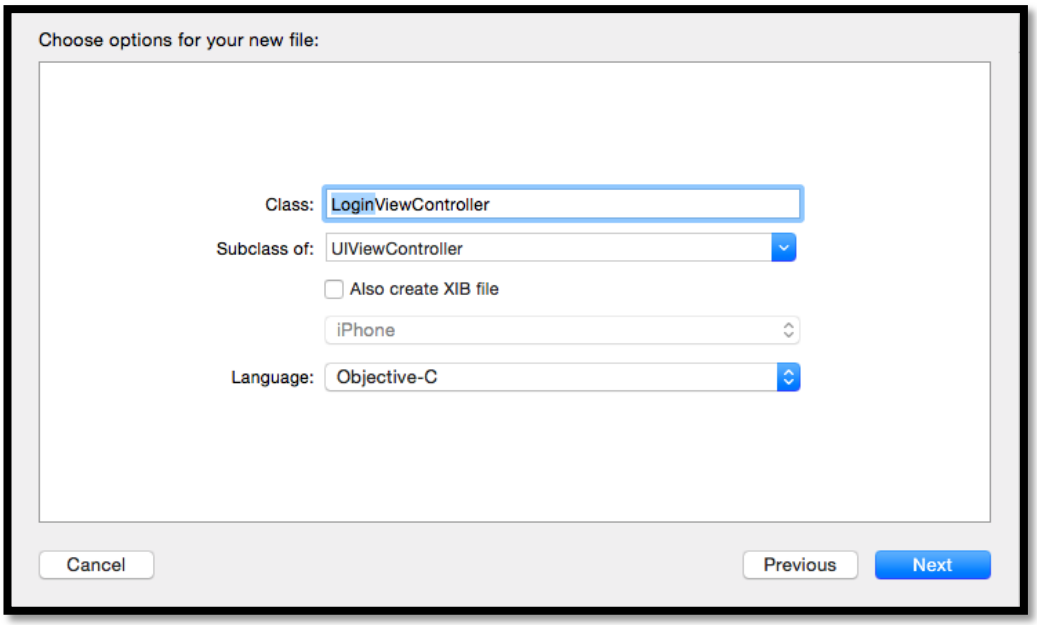


Figura 13.2 Creación del archivo LoginViewController de tipo UIViewController en Xcode. Fuente: propia

El archivo creado aparecerá en el Navegador del Proyecto en forma de dos archivos LoginViewController.h y LoginViewController.m; la cabecera de la clase es el archivo .h y el cuerpo de la clase el archivo .m; para enlazar la clase creada con la vista Login seleccionamos la vista en el storyboard y en la parte derecha de Xcode seleccionamos Show Identity Inspector, bajo la categoría Class seleccionamos de la lista desplegable la clase LoginViewController (figura 13.3)

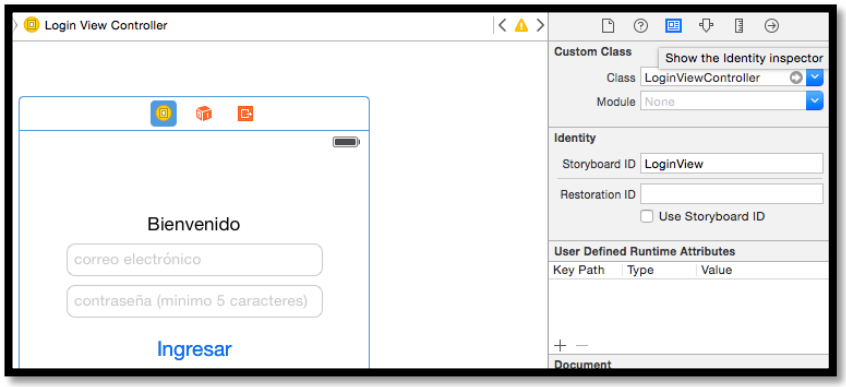


Figura 13.3 Enlazar Vista con su respectiva clase desde el storyboard. Fuente: propia

Una vez enlazada la vista con la respectiva clase procedemos a enlazar los controles de la vista con la clase creada, para esto, desde el storyboard seleccionamos el UITextField en el cual ingresaremos el correo electrónico y en la esquina superior derecha de Xcode seleccionamos la opción Show the Assitant Editor, la cual dividirá la pantalla de Xcode y podremos visualizar conjuntamente tanto la vista Login en el Storyboard como la clase perteneciente a dicha vista. Figura 13.4

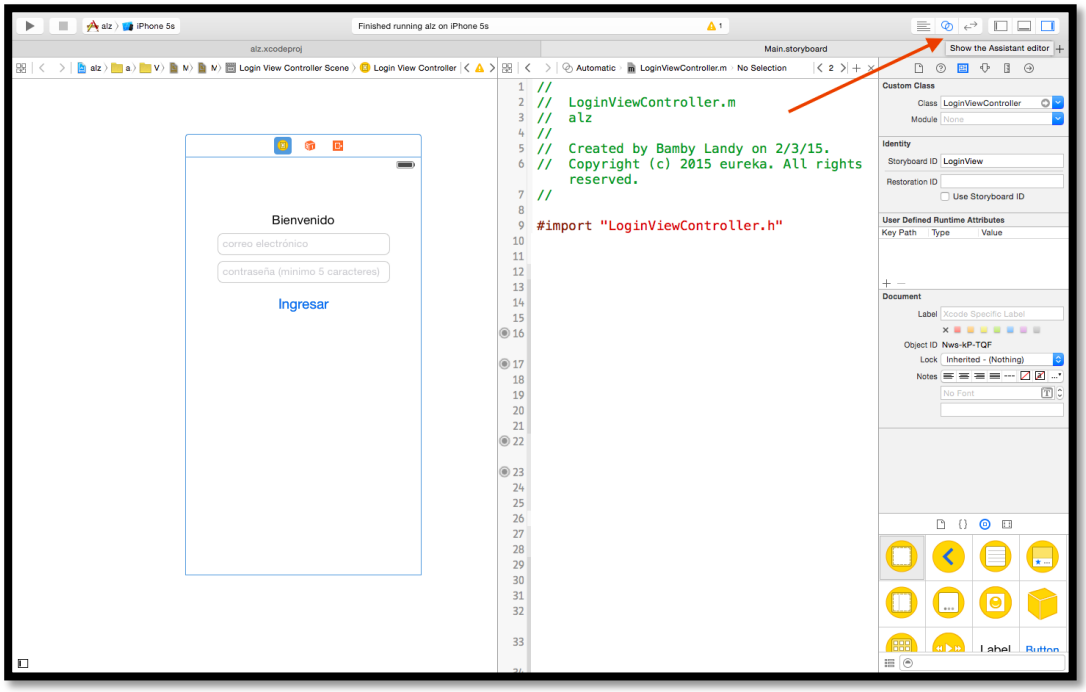


Figura 13.4 Opción Show Assitant Editor en Xcode. Fuente: propia

Nuevamente seleccionamos el UITextField del correo electrónico y al mismo tiempo presionamos la tecla “control” y arrastramos desde el control UITextField hacia la clase LoginViewController, podemos visualizar que una línea conecta el control con la clase (Figura 13.5).

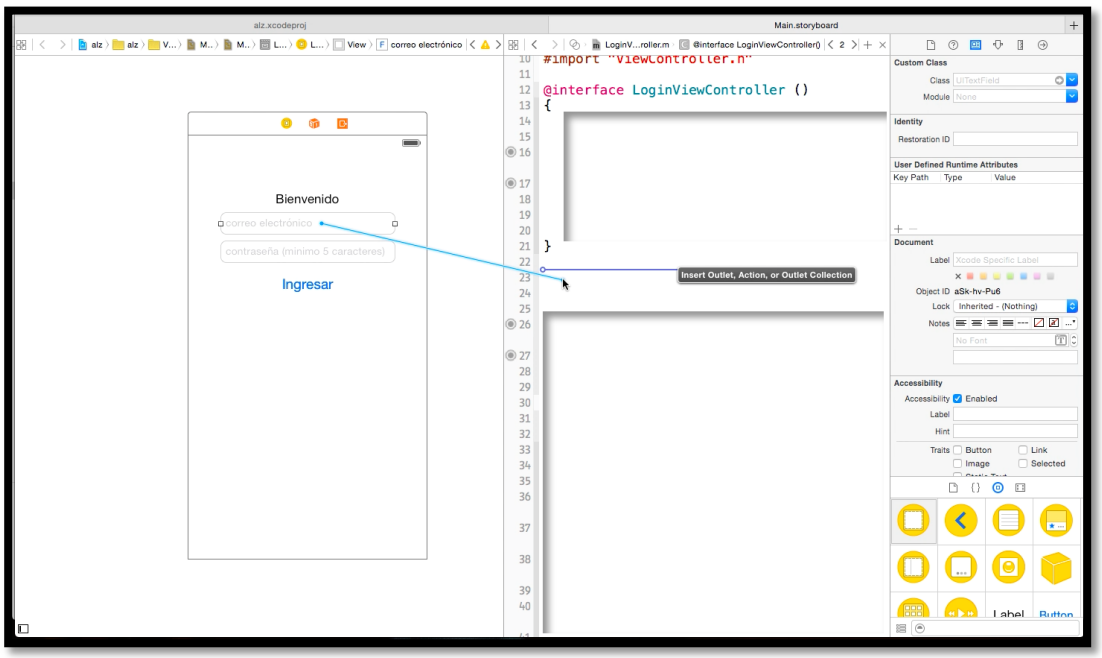


Figura 13.5. Creando conexión entre el control UITextField y la clase LoginViewController en Xcode. Fuente: propia

Soltamos la tecla en un espacio en blanco dentro de la clase y Xcode nos muestra la ventana con las características de dicha conexión (figura 13.6), dependiendo del tipo de control podemos establecer conexiones tipo Outlet o tipo Action. Las conexiones tipo Outlet nos permiten interactuar con las propiedades del objeto, es decir, podremos acceder vía código a propiedades como: titulo, color, posición en el storyboard, etc; mientras que las conexiones tipo Action nos permiten acceder a los métodos que el usuario puede realizar sobre dicho objeto, en el caso del UITextField que tenemos en la vista podemos acceder al método Did End On Exit, el cual nos permite controlar cuando el usuario sale del UITextField.

Para el UITextField de nuestra vista debemos crear los dos tipos de conexiones, tanto Outlet como Action. En la ventana que visualizamos con las características de la conexión seleccionamos Connection de tipo Outlet y damos el nombre txtCorreo a

nuestro control y click en Connect. Repetimos el procedimiento de arrastrar presionando la tecla control hacia la clase LoginViewControler para crear una conexión tipo Action, con el nombre textFieldReturn y event DidEndOnExit. Las dos conexiones realizadas las podemos visualizar en la Clase LoginViewController. Figura 13.7

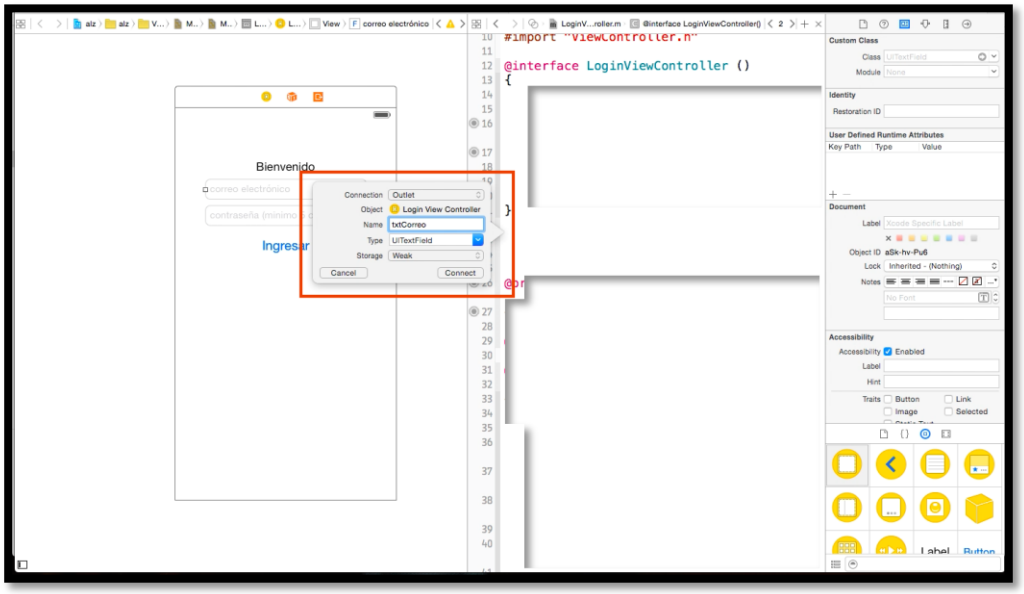


Figura 13.6 Conexión entre UITextField con su respectiva clase en Xcode. Fuente propia

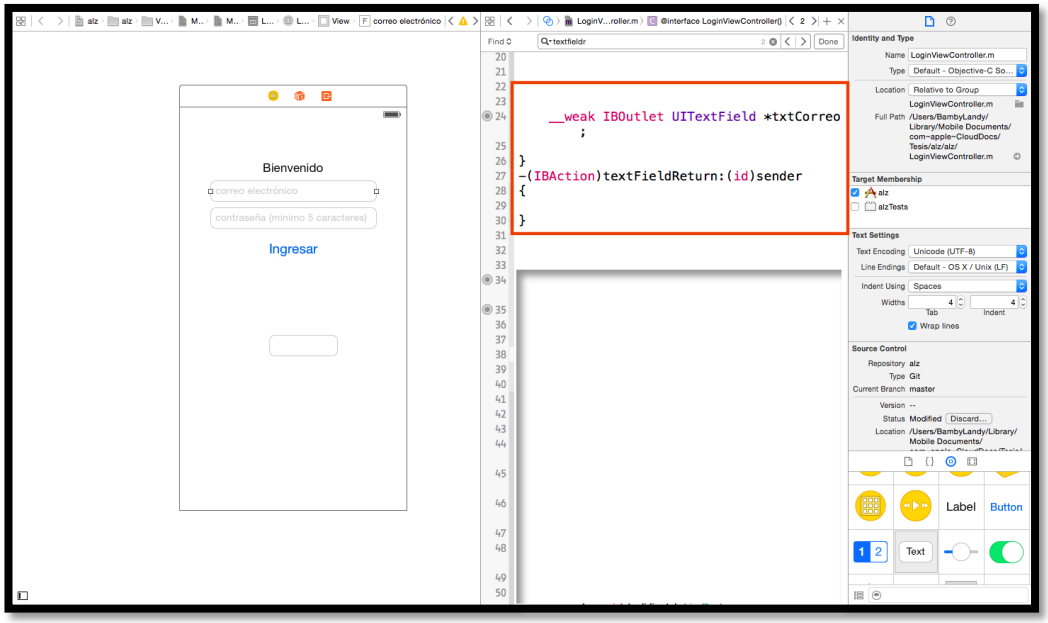


Figura 13.7 Creación de conexiones para UITextField en Xcode. Fuente: propia

Repetimos los pasos para creación de Outlet y Action para el UITextField del Password. Para el UIButton que permite al usuario ingresar al menú principal de igual manera debemos crear un Outlet y Action con un tipo de evento Touch Up Inside mediante el cual podemos validar los datos ingresados.

Los datos del usuario, tanto el mail como la contraseña los almacenaremos en un archivo de tipo Property List dentro de la aplicación, al cual podemos acceder de manera inmediata para obtener la información guardada dentro de este. Para agregar un archivo de tipo Property List seleccionamos dentro de Xcode el menu: File, New, File; dentro de la categoría iOS, seleccionamos Resource y Property List. (Figura 13.8); le damos el nombre Perfil.plist y lo almacenamos dentro de la carpeta del proyecto.

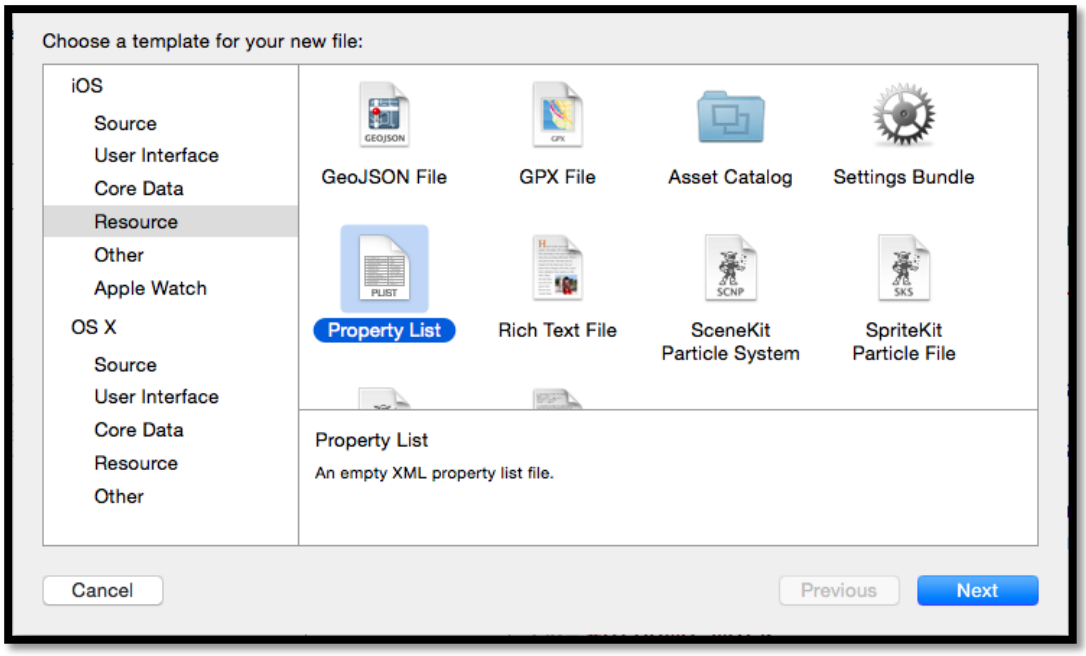


Figura 13.8 Creación de archivo Property List en Xcode. Fuente: propia

Con el archivo de tipo plist creado debemos leer sus contenido con el método que se puede visualizar en la figura 13.9 que lo agregamos en el archivo LoginViewController.m, este método nos devuelve la url con la ubicación del archivo plist, el cual lo almacenamos en una variable de tipo NSString con el nombre pathPlist.

```
-(NSString *)dataFilePath{

    NSError *error;

    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES); //1
    NSString *documentsDirectory = [paths objectAtIndex:0]; //2
    NSString *path = [documentsDirectory stringByAppendingPathComponent:@"Perfil.plist"]; //3

    NSFileManager *fileManager = [NSFileManager defaultManager];

    if (![fileManager fileExistsAtPath: path]) //si es q no existe - 4
    {
        NSString *bundle = [[NSBundle mainBundle] pathForResource:@"Perfil" ofType:@"plist"]; //5

        [fileManager copyItemAtPath:bundle toPath: path error:&error]; //6
        if (error) {
            NSLog(@"error copyItemAtPath %@", error);
        }
    }
    return path;
    /*
     1) Create a list of paths.
     2) Get a path to your documents directory from the list.
     3) Create a full file path.
     4) Check if file exists.
     5) Get a path to your plist created before in bundle directory (by Xcode).
     6) Copy this plist to your documents directory.

     */
}
```

Figura 13.9 Lectura de archivo plist en Xcode. Fuente: propia

Una vez leído el archivo plist podemos almacenar sus valores en variables de tipo NSString para el caso del mail y de la contraseña. Accedemos a dichos valores a través de una variable de tipo NSDictionary, la cual almacena toda la estructura del archivo plist con sus respectivas llaves, que para los campos que necesitamos serán “correo” y “clave”, la asignación de los valores se puede observar en la figura 13.10

```
dicData = [[NSMutableDictionary alloc]initWithContentsOfFile:pathPlist];

if (([dicData objectForKey:@"correo"] != nil) && ([dicData objectForKey:@"clave"] != nil))
{
    correo =[dicData objectForKey:@"correo"];
    clave = [dicData objectForKey:@"clave"];

    txtCorreo.text = correo;
}
```

Figura 13.10. Asignación de valores desde plist hacia variables. Fuente: propia

Dentro del método Login del UIButton validamos los datos ingresados para dar acceso al menú principal, con los valores leídos desde el plist validamos si son correctos para garantizar el acceso, a través del código del a figura 13.11 podemos validar los datos para posteriormente al menú principal.

```
- (IBAction)Login:(UIButton *)sender {
    if (txtClave.hasText && txtCorreo.hasText) {

        if (([dicData objectForKey:@"correo"] != nil) && ([dicData objectForKey:@"clave"] != nil))
        {
            //1. si ya existe el usuario buscar mail y clave en plist
            if ([correo isEqualToString:txtCorreo.text] && [clave isEqualToString:txtClave.text]) {
                NSLog(@"Login valido");
                [[NSUserDefaults standardUserDefaults] setBool:YES forKey:@"login" ];
                [[NSUserDefaults standardUserDefaults] synchronize];

                [[self presentingViewController]dismissViewControllerAnimated:YES completion:nil];

            }
            else {
                UIAlertView *validacion = [[UIAlertView alloc]initWithTitle:@"Login" message:@"Correo o clave mal ingresados" delegate:self cancelButtonTitle:@"OK"
                                           otherButtonTitles:nil, nil];
                [validacion show];
            }
        }
```

En la figura 13.11 podemos observar el método presentingViewController nos pemite desaparecer la vista Login y presentar al usuario la pantalla del menú principal.

3.2.2 Menú Principal

El menú principal, al ser la pantalla inicial de la aplicación brindará acceso directo a las siete opciones mas importes de la aplicación. Los accesos directos los visualizará el usuario a través de iconos intuitivos, brindando una idea clara y sencilla de lo que realiza cada uno de ellos, para ello, agregamos una nueva vista a nuestro storyboard conjuntamente con 7 botones de tipo UIButton desde la librería de Objetos hacia dentro de nuestra nueva vista. Al seleccionar uno de los UIButton agregados al storyboard podemos observar en la parte derecha de Xcode las propiedades del control seleccionado, dicha sección se denomina el inspector de atributos. Figura 14. Cabe mencionar que todas las pantallas de la aplicación estarán diseñadas para que solamente el Cuidador del paciente pueda ingresar. El Paciente tendrá acceso únicamente a una pantalla específicamente diseñada para él, la cuál se mostrará una vez que el Cuidador haya configurado la aplicación con los valores iniciales necesarios.

En el inspector de atributos “Attributes Inspector”, encontraremos propiedades para el UIButton que seleccionamos como: tipo de botón, estado inicial del botón, titulo, Fuente, color del texto, sombra del texto, imagen de fondo y muchas otras opciones que pueden aplicarse directamente desde esta sección y se aplicarán en tiempo real sobre el objeto en el storyboard del proyecto Xcode.

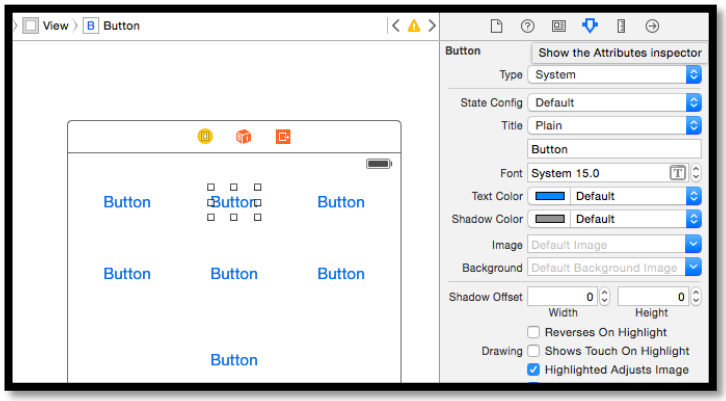


Figura 14. Inspector de Atributos. Fuente: propia

Para personalizar los botones de la pantalla inicial, haremos uso de la propiedad Image de este control, a la cual podemos acceder seleccionando el botón que necesitamos personalizar y dentro del Inspector de Atributos, en la propiedad Image seleccionamos la imagen deseada. Para hacer uso de imágenes en Xcode debemos agregarlas previamente al proyecto desde el menú File, Add files to Project. La pantalla de login, luego de personalizar cada uno de los botones que nos servirán como acceso directo a las diferentes opciones de la aplicación, la podemos visualizar en la figura 15.

Figura 15. Pantalla Login en Xcode. Fuente: propia.

En la Tabla 2 tenemos un listado de todos los controles utilizados en esta pantalla y las propiedades que hemos utilizado de cada uno de sus objetos.

Control/Propiedad	Type	Image	Title
UIButton Cuidador	Custom	Perfil.png	
UIButton Paciente	Custom	Paciente.png	
UIButton Definir Rango	Custom	Rango.png	
UIButton Recordatorios	Custom	Recordatorio.png	
UIButton Ayuda	Custom	Ayuda.png	
UIButton Configuración	Custom	Configuracion.png	
UIButton Iniciar	Custom		Iniciar
UILabel Cuidador			Cuidador
UILabel Paciente			Paciente
UILabel Definir Rango			Definir Rango
UILabel Recordatorios			Recordatorios
UILabel Ayuda			Ayuda
UILabel Configuración			Configuración

Tabla 2. Controles existentes en la pantalla Menu Principal. Fuente: propia.

A continuación seleccionamos nuestra vista del menú principal y seleccionamos en el menú la opción Editor, Embed In, Navigation Controller; lo cuál agregará una vista tipo Navigation View Controller a nuestra vista menú principal, lo que nos permitirá tener una jerarquía de vistas partiendo como vista padre desde el menú principal, así todas las vistas que se desprendan de ésta tendrán un navigation view controller que las permitirá regresar a su vista padre. Automáticamente la vista navigation view controller pasará a ser la vista inicial con la flecha gris apuntando hacia ella. (Figura 15.1)

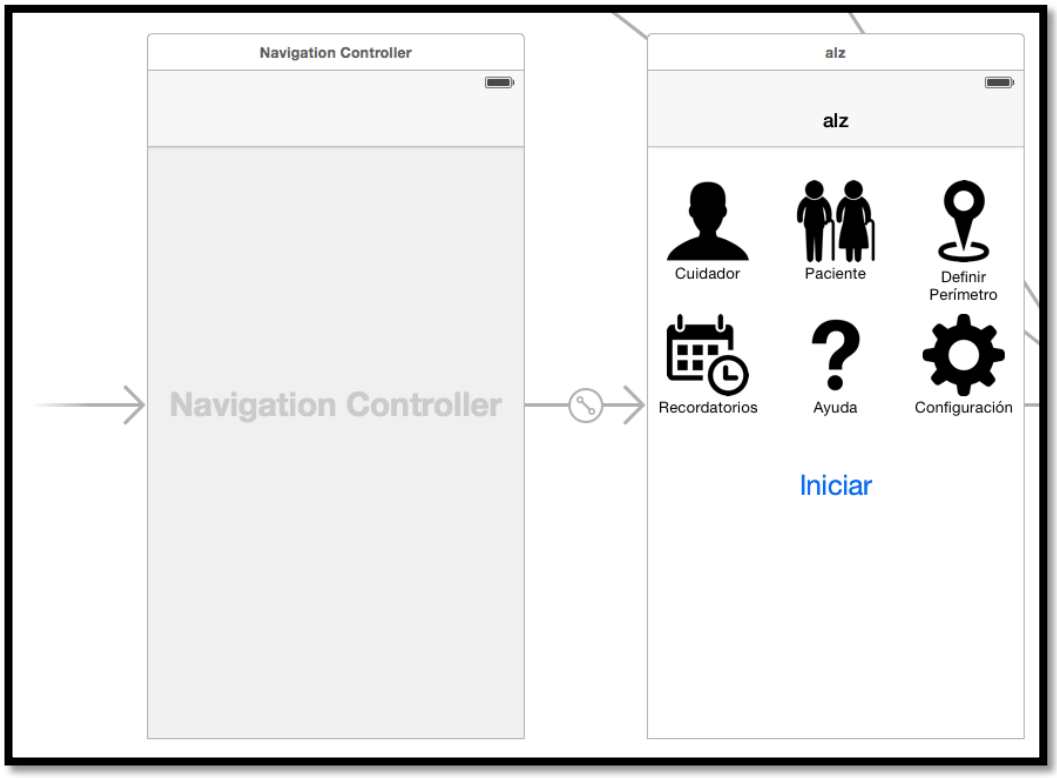


Figura 15.1 Agregar Navigation View Controller a la vista Menu Principal. Fuente: propia

3.2.3 Cuidador.

Es importante dedicar una pantalla al cuidador del paciente ya que en esta sección es donde va a ingresar sus datos personales, además de su parentesco con el paciente y su fotografía. Todos estos datos los podrá visualizar el paciente cuando solicite

ayuda remota desde la aplicación siendo de mucha ayuda para el visualizar la fotografía de su Cuidador, además de su nombre y su parentesco.

Agregamos una nueva vista al storyboard, arrastrando un control UIViewController desde la librería de objetos, esta ventana contará con los siguientes controles: UIImageView, UILabel, UITextField y UIPickerView; los mismos que nos permitirán visualizar la fotografía del cuidador, sus datos personales y el parentesco con el paciente. El cuidador podrá seleccionar su fotografía desde la galería de imágenes del dispositivo ó tomar una fotografía con la cámara (delantera o trasera) del dispositivo, la misma que se visualizará en el UIImageView ingresado en el Storyboard.

Los datos personales como Nombres, Apellidos, Parentesco y Género los ingresará desde los cuadros de texto UITextView además de la ayuda del control UIPickerView, el cual permite seleccionar un valor de una lista de posibilidades, los cuales se muestran al cuidador de forma gráfica representado por una rueda giratoria. En la figura 16 podemos observar el diseño final de la pantalla dedicada a la información del Cuidador. En la Tabla 3 tenemos la lista de componentes que forman parte de esta pantalla además de sus propiedades utilizadas.

Figura 16. Pantalla dedicada al Cuidador del paciente. Fuente: propia

Control/Propiedad	Title	Mode	User Interaction Enabled
UIImageView Perfil Cuidador		Aspect Fill	YES
UIlabel Nombre	Nombre completo		NO
UILabel Genero	Genero		YES
UILabel Parentesco	Parentesco		YES
UITextField	Nombres y apellidos		YES
UIPickerView			YES

Tabla 3. Controles existentes en la pantalla Cuidador. Fuente: propia.

Seleccionamos el botón “Cuidador” en la vista del menú principal en nuestro storyboard y presionando el botón control de nuestro computador hacemos click sobre dicho botón y arrastramos hacia la vista del Cuidador recién agregada, al soltar el click observaremos una ventana de opciones de conexión entre el botón y nuestra vista, seleccionamos la conexión de tipo “push” y nuestra vista se conectará automáticamente con el botón además de agregarse el navigation controller en la vista del Cuidador (figura 16.1), pasando a formar parte de una sub-vista del menú principal, de esta manera, cuando ejecutemos la aplicación y presionemos el botón “Cuidador” aparecerá automáticamente dicha vista con la opción de poder regresar a la vista menú principal.

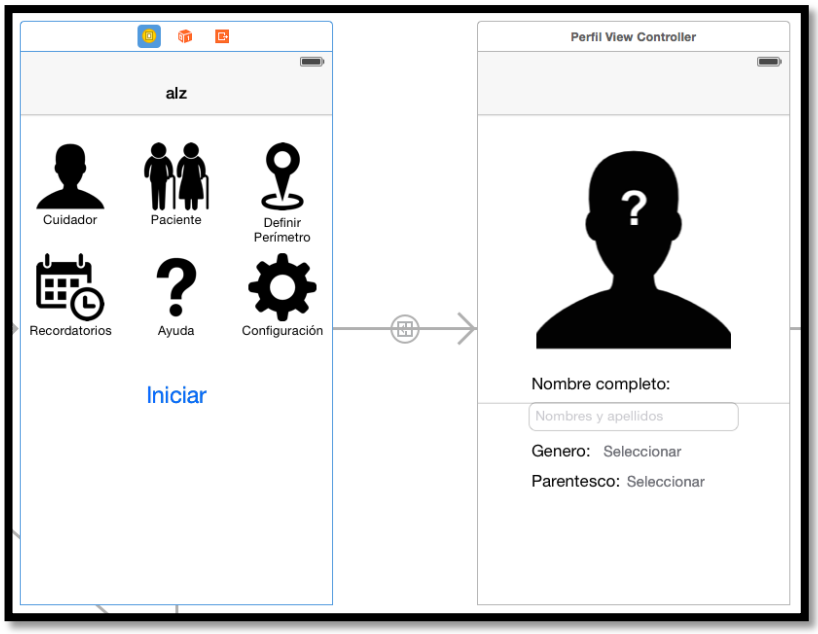


Figura 16.1 Conexión entre botón “Cuidador” del menú principal con la vista PerfilViewController. Fuente: propia

Toda la información registrada en esta vista la almacenaremos en el archivo Perfil.plist que creamos anteriormente dentro de la aplicación, de esta manera estarán disponibles de forma permanente durante toda la ejecución de la aplicación, de igual

manera, al reiniciar la aplicación los registros estarán disponibles, con la ventaja de poder editarlos fácilmente directamente desde la misma ventana del Cuidador.

Con todos los controles agregados en la vista del cuidador, creamos una nueva clase de tipo UIViewController con el nombre PerfilViewController, la cual agregamos a nuestra vista a través del Identity Inspector de Xcode y procedemos a crear las conexiones necesarias entre nuestros objetos y la vista del Cuidador; todas las conexiones necesarias se pueden visualizar en la figura 16.2.

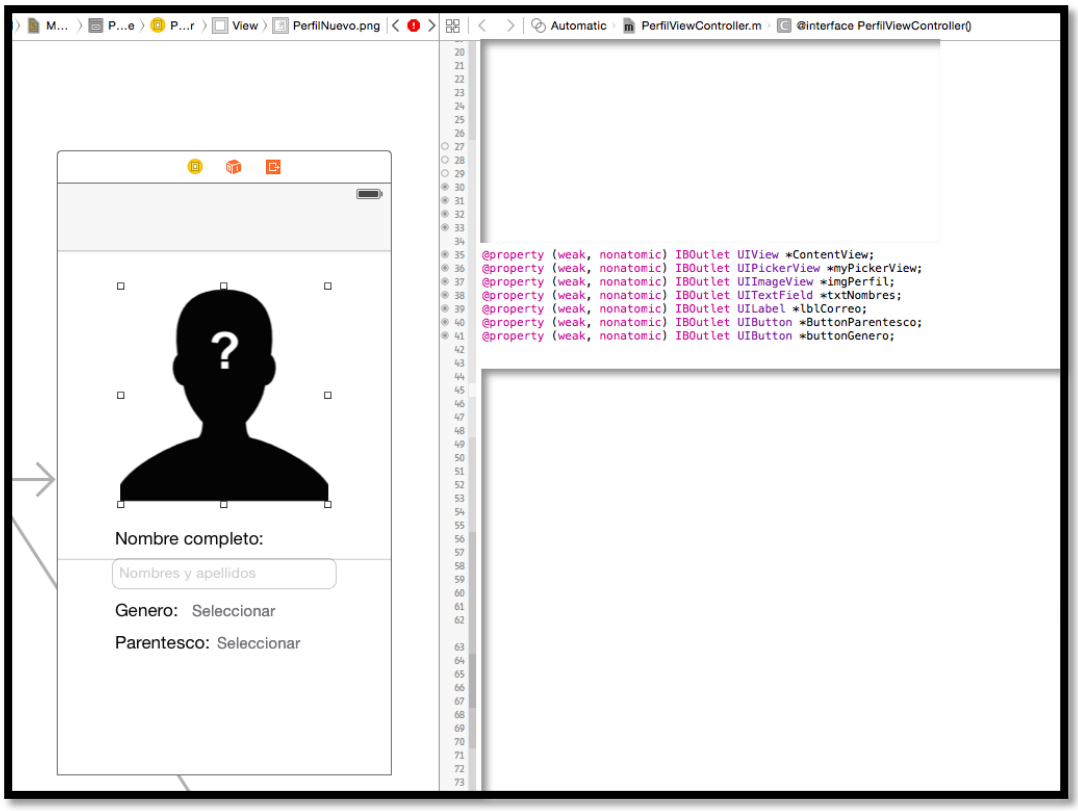


Figura 16.2 Conexiones entre controles y clase PerfilViewController. Fuente: propia

Adicionalmente debemos agregar un botón para poder guardar los datos ingresados en esta vista, el mismo que lo colocaremos en la cabecera de nuestra vista, es decir dentro del navigation view; el botón estará presente en la esquina superior derecha de

nuestra vista; dentro de la librería de objetos de Xcode buscamos un control de tipo UIBarButtonItem y lo arrastramos hasta la esquina superior derecha de nuestra vista; conectamos nuestro botón con la clase correspondiente a la vista con una conexión de tipo Action con el nombre Guardar.

Para guardar los datos en nuestro archivo Perfil.plist debemos registrar los datos previamente ingresados en una variable de tipo NSDictionary en la cuál podemos asignar cada valor a una llave específica. Este método lo podemos visualizar en la figura 16.2, en donde obtenemos los valores y los guardamos en Perfil.plist.

```
- (IBAction)GuardarPerfil:(UIBarButtonItem *)sender;
{
    NSLog(@"Guardar perfil");

    if ((self.txtNombres.hasText) && !([self.buttonGenero.titleLabel.text isEqualToString:@"Seleccionar"]) && !([self.ButtonParentesco.titleLabel.text
        isEqualToString:@"Seleccionar"])) {

        //1. copia datos en nuevo arreglo
        NewdicData = [[NSMutableDictionary alloc]init];
        [NewdicData setValue:self.txtNombres.text forKey:@"nombres"];
        [NewdicData setValue:self.ButtonParentesco.titleLabel.text forKey:@"parentesco"];
        [NewdicData setValue:self.buttonGenero.titleLabel.text forKey:@"genero"];
        [NewdicData setValue:correo forKey:@"correo"];
        [NewdicData setValue:[dicData objectForKey:@"clave"] forKey:@"clave"];

        //2. imagen
        NSData *data = UIImagePNGRepresentation(self.imgPerfil.image);

        [NewdicData setValue:data forKey:@"foto"];

        //
        if ([self writeDictionaryToPlist:NewdicData]) {
            //recordatorio - medicamento guardado
            HUD = [[MBProgressHUD alloc] initWithView:self.view];
            [self.view addSubview:HUD];
            HUD.delegate = self;

            [HUD showWhileExecuting:@selector(SaveSucess) onTarget:self withObject:nil animated:YES];

        }
        else
        {
            UIAlertView *ErrorGuardar = [[UIAlertView alloc]initWithTitle:@"Error" message:@"Se ha producido un error al intentar guadar sus datos" delegate:self
                cancelButtonTitle:@"Aceptar" otherButtonTitles:nil, nil];
            [ErrorGuardar show];
        }
    }

}
```

Figura 16.3 Almacenar datos en Perfil.plist. Fuente: propia

Una vez almacenados los datos se mostrará una alerta al Cuidador informado que los datos se han almacenado correctamente. Estos datos estarán almacenados de forma permanente en Perfil.plist y cada vez que el usuario ingrese nuevos valores se sobrescribirán sin ningún problema.

3.2.4 Paciente

A través de esta pantalla permitimos al Cuidador ingresar la fotografía y el nombre del Paciente, datos que se mostrarán al mismo al ejecutar la aplicación, ayudando a recordar sus datos en todo momento, ya que estarán disponibles de manera constante en la pantalla a la cuál tiene el acceso, ya que en la etapa inicial de la enfermedad, los pacientes tienden a olvidar este tipo de información.

Esta pantalla es similar a la pantalla dedicada para el Cuidador, se utilizarán los mismos controles, entre los que se encuentran: UIImageView para visualizar la fotografía del Paciente, además de UITextField en el cual ingresaremos los nombres y apellidos, también utilizamos un UILabel para indicar el texto que debe ingresar en el UITextField. El diseño de la vista es muy similar a la vista del Paciente como se puede observar en la figura 17.

Figura 17. Pantalla dedicada al Paciente. Fuente: propia

Los controles usados en la interfaz de la pantalla Paciente se encuentran listados en la Tabla 4.

Control/Propiedad	Title	Mode	User Interaction Enabled
UIImageView		Aspect Fill	YES
UITextField	Nombre del Paciente		YES
UILabel	Nombre:		NO

Tabla 4. Controles existentes en la pantalla Paciente. Fuente: propia.

De la misma manera, como lo hicimos en la vista Cuidador, enlazamos el icono correspondiente desde el menú principal hacia esta vista, para crear una conexión y que la vista Paciente sea una subvista del menú principal con un navigation controller que permita regresar de manera fácil hacia el menú principal. También agregamos una nueva clase al proyecto de tipo UIViewController con nombre PacienteViewController, la cual la enlazamos a nuestra vista para proceder a crear las conexiones necesarias con los siguientes controles: UIImageView y UITextField.

Como lo hicimos en la vista del Cuidador, agregamos un botón de tipo UIBarButtonItem en la parte superior derecha del navigation controller de esta vista, además de enlazarlo con una conexión de tipo Action en el evento Touch Up Inside, método que nos permitirá agregar el código necesario para almacenar los datos.

Para los datos del paciente creamos un archivo de tipo plist con el nombre Paciente.plist en el cual procederemos a almacenar la información ingresada en la vista respectiva; dentro del método del botón guardar ingresamos el código de la figura 17.1, el cual, de la misma manera que en el procedimiento para guardar los datos del Cuidador, registra los datos ingresados en una variable de tipo NSDictionary previo a almacernarlos en Paciente.plist; cada vez que el usuario ingrese nuevos valores y presione el botón "guardar" se sobre-escribirán los datos del archivo .plist.

```
-(void)Guardar
{
    NSLog(@"Guardar perfil");

    if (self.txtNombrePaciente.hasText) {

        //1. copia datos en nuevo arreglo
        NewdicData = [[NSMutableDictionary alloc]init];
        [NewdicData setValue:self.txtNombrePaciente.text forKey:@"nombres"];

        //2. imagen
        NSData *data = UIImagePNGRepresentation(self.imgPaciente.image);

        [NewdicData setValue:data forKey:@"foto"];

        //
        if ([self writeDictionaryToPlist:NewdicData]) {
            //recordatorio - medicamento guardado
            HUD = [[MBProgressHUD alloc] initWithView:self.view];
            [self.view addSubview:HUD];
            HUD.delegate = self;

            [HUD showWhileExecuting:@selector(SaveSucess) onTarget:self withObject:nil animated:YES];

        }
        else
        {
            UIAlertView *ErrorGuardar = [[UIAlertView alloc]initWithTitle:@"Error" message:@"Se ha producido un error al intentar guadar sus datos" delegate:self
                cancelButtonTitle:@"Aceptar" otherButtonTitles:nil, nil];
            [ErrorGuardar show];
        }
    }
}
```

Figura 17.1 Procedimiento para almacenar los datos en Paciente.plist. Fuente: propia

Una vez almacenada la información el usuario recibirá una notificación para informarle que el proceso ha sido exitoso, pudiendo de esta manera salir de la vista sin perder la información ingresada.

3.2.5 Definir Perímetro

Mediante los mapas que disponen los dispositivos iOS, podremos definir el rango de movimiento del paciente, dentro de un perímetro que el Cuidador deberá establecer. Para esta ventana hemos utilizado controles como UIMapView, UISlider, UILabel.

UIMapView es el control encargado de mostrar un mapa en tiempo real dentro de la ventana establecida, para esto hace uso de la librería MapKit. Figura 18.

Figura 18. Uso de mapas en iOS a través del contro UIMapView en Xcode. Fuente propia

La interfaz de esta ventana en particular, permite al Cuidador determinar su ubicación actual y crear un perímetro de movimiento permitido para el Paciente; este rango tendrá un mínimo de 100 metros hasta un máximo de 1000 mts, rango dentro del cuál el Paciente podrá movilizarse sin generar ningún tipo de alerta. Al sobrepasar dicho perímetro, la aplicación envía de forma automática una alerta vía

mail al correo electrónico del Cuidador, además de mostrar una alerta gráfica y sonora al Paciente informándole que ha excedido el perímetro permitido. Figura 18.

El control UISlider permite al Cuidador establecer el perímetro de una manera rápida y precisa, visualizando directamente en el UIMapView de forma gráfica el rango establecido además de permitirse acercar o alejar el mapa con un simple gesto denominado “pinch”, el cual viene integrado de forma nativa en el control UIMapView y consiste en pellizcar el mapa con dos dedos para alejar o acercar la vista según se desee.

El MapView es el control que ocupa la gran mayoría de la pantalla, de esta manera podemos permitir una visualización completa al Cuidador del perímetro que desea establecer. En la parte inferior de la pantalla se encuentra el UISlider, conjutamente con los controles UILabel, los cuales informan al Cuidador del valor exacto en metros establecido por el UISlider.

En la figura 19 podemos observar el diseño de la pantalla dedicada a registrar el perímetro de movimiento para el Paciente. Todos los controles utilizados en esta pantalla se encuentran listados en la Tabla 5, con sus propiedades que se han utilizado.

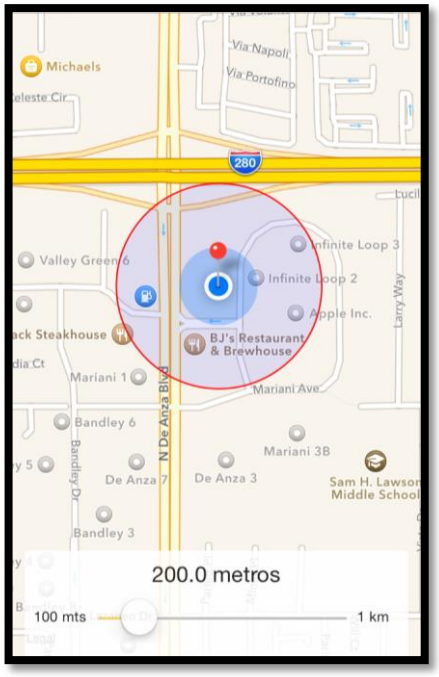


Figura 19. Pantalla Definir Perímetro diseñada en Xcode. Fuente: propia.

Control/Propiedad	Minimum – Maximum Value	Alpha	User Interaction enabled
UIMapView		1	YES
UIScrollView	100 – 1.000	1	YES
UILabel Valor minimo, Valor máximo, Valor actual		1	NO
UIView		0,85	NO

Tabla 5. Controles existentes en la pantalla Cuidador. Fuente: propia.

Para empezar a trabajar con mapas en Xcode debemos agregar dos frameworks que incorporan métodos necesarios que nos permiten utilizar todas las funcionalidades del dispositivo y sus mapas, estos frameworks son: MapKit y CoreLocation; para agregarlos nos ubicamos en la parte izquierda de Xcode y seleccionamos Project Navigator, luego dentro de Targets seleccionamos el proyecto en el que estamos trabajando, luego la pestaña Build Phases y dentro de ella desplegamos la opción Link Binary with Libraries, aquí agregamos los frameworks MapKit y CoreLocation como se puede observar en la figura 19.1.

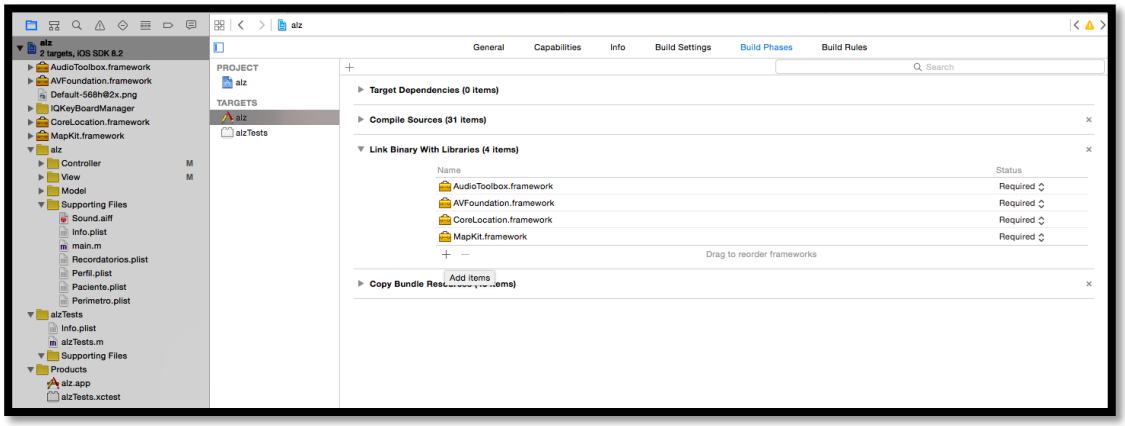


Figura 19.1 Agregamos los frameworks MapKit y CoreLocation en Xcode. Fuente: propia.

Con los frameworks incorporados al proyecto, creamos las conexiones entre todos los componentes agregados a la vista, además de agregar una nueva clase de tipo UIViewController con el nombre SetLocationViewController, la cual enlazaremos a la vista respectiva; además debemos conectar el botón respectivo del menú principal con la vista que hemos agregado, dando click en el botón “Definir Perímetro” del menú principal y presionando la tecla control de nuestro computador para arrastrar hacia la vista en la que estamos trabajando, al soltar el click, dentro de la ventana de opciones de conexión seleccionamos el tipo push, lo que enlazará las vistas.

Agregamos un UIBarButtionItem en el navigation controller, en la parte superior derecha, con el nombre “Guardar Perímetro” la cual le agregamos una conexión de tipo Action en el evento Touch Up Inside con el nombre “GuardarPerimetro”, método que nos permitirá guardar las coordenadas del perímetro establecido, dentro del cual debemos obtener las coordenadas de la ubicación actual del dispositivo para almacenarlo como se puede observar en la figura 19.2.

```
- (IBAction)GuardarPerimetro:(UIButton *)sender {

    NSNumber *latPin = [NSNumber numberWithDouble:myAnnotation.coordinate.latitude];
    NSString *latitudPin = [NSString stringWithFormat:@"%2@", latPin];

    NSNumber *longPin = [NSNumber numberWithDouble:myAnnotation.coordinate.longitude];
    NSString *longitudPin = [NSString stringWithFormat:@"%2@", longPin];

    //NSLog(@"perimetro: %d", [[NSString stringWithFormat:@"%1f", self.sliderDistancia.value]intValue]);
    NSString *perimetro = [NSString stringWithFormat:@"%d", [[NSString stringWithFormat:@"%1f", self.sliderDistancia.value]intValue]];

    dicData = [[NSMutableDictionary alloc]init];
    [dicData setValue:latitudPin forKey:@"latitud"];
    [dicData setValue:longitudPin forKey:@"longitud"];
    [dicData setValue:perimetro forKey:@"perimetro"];

    //
    if ([self writeDictionaryToPlist:dicData]) {
        //recordatorio - medicamento guardado
        HUD = [[MBProgressHUD alloc] initWithView:self.view];
        [self.view addSubview:HUD];
        HUD.delegate = self;

        [HUD showWhileExecuting:@selector(SaveSucess) onTarget:self withObject:nil animated:YES];

    }
    else
    {
        UIAlertView *ErrorGuardar = [[UIAlertView alloc]initWithTitle:@"Error" message:@"Se ha producido un error al intentar guadar sus datos" delegate:self
            cancelButtonTitle:@"Aceptar" otherButtonTitles:nil, nil];
        [ErrorGuardar show];
    }

}
```

Figura 19.2 Almacenamiento de coordenadas en Xcode. Fuente: propia.

Los datos se almacenarán en un archivo de tipo plist, el cual lo debemos crear y agregar al proyecto con el nombre Perimetro.plist permitiendo su lectura y escritura como se puede observar en el método WriteDictionaryToPlist de la figura 19.3.

El control UISlider que ingresamos nos permite establecer el perímetro de movimiento en un rango entre 100mts y 1km, para lo cual hemos configurado su valor mínimo como 100 y valor máximo como 1.000 en las propiedades del control dentro de Xcode. Adicionalmente agregamos una conexión de tipo Action en el evento Value Changed con nombre “ChangeDistance” dentro del cual configuramos el código que se puede visualizar en la figura 19.4

```
- (BOOL) writeDictionaryToPlist:(NSMutableDictionary *) plistDict
{
    NSError *error;

    NSDictionary *originalPlist = [NSDictionary dictionaryWithContentsOfFile:pathPlist];
    [originalPlist writeToFile:pathPlist atomically:YES];


    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES); //1
    NSString *documentsDirectory = [paths objectAtIndex:0]; //2
    NSString *path = [documentsDirectory stringByAppendingPathComponent:@"Perimetro.plist"]; //3
    //NSLog(@"path: %@",path);

    NSFileManager *fileManager = [NSFileManager defaultManager];

    if (![fileManager fileExistsAtPath: path]) //si es q no existe PlayList.plist - 4
    {
        NSString *bundle = [[NSBundle mainBundle] pathForResource:@"Perimetro" ofType:@"plist"]; //5

        [fileManager copyItemAtPath:bundle toPath: path error:&error]; //6
        return FALSE;
    }
    else
    {
        BOOL result = [plistDict writeToFile:path atomically:YES];

        return result;
    }

}
```

Figura 19.3 Método de escritura de datos en el archivo Perimetro.plist en Xcode. Fuente: propia

```
- (IBAction)ChangeDistance:(UISlider *)sender {
    lblDistancia.text = [[NSString stringWithFormat:@"%1.0f", sender.value] stringByAppendingString:@" mts"];

    NSString *perimetroactual = [NSString stringWithFormat:@"%1.0f", self.sliderDistancia.value];
    NSString *sub1 = [@"Perímetro actual (mts): " stringByAppendingString:perimetroactual];
    [myAnnotation setSubtitle:sub1];

    [mapView removeOverlay:circle];
    [self createCircle:centre Radio:sender.value];

}
```

Figura 19.4 Método ChangeDistance a través del cual se obtiene el valor del perímetro establecido con el contro UISlider en Xcode. Fuente: propia.

En el capítulo 3.3 se explica más detalladamente el uso del gps que viene incorporado en dispositivos iOS, a través del cual podemos obtener tanto latitud como longitud de la ubicación del dispositivo, con un rango de precisión aceptable, para poder generar el rango de movimiento permitido.

3.2.6 Recordatorios

Los recordatorios permitirán al paciente recibir alertas de forma visual y auditiva a la hora específica sobre tareas que debe realizar o medicamentos que debe tomar, facilitando el manejo de información para pacientes con alzheimer en etapa inicial, ya que en esta etapa se empiezan a manifestar problemas con la memoria a corto y mediano plazo, especialmente en recordar eventos, actividades o medicamentos que deben tomar.

Dentro de esta pantalla se ha considerado incluir una lista de todos los recordatorios ingresados, además de un filtro por cada uno de sus tipos: recordatorio o medicamento, para un fácil acceso a cada uno de ellos. Adicionalmente, esta pantalla incluye la opción para ingresar nuevos recordatorios, permitiendo al Cuidador ingresar toda la información necesaria para crear el recordatorio con la siguiente información: Tipo de recordatorio, nombre, descripción, alertar al cuidador, fecha y hora y dosis en el caso de que el tipo de recordatorio sea un medicamento.

Agregamos un nuevo UIViewController desde la librería de objetos hacia el storyboard, lo enlazamos con su respectivo ícono del menú principal para que forme parte de una subvista y su cabecera tenga un navigation controller que nos permite regresar de manera inmediata al menú principal; adicionalmente agregamos una nueva clase del tipo UITableViewController con el nombre RecordatoriosTableViewController y lo enlazamos a nuestra vista; esta clase contiene métodos propios del tipo TableViewController para manejar tablas, que en el caso de la pantalla de recordatorios nos serán de gran utilidad.

La opción de alertar al cuidador permite enviar una alerta vía mail al momento de ejecutarse el recordatorio, de esta manera tanto el paciente como el cuidador recibirán el recordatorio, vía notificación visual y sonora para el paciente directamente en el dispositivo y como una notificación vía mail al correo electrónico del cuidador con el cual se registró en la aplicación. La opción de alertar al cuidador es opcional, y la manejamos en la aplicación a través del control UISlider, el cual permite establecer dos estados, mediante un interruptor gráfico.

Una vez ingresados todos los datos del recordatorio se almacenarán en un archivo de tipo plist dentro de la aplicación, lo cual nos permite ingresar en cualquier momento a dicho archivo para poder leer toda la información necesario y mostrarla gráficamente al cuidador al acceder a la ventana Recordatorios, la cual se puede observar como ha sido diseñada en Xcode con el uso de storyboards en la figura 20.

Para la lista de medicamentos hemos implementado el control UIImageView para identificar a través de un icono, el tipo de recordatorio que se visualiza, un icono de color verde con un calendario y un reloj para recordatorios y otro icono de color rojo con una capsula y pastilla para el caso de recordatorios. Figura 20

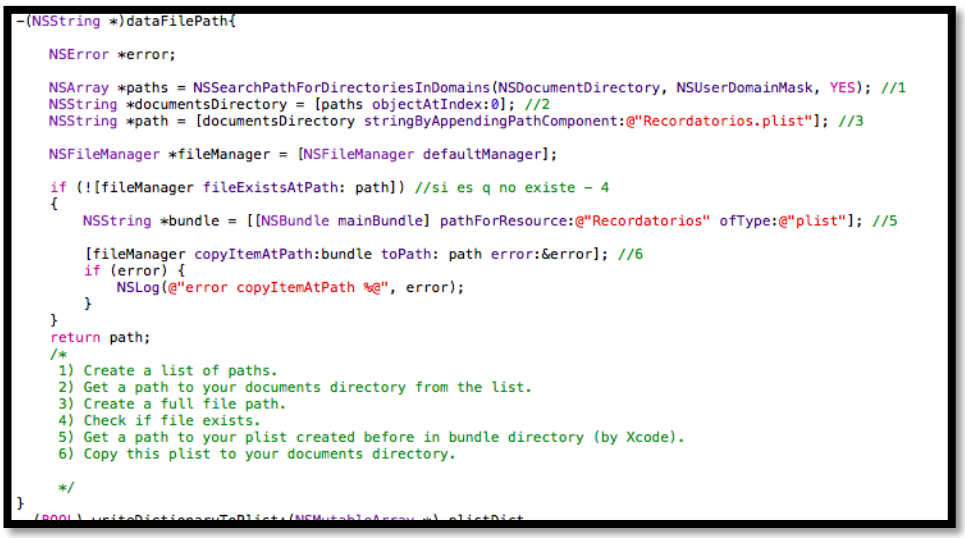
Figura 20. Pantalla Listado Medicamentos diseñada en Xcode. Fuente: propia

Todos los componentes utilizados en la pantalla de Lista de recordatorios se encuentran listados con sus propiedades que han sido utilizadas en la Tabla 6.

Control/ Propiedad	Content	User Interaction Enabled	Descripción
UITableView	Dynamic Prototypes	YES	Conjunto de UITableViewCell
UITableViewCell	Custom	YES	Visualiza todos los recordatorios
UIImageView	Aspect Fit	NO	Diferencia entre recordatorios y medicamentos a través de 2 tipos de iconos
UISegmented Controll	Plain	YES	Permite el filtrado de los recordatorios

Tabla 6. Controles existentes en la pantalla Lista de recordatorios. Fuente: propia.

Para empezar a trabajar con esta vista debemos leer la información que se encuentra en Recordatorios.plist con el método dataFilePath. Figura 20.1, el cual nos devuelve la dirección del archivo que la almacenaremos en una variable de tipo NSString con el nombre pathPlist y en una variable de tipo NSMutableArray con nombre tableData agregamos los valores del archivo.



```
-(NSString *)dataFilePath{

    NSError *error;

    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES); //1
    NSString *documentsDirectory = [paths objectAtIndex:0]; //2
    NSString *path = [documentsDirectory stringByAppendingPathComponent:@"Recordatorios.plist"]; //3

    NSFileManager *fileManager = [NSFileManager defaultManager];

    if (![fileManager fileExistsAtPath: path]) //si es q no existe - 4
    {
        NSString *bundle = [[NSBundle mainBundle] pathForResource:@"Recordatorios" ofType:@"plist"]; //5

        [fileManager copyItemAtPath:bundle toPath: path error:&error]; //6
        if (error) {
            NSLog(@"error copyItemAtPath %@", error);
        }
    }
    return path;
    /*
     1) Create a list of paths.
     2) Get a path to your documents directory from the list.
     3) Create a full file path.
     4) Check if file exists.
     5) Get a path to your plist created before in bundle directory (by Xcode).
     6) Copy this plist to your documents directory.

     */
}
```

Figura 20.1 Lectura del archivo Recordatorios.plist en Xcode. Fuente: propia

Dentro de la clase RecordatoriosViewController encontramos 3 métodos particulares que nos permiten controlar el número de secciones y la cantidad de elementos a cargar en la tabla, estos métodos son: numberOfRowsInSection y cellForRowAtIndexPath, en la figura 20.2 podemos observar como iniciar estos métodos con los valores y las validaciones correspondientes. Con estos dos métodos configuramos la tabla de recordatorios inicial, conforme se agreguen o eliminen recordatorios el valor de la tabla se ajustará automáticamente.

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{

    return [tableData count];
}
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"Cells";
    RecordatoriosTableViewCell *cell = (RecordatoriosTableViewCell *)[tableView dequeueReusableCellWithIdentifier:CellIdentifier forIndexPath:indexPath];

    cell.lblTituloRecordatorio.text = [[tableData objectAtIndex:indexPath.row]objectForKey:@"nombre"];
    cell.lblDescripcionRecordatorio.text =[[tableData objectAtIndex:indexPath.row]objectForKey:@"descripcion"];
    NSString *trecordatorio = [[tableData objectAtIndex:indexPath.row]objectForKey:@"tipo"];
    if ([trecordatorio isEqualToString:@"recordatorio"]) {
        [cell.imgTipoRecordatorio setImage:[UIImage imageNamed:@"bordeRecordatorio.png"]];
    }
    if ([trecordatorio isEqualToString:@"medicamento"]) {
        [cell.imgTipoRecordatorio setImage:[UIImage imageNamed:@"bordeMedicamento.png"]];
    }
    cell.lblFecha.text = [[tableData objectAtIndex:indexPath.row]objectForKey:@"fecha"];
    cell.lblDosis.text = [[tableData objectAtIndex:indexPath.row]objectForKey:@"dosis"];

    return cell;
```

Figura 20.2 Métodos numerOfRowsInSection y cellForRowAtIndexPath. Fuente: propia

Dentro de la pantalla de Lista de recordatorios el Cuidador puede ingresar nuevos recordatorios, para lo cual agregamos un botón de tipo UIBarButtonItem en la parte superior derecha del navigation controller que pertenece a esta vista; desde la librería de controles de Xcode seleccionamos este control y lo arrastramos hacia la esquina superior derecha de la vista Recordatorios (figura 20.3); al ubicarlo modificamos su nombre por Nuevo Recordatorio y lo enlazamos a la clase con una conexión de tipo Action en el evento Touch Up Inside que llamaremos AgregarRecordatorio, método dentro del cuál llamaremos a la nueva vista donde ingresar nuevos recordatorios.

Figura 20.3 UIBarButton dentro de la vista Recordatorios en Xcode. Fuente: propia

Creamos una nueva clase de tipo UIViewController con nombre NuevoRecordatorioViewController la cuál utilizaremos para la vista que nos permitirá ingresar nuevos recordatorios. Dentro del método AgregarRecordatorio que enlazamos al UIBarButtonItem de la vista RecordatoriosViewController llamamos a la vista donde vamos a ingresar los nuevos recordatorios con el código que se muestra en la figura 20.4.

Figura 20.4 Método para llamar a la nueva vista NuevoRecordatorio en Xcode. Fuente: propia

3.2.6.1 Nuevos recordatorios

En esta vista el Cuidador podrá ingresar nuevos recordatorios de dos tipos: Recordatorios y Medicamentos, los cuales se visualizarán en la vista Lista de Recordatorios. Empezamos por agregar una nueva vista al storyboard en la cual vamos a contar con los siguientes controles que debemos agregarlos también desde la librería de objetos de Xcode: UISegmentedControl el cual nos permitirá seleccionar el tipo de recordatorio para ingresar, UITextField para ingresar el nombre del recordatorio y su descripción y un UISwitch para indicar si el recordatorio se debe notificar al cuidador; adicionalmente agregamos un UIToolBar en la parte superior de la vista, y dentro del mismo agregamos dos UIBarButtonItem que nos permitirán cerrar la vista y guardar el recordatorio; la pantalla final con todos los controles agregados se puede visualizar en la figura 21.

Figura 21. Vista Agregar Recordatorios. Fuente: propia

Enlazamos todos los controles con la vista NuevoRecordatorioViewControles con conexiones de tipo Outlet; el contro UISegmetedControl lo enlazamos con una conexión de tipo Action en el método Value Changed con el nombre SeleccionarTipoRecordatorio, el cual nos permitirá reconocer la opción de recordatorio seleccionada por el cuidador.

El botón Guardar que ingresamos en la parte superior derecha del UIToolBar lo enlazamos con una conexión de tipo Action en el evento Touch Up Inside con el nombre GuardarRecordatorio; de igual manera el botón Cancelar lo enlazamos con una conexión del mismo tipo en el mismo evento con el nombre CerrarVista. Dentro del método GuardarRecordatorio recopilaremos toda la información ingresada en la vista para poder guardarla en el archivo Recordatorios.plist, mientras que en el

método CerrarVista ocultaremos la vista actual para regresar a la lista de recordatorios.

Todos los valores ingresados los almacenamos en la variable mutdict de tipo NSMutableDictionary con sus respectivas llaves, la cual almacenamos en el archivo Recordatorios.plist mediante el método writeDictionaryToPlist. Figura 21.1. Una vez ingresados los datos el cuidador tendrá la opción de cerrar la vista y regresar a la lista de recordatorios dando click en el botón cerrar, el cual invocará a su conexión establecida CerraVista con el método dissmissViewController la cual se observa en la figura 21.2

```
NSMutableDictionary *mutdict = [[NSMutableDictionary alloc]init];
if (trecordatorio==0) {
    [mutdict setValue:@"recordatorio" forKey:@"tipo"];
    [mutdict setValue:@"" forKey:@"dosis"];
}
if (trecordatorio==1) {
    [mutdict setValue:@"medicamento" forKey:@"tipo"];

    [mutdict setValue:[self.buttonDosis.titleLabel.text stringByReplacingOccurrencesOfString:@"Dosis: " withString:@""] forKey:@"dosis"];
}

[mutdict setValue:self.txtNombre.text forKey:@"nombre"];
[mutdict setValue:strDate forKey:@"fecha"];
//    [mutdict setValue: forKey:@"repetir"];

if (trecordatorio == 0) {
    [mutdict setValue:self.txtViewDescripcion.text forKey:@"descripcion"];
}
if (trecordatorio == 1) {
    NSString *desc1 = [self.txtViewDescripcion.text stringByAppendingString:@"\n"];
    NSString *desc2 = [desc1 stringByAppendingString:self.buttonDosis.titleLabel.text];
    [mutdict setValue:desc2 forKey:@"descripcion"];
}

//imagen
if (seleccionaFoto) {
    NSData *data = UIImagePNGRepresentation(self.imgCamara.image);

    [mutdict setValue:data forKey:@"foto"];
}

NSMutableArray *datos = [[NSMutableArray alloc]init];
[datos addObject:mutdict];

[tableData insertObject:mutdict atIndex:[tableData count]];

if ([self writeDictionaryToPlist:tableData]) {
    //recordatorio - medicamento guardado
    HUD = [[MBProgressHUD alloc] initWithView:self.contentView];
    [self.contentView addSubview:HUD];
    HUD.delegate = self;

    [HUD showWhileExecuting:@selector(SaveSucess) onTarget:self withObject:nil animated:YES];

    [self CrearNotificacion];
}
```

Figura 21.1 Almacenamiento de nuevos recordatorios en el archivo Recordatorios.plist. Fuente: propia

Figura 21.2 Método dissmissViewController que permite cerrar la vista actual y regresar a la vista inmediatamente anterior. Fuente: propia

Todos los componentes que comprenden la pantalla para ingreso de nuevos recordatorios además de sus propiedades los podemos observar en la tabla 7.

Control/Propiedad	Type	User Interaction Enabled	Descripción
UIToolBar	Default	NO	
UIBarButtonItem Cerrar	Custo	YES	Cierra la vista actual
UIBarButtonItem Guardar	Custom	YES	Guardar el recordatorio ingresado
UITextField Nombre	Plain	YES	Para ingreso del nombre del recordatorio
UITextField Descripción	Plain	YES	Para ingreso de la descripción del recordatorio
UISwitch Alertar Cuidador		YES	Activa/Desactiva la alerta para el cuiadador
UILabel Fecha y Hora	Default	YES	Ingreso de fecha y hora del recordatorio

Tabla 7. Controles y sus propiedades utilizadas en la vista Agregar Recordatorio. Fuente: propia

3.2.7 Pantalla dedicada al paciente.

El botón Iniciar que está en el menú principal será el encargado de mostrar la vista dedicada exclusivamente al paciente con alzheimer, la cual tendrá la función principal de ofrecer información como: hora, fecha, foto del paciente y sus nombres; datos que el paciente puede olvidar debido a la etapa inicial de su enfermedad, además de un único botón de libre acceso para el mismo con el texto "Solicitar Ayuda", el cual al ser presionado mostrará otra pantalla al paciente donde podrá visualizar la foto de su cuidador informando de que se ha enviado una solicitud de ayuda al mismo. Figura 22

Figura 22. Pantallas dedicadas al paciente. Fuente: propia

Para implementar la primera vista, agregamos un nuevo UIViewController al storyboard, además de una clase con el nombre InicioPaciente de tipo ViewController las cuales enlazamos desde la pestaña Show Identity Inspector de Xcode. Dentro de la vista agregada debemos añadir los siguientes controles desde la librería de objetos: UILabel para mostrar la hora del dispositivo y la fecha, UIImage en la cual visualizaremos la foto del paciente, UILabel para visualizar el nombre del paciente, UIButton el cual servirá para que el paciente solicite ayuda remota a su cuidador.

Enlazamos todos los controles con la vista respectiva, con conexiones de tipo Outlet, el único control sobre el cual necesitamos una acción en esta vista es el botón de ayuda, el cual enlazamos en el evento Touch Up Inside con el nombre del evento SolicitarAyuda, dentro del cual agregamos el código que se visualiza en la figura 22.1, que permitirá enviar la notificación vía mail al cuidador además de abrir la vista en donde se mostrará la foto del cuidador.

```
- (IBAction)SolicitarAyuda:(UIButton *)sender {

    [self performSelectorInBackground:@selector(NotificarCuidador:) withObject:[dicDataCuidador objectForKey:@"correo"]];

    //[self NotificarCuidador:[dicDataCuidador objectForKey:@"correo"]];

    SolicitarAyudaViewController *view = (SolicitarAyudaViewController *) [self.storyboard instantiateViewControllerWithIdentifier:@"SolicitarAyuda"];
    [self presentViewController:view animated:YES completion:nil];


}
-(void)NotificarCuidador:(NSString*)mail
{
    NSString *post = [NSString stringWithFormat:@"token=%@",mail];
    //cambiar subject mail a Ubicación Paciente
    NSString *post2 = [NSString stringWithFormat:@"&descripcion=%@",mapLink];
    NSString *postFinal = [post stringByAppendingString:post2];
    NSData *postData = [postFinal dataUsingEncoding:NSASCIIStringEncoding allowLossyConversion:YES];

    NSString *postLength = [NSString stringWithFormat:@"%lu", (unsigned long)[postData length]];

    NSMutableURLRequest *request = [[NSMutableURLRequest alloc] init];
    [request setURL:[NSURL URLWithString:urlPost2]];
    [request setHTTPMethod:@"POST"];
    [request setValue:postLength forHTTPHeaderField:@"Content-Length"];
    [request setValue:@"application/x-www-form-urlencoded" forHTTPHeaderField:@"Content-Type"];
    [request setHTTPBody:postData];

    NSURLResponse *response;
    NSError *error;
    NSData *POSTReply = [NSURLConnection sendSynchronousRequest:request returningResponse:&response error:&error];
    NSString *reply = [[NSString alloc] initWithBytes:[POSTReply bytes] length:[POSTReply length] encoding: NSASCIIStringEncoding];
    NSLog(@"POST REPLY: %@",reply);

}
```

Figura 22.1 Método SolicitarAyuda, el cual envía la notificación vía mail al Cuidador del paciente con alzheimer. Fuente: propia

Todos los controles implementados en esta vista, además de sus propiedades utilizadas se encuentran listadas en la tabla 8.

Control/Propiedad	Type	User Interaction Enabled	Descripción
UILabel Hora	Default	NO	Muestra la hora actual del dispositivo
UIImageView Paciente	Aspect Fill	NO	Muestra la fotografía del paciente
UILabel Nombre Paciente	Default	NO	Muestra el nombre del paciente
UIButton Solicitar Ayuda	Custom	YES	Botón para solicitar ayuda remota

Tabla 8. Controles y sus propiedades de la vista inicial del paciente. Fuente: propia

Al visualizar la siguiente pantalla, el paciente tendrá acceso a los nombres completos de su cuidador, además de su fotografía; para esto agregamos un nuevo UIViewController y una nueva clase del mismo tipo con nombre SolicitarAyuda las cuales enlazamos directamente desde el Identity Inspector de Xcode, además agregamos los siguientes controles: UILabel para mostrar el nombre del cuidador, un UIImageView que visualizará la fotografía del mismo y un único botón de tipo UIButton que permitirá regresar a la ventana anterior.

Los datos que se visualizan en esta pantalla la vista los obtiene del archivo Perfil.plist, a los cuales se accederán a través del método de lectura dataFilePath del cual obtenemos la dirección del archivo que lo almacenaremos en una variable de tipo NSString con nombre pathPlisth, además creamos una variable de tipo NSMutableDictionary con nombre dicData el cual almacenará todo el contenido del archivo Perfil.plist.

Una vez leída toda la información del archivo plist tenemos un solo método que nos permite cargar todos los datos en los controles respectivos, la variable de tipo NSMutableDictionary contiene toda la información necesaria, a la cuál debemos acceder llamando las llaves respectivas del archivo como se puede visualizar en la figura 22.2; el método InformacionCuidador lo llamamos inmediatamente después de leer el archivo plist, dentro del método ViewDidLoad.

```
-(void)InformacionCuidador
{
    NSString *string1 = [@"Hemos informado a " stringByAppendingString:[dicData objectForKey:@"nombres"]];
    NSString *string2 = [string1 stringByAppendingString:@", tu "];
    NSString *string3 = [string2 stringByAppendingString:[dicData objectForKey:@"parentesco"]];
    self.lblTitulo.text = string3;

    NSData *retrievedData = [NSData dataWithData:[dicData objectForKey:@"foto"]];
    if (retrievedData.length > 0) {

        self.imgCuidador.image = [UIImage imageWithData:retrievedData];
    }
    else self.imgCuidador.image = [UIImage imageNamed:@"PerfilNuevo.png"];
}
```

Figura 22.2 Método InformarCuidador que carga toda la información del archivo Perfil.plist en los controles respectivos. Fuente: propia

Enlazamos el botón de esta vista con una conexión de tipo Action en el método Touch Up Inside con el nombre Aceptar, método que nos permitirá regresar a la vista anterior cuando el paciente presione este botón; agregamos el método dissmissViewControllerAnimated dentro esta acción como lo podemos visualizar en la figura 22.3, lo cual cerrará esta vista y automáticamente nos llevará hacia la vista

inmediatamente anterior.

```
- (IBAction)Aceptar:(UIButton *)sender {

    [[self presentingViewController]dismissViewControllerAnimated:YES completion:nil];
}
```

Figura 22.3 Método Aceptar que nos permite regresar a la vista inmediatamente anterior. Fuente: propia

Todos los componentes utilizados en esta vista, además de sus propiedades se encuentran listados en la Tabla 9.

Control/Propiedad	Type	User Interaction Enabled	Descripción
UILabel Nombre Cuidador	Default	NO	Muestra el nombre del Cuidador al que se envió la notificación
UIImageView Foto Cuidador	Aspect Fill	NO	Muestra la foto del cuidador al cual se envió la notificación
UIButton Aceptar	Custom	YES	Permite regresar a la vista anterior

Tabla 9. Controles y sus propiedades utilizadas en la vista de solicitar ayuda. Fuente: propia

3.3 Uso del GPS en dispositivos iOS

Para poder definir el perímetro de movimiento permitido para el Paciente, debemos hacer uso del gps del dispositivo, siendo este el que nos permita ubicar el mismo de manera permanente mientras la aplicación se ejecute. Para obtener la ubicación actual del dispositivo creamos un objeto de tipo CLLocationManager con nombre locationManager, a través del cual podemos acceder tanto a latitud como longitud gracias al uso del gps interno.

Antes de acceder a la ubicación del dispositivo, debemos solicitar permiso de acceso al usuario, este permiso se visualiza en forma de alerta al ingresar a la aplicación a la cuál el usuario debe responder aceptando que la aplicación acceda a la ubicación, de no aceptarla el dispositivo no podrá acceder a la ubicación. Esta medida forma parte de las directivas de seguridad de iOS ya que pueden existir aplicaciones que hagan uso de esta característica sin autorización previa, lo que conlleva a una disminución drástica de la batería del dispositivo además de atentar contra la privacidad del usuario.

En el archivo de configuración plist que se incluye por defecto dentro del proyecto se deben incluir dos categorías adicionales, las cuales servirán para solicitar el permiso de acceso a la ubicación cuando se requiera, estas categorías son: NSLocationWhenInUseUsageDescription y NSLocationAlwaysUsageDescription; conjuntamente con cada una de estas categorías debemos agregar el texto que se mostrará al usuario al solicitar el permiso de acceso. Para agregar estas nuevas categorías seleccionamos el archivo .plist que viene por defecto en el proyecto, dentro del cual damos click derecho y seleccionamos la opción Add Row y agregamos la categoría correspondiente. En la figura 23 se puede observar las dos categorías agregadas al archivo plist del proyecto Xcode.

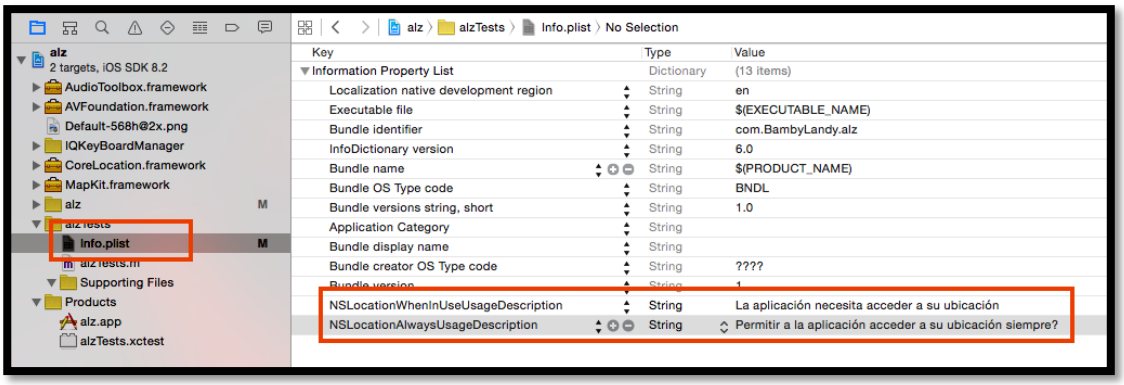


Figura 23. Categorías NSLocation para solicitar acceso a la ubicación del dispositivo. Fuente: propia.

Para mejorar la usabilidad de la aplicación, debemos solicitar el permiso de acceso a la ubicación solamente cuando se va a hacer uso del mismo, evitando solicitarlo inicialmente cuando el usuario abra por primera vez la aplicación ya que no se necesitará dicho permiso en primera instancia. La vista en donde establecemos el perímetro permitido para el paciente es en donde se necesita acceso a la ubicación del dispositivo, ésta vista será presentada al cuidador del paciente razón por la cual será él quien deba aceptar la solicitud; de otra manera, si solicitáramos acceso dentro de la pantalla dedicada al paciente posiblemente sea él quien niegue acceso al mismo y la aplicación no funcionará de la manera deseada.

Dentro del método que se carga al iniciar la vista, denominado ViewDidAppear ingresaremos la solicitud (figura 23.1), al aceptar la solicitud no se volverá a mostrar nuevamente la alerta si el usuario ingresa nuevamente a esta vista. La propiedad distanceFilter nos permite configurar la distancia que el dispositivo debe moverse horizontalmente para enviar los datos de su ubicación, en este caso hemos configurado dicha propiedad con el valor kCLDistanceFilterNone, el cuál envía constantemente notificaciones de su ubicación; por otro lado la propiedad

desiredAccurancy nos permite definir el nivel de exactitud al enviar las notificaciones, para la cual hemos fijado el valor kCLLocationAccuracyBest que nos brinda la mayor exactitud.

Por otro lado, si el usuario cancela la solicitud, al ingresar nuevamente a la misma vista tampoco se mostrará el mensaje con la solicitud para que permita acceder a la ubicación del dispositivo, en este caso se debe otorgar el permiso accediendo directamente a los Ajustes del dispositivo, dentro de la sección Privacidad y dentro de esa a Localización, donde encontrará el icono de la aplicación a la cual deberá acceder y activar el permiso necesario como se puede observar en la figura 23.2

```
-(void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:YES];

    self.locationManager.distanceFilter = kCLDistanceFilterNone;
    self.locationManager.desiredAccuracy = kCLLocationAccuracyBest;
    [self.locationManager startUpdatingLocation];
```

Figura 23.1 Solicitud de acceso a la ubicación del dispositivo dentro del método viewDidAppear en la clase SetLocationViewController.m. Fuente: propia

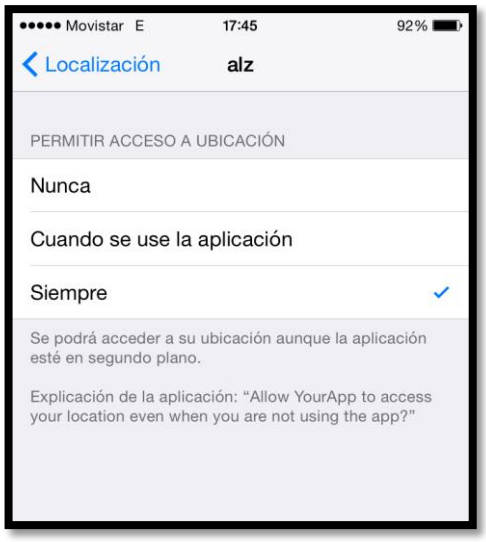


Figura 23.2 Activación del acceso a la ubicación del dispositivo desde los Ajustes del mismo. Fuente: propia

La solicitud se visualizará en forma de alerta sobre la pantalla en la cual se encuentra el usuario con dos opciones: Cancelar y Aceptar (figura 23.4); para que la aplicación pueda acceder a la ubicación del dispositivo de manera inmediata el usuario debe seleccionar la opción "Aceptar".

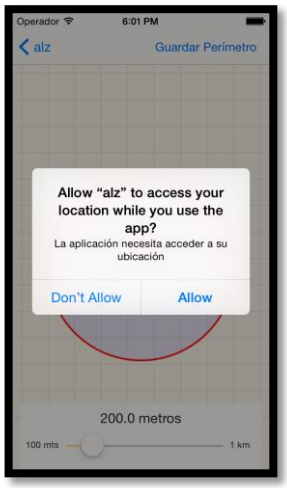


Figura 23.3 Solicitud de acceso a la ubicación del dispositivo iOS. Fuente: propia

A través de estos métodos la aplicación cumple con las normativas de privacidad y seguridad en el uso del gps en dispositivos iOS, previniendo el acceso sin autorización a la misma y ahorrando batería, ya que el uso del gps del dispositivo representa un consumo energético considerable en dispositivos móviles de todo tipo.

3.4 Alertas locales en iOS

iOS dispone de alertas locales, las cuales se ejecutan en una fecha y hora programada, para las cuales no se necesita conexión a internet o un servidor web como en el caso de las notificaciones remotas, siendo de gran ayuda para el proyecto que desarrollamos en la cual buscamos crear una aplicación que sea de gran ayuda al paciente y usuario sin la necesidad de una conexión constante a internet para recibir alertas de todo tipo.

Las notificaciones que recibirá tanto el paciente como el cuidador serán sobre actividades que debe realizar el paciente así como recordatorios de medicamentos que debe tomar, los mismos que serán ingresados a través de la vista Recordatorios que explicamos con anterioridad en la sección 3.2.6.1 Nuevos recordatorios.

Los recordatorios locales forman parte de la clase UILocalNotification que viene incorporada en la librería UIKit, la cuál se incorpora por defecto al crear un nuevo proyecto en Xcode sin necesidad de agregar librerías adicionales.

En iOS 8, una notificación local se visualizará de 2 maneras cuando el dispositivo se encuentra desbloqueado: en forma de tiras en el centro de notificaciones si es que la aplicación no se encuentra ejecutándose en primer plano, o en forma de alerta si se

encuentra ejecutándose en primer plano (Figura 24)

Figura 24. Modalidades de presentación de alertas locales en iOS. Fuente: Apple Inc, iOS 8.

Si el dispositivo se encuentra bloqueado, las alertas se visualizarán en la pantalla bloqueada del dispositivo, donde se podrá visualizar el ícono de la aplicación que la ejecutó, además del nombre de la misma y el detalle de la notificación. Figura 24.1

Una notificación local se inicializa con la clase UILocalNotification y podemos personalizar sus propiedades fireDate con la fecha en la que debe ejecutarse y alertBody con la descripción de la alerta. Una vez creada la registramos en el sistema con el método scheduleLocalNotification. Figura 24.2

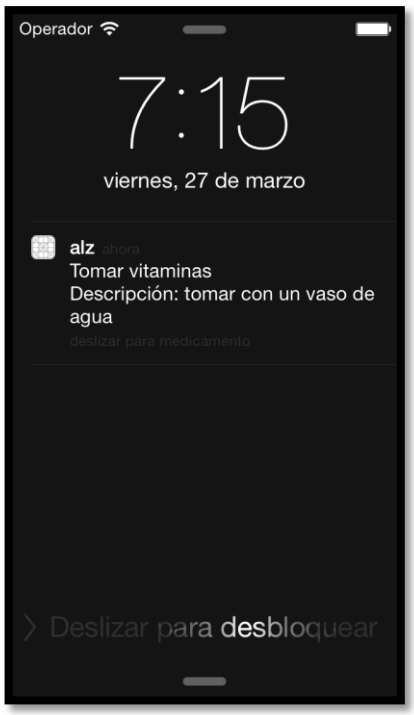


Figura 24.1 Presentación de alertas locales en iOS 8 con el dispositivo bloqueado. Fuente: propia

```
UILocalNotification *localNotif = [[UILocalNotification alloc] init];
localNotif.fireDate = fAlerta;
localNotif.alertBody = @"Recordatorio";
// Schedule the notification
[[UIApplication sharedApplication] scheduleLocalNotification:localNotif];
```

Figura 24.2 Inicialización de una alerta local en iOS 8. Fuente propia

Si el paciente recibe una notificación cuando el dispositivo se encuentra bloqueado podrá visualizarlo en la pantalla del dispositivo y al seleccionar la notificación ejecutará de manera inmediata la aplicación, ya sea que se encuentre abierta o en segundo plano.

3.5 Notificaciones vía mail

Adicional a las alertas locales, la aplicación permite el envío de notificaciones vía mail hacia el cuidador del paciente, permitiendo recibir alertas cuando el paciente excede el perímetro establecido y también de los recordatorios que hayan sido ingresados con la opción “alertar al cuidador”, de esta manera el cuidador estará informado de manera permanente sobre la ubicación del paciente y las actividades que debe realizar.

Para las notificaciones vía mail es necesario que el dispositivo tenga conexión a internet, puede ser por una red inalámbrica wi-fi o a través de un plan de datos celular ya que el envío de correos electrónicos se los realiza a través de un archivo en lenguaje php que se encuentra alojado en internet. El detalle del archivo php encargado del envío de las notificaciones vía mail se encuentra en la figura 25

```php
<?php
$token = $_POST['mail'];
$descrip = $_POST['descripcion'];
$to = $token;
 $subject = "Recordatorio alz";
 $body = $descrip;
 if (mail($to, $subject, $body)) {
   echo("<p>Email successfully sent!</p>");
  } else {
   echo("<p>Email delivery failed…</p>");
  }
?>
```

Figura 25: Archivo php encargado del envío de notificaciones vía mail. Fuente: propia.

La aplicación envía la solicitud en dos ocasiones específica, la primera cuando el paciente excede el perímetro de movimiento permitido, y la segunda cuando el recordatorio ha sido ingresado con la opción activada de notificar al cuidador, en ambos casos la llamada desde la aplicación es similar, especificando los valores a enviar en el detalle del mail en cada caso particular; como se puede observar en la figura 25.1, se realiza la solicitud de enviar la alerta vía mail en el caso de un recordatorio pasando como detalle del mail la descripción de la notificación, mientras que en la figura 25.2 la solicitud se ejecuta cuando el paciente ha excedido su rango de movimiento permitido, pasando como detalle del mail la ubicación en la que excedió el perímetro mediante un link de Google Maps. (Google Inc)

A través del método setHTTPMethod en Objective-C podemos enviar los detalles del mail como son: destinatario y detalle del mensaje en el formato necesario para que el archivo php de la figura 25 sea el encargado de enviar la notificación vía mail.

```
-(void)NotificarCuidador:(NSString*)mail
{
     NSString *post = [NSString stringWithFormat:@"token=%@",[dicData objectForKey:@"correo"]];
    NSString *post2 = [NSString stringWithFormat:@"&descripcion=%@",descripNotificacion];
    NSString *postFinal = [post stringByAppendingString:post2];
     NSData *postData = [postFinal dataUsingEncoding:NSASCIIStringEncoding allowLossyConversion:YES];

     NSString *postLength = [NSString stringWithFormat:@"%lu", (unsigned long)[postData length]];

     NSMutableURLRequest *request = [[NSMutableURLRequest alloc] init];
     [request setURL:[NSURL URLWithString:urlPost2]];
     [request setHTTPMethod:@"POST"];
     [request setValue:postLength forHTTPHeaderField:@"Content-Length"];
     [request setValue:@"application/x-www-form-urlencoded" forHTTPHeaderField:@"Content-Type"];
     [request setHTTPBody:postData];

     NSURLResponse *response;
     NSError *error;
     NSData *POSTReply = [NSURLConnection sendSynchronousRequest:request returningResponse:&response
         error:&error];
     NSString *reply = [[NSString alloc] initWithBytes:[POSTReply bytes] length:[POSTReply length]
         encoding: NSASCIIStringEncoding];
     NSLog(@"POST REPLY: %@",reply);

}
```

Figura 25.1 Método POST que envía los parámetros necesarios para las notificaciones vía mail que ejecutará el archivo php. Fuente: propia

```
-(void)NotificarUbicacionCuidador:(NSString*)mail
{
    NSString *post = [NSString stringWithFormat:@"token=%@",mail];
    //cambiar subject mail a Ubicación Paciente
    NSString *post2 = [NSString stringWithFormat:@"&descripcion=%@",mapLink];
    NSString *postFinal = [post stringByAppendingString:post2];
    NSData *postData = [postFinal dataUsingEncoding:NSASCIIStringEncoding allowLossyConversion:YES];

    NSString *postLength = [NSString stringWithFormat:@"%lu", (unsigned long)[postData length]];

    NSMutableURLRequest *request = [[NSMutableURLRequest alloc] init];
    [request setURL:[NSURL URLWithString:urlPost2]];
    [request setHTTPMethod:@"POST"];
    [request setValue:postLength forHTTPHeaderField:@"Content-Length"];
    [request setValue:@"application/x-www-form-urlencoded" forHTTPHeaderField:@"Content-Type"];
    [request setHTTPBody:postData];

    NSURLResponse *response;
    NSError *error;
    NSData *POSTReply = [NSURLConnection sendSynchronousRequest:request returningResponse:&response
     error:&error];
    NSString *reply = [[NSString alloc] initWithBytes:[POSTReply bytes] length:[POSTReply length]
     encoding: NSASCIIStringEncoding];
    NSLog(@"POST REPLY: %@",reply);

}
```

Figura 25.2 Método POST que envía la ubicación del paciente para la notificación vía mail que ejecutará el archivo php. Fuente: propia

Como se puede observar en las figuras 25.1 y 25.2, en ambos casos necesitamos una variable de tipo NSURL la cual contiene la url con la ubicación del archivo php que se encuentra alojado en un servidor en internet a la cual el método setHTPPMethod hace referencia para enviar la solicitud. Además, podemos visualizar en la figura 25.2 que enviamos en el campo &descripción una variable de tipo NSString denominada mapLink, la cual contiene la url de Google Maps con la ubicación en la que el paciente excedió el perímetro establecido, dicha url la obtenemos con la latitud y longitud de la ubicación del dispositivo, la cual podemos observar en la figura 25.3.

```
 //Google Map Link
 NSString *googleMapsURLString = [NSString stringWithFormat:@"http://maps.google.com/maps?q=
%1.6f,%1.6f", currentLocation.coordinate.latitude, currentLocation.coordinate.longitude];

 mapLink = googleMapsURLString;
 //////////
```

Figura 25.3 Creación del link en Google Maps con la ubicación del dispositivo en iOS 8. Fuente: propia

3.7 Conclusiones

La creación de una memoria técnica es pilar fundamental de este capítulo, en el cual se ha intentando dar una explicación técnica de todos los elementos que involucran el desarrollo de la aplicación móvil para pacientes con alzheimer en etapa inicial, la cual exige conocimientos de programación orientada a objetos en el lenguaje Objective-C y en php, tanto para la aplicación móvil como para la ejecución de módulos externos al desarrollo respectivamente.

En el desarrollo de este capítulo hemos aprovechado en su totalidad los controles nativos que brinda iOS para el desarrollo de la interfaz de la aplicación, además de los métodos respectivos de cada control en los cuales hemos demostrado de manera gráfica y a través de tablas, todos los controles utilizados, su funcionalidad dentro de cada pantalla, el tipo de interacción que va a desarrollar tanto el cuidador como el paciente así como las propiedades modificadas para adaptarse a los objetivos de este proyecto, con lo cual hemos obtenido una aplicación funcional, de rápida ejecución y de fácil manejo tanto para el cuidador como para el paciente que sufre de alzheimer en etapa inicial.

Desde el diseño de la interfaz a través de storyboards hasta la codificación necesaria para el funcionamiento de la aplicación han sido cubiertos en el capítulo 3, dando énfasis en el diseño de una interfaz fácil de usar como una codificación lo mas sencilla posible. Para mayor información en cuanto a controles iOS y programación en el lenguaje Objective-C recomendamos hacer uso de la documentación publicada por Apple Inc en su libro iOS Application Programming Guide, (Apple Inc) además de su libro Programming with Objective-C (Apple Inc)

CAPITULO 4

Evaluación de la aplicación en un contexto real.

Una parte importante de este proyecto se centra en la prueba del trabajo realizado directamente en dispositivos iOS para comprobar su rendimiento y el uso de sus características como el GPS para obtener la ubicación del dispositivo en tiempo real, las alertas locales, y otras; lo que nos permitiera observar cómo interactúan todos los actores involucrados en este trabajo: cuidadores, familiares y pacientes con Alzheimer en etapa inicial.

Para esta evaluación indagamos sobre la utilización de la aplicación mediante una etnografía enfocada a 3 personas con Alzheimer en etapa inicial. Se realizó una etnografía enfocada, basada fundamentalmente en la observación participante, con el fin de descubrir cómo los pacientes utilizan la aplicación en su vida cotidiana y cuál es su propia perspectiva (visión *emic*) (Salgado, 2014).

Utilizamos dispositivos reales iOS mediante los cuales pudimos observar la interacción usuario-aplicación y los posibles ajustes que fueran necesarios sobre la aplicación, en base al análisis de su uso en un contexto real, entre pacientes y cuidadores.

4.1 Etnografía enfocada sobre el uso de la aplicación.

La etnografía enfocada, caracterizada por un trabajo de campo de tiempo limitado e intensivo combinado con la exploración y el análisis extensivo de los datos generados, consiste básicamente en la observación de la actividad cotidiana de los involucrados dentro de este proceso, con el fin de comprender sus hábitos frente a la aplicación móvil desarrollada en este trabajo, mediante la cual podremos comprender la funcionalidad y el apoyo que brinda la misma a pacientes con Alzheimer en etapa inicial.

Para esta etnografía contamos con el apoyo del Dr. Fabián Guapisaca, geriatra del Hospital del Río en la ciudad de Cuenca, quien cuenta con un consultorio ubicado en el dicho hospital. El Dr. Guapisaca con su asesoramiento profesional nos supo guiar en la etnografía realizada a tres de sus pacientes con Alzheimer en etapa inicial. Visitamos personalmente en sus domicilios a los pacientes, quienes nos supieron acoger con mucho afecto y colaborar completamente con todas las preguntas y pruebas que se realizaron. La etnografía enfocada se realizó entre marzo y abril de 2015.

Para esta etnografía compartimos diversos momentos con los pacientes y cuidadores en sus actividades cotidianas como: paseos al aire libre en zonas cercanas a sus domicilios, juegos de memoria, y otros; siempre acompañados de sus respectivos custodios, quienes nos brindaron su ayuda en todo momento. Por razones de privacidad y por petición expresa de los tutores, no se publicará el nombre verdadero de los pacientes en esta tesis. Para identificar a los tres pacientes con los cuales realizamos la etnografía los nombraremos como Paciente 1, Paciente 2 y Paciente 3.

La modalidad de las pruebas consistía en probar dos características principales de la aplicación, las cuales fueron: alertar al cuidador si el paciente excedía el perímetro de movimiento establecido y hacer uso de los recordatorios programados por el cuidador para que el paciente pueda recibirlos en el dispositivo. La modalidad de las pruebas fue personal, conjuntamente entre cuidadores y pacientes.

Mediante un iPhone 6 con iOS 8.2 realizamos las pruebas con todos los pacientes, empezando por ingresar todos los datos principales que son necesarios para el correcto funcionamiento de la aplicación, tales como: datos del cuidador y datos del paciente. Una vez ingresados estos datos, el custodio estableció el perímetro de movimiento permitido, el cual por motivos de facilidad lo establecieron en el rango mínimo diseñado que es de 100 metros. Además se ingresaba un recordatorio en el cual se informaba al paciente de realizar alguna actividad o tomar algún medicamento a una hora específica.

Al probar el uso de la aplicación, el paciente se familiarizó inmediatamente con la interfaz de usuario que mostraba su fotografía, ingresada previamente por el cuidador. La característica del perímetro de movimiento permitido se puso a prueba en un paseo a pie entre el paciente y su cuidador, quienes transitaron un tramo de 100 metros aproximadamente cercano a su sitio de vivienda, lo cual permitió establecer la alerta de haber excedido el rango permitido además de la notificación vía mail al correo electrónico del cuidador, siendo una de las características más sorprendentes y útiles según comentarios de los cuidadores ya que en los Pacientes 1 y 3, habían sufrido la pérdida de memoria sobre su lugar de residencia, ocasionando preocupación a sus cuidadores y familiares en repetidas ocasiones ya que al salir de su domicilio para dirigirse a lugares cercanos para comprar víveres se habían perdido por horas, teniendo que recurrir en más de una ocasión a organizar su búsqueda con ayuda de vecinos del barrio y de elementos de la Policía Nacional, motivo por el cual

la ubicación del paciente en caso de exceder el perímetro establecido por el cuidador fue una de las características más útiles de las pruebas realizadas, según su propia perspectiva (visión *emic*).

Por otro lado, los recordatorios de actividades por realizar o medicamentos que debía tomar el paciente, se probaron directamente en sus domicilios. Para el efecto, se ingresaron recordatorios con un tiempo corto de espera entre 2 a 5 minutos, colocando el dispositivo a forma de collar con un lazo a los pacientes o en uno de sus bolsillos de abrigos o pantalones; al ejecutarse los recordatorios, gracias a la característica de vibración y sonido emitido por el dispositivo, los pacientes supieron responder de manera satisfactoria, a más que los cuidadores recibieron el recordatorio directamente en su correo electrónico. Éstos expresaron su satisfacción sobre estas notificaciones, ya que si el paciente por cualquier motivo olvidara visualizar la alerta, son los cuidadores quienes pueden tomar cualquier acción para notificar al paciente que debe cumplir determinada acción. Como caso particular, pudimos observarlo en el Paciente 1, que al generarse el recordatorio no lo escuchó oportunamente, siendo su cuidador quien, al recibir la notificación en su correo electrónico, pudo recordárselo de inmediato a la persona a su cargo.

Finalmente, otra de las características que se evaluó entre todos los pacientes, fue el botón de ayuda, a través del cual los pacientes al presionarlo enviaron una notificación vía mail a su respectivo cuidador, con la ubicación exacta de donde se realizó la petición. Mediante esta característica los cuidadores y familiares pudieron observar en tiempo real, la ubicación del paciente; todos expresaron que dicha característica es sumamente útil, ya que les brinda tranquilidad saber que en cualquier momento, el paciente podrá pedir ayuda remota desde el dispositivo sin mayores complicaciones. Esta característica la probaron con éxito los Pacientes 1, 2 y 3, además de recibir la respectiva notificación en el correo electrónico de sus

cuidadores.

4.2 Análisis de la indagación de campo.

En la Tabla 10 se expone los datos de los tres pacientes sobre los cuales se realizó la etnografía enfocada, quienes residen en la ciudad de Cuenca, Ecuador. Dos de los pacientes son de sexo femenino, siendo uno de ellos el de menor edad presentando síntomas de la enfermedad a la edad de 68 años, como problemas para recordar los nombres de sus familiares cercanos, además de problemas para ubicarse en tiempo y espacio olvidando frecuentemente su lugar de residencia y ubicación actual.

Identificación	Sexo	Edad	Tiempo de diagnóstico de Alzheimer	Síntomas iniciales
Paciente 1	Femenino	91 años	1 año	Problemas para recordar nombres de familiares. Dificultad al pronunciar palabras.
Paciente 2	Femenino	68 años	6 meses	Problemas para recordar nombres de familiares. Problemas de ubicación (tiempo, espacio).
Paciente 3	Masculino	74 años	1 año	Problemas para recordar nombres de familiares. Problemas de ubicación (tiempo, espacio).

Tabla 10. Características de los pacientes con quienes se realizó la etnografía enfocada.

Adicionalmente, recopilamos las impresiones de los cuidadores después de utilizar la aplicación, ya que son ellos quienes van a configurar inicialmente el dispositivo con los datos necesarios para su correcto funcionamiento y quienes van a monitorear las

notificaciones y alertas que se generen. Después de entrevistar a cada uno de ellos podemos observar que previamente ninguno de ellos había utilizado tecnologías de información para asistir el tratamiento de la enfermedad en pacientes con Alzheimer en etapa inicial, como se puede observar en la Tabla 11. En el caso del Paciente 1, su cuidador lleva un registro en un cuaderno con los problemas que van ocurriendo a diario, tanto de síntomas nuevos o concurrentes, como medicamentos que debe tomar el paciente, por lo cual resultó de gran ayuda para su cuidador la posibilidad de registrar los recordatorios en el dispositivo y que estos los pueda visualizar tanto el paciente como su custodio.

Como se puede observar en la Tabla 11, todos los custodios, sin excepción alguna, comentaron que utilizarían la aplicación móvil que se desarrolla en esta tesis para el tratamiento de la enfermedad con los pacientes de quienes son responsables, ya que facilitaría en gran medida el cuidado de los mismos gracias a características como la ubicación del paciente y las notificaciones vía mail, que según su propia perspectiva (visión *emic*), son las más atractivas de la aplicación móvil.

	Paciente 1	**Paciente 2**	**Paciente 3**
Parentesco	Hermana	Hija	Hija
¿Utiliza actualmente alguna aplicación para el tratamiento de la enfermedad?	No	No	No
¿Utilizaría cotidianamente la aplicación móvil que probó en esta experiencia?	Si	Si	Si
¿Qué fue lo que más le gustó de la aplicación probada?	Alertas para el cuidador sobre los recordatorios.	Alertas para el cuidador sobre los recordatorios. Ubicación del paciente si excede el perímetro establecido.	Ubicación del paciente si excede el perímetro establecido. Recordatorios tanto para el paciente como el cuidador.

Tabla 11. Encuesta realizada a los Cuidadores de pacientes con alzheimer después de probar la aplicación móvil.

4.3 Ajustes de la aplicación en base al análisis de su uso en un contexto real.

Tras analizar los resultados de los tres pacientes con quienes se realizó la etnografía enfocada, conjuntamente con sus cuidadores, obtuvimos una retroalimentación importante, la cual nos sirve para depurar la aplicación en busca de brindar mayores

y mejores prestaciones a la hora de facilitar el cuidado de pacientes con Alzheimer en etapa inicial.

En la Tabla 12, podemos observar los principales inconvenientes con los cuales los pacientes se encontraron al momento de utilizar la aplicación móvil, entre los que podemos destacar que el problema más común fue el de visualización del texto, ya que el Paciente 1 y 3 utilizaban lentes, lo cual les impedía tener una visualización en primera instancia, sin mayor esfuerzo de todo el texto que se muestra en la aplicación.

Para solventar dicho problema se podría incorporar un control dentro de la interfaz de la aplicación, el cual permita al cuidador establecer el tamaño de la fuente del texto que se emplea en la misma, de esta manera el paciente no tendría que solucionar dicho problema ya que su cuidador configuraría inicialmente el tamaño de la letra para evitar cualquier problema de lectura.

Paciente	Problemas al usar la aplicación
Paciente 1	El paciente tiene problemas con la visualización del texto. El paciente utiliza lentes para mejorar su visión.
Paciente 2	El paciente tiene problemas menores con la visualización del texto en el aplicación. No utiliza lentes.
Paciente 3	El paciente tiene problemas con la visualización del texto. El paciente utiliza lentes para mejorar su visión.

Tabla 12. Problemas que encontraron los pacientes al utilizar la aplicación móvil.

Para todos los custodios resultó sumamente sencillo el manejo y configuración inicial de la aplicación, desde el ingreso de datos tanto personales como del paciente hasta la definición del perímetro de movimiento como la creación de los recordatorios de actividades y los horarios de los medicamentos.

Una de las sugerencias que tuvimos por parte del cuidador del Paciente 3, fue la posibilidad de desarrollar esta aplicación para dispositivos móviles más económicos, los cuales funcionan bajo el sistema operativo Android, con lo cual se brindaría una oportunidad de instalar la aplicación en dispositivos con características de hardware similares a dispositivos iOS pero con un costo de adquisición más económico, facilitando el acceso a mas pacientes y cuidadores.

4.4 Conclusiones

Gracias a la etnografía enfocada que se realizó a través de tres pacientes con Alzheimer en etapa inicial, se constató la versatilidad de la aplicación, además de la ayuda que brinda tanto a cuidadores como a quienes sufren de la enfermedad, al facilitar su custodia, a más de disminuir el impacto de los síntomas que se presentan con la enfermedad en su etapa inicial, permitiendo al paciente contar con un asistente digital que brinde notificaciones oportuna a sus tutores sobre su ubicación y sus actividades por realizar.

Partimos de pruebas realizadas en un dispositivo iOS de última generación, mediante el cual los cuidadores pudieron configurar la aplicación con rapidez y sin mayores problemas para ponerla a disposición de sus pacientes en cuestión de minutos, configurando sus datos personales con la fotografía tanto del cuidador como la del paciente, lo que permite al paciente reconocerse visualmente tanto a sí mismo como a su cuidador, ya que en la etapa inicial de la enfermedad los pacientes suelen olvidar los rostros de sus seres queridos y también su propio nombre.

La totalidad de los custodios que probaron la aplicación aseguraron no haber utilizado previamente ninguna aplicación móvil que les permitiera de alguna manera facilitar el cuidado de sus pacientes. Los tutores afirmaron que estarían dispuestos a utilizar la aplicación móvil que se desarrolla para esta tesis ya que sus características, tales como obtener la ubicación del paciente en caso que exceda un perímetro establecido, notificar vía mail al cuidador cuando el paciente solicite ayuda remota, así como los recordatorios programados, hacen de la aplicación móvil un asistente digital de fácil uso tanto para pacientes como cuidadores. Según las propias expresiones de los custodios participantes en la etnografía, hubieran querido usar esta aplicación desde meses atrás, cuando sus pacientes recibieron el diagnóstico de Alzheimer; en el caso de uno de los pacientes, hubiera resultado de gran ayuda hace

un par de meses cuando lastimosamente perdió el conocimiento de su lugar de vivienda y estuvo extraviado por varias horas, en una zona cercana a su domicilio pero al no recordar la ubicación, no pudo regresar y sus familiares tuvieron que buscar ayuda de la Policía Nacional para poder dar con su paradero.

Para brindar un mejor asistente digital a través de la aplicación móvil que se desarrolla para esta tesis, los cuidadores han recomendado desarrollarla para dispositivos más económicos, siendo aquellos que se ejecutan bajo el sistema operativo Android, el cual brinda características similares en cuando al rendimiento de iOS, además que algunos de los dispositivos más contemporáneos que se ejecutan bajo esta plataforma, cuentan con las características de hardware necesarias que necesitamos en esta tesis como son el uso del GPS, además de conexión a internet y alertas locales.

CAPITULO 5.

Conclusiones.

Al ser una enfermedad degenerativa que no tiene cura hasta el momento, el Alzheimer representa un gran reto para la medicina moderna en tratar de buscar una cura definitiva, siendo un problema que afecta cada día a millones de personas en edad adulta alrededor del mundo, sin distinción de razas ni clases sociales, convirtiéndose en un problema que no solamente involucra a quienes la padecen si no a todo su entorno familiar.

iOS, el sistema operativo móvil con el cual trabajamos en esta tesis, se presenta como una solución muy potente y de gran aceptación a nivel mundial, sobre la cual se desarrolló la aplicación móvil para pacientes con alzheimer en etapa inicial, integrando tecnologías como el uso del gps en tiempo real, notificaciones locales para recordatorios tanto al paciente como a su cuidador, además de una interfaz táctil sobre la cual el usuario puede interactuar de una manera fácil, sin la asistencia de un tercero, con el objetivo de mejorar la calidad de vida de quienes padecen de la enfermedad como de quienes velan por el cuidado de los mismos.

A través de la memoria técnica implementada en el Capítulo 3, demostramos paso a paso el diseño y desarrollo del software en el entorno de desarrollo Xcode con el lenguaje de programación orientado a objetos denominado Objective-C, abarcando desde el diseño de interfaz de cada uno de sus ventanas disponibles, además de los controles utilizados y su código de programación, permitiendo conocer las

posibilidades de interacción entre el usuario y el control disponible en Xcode. Todos los controles utilizados en la aplicación tienen una interacción y un comportamiento definido dentro de iOS, permitiendo a los usuarios conocer de antemano su funcionamiento y como va a reaccionar el mismo en los diferentes escenarios de la aplicación. Para más información acerca de todos los controles disponibles en iOS y su interfaz de desarrollo Xcode pueden referirse a la documentación publicada por Apple Inc. en su libro iOS Application Programming Guide, (Apple Inc) además de su libro Programming with Objective-C (Apple Inc)

La aplicación cuenta con dos tipos de interacciones principalmente, una dedicada exclusivamente para el cuidador y otra para el paciente; siendo el cuidador quien se encarga de configurar la aplicación con la información necesaria para el correcto funcionamiento de la misma, con datos como el perfil del paciente, recordatorios sobre actividades que debe realizar así como medicamentos recetados al paciente, además de el perímetro de movimiento permitido dentro del cuál los custodios pueden desplazarse con tranquilidad, con la opción de notificación inmediata a sus cuidadores en el caso de que excedan dicho perímetro, generando una notificación vía mail al cuidador con la ubicación exacta localizada mediante el gps del dispositivo, con posibilidad de visualización inmediata de dicha posición a través de internet, demostrando el potencial del uso de dispositivos móviles y del gps incluido dentro de los mismos.

A través del uso de la cámara fotográfica del dispositivo móvil se permite integrar fotos del paciente y su cuidador en la aplicación, ayudando de esta manera a que el custodio tenga siempre presente una identificación visual tanto de su persona como de su cuidador, a quien el custodio puede recurrir en caso de necesitar ayuda, a través del botón de ayuda inmediata, siendo este el único botón disponible para el paciente en la ventana dedicada exclusivamente para él, minimizando la posibilidad de

interacción con otros controles que no son necesarios, además de evitar una posible experiencia de usuario negativa, considerando que el paciente es una persona de la tercera edad, a la cuál le debe resultar fácil e intuitivo, a primera vista, el uso de la aplicación móvil.

Uno de los síntomas en la etapa inicial del Alzheimer es el olvido constante de actividades a realizar por parte de quien sufre la enfermedad, necesitando ayuda de terceros para recordar sus actividades, generando una dependencia entre el paciente y su cuidador, dependencia que se minimiza con el uso de recordatorios dentro de la aplicación desarrollada para esta tesis, en la cuál el cuidador ingresa las actividades que debe realizar el paciente, así como medicamentos que debe tomar a horas determinada; dichos alertas no solamente se generan en el dispositivo del paciente, también se envían vía mail al cuidador permitiendo conocer las actividades que debe realizar su custodio.

Todas las funcionalidades de la aplicación se pusieron a prueba mediante la etnografía enfocada realizada a tres pacientes con alzheimer en etapa inicial, la cual se basó fundamentalmente en la observación del participante, con el fin de descubrir cómo los pacientes utilizan la aplicación en su vida cotidiana y cuál es su propia perspectiva (visión emic) (Salgado, 2014).

Los participantes en la etnografía fueron seleccionados aleatoriamente gracias a la ayuda del Dr. Fabián Guapisaca, geriatra del Hospital del Río en la ciudad de Cuenca, quien nos facilitó los datos de contacto de sus pacientes que cumplían con el requisito de tener alzheimer en etapa inicial, a los cuales visitamos en sus domicilios

y realizamos las pruebas respectivas con la aplicación ejecutándola en un dispositivo iOS de última generación, conjuntamente con sus cuidadores, quienes supieron informarnos que en su totalidad nunca habían utilizado un asistente electrónico para facilitar el tratamiento de la enfermedad de sus custodios, generando una primera impresión positiva en el uso de la aplicación entre pacientes y cuidadores.

Dentro de las funcionalidades de la aplicación que mayor aceptación tuvieron dentro de pacientes y cuidadores que participaron en la etnografía pudimos constatar que la posibilidad de definir un perímetro de movimiento para el paciente, así como el botón de ayuda fueron los más destacados por los usuarios, los cuales permitieron que sus cuidadores tengan una forma de controlar la ubicación de sus custodios en el caso de que éstos salgan de su domicilio o vecindario, previniendo situaciones en las que se habían visto involucrados muchos de los pacientes que realizaron la etnografía cuando habían salido de sus viviendas y se habían extraviado al perder la memoria sobre su lugar de residencia, generando preocupación entre sus grupo familiar, que vieron en esta característica de la aplicación una solución efectiva para no volver a pasar por esta penosa situación, funcionalidad que se replica en el accionar del botón de ayuda que está siempre disponible para el paciente, el cual al ser presionado por el custodio genera la misma notificación vía mail a su cuidador con la ubicación exacta de donde fue generada.

La aplicación móvil para pacientes con alzheimer en etapa inicial, desarrollada en esta tesis cumple con las expectativas para ayudar a pacientes y cuidadores a facilitar el tratamiento de esta enfermedad degenerativa, con un asistente móvil que permite el monitoreo de sus actividades y minimizando la dependencia de los pacientes a contar siempre con un cuidador para actividades que, en la etapa inicial de esta enfermedad, pueden realizarlas con una dependencia parcial pero siempre con la seguridad de contar con la ayuda inmediata de sus familiares si lo requieren.

Referencias

Eisendrath S, Lichtmacher J. Trastornos psiquiátricos. Delirio, demencia y otros trastornos cognitivos. Mexico: El Manual Moderno, 2006.

Apple Inc. iOS Application Programming Guide. Cuppertino: Apple Inc, 2010.

Google Inc. Google Maps. 27 de Marzo de 2015. 27 de Marzo de 2015 <maps.google.com>.

Martín Fidel Romano, Maria Daniela Nissen, Noelia Maria Del Huerto Paredes Dr. Carlos Alberto Parquet. «Enfermedad de Alzheimer.» Revista de Posgrado de la Via Catedra de Medicina No. 175 (2007): 12.

Isaacs, Ellen. "Ethnography." TEDx Broadway. New York: TED, 2013.

Apple Inc. «Programming with Objective C.» Inc, Apple. Programming with Objective C. Cuppertino: Apple Inc, 2014. 127.

Salgado, Francisco. Etnografía enfocada en redes de innovación latinoamericanas. . Quito: Universidad Andina Simon Bolivar, 2014.

Glosario.

Alzheimer: Enfermedad mental progresiva que se caracteriza por una degeneración de las células nerviosas del cerebro y una disminución de la masa cerebral.

Apple Inc: Empresa multinacional estadounidense que diseña y produce equipos electrónicos y software, con sede en Cupertino, California.

App Store: Servicio para el iPhone, el iPod Touch, el iPad y Mac OS X Snow Leopard o posterior, creado por Apple Inc, que permite a los usuarios buscar y descargar aplicaciones informáticas de iTunes Store o Mac App Store en el caso de Mac OS X, desarrolladas con el iPhone SDK y publicadas por Apple.

Darwin BSD: Sistema que subyace en Mac OS X, cuya primera versión final salió en el año 2001 para funcionar en computadoras Macintosh.

e-Salud: Práctica de cuidados sanitarios apoyada en tecnologías de la información y las comunicaciones (TIC).

GPS: Sistema americano de navegación y localización mediante satélites.

iOS: Sistema operativo móvil de Apple. Originalmente desarrollado para el iPhone (iPhone OS), siendo después usado en dispositivos como el iPod Touch, iPad y el Apple TV. La interfaz de usuario de iOS está basada en el concepto de manipulación directa, usando gestos multitáctiles.

iPad: Línea de tabletas digitales portátiles diseñadas y comercializadas por Apple Inc.

iPhone: Línea de teléfonos inteligentes diseñada y comercializada por Apple Inc.

iPod: Línea de reproductores de audio digital portátiles diseñados y comercializados por Apple Inc.

Objective-C: Lenguaje orientado a objetos que desciende del lenguaje de programación C.

OS X: Nombre de una línea de sistemas operativos gráficos desarrollados y vendidos por la compañía Apple Inc.

SDK: (Software Development Kit) Conjunto de herramientas y programas de desarrollo que permite al programador crear aplicaciones para un determinado paquete de software.

Storyboard: Conjunto de ilustraciones mostradas en secuencia con el objetivo de servir de guía para entender una historia.

UNIX: Sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969, por un grupo de empleados de los laboratorios Bell de AT&T.

Xcode: Entorno de desarrollo integrado (IDE, en sus siglas en inglés) de Apple Inc. y se suministra gratuitamente junto con Mac OS X. Xcode trabaja conjuntamente con Interface Builder, una herencia de NeXT, una herramienta gráfica para la creación de interfaces de usuario.

Bibliografía.

Apple Inc. «Programming with Objective C.» Inc, Apple. Programming with Objective C. Cuppertino: Apple Inc, 2014. 127.

Apple Inc. iOS Application Programming Guide. Cuppertino: Apple Inc, 2010.

Doctora Jenny Ríos Coello, Secretaria de la Facultad de Ciencias de la Administración de la Universidad del Azuay,

C E R T I F I C A:

Que, el H. Consejo de Facultad en sesión realizada el 08 de enero del 2015 conoció la petición del estudiante **Juan Sebastián Landy Ríos** con código, que denuncian su trabajo de titulación (tesis) **"APLICACIÓN MOVIL PARA PERSONAS CON ALZHEIMER EN ETAPA INICIAL"**, previa a la obtención del Grado de Ingeniero de Sistemas. El Consejo de Facultad acoge el informe de la Junta Académica y aprueba la denuncia de tesis. Designa como Director al ingeniero Francisco Salgado Arteaga y como miembros del Tribunal Examinador a los ingenieros Luis Calderón Peralta y Fernando Balarezo Rodríguez.

La Disposición Transitoria Quinta literal b) del Reglamento de Régimen Académico aprobado el 21 de noviembre de 2013 y reformado el 9 de abril de 2014, establece: " Quienes finalizaron sus estudios a partir del 21 de noviembre de 2008 podrán titularse … en un plazo máximo de 18 meses a partir de la vigencia del presente Reglamento…" estos es hasta el **20 DE MAYO DE 2015**.

Cuenca, enero 9 de 2015

UNIVERSIDAD DEL AZUAY
FACULTAD DE ADMINISTRACIÓN
SECRETARIA

CONVOCATORIA

Por disposición de la Junta Académica de Ingeniería de Sistemas y Telemática, se convoca a los Miembros del Tribunal Examinador, a la sustentación del Protocolo del Trabajo de Titulación **"Aplicación móvil para personas con Alzheimer en etapa inicial"**, presentado por el estudiante Juan Sebastián Landy Ríos con código 39747, previa a la a la obtención del grado de Ingeniero de Sistemas, para el día **MARTES 9 DE DICIEMBRE 2014 A LAS 18H30.**

Cuenca, 3 de diciembre de 2014

Dra. Jenny Ríos Coello
Secretaria de la Facultad

Ing. Francisco Salgado Arteaga .....

Ing. Luis Calderón Peralta .....

Ing. Fernando Balarezo Rodríguez .....

Oficio Nro. 142-2014-DIST-UDA

Cuenca, 28 de Noviembre de 2014

Señor Ingeniero
Xavier Ortega Vázquez
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMNISTRACIÓN
Presente.-

De nuestras consideraciones:

La Junta Académica de la Escuela de Ingeniería de Sistemas y Telemática, reunida el día 28 de Noviembre del 2014, recibió el proyecto de tesis titulado "Aplicación móvil para personas con Alzheimer en etapa inicial", presentada por el estudiante Juan Sebastián Landy Ríos, estudiante de la Escuela de Ingeniería de Sistemas y Telemática, y revisado por Francisco Salgado Ph.D. previo a la obtención del título de Ingeniero de Sistemas.

La Junta solicita por su digno intermedio notificar al tribunal designado y determinar lugar, fecha y hora de sustentación.

Por lo expuesto, y de conformidad con el Reglamento de Graduación de la Facultad, recomienda como director y responsable de aplicar cualquier modificación al diseño del trabajo de graduación posterior a Francisco Salgado Ph.D. y como miembros del Tribunal al Ing. Luis Calderón y Ing. Fernando Balarezo.

Atentamente,

Ing. Marcos Orellana Cordero
Director Escuela de Ingeniería de Sistemas y Telemática
Universidad del Azuay

Sustentación del Diseño de Tesis (DOCTORA JENNY RIOS COELLO)

Fecha: 02-12-2014

ESCUELA DE INGENIERIA DE SISTEMAS

Diseños de Tesis
Escuela de Ingeniería de Sistemas

Estudiante: Juan Sebastián Landy Ríos con código 39747.
Tema: "APLICACIÓN MOVIL PARA PERSONAS CON ALZHEIMER EN ETAPA INICIAL"
Para: La obtención del título de Ingeniero de Sistemas
Director: Ing. Francisco Salgado
Tribunal: Ing. Luis Calderón
Tribunal: Ing. Fernando Balarezo

DIA: Martes.

FECHA: 9 De Diciembre De 2014

HORA: 18h30.

UNIVERSIDAD DEL AZUAY

ACTA
SUSTENTACIÓN DE PROTOCOLO/DENUNCIA DEL TRABAJO DE TITULACIÓN

1.1 **Nombre del estudiante:** Juan Sebastián Landy Ríos
1.1.1 Código 39747
1.2 **Director sugerido**: Ing. Francisco Salgado Arteaga
1.3 **Codirector (opcional):** _____
1.4 **Tribunal:** Ing. Luis Calderón Peralta / Ing. Fernando Balarezo Rodríguez
1.5 **Título propuesto:** "Aplicación móvil para personas con Alzheimer en etapa inicial"
1.6 **Resolución**:

1.6.1 Aceptado sin modificaciones ✓

1.6.2 Aceptado con las siguientes modificaciones:

_____

_____

_____

- Responsable de dar seguimiento a las modificaciones: Ing. Francisco Salgado Arteaga

1.6.3 No aceptado
- Justificación:

_____
_____
_____

Tribunal

Francisco Salgado
.....
Ing. Francisco Salgado Arteaga

.....
Ing. Luis Calderón Peralta

.....
Ing. Fernando Balarezo Rodríguez

.....
Sr. Juan Sebastián Landy Ríos

Dra. Jenny Ríos Coello
Secretario de Facultad

Fecha de sustentación: Martes 9 de diciembre de 2014

18h30

UNIVERSIDAD DEL AZUAY

RÚBRICA PARA LA EVALUACIÓN DEL PROTOCOLO DE TRABAJO DE TITULACIÓN

1.1 Nombre del estudiante: Juan Sebastián Landy Ríos
1.1.1 **Código** 39747
1.2 Director sugerido: Ing. Francisco Salgado Arteaga
1.3 Codirector (opcional):.
1.4 Título propuesto: "Aplicación móvil para personas con Alzheimer en Etapa Inicial"
1.5 Revisores (tribunal): Ing. Luis Calderón Peralta / Ing. Fernando Balarezo Rodríguez
1.6 Recomendaciones generales de la revisión:

	Cumple totalmente	Cumple parcialmente	No cumple	Observaciones (*)
Línea de investigación				
1. ¿El contenido se enmarca en la línea de investigación seleccionada?	✓			
Título Propuesto				
2. ¿Es informativo?	✓			
3. ¿Es conciso?	✓			
Estado del arte				
4. ¿Identifica claramente el contexto histórico, científico, global y regional del tema del trabajo?	✓			
5. ¿Describe la teoría en la que se enmarca el trabajo	✓			
6. ¿Describe los trabajos relacionados más relevantes?	✓			
7. ¿Utiliza citas bibliográficas?	✓			
Problemática y/o pregunta de investigación				
8. ¿Presenta una descripción precisa y clara?	✓			
9. ¿Tiene relevancia profesional y social?	✓			
Hipótesis (opcional)				
10.¿Se expresa de forma clara?	N/A			No aplica
11.¿Es factible de verificación?	N/A			No aplica
Objetivo general				
12.¿Concuerda con el problema formulado?	✓			
13.¿Se encuentra redactado en tiempo verbal infinitivo?	✓			
Objetivos específicos				
14.¿Concuerdan con el objetivo	✓			

general?				
15.¿Son comprobables cualitativa o cuantitativamente?	✓			
Metodología				
16.¿Se encuentran disponibles los datos y materiales mencionados?				N/A
17.¿Las actividades se presentan siguiendo una secuencia lógica?	✓			
18.¿Las actividades permitirán la consecución de los objetivos específicos planteados?	✓			
19.¿Los datos, materiales y actividades mencionadas son adecuados para resolver el problema formulado?				N/A
Resultados esperados				
20.¿Son relevantes para resolver o contribuir con el problema formulado?				
21.¿Concuerdan con los objetivos específicos?	✓			
22.¿Se detalla la forma de presentación de los resultados?	✓			
23.¿Los resultados esperados son consecuencia, en todos los casos, de las actividades mencionadas?	✓			
Supuestos y riesgos				
24.¿Se mencionan los supuestos y riesgos más relevantes?				N/A
25.¿Es conveniente llevar a cabo el trabajo dado los supuestos y riesgos mencionados?				
Presupuesto				
26.¿El presupuesto es razonable?	✓			
27.¿Se consideran los rubros más relevantes?	✓			
Cronograma				
28.¿Los plazos para las actividades son realistas?	✓			
Referencias				
29.¿Se siguen las recomendaciones de normas internacionales para citar?	✓			
Expresión escrita				
30.¿La redacción es clara y fácilmente comprensible?	✓			
31.¿El texto se encuentra libre de faltas ortográficas?	✓			

(*) Breve justificación, explicación o recomendación.

- Opcional cuando cumple totalmente,

- Obligatorio cuando cumple parcialmente y NO cumple.

.....

.....

.....

.....	.....	.....
Ing. Francisco Salgado A.	Ing. Luis Calderón P.	Ing. Fernando Balarezo R.

UNIVERSIDAD DEL AZUAY

DOCTORA JENNY RIOS COELLO SECRETARIA DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACION

C E R T I F I C A:

Que, el Señor **Juan Sebastián Landy Ríos**, inició sus estudios en la Escuela de Ingeniería de Sistemas, ingresó el 19 de Septiembre de 2005 y egresó de la Facultad el 31 de Julio de 2012.

Cuenca, 04 de Diciembre de 2014

UNIVERSIDAD DEL AZUAY FACULTAD DE ADMINISTRACION SECRETARIA

Derecho N° 111802

vcf

UNIVERSIDAD DEL AZUAY

Cuenca, 28 de noviembre 2014

Sr. Ingeniero
Xavier Ortega Vázquez
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
Presente.-

De mis consideraciones:

Yo, Juan Sebastián Landy Ríos, estudiante egresado de la carrera de Ing. Sistemas, con código 39747, solicito a usted aprobar mi proyecto de tesis titulado "Aplicación móvil para personas con Alzheimer en etapa inicial", revisado por Francisco Salgado Ph.D, previo a la obtención del título de Ingeniero de Sistemas.

Por la favorable acogida que se sirva dar a la presente solicitud le anticipo mis agradecimientos.

Atentamente,

Juan Sebastián Landy Ríos
Código: 39747

UNIVERSIDAD DEL AZUAY

Cuenca, noviembre 28 de 2014

Ingeniero

Xavier Ortega Vásquez

Decano de la Facultad de Ciencias de la Administración

Presente

De mi consideración:

Por la presente, me permito informarle que he revisado el diseño de trabajo de titulación presentado por el estudiante **Juan Sebastián Landy Ríos**, con el tema *"Aplicación móvil para personas con Alzheimer en etapa inicial"* como requisito previo para la obtención del título de Ingeniero en Sistemas y Telemática.

El diseño del trabajo presenta una estructura teórica, metodológica y técnica coherente para la realización de la investigación propuesta. Por lo expuesto, emito informe favorable y recomiendo su aprobación.

Atentamente

Francisco Salgado

Prof. Francisco Salgado Arteaga, Ph. D.

UNIVERSIDAD DEL AZUAY

Facultad de Ciencias de la Administración
Escuela de Ingeniería de Sistemas y Telemática

Oficio Nro. 142-2014-DIST-UDA

Cuenca, 28 de Noviembre de 2014

Señor Ingeniero
Xavier Ortega Vázquez
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMNISTRACIÓN
Presente.-

De nuestras consideraciones:

La Junta Académica de la Escuela de Ingeniería de Sistemas y Telemática, reunida el día 28 de Noviembre del 2014, recibió el proyecto de tesis titulado "Aplicación móvil para personas con Alzheimer en etapa inicial", presentada por el estudiante Juan Sebastián Landy Ríos, estudiante de la Escuela de Ingeniería de Sistemas y Telemática, y revisado por Francisco Salgado Ph.D. previo a la obtención del título de Ingeniero de Sistemas.

La Junta solicita por su digno intermedio notificar al tribunal designado y determinar lugar, fecha y hora de sustentación.

Por lo expuesto, y de conformidad con el Reglamento de Graduación de la Facultad, recomienda como director y responsable de aplicar cualquier modificación al diseño del trabajo de graduación posterior a Francisco Salgado Ph.D. y como miembros del Tribunal al Ing. Luis Calderón y Ing. Fernando Balarezo.

Atentamente,

Ing. Marcos Orellana Cordero
Director Escuela de Ingeniería de Sistemas y Telemática
Universidad del Azuay

UNIVERSIDAD DEL AZUAY

Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas

Aplicación móvil para personas con Alzheimer en etapa inicial.

Protocolo de Trabajo de Titulación

Autor:

Juan Sebastián Landy Ríos.

Director de Tesis:

Prof. Dr. Francisco Salgado

Cuenca - Ecuador

2014

UNIVERSIDAD DEL AZUAY

1. DATOS GENERALES

1.1. **Nombre del estudiante**: Juan Sebastián Landy Ríos

1.1.1. **Código**: 39747

1.1.2. **Contacto**: 07-4191305, 0984179941, jlandyr@gmail.com

1.2. **Director Sugerido**: Salgado, Francisco, Prof. Dr.

1.2.1. **Contacto**: 099-948-1332, francisco.salgado@fulbrightmail.org

1.3. **Co-director sugerido (opcional):**

1.4. **Asesor Metodológico (opcional):**

1.5. **Tribunal designado:**

1.6. **Aprobación:**

1.7. **Línea de investigación de la Carrera**: Informática de computadores

1.7.1. **Código UNESCO**: 1203.17

1.7.2. Tipo de trabajo

1.7.2.1. Proyecto Técnico

1.7.2.2. Investigación formativa

1.8. **Área de estudio**: Programación para dispositivos móviles

1.9. **Titulo propuesto**: Aplicación móvil para personas con Alzheimer en etapa inicial.

1.10. **Subtitulo**: (opcional)

1.11. **Estado del proyecto:** Nuevo.

2. CONTENIDO

2.1. **Motivación de la investigación:** Brindar a las personas con Alzheimer en etapa inicial, un asistente digital para solventar parcialmente los inconvenientes de esta enfermedad.

2.2. **Problemática:** Las personas que padecen Alzheimer son incapaces de realizar tareas cotidianas sin la ayuda de una persona, privándose de su autonomía. En etapas leves y medianas no disponen de un asistente electrónico que les permita realizar sus tareas básicas y no frustrarse por la pérdida de memoria, ubicación, noción del tiempo, etc. No solo los pacientes con Alzheimer son quienes sufren, sino también sus familiares, ya que no pueden dejarlos solos por temor a que por su pérdida de memoria puedan necesitar ayuda y no tener a quien recurrir.

2.3. **Pregunta de investigación:** ¿Como una aplicación móvil puede ayudar a personas con Alzheimer en etapa inicial en sus tareas básicas?

2.4. **Resumen:** La creación de un asistente virtual para quienes sufren de Alzheimer será de gran ayuda en la incorporación de los mismo dentro de la sociedad, permitiéndoles su autonomía parcial con la seguridad de que sus seres queridos tendrán conocimiento de su ubicación y monitoreo constante. Este trabajo permitirá conocer de que manera las aplicaciones móviles pueden ser una ayuda para personas con discapacidades físicas o mentales.

2.5. **Estado del Arte y Marco Teórico:**

La enfermedad de Alzheimer es un trastorno neurológico que provoca la muerte de las células nerviosas del cerebro. Por lo general, la enfermedad de Alzheimer comienza paulatinamente y sus primeros síntomas pueden atribuirse a la vejez o al olvido común. (Martín Fidel Romano, 2007)

A medida en que avanza la enfermedad, se van deteriorando las capacidades cognitivas, entre ellas la capacidad para tomar decisiones y

llevar a cabo las tareas cotidianas, dificultades de orientación, problemas de lenguaje, y pueden surgir modificaciones de la personalidad, así como conductas problemáticas. La pérdida de memoria llega hasta el no reconocimiento de familiares o el olvido de habilidades normales para el individuo. (Martín Fidel Romano, 2007)

Las tres etapas que figuran a continuación representan la progresión general de la enfermedad.

Etapa 1 (inicial): Temprano en la enfermedad, las personas con Alzheimer tienden a ser menos enérgicas y espontáneas. Muestran pérdida mínima de la memoria y cambios de humor, y son lentos para aprender y reaccionar. Ellos se hacen aislados, evitan la gente y nuevos lugares y prefieren lo familiar.

Etapa 2 (moderada): En esta etapa, la persona con la enfermedad de Alzheimer está claramente convirtiéndose en discapacitada. Los individuos pueden todavía realizar tareas simples independientemente, pero pueden necesitar la ayuda con actividades más complicadas. Ellos olvidan los acontecimientos recientes y su historia personal, y cada vez son más desorientados y desconectados de la realidad.

Etapa 3 (grave): Durante esta fase final, la gente puede perder la capacidad para alimentarse a sí misma, hablar, reconocer personas y el control de las funciones corporales. Su memoria se agrava y puede llegar a ser casi inexistente. La atención constante es típicamente necesaria. En un estado físico debilitado, el paciente puede llegar a ser vulnerable a otras enfermedades y problemas respiratorios, sobre todo cuando tiene que estar confinado a la cama. (Martín Fidel Romano, 2007)

El uso de aplicaciones móviles en el ámbito de la salud se remonta desde el año 2008, cuando Apple liberó su SDK para desarrolladores. Desde entonces han sido varias las soluciones que se han creado para solventar distintos problemas en cuanto a salud se refiere.

Objective-C es el lenguaje de programación principal que se utiliza para el desarrollo de aplicaciones OS X e iOS. Es un superconjunto del lenguaje de programación C y proporciona capacidades orientadas a objetos y un tiempo de ejecución dinámico. (Apple Inc, 2014)

Xcode es el IDE utilizado para el desarrollo de aplicaciones OS X e iOS, en el se escribe el código, se diseña las interfaces gráficas, se prueba y depuran las aplicaciones. (Apple Inc, 2014)

2.6. **Hipótesis (opcional)**

2.7. **Objetivo general:** Desarrollar una aplicación iOS para brindar autonomía parcial a personas con Alzheimer en la etapa inicial de la enfermedad.

2.8. **Objetivos específicos:**

2.8.1. Desarrollar una aplicación iOS para iPhone que integre:

2.8.1.1. Geolocalizar al usuario estableciendo un perímetro de tránsito.

2.8.1.2. Alertar al acompañante del paciente en caso que éste último salga del perímetro de tránsito establecido.

2.8.1.3. Alertar con actividades por realizar ó medicamentos que deben tomar los pacientes.

2.8.1.4. Solicitar ayuda remota desde la aplicación.

2.8.2. Desarrollar una memoria técnica de la aplicación en el lenguaje de programación utilizado "Objective C".

2.8.3. Analizar las características de la aplicación en la vida real.

2.8.3.1. Indagar sobre la utilización de aplicación mediante una etnografía enfocada a 3 personas con Alzheimer en etapa inicial. Se realizará fundamentalmente en base a la técnica de observación participante, con el fin de descubrir cómo los pacientes utilizan la aplicación en su vida cotidiana (Isaacs, 2013; Salgado, 2014)

2.8.3.2. Analizar los datos de la indagación de campo (etnografía) para determinar las correcciones o ajustes que deben hacerse a la aplicación.

2.9. **Metodología:** Se ha visto conveniente que para la creación de la aplicación iOS se debe realizar las siguientes etapas de investigación y desarrollo.

Investigación.

- Se utilizará el método de análisis orientado a objetos para el modelado del problema y de su posterior solución.
- Recopilación y análisis bibliográfico, publicaciones, artículos, etc, relacionados con iOS, lo cuál permitirá conocer los preceptos teóricos necesarios para entender el funcionamiento del sistema operativo para móviles iOS.
- Analizar las necesidades básicas en pacientes con Alzheimer que podrían solventarse con el uso de la aplicación móvil.

Desarrollo.

- El desarrollo de la aplicación se realizará en el lenguaje de programación Objective-C.
- Se realizará un diagrama de casos de uso que represente el comportamiento entre el usuario y la aplicación.
- Se desarrollará una interfaz intuitiva para facilitar el uso de la aplicación a los pacientes con Alzheimer y sus acompañantes.

Pruebas de la aplicación.

- Se realizarán pruebas en pacientes con Alzheimer en su etapa inicial para comprobar en que medida las características de la app resultan útiles.

2.10. **Alcances y resultados esperados:** Al finar el trabajo se obtendrá la aplicación móvil que se podrá instalar en dispositivos iOS con el sistema

operativo iOS 8 o superior. Además, se presentará un cuadro con información estadística acerca de las pruebas realizadas en pacientes con Alzheimer en el cuál se podrá observar en que medida, las características de la app, son útiles en los pacientes.

2.11. Supuestos y riesgos:

2.11.1. No contar con pacientes con Alzheimer en etapa inicial para las pruebas: En este caso, proponemos realizar las pruebas con voluntarios, los cuáles simularán tener los síntomas iniciales de la enfermedad.

2.12. Presupuesto:

Rubro – denominación	Costo USD (detalle)	Justificación ¿para qué?
Computador portátil MacBook Air	$1100	Desarrollo del documento escrito. Tabulación de las pruebas de la aplicación realizadas. Programación de la aplicación.
Smartphone iPhone 5S	$500	Pruebas de la aplicación
Plan de internet móvil para el Smartphone	$20	Pruebas de la aplicación que necesitan conexión a internet
TOTAL	$1620	

2.13. **Financiamiento:** La fuente del financiamiento para este proyecto serán provistos por el autor del mismo.

2.14. Esquema tentativo:

Introducción El Alzheimer.

1.1. Introducción.

1.2. Desarrollo de la enfermedad.

1.3 Etapas de la enfermedad.

2. Sistema Operativo iOS.

UNIVERSIDAD DEL AZUAY

2.1. Introducción.
2.2. Acerca del desarrollo iOS.
2.2.1. Arquitectura de iOS.
2.2.2. Que contiene el iOS SDK?
3. Lenguaje Objective-C
3.1. Introducción
3.2. Ambiente de ejecución
3.2.1 Objetos
3.2.2 Clases
4. Ambiente de desarrollo Xcode.
4.1. Introducción
4.2. Características.
5. Guía de Interfaz de la aplicación.
5.1. Introducción.
5.2. Características de la plataforma.
5.3. Principios de la interfaz.
6. Manual de Desarrollo iOS.
6.1. Creación del proyecto en Xcode.
6.2. Uso de Storyboards.
6.3. Servicios de localización.
6.4. Pruebas en el dispositivo iOS.
7. Evaluación de la aplicación en un contexto real
7.1. Etnografía enfocada sobre el uso de la aplicación
7.2. Análisis de la indagación de campo
7.3. Ajustes de la aplicación en base al análisis de su uso en un contexto real.
8. Conclusiones.
9. Bibliografía.

2.15. **Cronograma**

2.16 Referencias

Objetivo Específico	Actividad	Resultado esperado	Tiempo (semanas)
Geolocalizar al usuario estableciendo un perímetro de tránsito.	Desarrollar del sistema de localización del Smartphone para establecer la ubicación del usuario y su campo de acción permitido	La aplicación permitirá ubicar al paciente con Alzheimer a través del Smartphone.	2
Alertar al acompañante del paciente en caso que exceda el perímetro de acción permitido.	Desarrollar del sistema de notificaciones.	El cuidador recibirá una notificación vía mail en el caso que el paciente exceda su rango de acción permitido.	2
Crear una lista de recordatorios para el paciente con Alzheimer	Desarrollar de la lista de recordatorios	El cuidador puede agregar, editar o eliminar ítems de dicha lista	1
Solicitar ayuda remota desde la aplicación.	Integrar un sistema de ayuda remota dentro de la aplicación iOS	El paciente podrá solicitar ayuda remota al presionar un botón, el cuidador recibirá la ubicación exacta en la que se generó la alerta	2
Desarrollar una memoria técnica de la aplicación en el lenguaje de programación utilizado "Objective C".	Creación del manual conjuntamente con el desarrollo de la aplicación	Un manual de referencia en la creación de la aplicación	2
Analizar las características de la aplicación en la vida real	Etnografía enfocada a pacientes con Alzheimer en etapa inicial, por medio de observación participante del uso de la aplicación en la realidad	Datos procedentes de la indagación de campo, sobre la utilización de la aplicación por pacientes concretos en la vida cotidiana.	2

Martín Fidel Romano, M. D. (2007). Enfermedad de Alzheimer. *Revista de Posgrado de la Vía Catedra de Medicina No. 175* , 12.

Isaacs, E. (2013). Ethnography. *TEDx Broadway*. New York: TED.

Apple Inc. (2014). Programming with Objective C. En A. Inc, *Programming with Objective C* (pág. 127). Cuppertino, California, EEUU: Apple Inc.

Salgado, F. (2014). Etnografía Enfocada en *Redes de innovación latinoamericanas*. Quito, Universidad Andina Simón Bolívar, Sede Ecuador.

2.16. **Anexos:**

2.17. **Firma de responsabilidad (estudiante)**

Francisco Salgado

2.18. **Firma de responsabilidad (director sugerido)**