Universidad Carlos III de Madrid
Escuela politécnica superior

Ingeniería técnica en informática de gestión
Proyecto de fin de carrera

Desarrollo de una aplicación móvil con cifrado de datos por geo posición

Autor: Jesús Martín Chicharro Gallego
Tutor: Jorge Blasco Alís

Octubre 2013

Agradecimientos

No quería empezar este documento sin hacer un inciso sobre las personas que me han ayudado a llegar donde estoy actualmente. En primer lugar quiero agradecer al tutor, Jorge Blasco Alís, la comprensión y aguante durante todo este periodo de tiempo y la implicación que siempre ha mantenido en este proyecto.

Sin embargo, el mayor agradecimiento, como no podría ser de otra forma, se lo tengo que dar a mi familia:

- A mis padres, Andrés Chicharro y Mercedes Gallego, por haber cimentado mi personalidad e inculcarme unos valores que han hecho que llegue a convertirme en la persona que soy hoy. Además de ofrecerme el cariño y comprensión durante todo este largo camino.
- A mi hermano, Raúl Chicharro, por estar siempre cuando le he necesitado sin pedir nada a cambio.
- Abuelos, tíos, primos, por haberme apoyado durante todos estos años y que me pueda sentir orgulloso de pertenecer a esta familia.

No podía terminar de escribir este apartado sin acordarme de mi otra familia, mis amigos. Gracias a todos los Illescanos/as, ya sean de nacimiento o de adopción por haberme aguantado, y lo que os queda, todos estos años. Siempre me habéis motivado para que llegase a donde he llegado hoy.

Muchas gracias,
Jesús Martín Chicharro Gallego

Índice

Índice de figuras

Índice de tablas

Índice de Requisitos

Índice de Pruebas

Capítulo 1: Introducción

1.1 Motivación

La seguridad de los datos es una necesidad básica, se pueden utilizar estos datos para usurpar la identidad de otro individuo o empresa, estafar, en definitiva realizar actividades delictivas sin que el dueño de los datos sea consciente de ello . Hoy en día se maneja una cantidad de información enorme, por ello se busca la utilización de aplicaciones que puedan guardar información sensible. Uno de estos contenedores de datos pueden ser dispositivos móviles, como por ejemplo *iPhone, iPad*, dispositivos con sistema operativo Android, Blackberry, etc.

Un ejemplo de dispositivos herméticos es la gama de movilidad de *Apple, iPhone, iPod e iPad*. Disponen de un hardware con una capacidad computacional que permite ejecutar tareas como si de un ordenador de sobremesa se tratase, es decir, que se pueden implantar técnicas criptográficas estándar para la protección de los datos.

Además existen multitud de algoritmos de cifrado en el mercado. Los usuarios más avanzados ven la necesidad de poder elegir que algoritmo van a utilizar para cifrar sus datos mientras que para el resto de usuarios el poder elegir que algoritmo utilizar ofrece más versatilidad a la hora de utilizar un algoritmo libre de errores.

Por tanto, se plantea una aplicación que sea capaz de proteger los datos de forma transparente para el usuario y que además sea configurable. Por ello se realizará un estudio del panorama actual comparando el esfuerzo de la realización de la aplicación con los distintos tipos de tecnologías disponibles, Figura 2, teniendo en cuenta además, el esfuerzo de aprendizaje de estas tecnologías.

Figura 2: Tecnologías

La motivación de este proyecto surgió por la necesidad de tener un contenedor de datos en dispositivos móviles y que además fuese seguro.

Con el auge de los dispositivos móviles y la necesidad de tener los datos siempre disponibles este tipo de aplicaciones son de gran utilidad en escenarios empresariales, como por ejemplo, los técnicos de sistemas o de soporte que necesitan acceder a gran número de sistemas informáticos con credenciales que no pueden estar a la vista.

Figura 3: Cifrado de los datos

Para asegurarse que los puntos anteriores se puedan llevar a cabo se plantea la búsqueda de un método de cifrado, Figura 3, acorde a las necesidades, que los datos no puedan ser interpretados por un humano y/o máquina.

Además del sector empresarial, en un escenario personal. También existe la necesidad de tener los datos siempre disponibles. Por ello se ha querido generar una aplicación con gran capacidad de adaptación del usuario. La ventaja que ofrece la autenticación con el GPS es la separación que puede ofrecer para la parte personal y laboral.

Además ofrece la posibilidad de acceder a los datos con el modo tradicional de usuario y contraseña en el caso de que el usuario no se encuentre en la posición del GPS, Figura 4.

Figura 4: Geo localización en un dispositivo iOS

Gracias a los sistemas de distribución que hay disponibles actualmente, incluso dentro de las empresas con los productos destinados a la administración remota de los dispositivos, este tipo de aplicaciones ofrece una gran oportunidad de crecimiento económico a grandes, pequeñas empresas e incluso a desarrolladores en solitario. La inversión que se necesita es muy baja en comparación a aplicaciones para otras plataformas. La posibilidad de que el producto llegue al mayor número de usuarios es enorme, debido a que Apple incorpora en sus dispositivos una aplicación que permite la búsqueda y compra de aplicaciones desde el propio terminal (App Store, Figura 5).

Figura 5: App Store e iPhone

Además los beneficios económicos no se producen únicamente con la compra del producto sino que se pueden añadir ganancias mediante la publicidad incrustada en la aplicación. También permite la actualización del producto de forma sencilla y transparente para el usuario.

En un mundo en el que los datos tienen que estar disponibles de forma inmediata, actualizada y segura, las aplicaciones para dispositivos móviles han crecido de manera exponencial por lo que introducirse en este mercado a corto plazo ofrece unas posibilidades de crecimiento muy altas.

A todo esto se suman los conceptos de "nube de información", donde las posibilidades son infinitas. Guardar información en la "nube" o ejecutar funciones remotas permiten al dispositivo transformarse en increíbles centros de datos donde lo importante es la información siendo independiente del medio donde se acceda a estos.

Con lo que el perfil de "Especialista en Movilidad" se vuelve indispensable en el sector empresarial actual.

1.2 Objetivos

Los objetivos que se van a alcanzar con el desarrollo de la aplicación son los siguientes:

1. Ser capaz de establecer la posición del usuario para ofrecer servicios derivados de ella.

   Se realizará un estudio sobre los métodos y/o herramientas que ofrece el dispositivo para obtener la geo localización del usuario, que se utilizará en procesos internos de la aplicación. Además de obtener la posición se debe asegurar que ésta sea fidedigna y no sea manipulable.

2. Gestor de información. La aplicación deberá poder gestionar la información que el usuario quiera almacenar utilizando para ello una interfaz amigable y de fácil manejo. Se implementará una herramienta de control y gestión de la información añadida a la aplicación.

   La aplicación está dotada de un gestor de carpetas para categorizar los distintos tipos de datos que almacena. Además cada nuevo objeto se puede personalizar añadiéndole atributos.

3. Sistema de seguridad. La aplicación deberá garantizar que la información que se almacena en el dispositivo no se verá comprometida. Además de las implementaciones propias del proyecto se aplicarán mecanismos propios del dispositivo para aumentar la seguridad de la aplicación.

   La autenticación se llevara a cabo mediante dos métodos:

   - Posición GPS actual del dispositivo.
   - Mediante par de claves Usuario/Contraseña.

   La posición del GPS es el modo de autenticación por defecto. En caso de que esta autenticación falle debido a que el dispositivo no se encuentre en la posición correcta existe la posibilidad de acceder a los datos mediante un par de claves Usuario/Contraseña.

4. Aplicación multi usuario.

   Se ha pensado en la posible expansión del producto a otros dispositivos móviles compartidos. Como puede ser un Tablet, tanto a nivel empresarial como en el hogar. Además de poder utilizar varias cuentas y que éstas sean independientes aunque las utilice el mismo usuario, como por ejemplo, una cuenta para el ocio y otra para el trabajo.

5. Aprender a desarrollar una aplicación para el sistema operativo iOS.

   El producto final es una aplicación compatible con todos los dispositivos móviles basados en iOS. Siendo un entorno seguro de almacenamiento de información confidencial altamente configurable. Entre estas opciones se encuentran el tipo de autenticación y el algoritmo de encriptación que utilizará la aplicación.

1.3 Contenido de la memoria

La memoria está estructurada de la siguiente forma:

- Capítulo 1: Introducción.

  El propósito de este capítulo es sentar las bases para el lector sobre el desarrollo del proyecto.

- Capítulo 2: Análisis de los componentes del proyecto.

  En este capítulo se recoge el análisis de las funcionalidades que componen la aplicación a desarrollar. Se realizará un estudio de mercado de las soluciones de la competencia. Además de analizar los motivos por los que se ha elegido la plataforma móvil, se realizará un estudio técnico de los distintos componentes que se integrarán en la aplicación como son las opciones existentes en sistemas de localización, bases de datos y tipos de cifrado. Por último se detallan los requisitos y las pruebas de aceptación del producto.

- Capítulo 3: Diseño de la aplicación.

  En este capítulo se detalla a bajo nivel el desarrollo de la aplicación. Desarrollando el diseño técnico que detalla el cómo se van a implementar los distintos componentes de la aplicación.

- Capítulo 4: Implementación del sistema.

  Se detalla cómo se han implementado los diseños técnicos explicados en el capítulo anterior.

- Capítulo 5: Gestión del proyecto.

  En este capítulo se detalla todo el trabajo de la jefatura del proyecto. Estableciendo planificaciones, presupuestos así como un estudio de mercado y posibilidades de venta del producto.

- Capítulo 6: Conclusión y líneas futuras.

  Se establecen las ideas adquiridas a la finalización del proyecto y los pasos a seguir en el futuro, así como nuevos desarrollos dentro del proyecto.

- Capítulo 7: Glosario de términos.

  En esta parte del documento se realiza una descripción de los términos técnicos que se utilizan dentro del mismo.

Capítulo 2: Análisis

2.1 Introducción

El desarrollo de un proyecto se lleva a cabo por la necesidad de resolver una problemática. En este capítulo se buscará la información necesaria para acometer dicho proyecto. A continuación se describen los principales puntos de los que se ha realizado un análisis, así como de los caso de uso, requisitos y pruebas:

- Smartphone. Comparativa de dispositivos.
- Sistemas de localización.
- Gestores de bases de datos móviles.
- Tipos de cifrado.
- Competencia. Aplicaciones similares en el mercado.
- Casos de uso.
- Requisitos software.
- Casos de prueba.

2.2 Comparativa de plataformas móviles

El primer paso a la hora de elegir una plataforma en la que desarrollar es comprobar que el tipo de dispositivo, y a su vez, de usuario cumple una serie de características que se ajusten a la identidad de la aplicación. Por ello se han estudiado los puntos clave de los dispositivos que hoy triunfan en el mercado.

2.2.1 Crecimiento

Siguiendo un estudio comparativo [1], entre la cuota de mercado que tenían en Julio de 2010 con la que presentan en el mismo mes de 2011 en la zona euro (Alemania, Francia, España, Inglaterra e Italia) se obtienen los siguientes resultados:

Plataforma Smartphone	Julio 2010	Julio 2011	%Variación
Symbian	53.9%	37.8%	-16.1
Google	6.0%	22.3%	+16.2
Apple	19.0%	20.3%	+1.2
RIM	8.0%	9.4%	+1.5
Microsoft	11.5%	6.7%	-4.8

Tabla 1: Crecimiento de los dispositivos

Con esta comparativa se comprueba que los dispositivos que utilizan el sistema operativo de Google, Android. Han visto aumentada su cuota de mercado,

Tabla 1. Los fabricantes principales de dispositivos con este sistema operativo son HTC y Samsung.

En el otro lado encontramos a los dispositivos que utilizan Symbian, principalmente Nokia. De hecho, y es el motivo principal por el que se ha desechado la idea de desarrollar la aplicación en Symbian, es la reciente alianza entre Microsoft y Nokia para que ésta última instale Windows Phone 7/ Windows 8 en la serie Lumia y el nuevo sistema operativo de Nokia llamado Asha, dejando en el olvido a Symbian.

Como conclusión, si solo tenemos en cuenta estos datos sería lógico pensar que la plataforma a elegir para el desarrollo de la aplicación es la creada por Google. Sin embargo, y como veremos más adelante, la rentabilidad que ofrece esta plataforma para el desarrollador es inferior a otras plataformas con menor cuota de mercado. Gracias a este estudio se ha descartado el desarrollo en las siguientes plataformas.

- **Symbian:** Sistema operativo anticuado y que pronto se dejará de utilizar.
- **Microsoft:** La cuota de mercado es bastante pequeña. A la espera de que Windows Phone/Windows 8 pueda competir con iOS o Android.

2.2.2 Sistema de distribución de aplicaciones

Un punto importante a la hora de elegir un sistema en el que desarrollar es comprobar las facilidades que ofrece el dispositivo para ofertar tu aplicación al cliente. En este caso se van a estudiar los modelos de distribución de las plataformas iOS, Android y BB.

- **App Store (iOS)**, Figura 7: Es un servicio que permite buscar aplicaciones, que Apple permite instalar, para los dispositivos con iOS instalado. Una ventaja a la hora de distribuir aplicaciones es que desde el propio dispositivo puedes acceder al App Store y comprar e instalar las aplicaciones de forma transparente al usuario. A esto se le une que un tercio de las aplicaciones en el App Store son de pago y las aplicaciones favoritas de los usuarios también son de pago. Un aliciente a la hora de elegir plataforma, que perfila a los usuarios del dispositivo.

Figura 7: App Store logo

- **Google Play (Android):** El gran competidor del App Store, sin embargo al contrario de éste, la mayoría de las aplicaciones son gratuitas. Solo un 1,3% de las aplicaciones son de pago, según Piper Jaffray [2] Tabla 2. Y en muchos casos, gracias a la facilidad que ofrecen los dispositivos para instalar sistemas operativos modificados, se facilita al usuario la utilización de aplicaciones bajadas de forma ilegal. Por lo que para el desarrollador este sistema de distribución no es tan efectivo como el de los competidores. Aunque para este sistema existen más métodos de pago, a parte de los habituales, tarjeta de crédito, PayPal se le suma el poder añadir el pago de las aplicaciones a la factura del operador.

	Aplicaciones descargadas	Ingresos brutos	Ingresos desarrolladores	% Aplicaciones de pago	Precio Medio
Android Market	6.750.000.000	341.765.335 $	239.235.734 $	1.3%	$3,79
App Store	18.566.331.811	4.939.611.127 $	3.457.727.789 $	13.5%	$2.01

Tabla 2: Estimación Piper Jaffray [2].

- **BlackBerry App World (BlackBerry)**, Figura 8: Dentro del marco empresarial, aplicaciones internas para empleados, el desarrollo de aplicaciones para esta plataforma está muy extendido, sin embargo, fuera de éste la plataforma está perdiendo clientes y por consiguiente consumidores de aplicaciones.

Figura 8: BB App World

En conclusión, se descarta el desarrollo para BlackBerry debido a que el consumo de aplicaciones es menor que la de sus competidores.

2.2.3 Análisis económico

En todo proyecto el objetivo final es conseguir beneficios. Por este motivo es importante hacer un estudio de la viabilidad y de la inversión necesaria para acometer dicho proyecto.

En el caso que nos ocupa y habiendo limitado la búsqueda a dos posibles plataformas, Android o iOS. Se tendrán en cuenta las siguientes características a la hora de emprender el proyecto.

- Inversión inicial.
- Mantenimiento de la aplicación.
- Modelo de ingresos.
- Piratería.

2.2.3.1 Inversión Inicial

En cualquiera de los dos casos se supone que partimos de cero.

- En el caso de la plataforma de Apple, es necesaria la compra de un ordenador MAC con el que desarrollar. Además a la hora de publicar la aplicación es necesario pertenecer a la comunidad de desarrolladores de Apple. Para acceder a dicha comunidad es necesario pagar una cuota de 90€ anuales. Este pago te permite la distribución de las aplicaciones a través del App Store, previo paso por el comité de aceptación de Apple. El IDE de desarrollo y el SDK se pueden bajar de forma gratuita desde el portal de desarrollador de Apple.

- En Android, Figura 9, no existe la limitación del sistema en el que desarrollar. La herramienta de desarrollo puede ser Eclipse y el SDK se descarga de forma gratuita desde la página de Android. Además del gasto del equipo de desarrollo hay que añadir el coste por desplegar aplicaciones al Market de Android que es un único pago de 25$.

Figura 9: Android Market

Además hay que tener en cuenta el sistema de aceptación de aplicaciones por cada una de las empresas.

- En Apple la aplicación debe pasar por un rígido control de contenidos y rendimiento, para asegurar la calidad de las aplicaciones disponibles en el App Store. Esto que es un punto a favor desde la vista del usuario, puede provocar grandes dolores de cabeza al desarrollador. El tiempo medio de espera de aprobación de una aplicación está en torno a 1 semana, pero se puede alargar incluso meses. Con lo que la perdida de dinero puede ser notable.

- Para Google este sistema no ofrece la agilidad necesaria para que los desarrolladores publiquen sus proyectos en el menor tiempo posible. Las aplicaciones no deben pasar tantos filtros, solo pasan controles de seguridad para asegurarse que la aplicación no accede a partes del sistema para la que no está diseñada. La tasa de rechazo de aplicaciones es del 1%.

En ambos casos se hace un reparto de los beneficios entre el desarrollador y las empresas. Normalmente el reparto es del 70% para el desarrollador y el 30% para la compañía. Sin embargo también hay que pagar las tasas de la tarjeta de crédito que se restan del 70% de beneficio del desarrollador. Estas tasas suelen ser del 1-2% del beneficio.

2.2.3.2 Mantenimiento de la aplicación

Para las aplicaciones para el sistema operativo Android, una vez subida a Google Play, no se produce ningún gasto por el mantenimiento de ésta. Sin embargo, en el caso de Apple, si se deja de mantener la cuenta de desarrollador asociada a la aplicación, la aplicación desaparece del App Store. Por lo que conlleva un gasto anual de 90€.

2.2.3.3 Modelo de ingresos

En los dos casos la principal forma de general ingresos es mediante la venta de la aplicación. Sin embargo también existe la posibilidad de utilizar publicidad en la aplicación.

- **iAd en Apple:** Se basa en añadir unos espacios dedicados a publicidad en la aplicación que permiten al usuario visitar la publicidad que se muestra. Los beneficios derivados de esta práctica se basan en accesos a la publicidad. Por cada 1000 impresiones Apple paga en torno a 15-30$. Varía dependiendo del éxito de la aplicación.

- **AdMod en Google:** Se basa en el mismo principio que iAd. Sin embargo el importe obtenido por cada 1000 impresiones es menor. En torno a 1$.

En cuanto a los datos de ventas de aplicaciones. En Android en torno al 97% de las aplicaciones son gratuitas. Esto quiere decir que el usuario de Android es reacio a pagar por una aplicación. En muchos casos es debido a la facilidad que existe en este dispositivo para obtener aplicaciones de manera ilegal.

El usuario de iOS compra de media 83 aplicaciones en el App Store. Esto sumado a los datos de gratuidad de Android pone en cabeza en cuanto a beneficios a los dispositivos de Apple. Además de la posibilidad de lanzar una aplicación con las dos modalidades. De pago y otra gratuita con anuncios. En Android casi estas limitado a esta última versión a la hora de lanzar una aplicación.

En 2011 las descargas desde el App Store se han incrementado en un 61% en 2011 y su precio ha aumentado un 14%. El precio medio es de 1,44$.

Según Google el 80% de las aplicaciones de pago descargadas desde la plataforma Android solo fueron descargadas 100 veces. Con lo que recuperar la inversión en este caso es casi imposible.

Como conclusión, la plataforma de Apple ofrece más posibilidades de ingresos que la de Google. Esto es un peso muy importante a la hora de elegir la plataforma en la que desarrollar nuestra aplicación.

2.2.3.4 Piratería

Toda aplicación está expuesta a sufrir el intento de obtención de forma ilegal. Y este riesgo aumenta en la plataforma de Google. El ratio de descargas ilegales entre Android e iOS es de un 14:1 [2].

Sin embargo, el porcentaje de los dispositivos en los que se ha utilizado el método de liberación en Apple apenas superan el 13%, que permiten utilizar aplicaciones descargadas ilegalmente. Aunque sigue siendo un problema para los desarrolladores, éste es menor que en los dispositivos que utilizan Android como sistema operativo.

Según un estudio publicado en *Yankee Group* [3], el 53% de los desarrolladores de la plataforma de Google ve la piratería un problema grave, de los cuales el 27% no elige Android como plataforma principal de negocio debido a que considera este problema como un escollo imposible de salvar. Además de esto, tres cuartas partes de los desarrolladores creen que es muy fácil descargar una aplicación de Google Play, descompilarla, realizar modificaciones sobre ésta y republicarla, de los cuales la mitad ven esta práctica como extremadamente fácil. De hecho, es una práctica habitual entre los usuarios de Android descompilar clases del sistema operativo para modificar aspectos de éste [3].

2.2.4 Hardware

Los dispositivos móviles están aumentando muy rápido su potencia, incluso se empieza a realizar comparativa con algunos ordenadores portátiles y/o sobremesa. Por lo que cada vez es más importante buscar software específico que pueda hacer uso de toda la capacidad que puede ofrecer.

Gracias a iOS, se puede obtener todo el potencial que ofrece el hardware de los dispositivos de Apple. Esto no ocurre de igual forma en los dispositivos con Android. Con más de 60 fabricantes distintos, utilizando componentes distintos, hace que Android no se focalice en un hardware específico. Haciendo para el desarrollador una tarea más pesada, debido a que se deben generar, por ejemplo, gráficos para los distintos hardware, etc.

Además el dispositivo debe de ser capaz de obtener la posición en la que se encuentra. Es decir, debe cumplir algunas características hardware como son disponer de algún tipo de conexión como las que se muestran a continuación:

- Conexión GPRS y EDGE (2.5G y hasta 384 Kbps).
- Conexión UMTS (3G y hasta 3.6 Mbps).
- Conexión HSDPA (3.5G y hasta 7.2 Mbps).
- Conexión WIFI.

2.2.5 Elección del dispositivo

Aunando todos los puntos fuertes de cada una de las dos plataformas líderes en el mercado tenemos que:

- **Android:** Líder en cuanto a cuota de mercado peca por una gran fragmentación de dispositivos y que gracias a contar con dispositivos relativamente económicos y un gran sistema operativo ha conseguido superar en cuota de mercado a Apple, sin embargo el mercado para los desarrolladores Android es bastante complicado, debido a que el consumidor de aplicaciones de este sistema operativo es menos receptivo a las aplicaciones de pago.

- **iOS:** En este caso el peso del App Store prevalece sobre sus competidores, en cuanto a número de descargas. El usuario de iOS gasta más dinero en aplicaciones, comparándolo con el resto de plataformas. A esto se le suma la baja fragmentación de dispositivos, un sistema operativo optimizado y sencillo en su utilización.

Por lo tanto la elección del dispositivo de desarrollo está clara. No hay dudas que Android es un digno competidor a nivel de usuario para Apple, pero desde el punto de vista del desarrollador, y también desde el punto de vista personal, iOS supera a todos sus rivales.

El sistema operativo en el cual se va a desarrollar la aplicación es iOS 3+. Esto hace que la aplicación sea compatible con todos los sistemas operativos hasta el actual iOS 7.

2.3 Sistemas de localización

Un peso importante en el proyecto es el sistema de identificación mediante geo posicionamiento. En los dispositivos basados en iOS podemos obtener la posición utilizando los servicios de localización. También conocidos como LBS.

2.3.1 Aplicaciones LBS

Estas aplicaciones obtienen la posición, latitud, longitud y altitud, del hardware que la ejecuta. En la mayoría de los casos deben ser dispositivos móviles capaces de obtener direcciones IP. Estas aplicaciones se componen de cinco elementos.

- **Dispositivo Móvil:** En algunos casos es el que inicia la petición de localización, pero en otros la petición se hace mediante una acción de push hacia el dispositivo.
- **Red de comunicación:** Es el canal de transmisión de datos entre el dispositivo y el resto del mundo.
- **Componente de posicionamiento:** Se encarga de establecer la posición física del dispositivo. Principalmente son el GPS, red de comunicación móvil o redes WIFI.
- **Proveedor de servicios:** Procesa la petición del usuario y se encarga de devolverle los datos provenientes del contenedor de contenidos.
- **Proveedor de contenidos:** Es el encargado de almacenar la información.

Para la aplicación que se desarrolla en este proyecto el proveedor de servicios y el proveedor de contenidos son el propio dispositivo ya que la información está contenida en el teléfono y la petición una vez obtenida la posición se auto gestiona.

2.3.2 Técnicas de localización

Un dispositivo iPhone tiene disponible tres tipos de localización.

- **A-GPS:** Se basa en obtener la posición del dispositivo triangulando a través de tres satélites. Este tipo de localización es la más precisa de las tres.
- **Mediante WPS** (*WIFI Positioning System*): Este sistema se basa en la triangulación a través de redes WIFI. Se realiza a través de la dirección MAC del dispositivo y su posición relativa a tres redes (*Hotspot*). Suple la carencia del posicionamiento GPS en interiores. Sin embargo, la precisión es inferior a la del GPS, en torno a 20 metros menos.

- **Mediante torres de telefonía** (*Cell ID*): En este caso, se necesita de tres antenas de telefonía para realizar la triangulación para obtener la posición. Este tipo de localización es el menos preciso de los tres, en torno a 200 metros.

Para el tipo de aplicación que nos ocupa, ya que necesitamos una cierta precisión a la hora de obtener la posición del dispositivo, los métodos de localización que utilizará son A-GPS y WPS. No se deshecha la opción de triangulación mediante las torres de telefonía, que aunque da una precisión menor, en muchos casos es suficiente.

2.4 Gestores de datos móviles

El proyecto que vamos a acometer, en líneas generales, es un contenedor de información. Los dispositivos móviles, aunque cada vez en menor medida, son dispositivos con características muy limitadas. Por ello se deben gestionar los recursos de forma que estas limitaciones no sean un problema a la hora de utilizar la aplicación.

Los datos se han de almacenar en el dispositivo y de forma permanente, es decir, deben mantenerse entre ejecuciones. Por ello se ha realizado un estudio de los tipos de almacenamiento permanente que ofrece el dispositivo.

- **Core Data, Figura 12:** Permite la organización de los datos siguiendo un modelo relacional entidad-atributo, muy parecido al diagrama de clases. iOS automáticamente cifra estos datos para que no sean accesibles. Sin embargo, no permite una configuración exhaustiva de este cifrado, por lo que esta opción es descartada debido a que en nuestro proyecto se busca poder manipular el tipo de cifrado que se realiza a los datos.

Figura 12: Core Data

- **Sqlite 3.0:** Es un sistema gestor de bases de datos que gracias a los pocos recursos que consume del sistema, es una de las opciones más utilizadas en cuanto a almacenamiento de información persistente. De hecho, Core Data gestiona una base de datos Sqlite.

- **UltraliteJ:** Es un sistema gestor de bases de datos que utilizado junto a productos como *Mobilink* de *Sybase*, permite realizar replicación entre bases de datos consolidadas y bases de datos en dispositivos móviles. Debido a que en este proyecto no se plantea la sincronización con bases de datos externas se descarta esta opción.

Como conclusión, el sistema que se va a utilizar para gestionar la base de datos es un sistema Sqlite 3.0. Este sistema no cifra la información de la base de datos, lo que nos permitirá crear una clase para la gestión de ésta con la que elegir el tipo de cifrado que el usuario quiere utilizar.

2.5 Tipos de cifrado

En la actualidad los dispositivos móviles tienen unas características hardware cercanas a algunos ordenadores de sobremesa.

El sistema operativo de Apple ofrece al desarrollador los siguientes tipos de cifrados para utilizar en su aplicación:

- AES.
- DES.
- 3DES.
- CAST.
- RC4.
- RC2.

2.5.1 Cifrado AES

Estándar de criptografía simétrica. Se caracteriza por ser inmune a los ataques conocidos, tener un diseño sencillo y poder implementarse en la mayoría de los escenarios conocidos.

2.5.3 Cifrado 3DES

El algoritmo DES tiene el problema de que la capacidad de cálculo actual es tan elevada que es posible romper la seguridad. Para añadir más seguridad al cifrado DES se creó en 1990 un sistema de generación de claves que aumentaba la complejidad del cálculo de la clave aumentando el tamaño de ésta a 112 bits.

Este sistema necesita de más recursos para el cifrado y descifrado puesto que concatena el cifrado con 3 claves distintas. Además pasa de un clave en el sistema DES a 3 claves en el sistema 3DES

Actualmente este cifrado ha sido sustituido por el conocido como AES.

2.5.4 Cifrado CAST

Este sistema sigue un algoritmo de cifrado muy similar al utilizado por DES. Sin embargo utiliza claves con longitudes mayores a este, 128 y 256 bits. Fue candidato a (AES) y se utiliza como estándar en varios productos como GPG y PGP.

Para cifrar se siguen los siguientes pasos:

- Computación de la clave en 16 sub-claves.
- Parte el texto en bloques de 64 bits y los separa en sub-bloques de 32 bits llamados L_0 y R_0.
- Se realiza una iteración de 16 rondas en las cuales se llevan a cabo una serie de operaciones sobre los bloques.
- Por último se realiza un intercambio de L_{16} y R_{16} y se concatenan obteniendo el texto cifrado.

El descifrado utiliza el mismo algoritmo pero siguiendo los pasos al revés.

Las rondas que se producen el paso tres no son todas iguales. Existen 3 tipos de rondas:

Tipo 1:

$$I = ((K_{mi} + D) <<< K_{ri})$$

$$f = ((S_1[I_a] \oplus S_2[I_b]) - S3[I_c]) + S_4[I_d]$$

Tipo 2:

$$I = ((K_{mi} \oplus D) <<< K_{ri})$$

$$f = ((S_1[I_a] - S_2[I_b]) + S_3[I_c]) \oplus S_4[I_d]$$

Tipo 3:

$$I = ((K_{mi} - D) <<< K_{ri})$$

$$f = ((S_1[I_a] + S_2[I_b]) \oplus S_3[I_c]) - S_4[I_d]$$

Rondas 1, 4, 7, 10, 13, y 16 son del tipo 1.
Rondas 2, 5, 8, 11, y 14 son del tipo 2.
Rondas 3, 6, 9, 12, y 15 son del tipo 3.

La descripción del algoritmo ha sido consultada en un artículo publicado por la Universidad de Michigan [5].

2.5.5 Cifrado RC4

Es un sistema de cifrado simétrico de flujo y a diferencia de los anteriores el algoritmo de cifrado y descifrado era secreto. Solo se tenía acceso al binario de éste. El RC4 es comúnmente utilizado en protocolos como WEP y WAP en sistemas de comunicación inalámbrica y en SSL.

Se caracteriza por ser un generador de números pseudo aleatorio a partir de una clave de 256 bits. Es uno de los más rápidos del mercado.

Se basa en generar una clave de flujo a la que se le realiza una operación XOR con el texto en plano lo que genera el texto cifrado.

El algoritmo se compone de dos fases:

- Creación de la llave a través de la clave, Key Scheduling Algorithm (KSA):

```
j = 0
for i = 0 .. 255
      S[i] = i
for i =0 .. 255
      j = (j + S[i]) + key [i mod key_length] ) mod 256
```

- Pseudo-Random Generation Algorithm (PRGA):
  Operación XOR entre el texto y la llave generada.

Este cifrado junto a RC2 es uno de los más débiles de los que ofrece el sistema operativo y no es recomendable su utilización.

La descripción del algoritmo ha sido obtenida de un artículo publicado por Simon Josefsson [6].

2.5.6 Cifrado RC2

Se basa en un algoritmo de cifrado de bloque con una clave de 64 bits. Este algoritmo es el más inseguro de los comentados en este documento.

2.6 Competencia. Alternativas en el mercado

Todas las aplicaciones existentes en el mercado ofrecen la posibilidad de guardar los datos del dispositivo de forma segura utilizando una clave maestra.

A la hora de desarrollar una nueva aplicación es importante conocer los elementos diferenciadores de nuestra aplicación con las ya existentes en el mercado. Por ello se han definido algunos elementos que no se han encontrado en otras aplicaciones.

- Sistema de autenticación mediante posición geográfica.
- Sistema de cifrado a elección del usuario.
- Sistema de gestión de categorías.
- Utilización de una interfaz amigable en cuanto a uso para el usuario.

Es importante conocer las aplicaciones que ya hay en el mercado para que al finalizar el proyecto se haya creado una aplicación competitiva. Por ello se han estudiado las aplicaciones que tienen una funcionalidad similar a la que se va a desarrollar. Entre ellas tenemos:

2.6.1 1Password

Posiblemente la aplicación, Figura 19, más extendida entre los usuarios de dispositivos Apple. Entre las características que ofrece:

- Posibilidad de almacenamiento de contraseñas en el dispositivo.
- Sincronización con cuentas Dropbox.
- Sistema gestor de contenidos.
- Cifrado de copias de seguridad.
- Sistema de seguridad con clave de 4 dígitos o clave maestra.

Uno de los puntos negativos que tiene la aplicación es la integración de un navegador web embebido para alguna funcionalidad que hace la usabilidad empeore. Produciendo algunos casos de error molestos para el usuario, como pedir los datos de identificación varias veces durante la ejecución de ésta.

Figura 19: 1Password

2.6.2 LastPass

Sistema basado en la sincronización dispositivo-servidor. Utilizado para identificación automática con sitios web. No es una aplicación nativa y se basa en el navegador web.

2.7 Diagrama de casos de uso

A continuación se especifica la funcionalidad del sistema a partir del diagrama de casos de uso. Se ha extendido el diagrama ya que algunos casos de uso, Figura 20, engloban a los referidos.

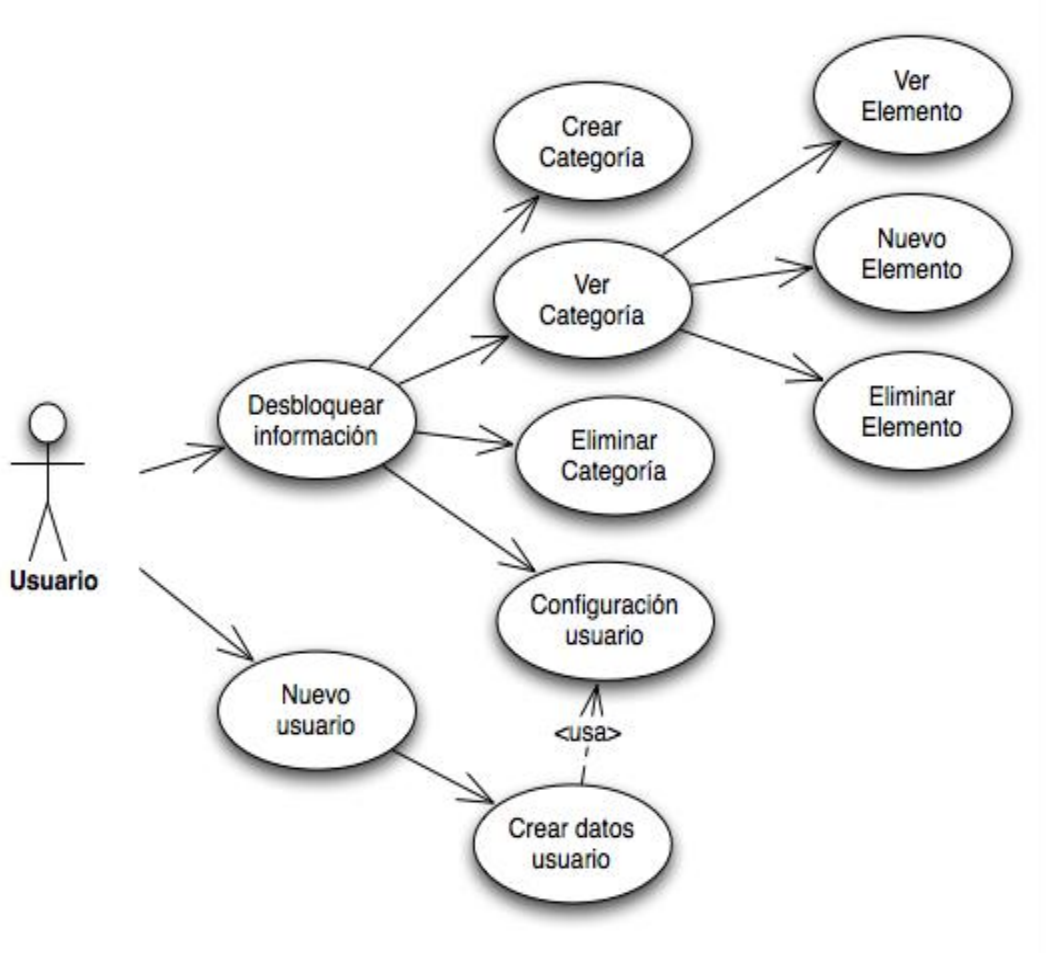

Figura 20: Diagrama de casos de uso

Al iniciar la aplicación el usuario puede o bien crear un usuario con una configuración especifica o bien desbloquear los datos relativos al usuario. Una vez se ha identificado el usuario correctamente, el usuario hace uso del gestor de contenidos. Este gestor se encarga de categorizar la información que se almacena en él.

2.8 Requisitos software

La aplicación a desarrollar debe cumplir unos requisitos software que se exponen a continuación.

2.8.1 Formato de la especificación de requisitos

A continuación se detalla el formato en el que se van a especificar los requisitos de la aplicación.

- **Identificador**: Valor unívoco identificativo del requisito.
- **Descripción**: Breve comentario explicativo del requisito.
- **Prioridad**: Relativo a la importancia del requisito.
- **Verificabilidad**: Prueba que desarrolla el requisito.

2.8.2 Requisitos funcionales

Identificador	RF-01
Descripción	El programa permitirá crear nuevos usuarios
Prioridad	Alta
Verificabilidad	CP-04,CP-06

Requisito 1: Creación de usuario

Identificador	RF-02
Descripción	El programa mostrará la ubicación actual del dispositivo.
Prioridad	Baja
Verificabilidad	CP-01,CP-02,CP-03,CP-04

Requisito 2: Ubicación actual

Identificador	RF-03
Descripción	El programa debe de permitir la identificación a través de la posición.
Prioridad	Alta
Verificabilidad	CP-06,CP-07

Requisito 3: Autenticación

Identificador	RF-04
Descripción	El programa debe permitir la identificación del usuario mediante usuario/contraseña.
Prioridad	Alta
Verificabilidad	CP-02, CP-03, CP-04,CP-06

Requisito 4: Autenticación secundaria

Identificador	RF-05
Descripción	El programa debe permitir la configuración de cifrado de la base de datos.
Prioridad	Alta
Verificabilidad	CP-04

Requisito 5: Cifrado de la base datos

Identificador	RF-06
Descripción	El programa debe poder gestionar los elementos del usuario en categorías.
Prioridad	Media
Verificabilidad	CP-10,CP-11

Requisito 6: Gestión categorías

Identificador	RF-07
Descripción	El programa permitirá crear categorías.
Prioridad	Media
Verificabilidad	CP-10,CP-11

Requisito 7: Creación de categorías

Identificador	RF-08
Descripción	El programa debe permitir la creación de nuevos elementos. Los campos a informar son: – Tipo de categoría. – Usuario. – Contraseña. – Descripción. – URL. – Nota. – Pin. – Número secreto. – Correo electrónico. – Teléfono.
Prioridad	Media
Verificabilidad	CP-12,CP-13,CP-14,CP-15

Requisito 8: Creación de elementos

Identificador	RF-09
Descripción	El programa debe permitir la configuración de los elementos.
Prioridad	Media
Verificabilidad	CP-12,CP-13

Requisito 9: Configuración de elementos

Identificador	RF-10
Descripción	El programa debe permitir el borrado de categorías.
Prioridad	Media
Verificabilidad	CP-16,CP-17

Requisito 10: Borrado de categorías

Identificador	RF-11
Descripción	El programa debe permitir el borrado de elementos.
Prioridad	Media
Verificabilidad	CP-18

Requisito 11: Borrado de elementos

Identificador	RF-12
Descripción	El programa debe permitir la configuración de los usuarios.
Prioridad	Alta
Verificabilidad	CP-6,CP-19,CP-20

Requisito 12: Configuración de usuarios

Identificador	RF-13
Descripción	Los datos de configuración propios de la aplicación deben almacenarse en un repositorio propio del dispositivo.
Prioridad	Alta
Verificabilidad	CP-5, CP-7,CP-8 CP-9

Requisito 13: Repositorio de la configuración

Identificador	RF-14
Descripción	Al borrarse una categoría con elementos. Los elementos deben eliminarse también.
Prioridad	Media
Verificabilidad	CP-16

Requisito 14: Orden las categorías

Identificador	RF-15
Descripción	El programa permitirá la modificación de elementos.
Prioridad	Media
Verificabilidad	CP-15

Requisito 15: Orden de los elementos

2.8.3 Requisitos no funcionales

Identificador	RNF-01
Descripción	El programa debe funcionar en todos los dispositivos basados en el sistema operativo iOS versión 3.0 o superior.
Prioridad	Alta
Verificabilidad	--

Requisito No funcional 1: Compatibilidad de sistema

Identificador	RNF-02
Descripción	No se utilizarán librerías externas al SDK para el desarrollo de la aplicación.
Prioridad	Media
Verificabilidad	--

Requisito No funcional 2: Librerías externas

Identificador	RNF-03
Descripción	La aplicación debe ser nativa.
Prioridad	Alta
Verificabilidad	--

Requisito No funcional 3: Aplicación nativa

Identificador	RNF-04
Descripción	Se deben seguir los estándares de diseño especificados por Apple en "iPhone Human Interface Guidelines".
Prioridad	Alta
Verificabilidad	--

Requisito No funcional 4: Estándares de diseño

Identificador	RNF-05
Descripción	Debe ser compatible con versiones de iOS 3.0 y posteriores.
Prioridad	Alta
Verificabilidad	CP-01 … CP-22

Requisito No funcional 5: Compatibilidad de versiones

Identificador	RNF-06
Descripción	Se utilizará la capa de mapas de "Google Maps" para el posicionamiento del dispositivo.
Prioridad	Baja
Verificabilidad	CP-01,CP-02,CP-03,CP-04,CP-05

Requisito No funcional 6: Utilización de mapas Google

Identificador	RNF-07
Descripción	El idioma será el castellano.
Prioridad	Baja
Verificabilidad	CP-01 … CP-22

Requisito No funcional 7: Idioma

Identificador	RNF-08
Descripción	Las categorías deben ordenarse alfabéticamente.
Prioridad	Baja
Verificabilidad	CP-21

Requisito No Funcional 8 Ordenación de las categorías

Identificador	RNF-09
Descripción	Los elementos deben ordenarse alfabéticamente.
Prioridad	Baja
Verificabilidad	CP-22

Requisito No Funcional 9: Ordenación de los elementos

Identificador	RNF-10
Descripción	La generación de la clave de acceso estará compuesta por el usuario y la posición codificada del usuario.
Prioridad	Media
Verificabilidad	--

Requisito No Funcional 10: Eliminación en cascada de elementos

2.9 Casos de prueba

Se diseñará un plan de pruebas exhaustivo. La aplicación debe cumplir todos los requisitos expuestos en el punto **2.7**. Existen algunas pruebas en lo relativo a la cobertura GPS que se han realizado teniendo en cuenta los distintos niveles de fuerza de la señal.

2.9.1 Formato de la especificación de los casos de pruebas

A continuación se detalla el formato en el que se van a especificar los casos de prueba.

- **Identificador**: Valor unívoco identificativo de la prueba.
- **Descripción**: Breve comentario explicativo de la prueba.

Identificador	CP-01
Descripción	Inicio de la aplicación. Intento de identificación mediante usuario erróneo. Se comprobará que produce un error de identificación.

Prueba 1: Identificación en la aplicación

Identificador	CP-02
Descripción	Inicio de la aplicación. Intento de identificación mediante posición GPS errónea. Se comprobará que produce un error de identificación.

Prueba 2: Posición GPS incorrecta

Identificador	CP-03
Descripción	Inicio de la aplicación. Intento de identificación mediante usuario/contraseña erróneo. Se comprobará que produce un error de identificación.

Prueba 3: Identificación Usuario/Contraseña incorrecta

Identificador	CP-04
Descripción	Inicio de la aplicación. Creación de usuario utilizando la configuración por defecto. Se comprobará que la creación del usuario se lleva a cabo utilizando dicha configuración.

Prueba 4: Creación de usuario con configuración por defecto

Identificador	CP-05
Descripción	Inicio de la aplicación. Creación de usuario sin insertar ningún valor en los campos. Se comprobará que produce un error en la creación.

Prueba 5: Creación de usuario vacío

Identificador	CP-06
Descripción	Inicio de la aplicación. Creación de usuario utilizando unos valores en uso. (Creación de doble usuario). Se comprobará que produce un error en la creación.

Prueba 6: Creación de usuario existente

Identificador	CP-07
Descripción	Inicio de la aplicación. Identificación correcta por primera vez en el historial del usuario. Se muestran las categorías por defecto establecidas en la aplicación. Se comprobará que aparecen las categorías por defecto establecidas.

Prueba 7: Muestreo de categorías por defecto

Identificador	CP-08
Descripción	Inicio de la aplicación. Identificación correcta. Creación de una nueva categoría sin valores en la creación. Se comprobará que produce un error a la hora de la creación.

Prueba 8: Creación de categoría

Identificador	CP-09
Descripción	Inicio de la aplicación. Identificación correcta. Creación de una nueva categoría sin valores en la creación. Se comprobará que produce un error a la hora de la creación.

Prueba 9: Creación de categoría vacía

Identificador	CP-10
Descripción	Inicio de la aplicación. Identificación correcta. Creación de una nueva categoría con valores correctos. Se comprobará que se crea la categoría correctamente.

Prueba 10: Creación de categoría correcta

Identificador	CP-11
Descripción	Inicio de la aplicación. Identificación correcta. Creación de una nueva categoría con título ya existente. Se comprobará que produce un error a la hora de la creación.

Prueba 11: Creación de categoría existente

Identificador	CP-12
Descripción	Inicio de la aplicación. Identificación correcta. Comprobación del contador de elementos de la categoría. Se comprobará si el contador recoge la cantidad exacta de elementos pertenecientes a la categoría.

Prueba 12: Mostrar contador de elementos en las categorías

Identificador	CP-13
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Creación de un nuevo elemento sin introducir valores. Se comprobará que no se produce la creación del elemento.

Prueba 13: Creación de elemento

Identificador	CP-14
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Navegación hacia un elemento. Se elimina dicho elemento. Se comprobará que se elimina correctamente

Prueba 14: Eliminación de elementos

Identificador	CP-15
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Navegación hacia un elemento. Se modificará el elemento Se comprobará que se modifica correctamente.

Prueba 15: Modificación de elementos

Identificador	CP-16
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Se eliminará la categoría creada por el usuario. Se comprobará que se elimina correctamente la categoría creada por el usuario.

Prueba 16: Eliminación correcta de la categoría

Identificador	CP-17
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Se eliminará una categoría creada por la aplicación No es posible la eliminación de la categoría.

Prueba 17: Eliminación fallida de categoría por defecto

Identificador	CP-18
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Navegación hacia un elemento. Borrado de un elemento. Se comprobará que se elimina correctamente.

Prueba 18: Eliminación de elemento

Identificador	CP-19
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a la sección de configuración. Guardar la configuración sin modificar ninguna opción. Se comprobará que no se producen cambios en la configuración.

Prueba 19: Modificación de la configuración

Identificador	CP-20
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a la sección de configuración. Modificar la opción de cifrado. Se comprobará que se guardan los cambios correctamente.

Prueba 20: Modificar la opción de cifrado

Identificador	CP-21
Descripción	Inicio de la aplicación. Identificación correcta. Comprobación del listado. Se comprobará que el listado está ordenado alfabéticamente.

Prueba 21: Comprobación de ordenación del listado

Identificador	CP-22
Descripción	Inicio de la aplicación. Identificación correcta. Navegación hacia una categoría. Comprobación del listado. Se comprobará que el listado está ordenado alfabéticamente.

Prueba 22: Comprobación de ordenación de elementos

A continuación se muestra la matriz de trazabilidad entre requisitos y pruebas, Tabla 3.

R\P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
RF-1				X		X																
RF-2	X	X	X	X																		
RF-3						X	X															
RF-4		X	X	X		X																
RF-5				X																		
RF-6										X	X											
RF-7										X	X											
RF-8												X	X	X	X							
RF-9												X	X									
RF-10																X	X					
RF-11																		X				
RF-12																			X	X		
RF-13					X		X	X	X													
RF-14																					X	
RF-15																						X

Tabla 3: Matriz de trazabilidad entre requisitos y pruebas

Capítulo 3: Diseño

3.1 Introducción

Una vez analizada la problemática que nos ocupa, se ha de diseñar la solución que posteriormente se implementará. En este capítulo se detalla en profundidad el diseño de la solución. Como apartados claves encontramos:

- Arquitectura del software.
- Controladores de vista.
- Diseño de vistas personalizadas.
- Diseño de la base de datos.
- Diagramas de secuencia.

3.2 Arquitectura del software

El patrón de diseño de la arquitectura del software que se utilizará en este proyecto es la Modelo, vista, controlador, (MVC). Este patrón se caracteriza por dividir las clases que componen el proyecto en tres grupos:

- **Modelo:** Son las clases encargadas de la gestión de los datos que caracterizan a la aplicación.
- **Vista:** Se encargan de mostrar la información al usuario. Este tipo de clases deben ser genéricas e independientes del modelo.
- **Controlador:** Su labor es la de traducir los datos del modelo y posteriormente configurar las clases de la vista para mostrar dichos datos.

La comunicación entre grupos de clase está limitada. Las clases relativas al modelo no pueden realizar ninguna comunicación con las de la vista. El motivo de esta limitación es la de generar código reutilizable y cuyo mantenimiento sea sencillo. En el siguiente diagrama, Figura 21, se muestra la intercomunicación entre los grupos de clases que componen el proyecto.

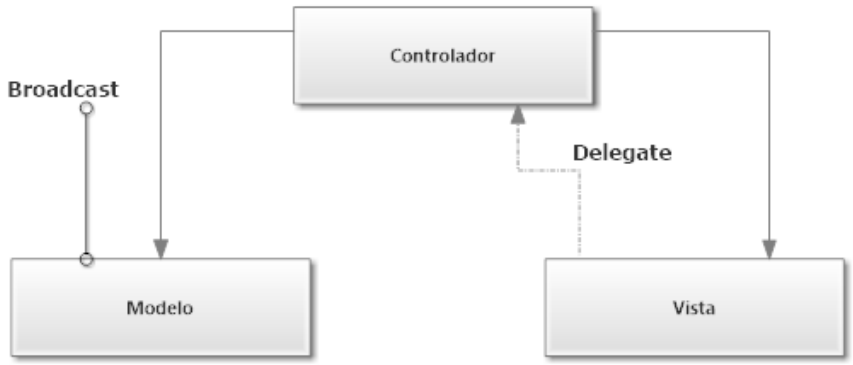

Figura 21: Arquitectura del Software MVC

Como se muestra en el gráfico la comunicación entre clases se limita a:

- Controlador -> Vista.
- Controlador -> Modelo.

Sin embargo existen tipos especiales de comunicación entre los grupos de clases.

- **Modelo -> Controlador:** Es posible esta comunicación gracias al centro de notificaciones (*NSNotificationCenter*). Su misión es la de dar a conocer al controlador cambios en el modelo. El paso de mensaje en este tipo de comunicación se produce a través del objeto *NSNotification* y es gestionado por *NSNotificationCenter*.

- **Vista -> Controlador:** Aunque no se produce una comunicación el sistema de delegación permite que las vistas sean totalmente genéricas y sea el controlador el que implemente la funcionalidad. Existen tres tipos de delegación:

    - *Target-Action*: Se lanza un mensaje al controlador al realizar una acción.
    - *Delegate*: El controlador se encarga de implementar funcionalidad de la vista o de otro controlador.
    - *Data Source*: El controlador se encarga de ofrecer los datos a la vista o a otro controlador.

A continuación se describe el diseño arquitectónico de la aplicación, Figura 22.

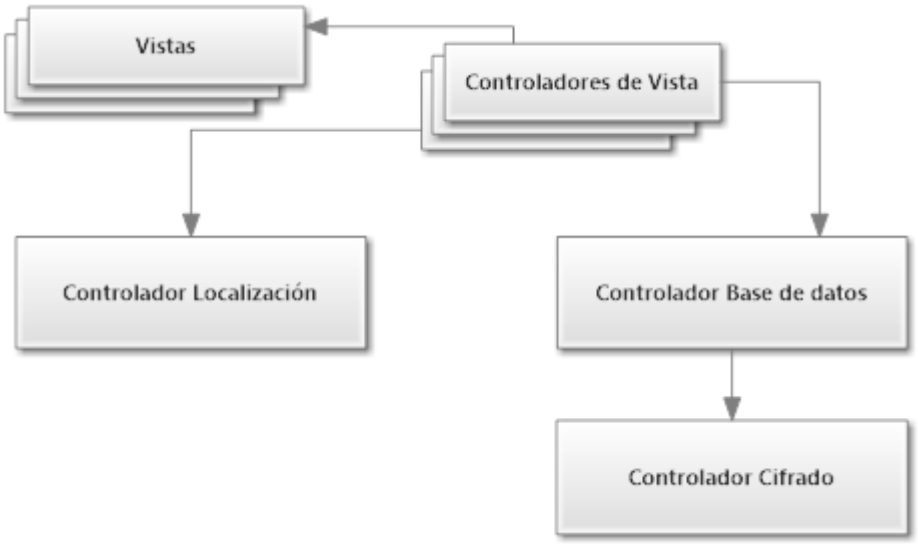

Figura 22: Arquitectura de controladores

3.3 Diseño de los controladores de vista.

En este apartado se detallará el diseño de los controladores de vista a implementar. Éstos seguirán la guía proporcionada por Apple, *iPhone Human Interface Guidelines,* para que la usabilidad de la aplicación sea óptima.

Los controladores de vista se pueden agrupar dependiendo de la funcionalidad que ofrecen a la aplicación:

- Identificación.
- Creación usuario.
- Gestor de contenidos.

Además de los controladores diseñados para la realización del proyecto. Se utilizarán controladores de vista básicos que ofrece el SDK de iOS, como por ejemplo *UITableViewController*, que se basa en un listado compuesto por celdas.

3.3.1 Identificación

Estos controladores se encargan de mostrar al usuario las herramientas necesarias para interactuar con el usuario.

El controlador principal gestionará los sub-controladores. Se utilizará los siguientes controladores:

- *UITableViewController*: Encargado de contener la información a introducir por el usuario. En caso de que la identificación por "Usuario/GPS" no sea satisfactoria se mostrará la opción de identificación mediante "Usuario/Clave".
- *MapViewController:* Mostrará la información relativa a la posición del GPS dibujando un mapa en pantalla.

Además la misión de esta clase es la de realizar los chequeos de autenticación del usuario. Bloqueando, en caso de que se produzcan más de 3 intentos de identificación erróneos, el acceso a la aplicación. Aumentando el tiempo de bloqueo en caso de que se produzcan más intentos.

El esquema de la clase es el siguiente:

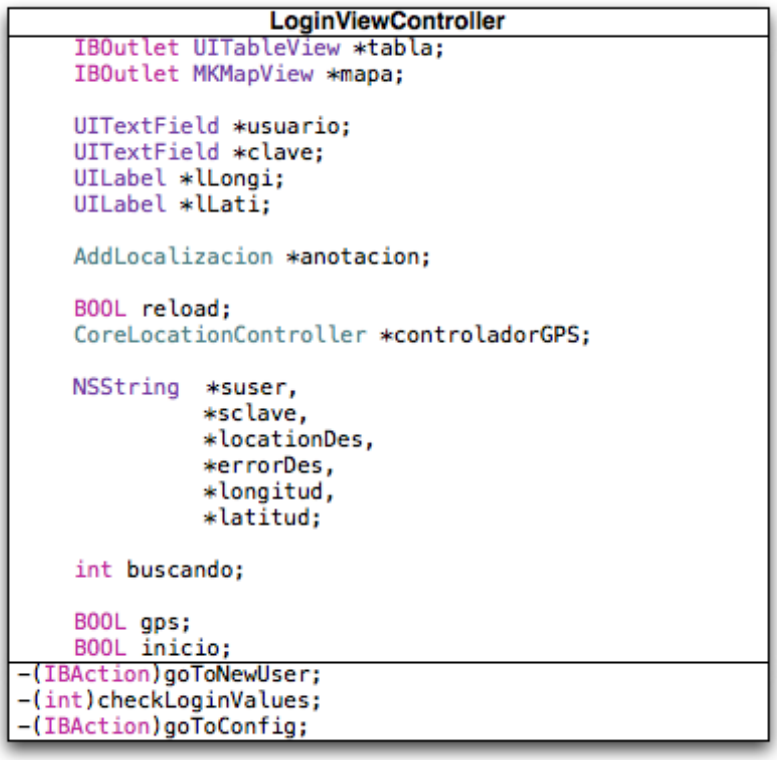

Figura 23: Clase LoginViewController

Clase	LoginViewController
Atributos	
tabla	Contenedor de las opciones de identificación.
mapa	Muestra la posición actual del dispositivo.
usuario	UITextfield en el que el usuario deberá introducir su usuario.
Clave	UITextfield en el que el usuario deberá introducir su clave.
lLongi	UILabel en el que se muestra la longitud en la que se encuentra el dispositivo.
lLati	UILabel en el que se muestra la latitud en la que se encuentra el dispositivo
anotacion	Objeto encargado de la gestión de las anotaciones del mapa.
Reload	Control de carga de la vista.
controladorGPS	Controlador encargado de obtener la posición relativa del dispositivo.
suser	Valor introducido por el usuario para la identificación.
sclave	Valor introducido por el usuario para la identificación.
locationDes	Descripción de la localización.

errorDes	Descripción del error a la hora de obtener la posición.
longitud	Valor de la longitud del dispositivo.
latitud	Valor de la latitud del dispositivo.
buscando	Valor de control de inicio y apagado del controladorGPS.
gps	Valor de control de la utilización del gps.
inicio	Valor de control de primera ejecución.
Métodos	
goToNewUser()	Navegación hacia la creación de un nuevo usuario.
checkLoginValue()	Control de identificación devolviendo un valor int dependiendo del resultado.
goToConfig()	Navegación hacia la configuración.

Tabla 2: LoginViewController

3.3.2 Creación de usuarios

Además el usuario debe poder registrarse en la aplicación. Para ello se han diseñado una serie de controladores que permiten esto. En este apartado se detalla el diseño de las clases que van a gestionar el registro de usuarios. Las clases implicadas en esta funcionalidad son las siguientes:

- *UserViewController*: Permite al usuario introducir los datos relativos a la identificación. Así como mostrar la posición GPS que se va a utilizar a la hora de identificarse.
- *ConfigViewController*: Se encarga de añadir la configuración de cifrado.

3.3.2.1 UserViewController:

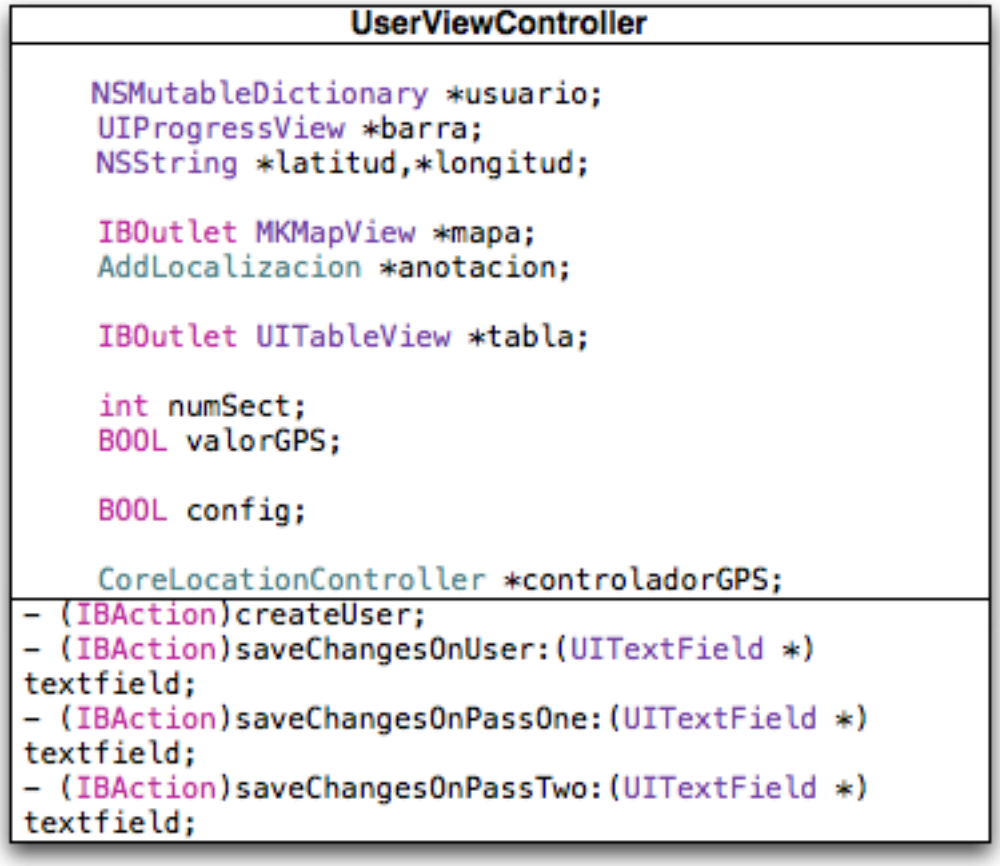

Figura 23: Clase UserViewController

Clase	UserViewController
Atributos	
usuario	Valor introducido por el usuario como identificador unívoco.
Barra	Control de seguridad de la clave.
latitud	Contenedor relativo al valor de la latitud actual.
longitud	Contenedor relativo al valor de la longitud actual.
mapa	Muestra la posición actual del dispositivo.
anotacion	Objeto encargado de la gestión de las anotaciones del mapa.
tabla	Contenedor de las opciones de identificación.
numSect	Número de secciones del contenedor tabla.
valorGPS	Valor de control relativo al actual valor obtenido por el GPS
config	Valor de control relativo a la configuración.

controladorGPS	Controlador para la localización del dispositivo.
Métodos	
createUser()	Acción del botón crear usuario.
saveChangesOnUser()	Al introducir un valor en el campo usuario este método se encarga de guardar el valor.
saveChangesOnPassOne()	Se encarga de guardar los cambios introducidos por el usuario en el campo clave uno.
saveChangesOnPassTwo()	Se encarga de guardar los cambios introducidos por el usuario en el campo clave dos.

Tabla 3: UserViewController

3.3.2.2 ConfigViewController

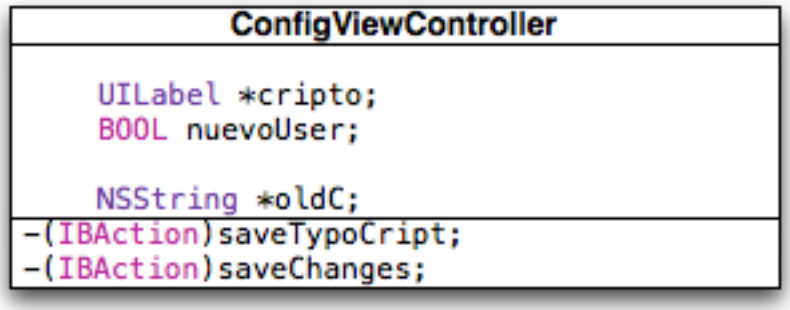

Figura 24: Clase ConfigViewController

Clase	ConfigViewController
Atributos	
cripto	Contenedor encargado de mostrar la etiqueta de configuración.
nuevoUser	Control de configuración para nuevos usuarios o usuarios ya existentes.
oldC	Control de configuración. Se comprueba si la configuración no se ha modificado para evitar sobrecarga.
Métodos	
saveTypoCript()	Salva cualquier cambio producido en la opción de selección de algoritmo de cifrado.
saveChanges()	Salvar cambios.

Tabla 4: ConfigViewController

3.3.3 Gestor de contenidos

El grueso de controladores se encarga de gestionar los contenidos que se muestran para el usuario. Las clases que realizan esta función son las siguientes:

- *CategoryViewController*: Permite la visualización de categorías al usuario.
- *SelectCategoryViewController*: Detalle de la categoría seleccionada por el usuario.
- *ItemViewController*: Detalle del objeto seleccionado por el usuario.
- *ConfigViewController*: Se encarga de mostrar las opciones de configuración de usuario.
- *NewCategoryViewController*: Permite al usuario crear una nueva categoría
- *NewItemViewController*: Permite al usuario crear un nuevo objeto
- *CustomItemViewController*: Permite al usuario configurar el nuevo objeto.

3.3.3.1CategoryViewController

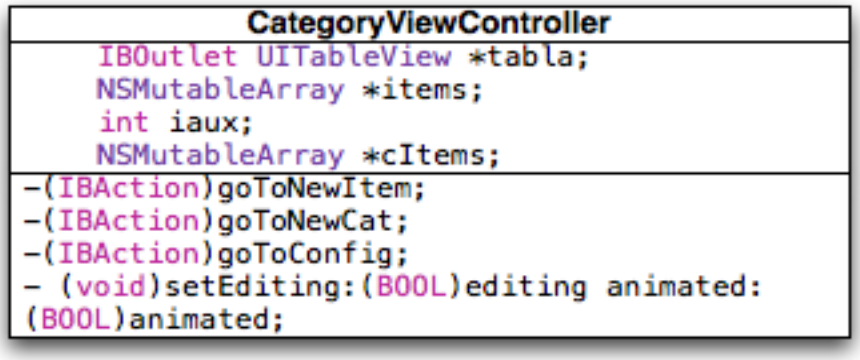

Figura 25: Clase CategoryViewController

Clase	CategoryViewController
Atributos	
table	Contenedor de los datos que se tienen que mostrar al usuario.
items	Lista de datos.
iaux	Contador auxiliar.
cItems	Lista con el recuento de elementos de cada categoría.
Métodos	
goToNewItem()	Navegación hacia la creación de un nuevo elemento.
goToNewCat()	Navegación hacia la creación de una nueva categoría.
goToConfig()	Navegación hacia la sección de configuración

	de la aplicación.
setEditing()	Modifica el aspecto del contenedor "table" para modificar su contenido.

Tabla 5: CategoryViewController

3.3.3.2 SelectCategoryViewController

Figura 26: Clase SelectCategoryViewController

Clase	SelectCategoryViewController
Atributos	
idCat	Identificador de la categoría.
elements	Listado con los elementos a mostrar.

Tabla 6: SelectCategoryViewController

3.3.3.3 ItemsViewController

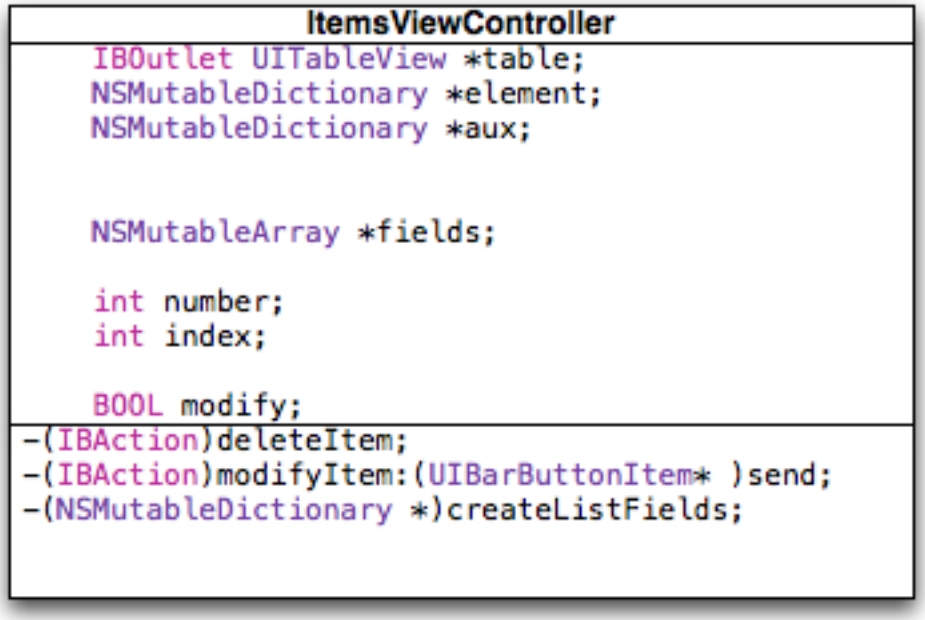

Figura 27: Clase ItemViewController

Clase	ItemsViewController
Atributos	
table	Contenedor encargado de mostrar los elementos de la categoría.
element	Contenedor de los datos.
aux	Contenedor auxiliar.
fields	Campos a mostrar.
number	Contador de elemento.
index	Puntero al índice.
modify	Valor de control de modificación.
Métodos	
deleteItem()	Elimina el elemento seleccionado.
modifyItem()	Modifica el elemento seleccionado.
createListFields()	Crea los campos que se muestran al usuario.

Tabla 7: ItemsViewController

3.3.3.4 NewCategoryViewController

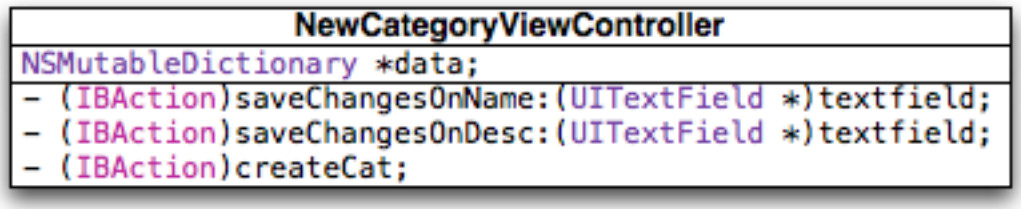

Figura 28: Clase NewCategoryViewController

Clase	NewCategoryViewController
Atributos	
data	Contenedor auxiliar para el almacenamiento de los datos.
Métodos	
saveChangesOnName()	Salva cualquier cambio producido en el nombre.
saveChangesOnDesc()	Salva cualquier cambio producido en la descripción.
createCat()	Método encargado de la creación de la categoría.

Tabla 8: NewCategoryViewController

3.3.3.5 NewItemViewController

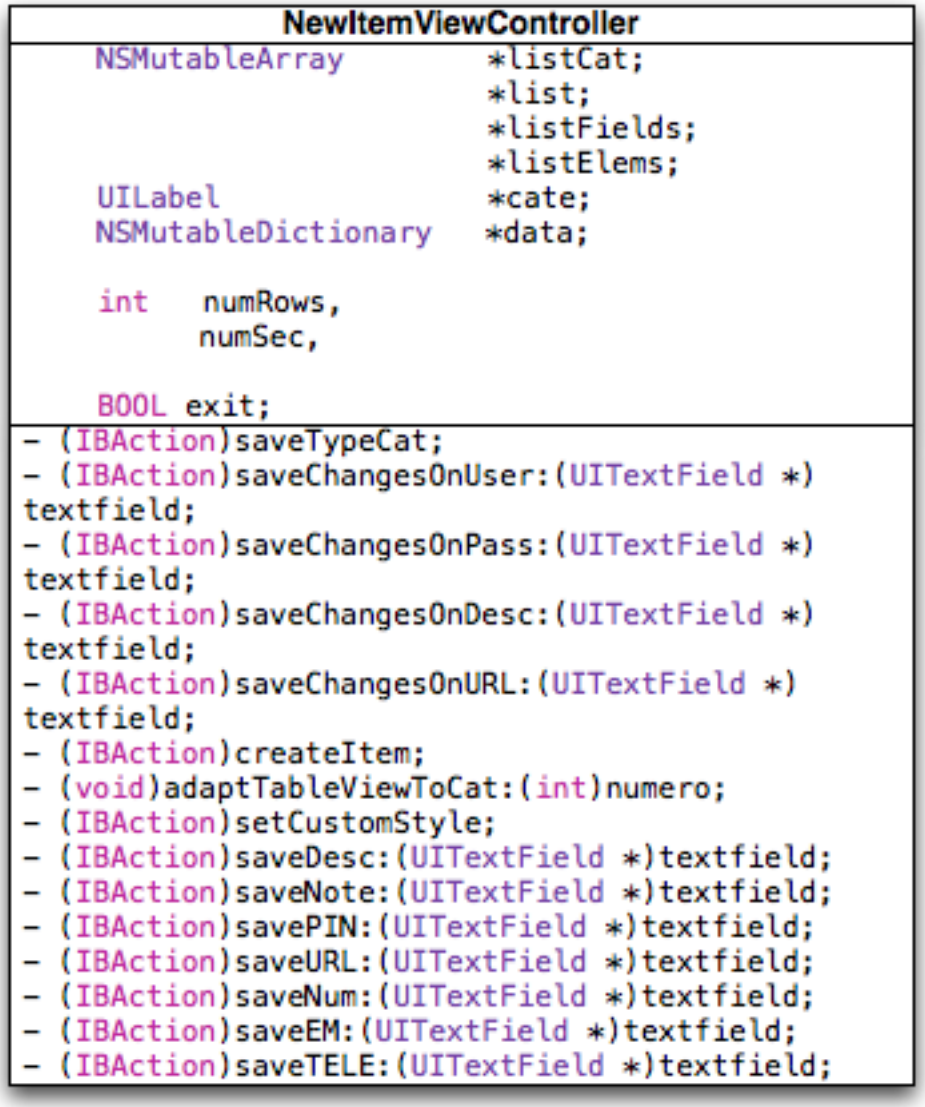

Figura 29: Clase NewItemViewController

Clase	NewItemViewController
Atributos	
listCat	Lista de categoría disponibles para la selección.
list	Variable de control que gestiona que los datos estén bien introducidos.
listFields	Listado de campos del nuevo elemento
listElements	Listado de valores del nuevo elemento.
Cat	Identificador de la categoría
Data	Datos asociados al elemento.
numRows	Número de filas a mostrar por el contenedor.
numSec	Número de secciones a mostrar por el

	contenedor.
exit	Variable de control para la salida del Controlador.
Métodos	
saveTypoCat()	Salva el tipo de categoría.
saveChangesOnUser()	Salva cambios en el usuario.
saveChangesOnPass()	Salva cambios en la contraseña.
saveChangesOnDesc()	Salva cambios en la descripción.
saveChangesOnURL()	Guarda cambios en la URL.
createItem()	Crea el nuevo elemento.
adaptTableViewToCat()	Adapta el contenedor para los nuevos campos.
setCustomStyle()	Establece el nuevo estilo.
saveNote()	Salva cambios en la nota.
savePIN()	Salva cambios en el pin.
saveURL()	Salva cambios en la URL.
saveNum()	Salva cambios en el número secreto.
saveEM()	Salva cambios en el correo electrónico
saveTele()	Salva cambios en el teléfono.

Tabla 9: NewItemViewController

3.3.3.6 CustomItemViewController

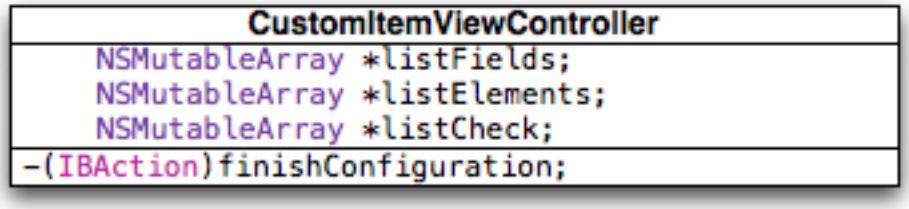

Figura 30: CustomItemViewController

Clase	CustomViewController
Atributos	
listFields	Listado de los campos del nuevo elemento.
listElements	Listado del valor de los valores de los campos para el nuevo elemento.
listChecks	Listado de control de selección de elementos.
Métodos	
finishConfiguration ()	Termina con la configuración del nuevo elemento.

Tabla 10: CustomViewController

3.4 Diseño del modelo

En este apartado se detalle el diseño de las clases encargadas de la gestión del modelo de la aplicación, las actividades relativas al modelo son:

- Gestión de la base de datos.
- Gestión de la localización.
- Gestión del cifrado.

3.4.1 Gestión de la base de datos

Esta clase se encarga del mantenimiento de la base de datos. Como principales funciones tiene la de crear la base de datos, inserciones, modificaciones, consultas y borrados.

DataBaseController

```
    sqlite3 *db;
-(NSString *) filePath;
-(NSString *) filePathWithName:(NSString *)nombre;

-(void) openDB;
-(void) openDbForCreate:  (NSString *)nombre;
-(void) openDbWithName:       (NSString *)nombre;

-(void) changeCypher:         (NSString *)nC and:(NSString *)oC withUser:(NSString *)
nombre;
-(void) changePassword:        (NSString *)cypher withGPS:(NSString *)GPS withUser:
(NSString *)user;

-(BOOL) createUser:     (NSString *)userid;
-(void) insertIntoUser: (NSDictionary *)datos;

-(NSMutableArray *)selectAllFromUser;
-(void) initiateBD;

-(void) closeBdWithUser:    (NSString *)name;
-(void)closeBdWithConfig:   (NSMutableDictionary *)config;

-(BOOL) insertIntoCat:      (NSMutableDictionary *)data withUser:(NSString *)iduser;
-(BOOL) insertIntoItem:     (NSMutableDictionary *)data withUser:(NSString *)iduser;
-(void) insertIntoConfig:   (NSDictionary *)data withUser:(NSString *)iduser;

-(NSMutableDictionary *)selectIdFromUser:       (NSString *)userid;
-(NSMutableArray *)selectAllItemsFromCat:       (NSString *)cat withUser:(NSString *)
iduser;
-(NSMutableArray *)selectAllFromCat:            (NSString *)iduser;
-(NSMutableDictionary *)selectConfigForUser:    (NSString *)iduser;

-(void)modifyConfig:        (NSDictionary *)data withUser:(NSString *)user;

-(void)deleteFromCat:       (NSString *)name ofUser:(NSString *)user;
-(void)deleteAllItemsOfCat: (NSString *)nameCat ofUser:(NSString *)user;
-(void)deleteItemFromCat:   (NSString *)nameItem ofUser:(NSString *)user;
```

Figura 31: Clase DataBaseController

Clase	DataBaseController
Atributos	
db	Contenedor de la base de datos.
Métodos	
filePath ()	Obtiene la ruta en el sistema de archivos de la base de datos por defecto.
filePathWithName()	Obtiene la ruta en el sistema de archivos de la base de datos con un nombre determinado.
openDB()	Abre la base de datos por defecto.
openDBForCreate()	Crea la base de datos en caso de que ésta no exista.
openDBWithName()	Abre la base de datos a partir del nombre.
changeCypher()	Cambiar el tipo de cifrado de la base de datos.
changePassword()	Cambia la contraseña que se utiliza como parte del cifrado.
createUser()	Crea una entrada en la tabla usuario.
insertIntoUser()	Inserta datos relacionados con el usuario.
selectAllFromUser()	Selecciona toda la información de la tabla usuario.
initiateDB()	Inicializa la base de datos.
closeDBWithUser()	Cierra la base de datos del usuario.
closeBDWithConfig()	Cierra la base de datos utilizando la configuración pasada.
insertIntoCat()	Inserta una entrada en la tabla Categorías.
insertIntoItem()	Inserta una entrada en la tabla Elemento.
insertIntoConfig()	Inserta una entrada en la tabla de configuración.
selectIDFormUser()	Obtiene el identificador de un usuario dado.
selectAllFromCat()	Obtiene todos los datos de la tabla Categorías.
selectConfigForUser()	Obtiene los datos de configuración relativos al usuario dado.
modifyConfig()	Modifica los datos relativos a la configuración de un usuario.
deleteFromCat()	Borra un elemento de la tabla Categoría.
deleteAllitemsOfCat()	Borra todos los elementos de una categoría.
deleteItemFromCat()	Borra un elemento dado el elemento.

Tabla 11: DataBaseController

3.4.2 Gestión de la localización

Se implementará la funcionalidad de localización. Para ello se utilizará un protocolo. Este protocolo gestiona el comportamiento del controlador cuando recibe una actualización de la posición relativa del dispositivo.

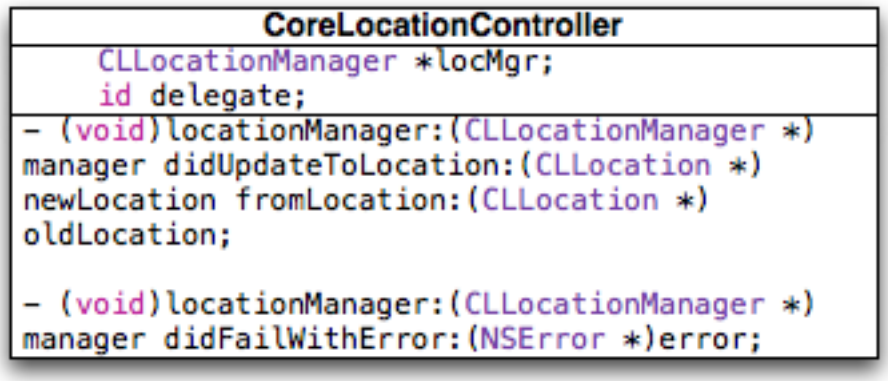

Figura 32: Clase CoreLocationController

Clase	CoreLocationController
Atributos	
locMgr	Objeto gestor de la localización.
delegate	Clase encargada de implementar los métodos del protocolo.
Métodos	
didUpdateToLocation ()	En caso de que se produzca una variación de la posición, este método ejecuta el del delegado.
didFailWithError()	Lanza un error en caso de que no se pueda determinar la localización.

Tabla 12: CoreLocationController

3.4.3 Gestión del cifrado

Esta clase se encarga de realizar el cifrado a los datos que se le especifiquen. Los métodos de cifrado son los que ofrece el SDK de iOS ya comentados en el Capítulo 2.

CryptoController

```
NSData *bData;
-(BOOL)addKeyToChain:(NSString *)name;
-(id)retrieveKeyFromChain:(NSString *)name;
// AES
- (NSData *)aesEncrypt:           (NSString *)key withData:(NSData*)
datos;
- (NSData *)aesDecryptWithKey:    (NSString *)key withData:(NSData*)
datos;
// tDES
- (NSData *)tDESEncryptWithKey:   (NSString *)key withData:(NSData*)
datos;
- (NSData *)tDESDecryptWithKey:   (NSString *)key withData:(NSData*)
datos;
// CAST
- (NSData *)CASTEncryptWithKey:   (NSString *)key withData:(NSData*)
datos;
- (NSData *)CASTDecryptWithKey:   (NSString *)key withData:(NSData*)
datos;
// DES
- (NSData *)DESEncryptWithKey:    (NSString *)key withData:(NSData*)
datos;
- (NSData *)DESDecryptWithKey:    (NSString *)key withData:(NSData*)
datos;
// RC4
- (NSData *)RC4EncryptWithKey:    (NSString *)key withData:(NSData*)
datos;
- (NSData *)RC4DecryptWithKey:    (NSString *)key withData:(NSData*)
datos;
// RC2
- (NSData *)RC2EncryptWithKey:    (NSString *)key withData:(NSData*)
datos;
- (NSData *)RC2DecryptWithKey:    (NSString *)key withData:(NSData*)
datos;
```

Figura 33: Clase CryptoLocationController

Clase	CryptoLocationController
Atributos	
bData	Datos a cifrar.
Métodos	
addKeyToChain ()	Introduce un nuevo elemento en el llavero del dispositivo.
retrieveKeyFromChain()	Obtiene un elemento del llavero del dispositivo.
aesEncrypt	Cifrado del mensaje utilizando el algoritmo AES.
aesDecryptWithKey()	Descifrado del mensaje utilizando el algoritmo AES.

tDESEncryptWithKey()	Cifrado del mensaje utilizando el algoritmo Triple DES.
tDESDecryptWithKey()	Descifrado del mensaje utilizando el algoritmo triple DES.
tCASTEncryptWithKey()	Cifrado del mensaje utilizando el algoritmo CAST.
tCASTDecryptWithKey()	Descifrado del mensaje utilizando el algoritmo CAST.
DESEncryptWithKey()	Cifrado del mensaje utilizando el algoritmo DES.
DESDecryptWithKey()	Descifrado del mensaje utilizando el algoritmo DES.
RC4EncryptWithKey()	Cifrado del mensaje utilizando el algoritmo RC4.
RC4DecryptWithKey()	Descifrado del mensaje utilizando el algoritmo RC4.
RC2EncryptWithKey()	Cifrado del mensaje utilizando el algoritmo RC2.
RC2DecryptWithKey()	Descifrado del mensaje utilizando el algoritmo RC2.

Tabla 13: CryptoLocationController

3.4.4 Diseño de las vistas personalizadas

Además se generarán diseños propios a las vistas que ofrece UIKit. Por ello se necesitan clases que modifiquen la creación de dichos elementos. Las vistas propias utilizadas son las siguientes:

- *ClearLabelsCellView*: Genera celdas para las tablas sin formato.
- *GradientView*: Genera el color de gradiente que posteriormente se utilizará en las celdas.
- *ShadowTableView*: Es una Subclase de *UITableView* que implementa los cambios de estilo.

3.5 Diseño del modelo de datos

Los datos deben almacenarse de forma permanente en el dispositivo. Para ello se utilizará el sistema gestor de base de datos Sqlite. La base de datos deberá almacenar datos sobre:

- Usuarios.
- Categorías.
- Objetos.
- Configuraciones.

Puesto que se necesita que la base de datos este cifrada. El sistema creará una base de datos por usuario y una base de datos para la propia aplicación. Ésta última contendrá información sobre:

- Usuarios de la aplicación.
- Configuración de los usuarios.

El diseño de la base de datos quedará de la siguiente manera:

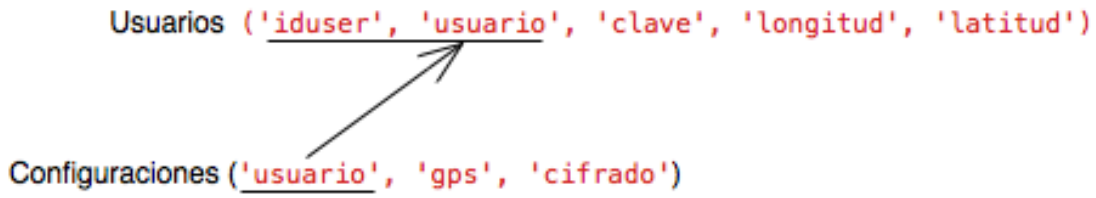

Figura 34: Diseño de la base de datos CORE

Para guardar la información relativa a los datos del usuario se generará una base de datos por cada usuario. El diseño de la base de datos será el siguiente.

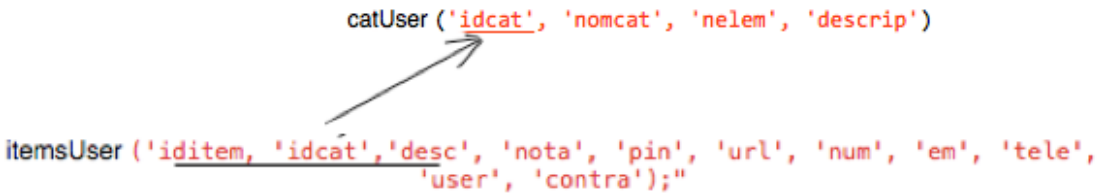

Figura 35: Diseño de la base de datos por Usuario

En este caso, el nombre de las tablas se generará de forma dinámica. Se le añadirá el nombre de usuario a dichas tablas.

3.6 Diagramas de secuencia

A continuación se detallaran los diagramas de secuencia de la funcionalidad principal. Los casos que se van a explicar son los siguientes:

- Inicio de la aplicación.
- Identificación en la aplicación.
- Creación de usuario.
- Creación de categoría.
- Creación de elemento.

3.6.1 Inicio de la aplicación

La aplicación debe ser iniciada por el usuario. Una vez iniciada se configuran los servicios básicos de la aplicación y se generan los contenedores de los datos, Figura 36.

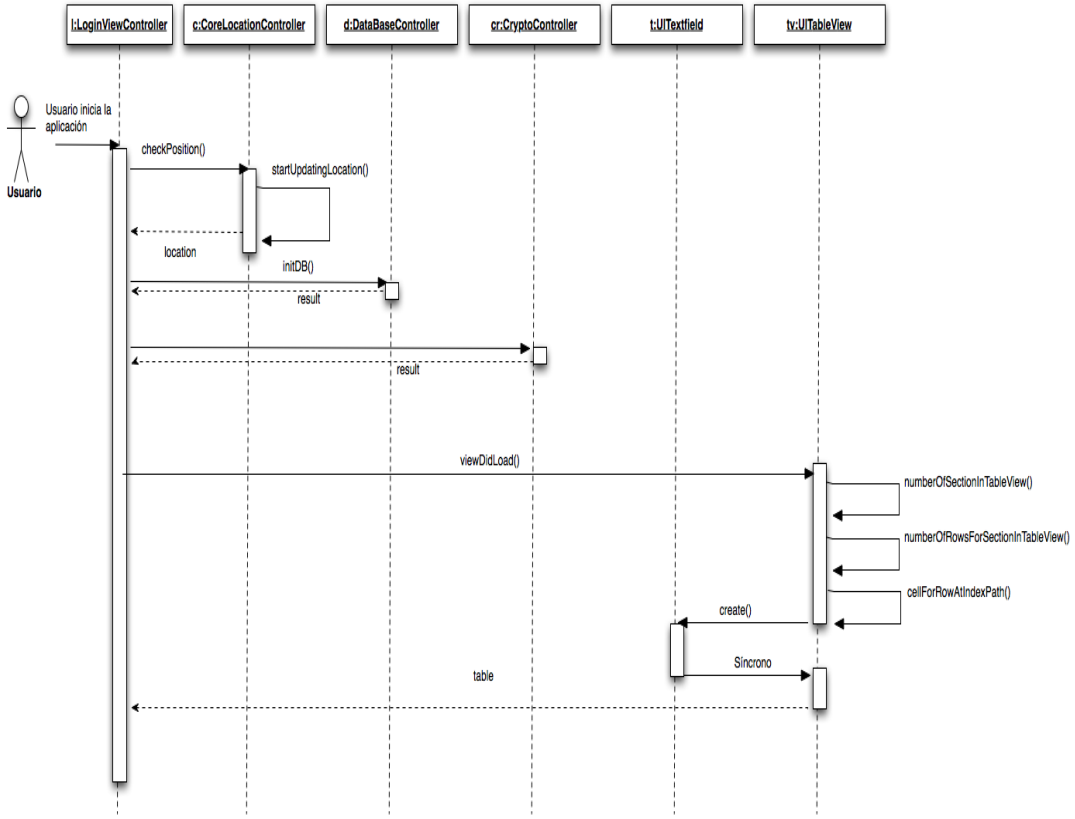

Figura 36: Diagrama de secuencia de Inicio de la aplicación

3.6.2 Identificación en la aplicación

Diagrama de secuencia en el caso de que se consiga identificar correctamente al usuario utilizando la posición del GPS, Figura 37.

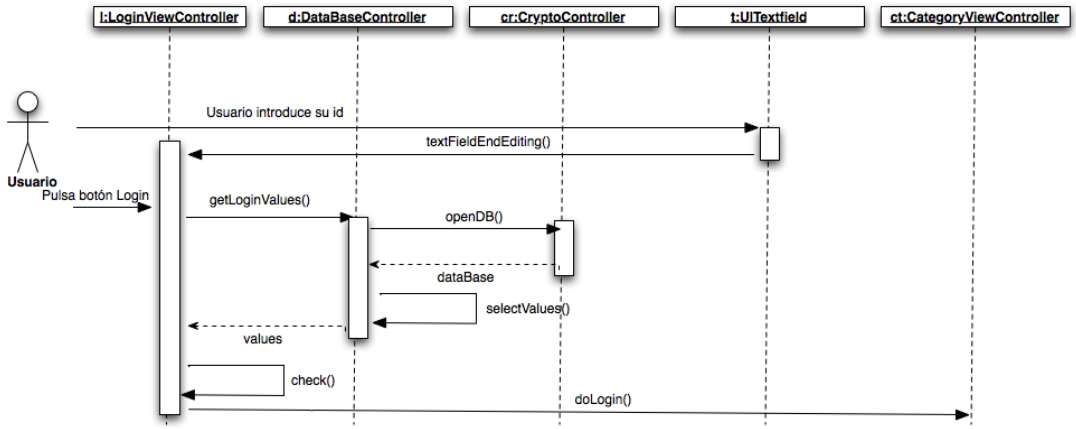

Figura 37: Diagrama de secuencia identificación de la aplicación

3.6.3 Creación de usuario

Diagrama de secuencia para la creación de un nuevo usuario, Figura 38.

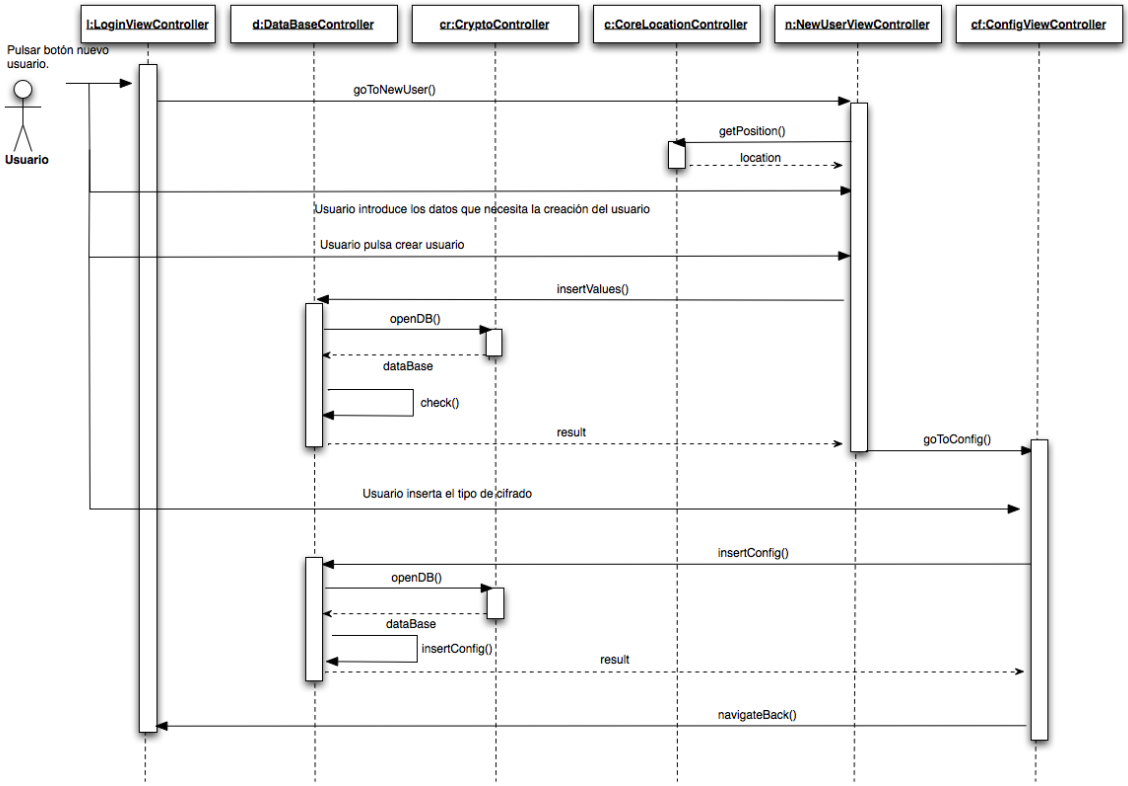

Figura 38: Diagrama de secuencia creación de usuario

3.6.4 Creación de categoría

Diagrama de secuencia que reproduce la acción de creación de una categoría una vez que el usuario está identificado, Figura 39.

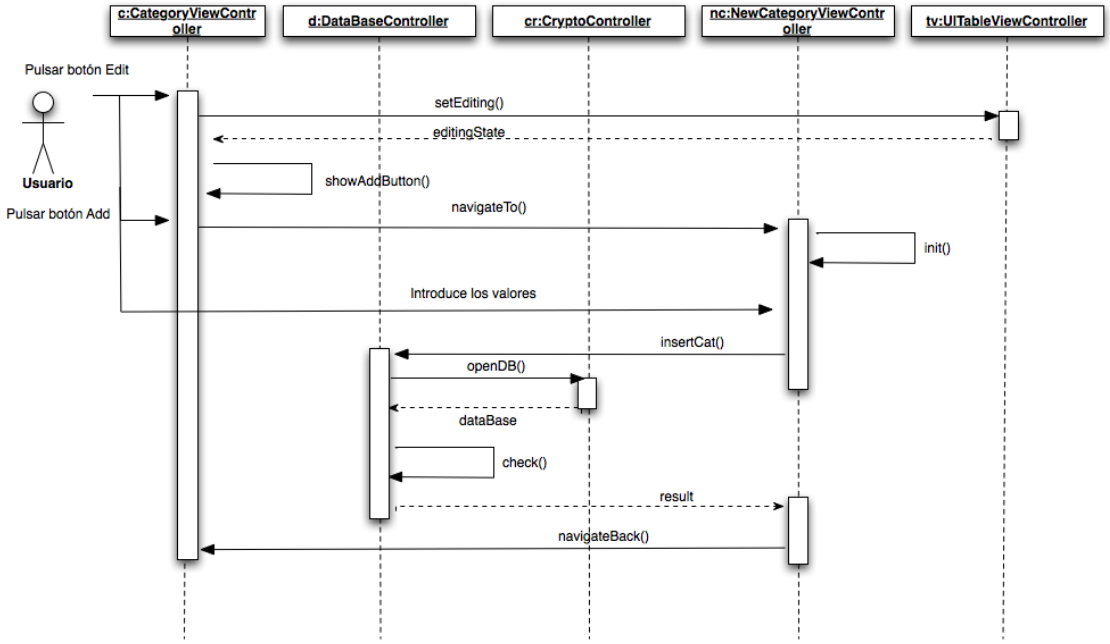

Figura 39: Diagrama de secuencia creación de categoría

3.6.4 Creación de usuario

Diagrama de secuencia para la creación de un nuevo usuario, Figura 40.

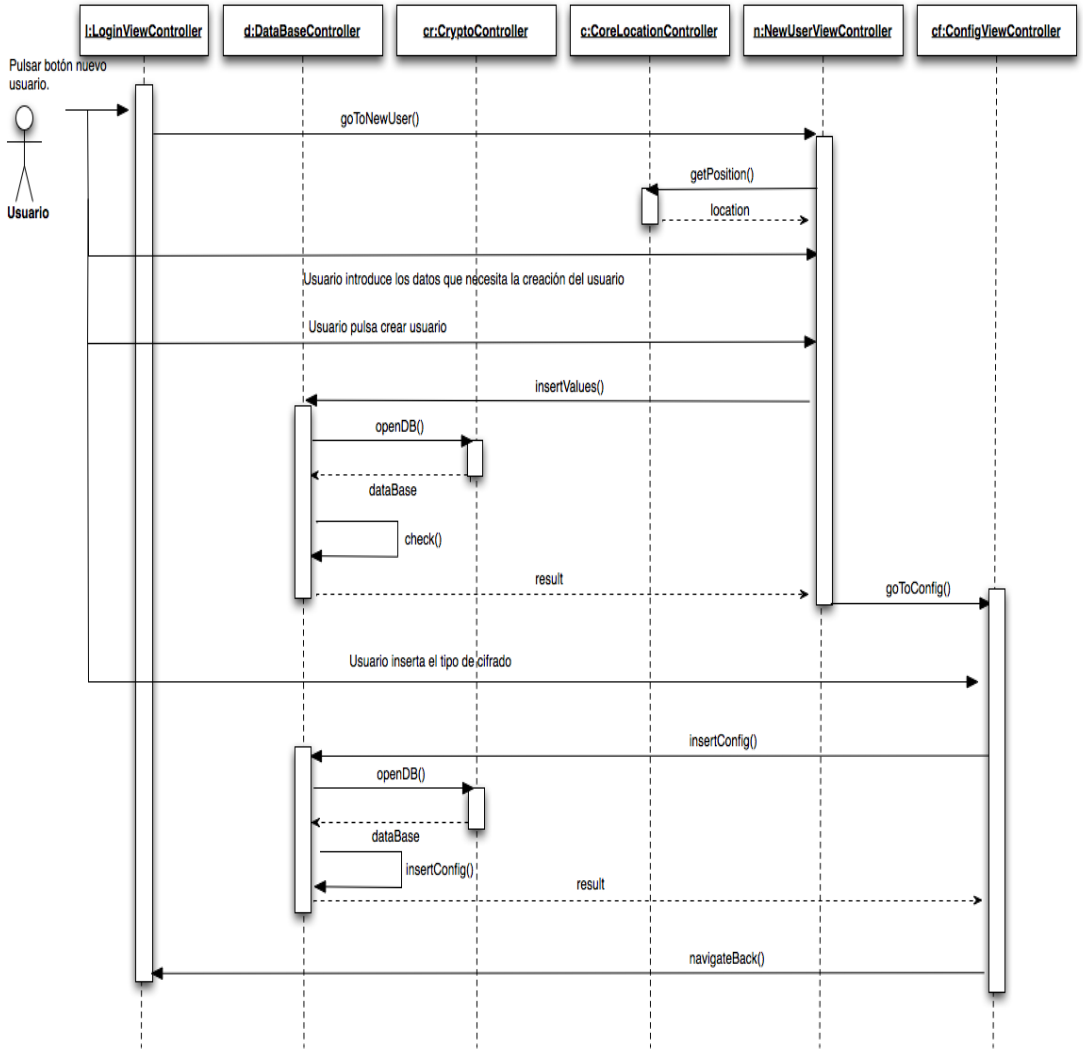

Figura 40: Diagrama de secuencia creación de usuario

3.6.5 Creación de elemento

Diagrama de secuencia que reproduce la acción de creación de un nuevo elemento una vez que se ha identificado correctamente al usuario, Figura 41.

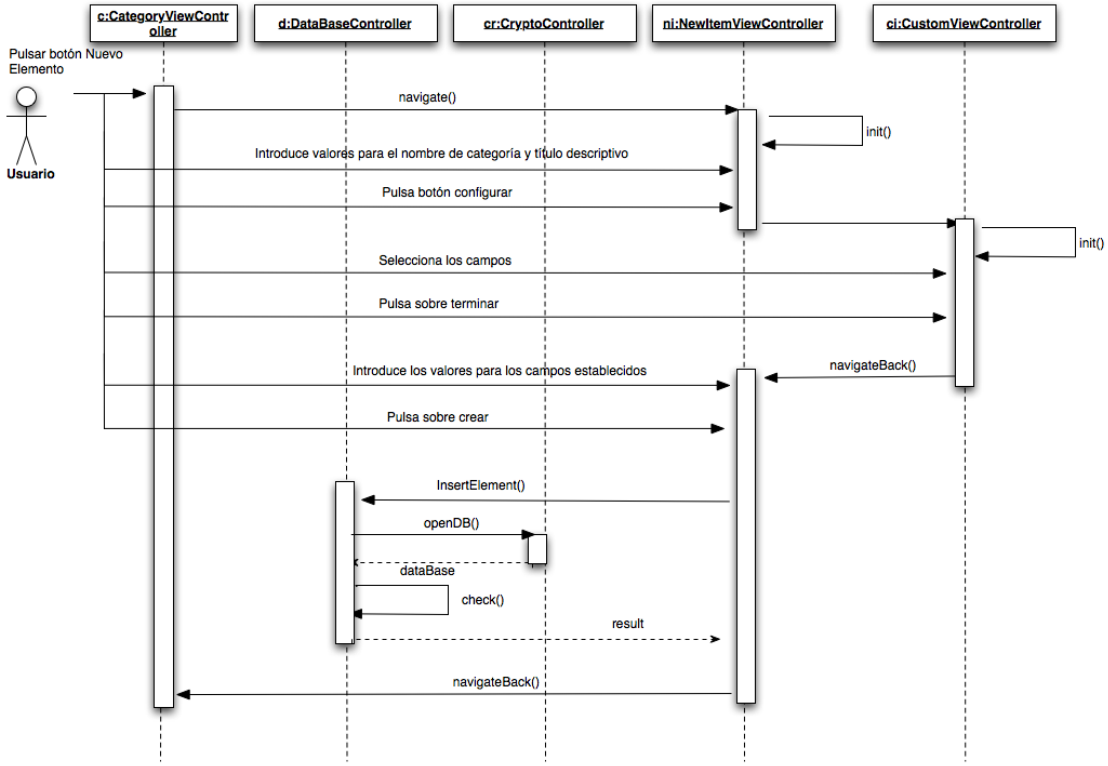

Figura 41: Diagrama de secuencia creación de elemento

3.7 Diseño del sistema de seguridad

Se ha diseñado un sistema de seguridad para evitar ataques a la aplicación.

- Sistema de identificación con tiempo de bloqueo progresivo. Se permite 3 intentos de acceso a la aplicación erróneos. En caso de que se supere este número de intentos, se bloquea la posibilidad de intentarlo más durante un periodo de 5 minutos. Por cada nuevo intento fallido se multiplicara este tiempo por el número de intentos. Este sistema permite evitar un ataque basado en la fuerza bruta al sistema de identificación de la aplicación.

  *Tiempo Bloqueo = nº Intentos Fallidos * 300 (segundos)*.

- Cifrado de la base de datos. Además el sistema de cifrado de la base de datos se base en una clave generada a partir de la localización (Longitud y Latitud), una clave generada por la aplicación en cada instalación de ésta y la clave privada del usuario, que se almacena en el llavero del dispositivo. Dado que no es posible evitar ataques por fuerza bruta a la

base de datos de la aplicación, es posible obtener las bases de datos del dispositivo utilizando herramientas como *iPhone Explorer*, se ha diseñado este sistema de claves para evitar que se pueda obtener acceso a la información contenida en la base de datos mediante un ataque por fuerza bruta utilizando herramientas externas.

Capítulo 4: Implementación

4.1 Introducción

En este capítulo se trata de los aspectos más importantes de la ejecución del diseño y además se tratarán los casos de prueba detallados en el capítulo 2.

4.2 Aspectos importantes de la implementación

A continuación se detallarán los aspectos más relevantes en cuanto a la implementación. Puesto que la aplicación se basa en la funcionalidad de localización geo espacial y cifrado de los datos, estos serán los aspectos principales que se tratarán en esta sección.

4.2.1 Localización

A la hora de utilizar el posicionamiento relativo del dispositivo, el SDK de iOS proporciona las herramientas para el acceso a dicha información.

La clase que implementa la interfaz para obtener la localización es "*CoreLocationController*". Esta clase está definida por:

- *CLLocationManager*.
- *CLLocationManagerDelegate*.

4.2.1.1 CLLocationManager

Esta clase define el interfaz y la configuración para obtener los datos de la localización hacia la aplicación.

La interfaz se compone de estos dos métodos, Figura 42:

```
- (void)locationManager:(CLLocationManager *)manager didUpdateToLocation:(CLLocation
*)newLocation fromLocation:(CLLocation *)oldLocation;


- (void)locationManager:(CLLocationManager *)manager didFailWithError:(NSError *)error;
```

Figura 42: Métodos CLLocationManager

Estos métodos se encargan del control de cambios en la posición dada por el valor de la localización. Además del control de errores en caso de que no se pueda obtener la posición.

Además se puede especificar la precisión del resultado. Para ello se define un valor del controlador del siguiente modo, Figura 43:

```
locManager.desiredAccuracy = kCLLocationAccuracyBest;
```

Figura 43: Fragmento de código locManager

4.2.1.2 CLLocationManagerDelegate

Está constituido por la clase que va a implementar el protocolo dado por "*CoreLocationController*". Dicho protocolo está definido por los siguientes métodos, Figura 44:

```
@protocol CoreLocationControllerDelegate
@required
- (void)locationUpdate:(CLLocation *)location;
- (void)locationError:(NSError *)error;
@end
```

Figura 44: Métodos CLLocationManagerDelegate

- *locationUpdate*(): La clase delegada debe implementar este método. Cada vez que se produce un cambio en la posición del GPS, la clase encargada de la gestión de la localización (*CoreLocationController*), se realiza una llamada a este método en el delegado.
- *locationError*(): En caso de producirse un error el controlador realiza una llamada a este método.

El tipo de la variable de retorno de la localización es "*CLLocation*". Para acceder al valor tipo "*Float*" de la longitud y latitud se utiliza la notación del punto, Figura 45.

```
newLocationLatitude = location.latitude;
newLocationLongitude = location.longitude;
```

Figura 45: Asignación de localización

4.2.2 Cifrado de datos

Un punto importante de la aplicación es el cifrado de los datos. Del mismo modo que en la localización, para esta funcionalidad se van a utilizar las herramientas que ofrece el SDK de iOS. Los algoritmos de cifrado que se van a implementar en la aplicación están detallados en el Capítulo 2.

La clave de los datos que se va a utilizar se compone de:

- Clave alfanumérica generada aleatoriamente para la base de datos propia de la aplicación.
- Clave alfanumérica generada por la geo posición concatenada con la clave de usuario.

4.2.2.1 Clave propia de la aplicación.

Se utilizará en el cifrado de la base de datos que contiene los usuarios y las configuraciones. La clase que implementa esta funcionalidad es "*CryptoController*", Figura 46.

```
- (NSMutableString *)createKeyForCore;
```

Figura 46: Método de creación de clave de la aplicación

Esta clave una vez generada se almacena en el repositorio que ofrece para cada aplicación el dispositivo móvil. Este repositorio se denomina "Llavero" y está cifrado.

La clase que implementa esta funcionalidad es "*CryptoController*" y los métodos que se encargan de esta funcionalidad son, Figura 47:

```
- (id)retrieveKeyFromChain:(NSString *)name;

- (BOOL)addKeyToChain:(NSString *)name;
```

Figura 47: Métodos de obtención de clave

El método "*retrieveKeyFromChain*()" se encarga de obtener la clave del llavero. Básicamente obtiene una copia del valor almacenado con este método, lo transforma en un tipo "*NSMutableDictionary*", Figura 48.

```
OSStatus status = SecItemCopyMatching((CFDictionaryRef)searchDictionary,
                                      (CFTypeRef *)&result);
```

Figura 48: Extracción de la clave

El método "*addKeyToChain*()" se encarga de almacenar la clave en el llavero. La clave debe estar almacenada en una estructura tipo "*NSMutableDictionary*", Figura 49.

```
OSStatus status = SecItemAdd((CFDictionaryRef)dictionary, NULL);
```

Figura 49: Añadir la clave a la herramienta de Llavero

4.2.2.2 Clave generada para el usuario.

Esta clave se genera cada vez que se crea un usuario utilizando los datos introducidos por el usuario y la posición del dispositivo a la hora de crear el usuario.

Esta clave se almacena en la base de datos de la aplicación. La clase encargada de generar la clave es "*CryptoController*". El método que implementa la funcionalidad es, Figura 50:

```
-(NSMutableString *)createKeyForUser:(NSDictionary *)data;
```

Figura 50: Generación de la clave para el usuario

Este método recibe como parámetro los valores del usuario y devuelve la clave generada.

Dentro de los parámetros contenidos en "data" se encuentra el tipo de cifrado. La clase que implementa los distintos tipos de cifrado es "*CryptoController*" en los métodos, Figura 51:

```
- (NSData *)aesEncrypt:(NSString *)key withData:(NSData*)datos;

- (NSData *)aesDecryptWithKey:(NSString *)key withData:(NSData*)datos;

- (NSData *)tDESEncryptWithKey:(NSString *)key withData:(NSData*)datos;

- (NSData *)tDESDecryptWithKey:(NSString *)key withData:(NSData*)datos;

- (NSData *)CASTEncryptWithKey:(NSString *)key withData:(NSData*)datos;

- (NSData *)CASTDecryptWithKey:(NSString *)key withData:(NSData*)datos;

- (NSData *)DESEncryptWithKey:(NSString *)key withData:(NSData*)datos;

- (NSData *)DESDecryptWithKey:(NSString *)key withData:(NSData*)datos;

- (NSData *)RC4EncryptWithKey:(NSString *)key withData:(NSData*)datos;

- (NSData *)RC4DecryptWithKey:(NSString *)key withData:(NSData*)datos;
```

Figura 51: Métodos de cifrado

Son los métodos de cifrado y descifrado para los distintos algoritmos.

4.2.3 Gestor de base de datos.

Además es necesario un gestor para la base de datos. Esta funcionalidad está implementada en la clase "*DataBaseController*". Se encargará de, Figura 52:

- Inicialización de las bases de datos, usuario y aplicación.
- Apertura de la base de datos. Control del descifrado.

```
- (NSString *) filePathWithName:(NSString *)name;

- (void) openDbWithName:(NSString *)name;
```

Figura 52: Métodos de apertura de la base de datos

Son los métodos encargados de la apertura de la base de datos. El primer método obtiene la ruta de la base de datos en el sistema de ficheros a partir del nombre de la base de datos.
El segundo método permite abrir la base de datos para obtener los datos de ella.

- Clausura de la base de datos. Control del cifrado, Figura 53.

```
- (void)closeBdWithConfig:(NSMutableDictionary *)config
```

Figura 53: Método de cierre de la base de datos

Cierra la base de datos utilizando una configuración determinada.

- Inserción.

```
- (void) insertIntoUser:(NSDictionary *)data;

- (BOOL) insertIntoCat:(NSMutableDictionary *)data withUser:(NSString *)iduser;

- (BOOL) insertIntoItem:(NSMutableDictionary *)data withUser:(NSString *)iduser;

- (void) insertIntoConfig:(NSDictionary *)data withUser:(NSString *)iduser;
```

Figura 54: Métodos de inserción en la base de datos

Estos métodos se encargan de la inserción de datos en la base de datos, Figura 54.

- Modificación.

```
- (void)modifyConfig:(NSDictionary *)data withUser:(NSString *)user;
```

Figura 55: Métodos de modificación de configuración

Permite la modificación de la configuración del usuario. Estos métodos utilizan las funciones de acceso a la base de datos, Figura 55.

- Eliminación.

```
- (void)deleteFromCat:(NSString *)name ofUser:(NSString *)user;
- (void)deleteAllItemsOfCat:(NSString *)nameCat ofUser:(NSString *)user;
- (void)deleteItemFromCat:(NSString *)nameItem ofUser:(NSString *)user;
```

Figura 56: Métodos de eliminación de elementos

Permite el borrado de elementos de la base de datos, Figura 56.

- Cambiar cifrado de la base de datos.

```
-(void) changeCypherDB:        (NSString *)nC and:(NSString *)oC
withUser:(NSString *)name;
```

Figura 57: Método de cambio de cifrado de la base de datos

Este método permite cambiar el cifrado a la base de datos. Para ello es necesaria la configuración de cifrado original. Una vez se puede descifrar los datos se vuelve a cifrar con los datos de configuración nuevos, Figura 57.

4.3 Pruebas de aceptación

El objetivo del plan de pruebas es asegurar el nivel de calidad del aplicativo y que el paso a producción de éste tenga el mínimo número de incidencias posibles. El encargado de realizar el plan de pruebas es Jesús M. Chicharro Gallego.

4.3.1 Hardware

Los sistemas donde se han realizado el plan de pruebas son los siguientes.

Sistema	Sistema operativo de las pruebas
MacBook Pro 15"	Simulador iOS 4,iOS 5, iOS6
iPhone 4S	iOS 6

Tabla 14: Sistemas de pruebas

4.3.2 Resultados de las pruebas

El resultado de los casos de prueba se detalla a continuación.

Identificador	CP-01
Descripción	Inicio de la aplicación. Intento de identificación mediante usuario erróneo. Se comprobará que produce un error de identificación.
Resultado	Éxito.

Resultado prueba 1: Identificación en la aplicación

Identificador	CP-02
Descripción	Inicio de la aplicación. Intento de identificación mediante posición GPS errónea. Se comprobará que produce un error de identificación.
Resultado	Éxito.

Resultado prueba 2: Posición GPS incorrecta

Identificador	CP-03
Descripción	Inicio de la aplicación. Intento de identificación mediante usuario/contraseña erróneo. Se comprobará que produce un error de identificación.
Resultado	Éxito.

Resultado prueba 3: Identificación Usuario/Contraseña incorrecta

Identificador	CP-04
Descripción	Inicio de la aplicación. Creación de usuario utilizando la configuración por defecto. Se comprobará que la creación del usuario se lleva a cabo utilizando dicha configuración.
Resultado	Éxito.

Resultado prueba 4: Creación de usuario con configuración por defecto

Identificador	CP-05
Descripción	Inicio de la aplicación. Creación de usuario sin insertar ningún valor en los campos. Se comprobará que produce un error en la creación.
Resultado	Éxito.

Resultado prueba 5: Creación de usuario vacío

Identificador	CP-06
Descripción	Inicio de la aplicación. Creación de usuario utilizando unos valores en uso. (Creación de doble usuario). Se comprobará que produce un error en la creación.
Resultado	Éxito.

Resultado prueba 6: Creación de usuario existente

Identificador	CP-07
Descripción	Inicio de la aplicación. Identificación correcta por primera vez en el historial del usuario. Se muestran las categorías por defecto establecidas en la aplicación. Se comprobará que aparecen las categorías por defecto establecidas.
Resultado	Éxito.

Resultado prueba 7: Muestreo de categorías por defecto

Identificador	CP-08
Descripción	Inicio de la aplicación. Identificación correcta. Creación de una nueva categoría sin valores en la creación. Se comprobará que produce un error a la hora de la creación.
Resultado	Éxito.

Resultado prueba 8: Creación de categoría

Identificador	CP-09
Descripción	Inicio de la aplicación. Identificación correcta. Creación de una nueva categoría sin valores en la creación. Se comprobará que produce un error a la hora de la creación.
Resultado	Éxito.

Resultado prueba 9: Creación de categoría vacía

Identificador	CP-10
Descripción	Inicio de la aplicación. Identificación correcta. Creación de una nueva categoría con valores correctos. Se comprobará que se crea la categoría correctamente.
Resultado	Éxito.

Resultado prueba 10: Creación de categoría correcta

Identificador	CP-11
Descripción	Inicio de la aplicación. Identificación correcta. Creación de una nueva categoría con título ya existente. Se comprobará que produce un error a la hora de la creación.
Resultado	Éxito.

Resultado prueba 11: Creación de categoría existente

Identificador	CP-12
Descripción	Inicio de la aplicación. Identificación correcta. Comprobación del contador de elementos de la categoría. Se comprobará si el contador recoge la cantidad exacta de elementos pertenecientes a la categoría.
Resultado	Éxito.

Resultado prueba 12: Mostrar contador de elementos en las categorías

Identificador	CP-13
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Creación de un nuevo elemento sin introducir valores. Se comprobará que no se produce la creación del elemento.
Resultado	Éxito.

Resultado prueba 13: Creación de elemento

Identificador	CP-14
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Navegación hacia un elemento. Se elimina dicho elemento. Se comprobará que se elimina correctamente.
Resultado	Éxito.

Resultado prueba 14: Eliminación de elementos

Identificador	CP-15
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Navegación hacia un elemento. Se modificará el elemento Se comprobará que se modifica correctamente.
Resultado	Éxito.

Resultado prueba 15: Modificación de elementos

Identificador	CP-16
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Se eliminará la categoría creada por el usuario. Se comprobará que se elimina correctamente la categoría creada por el usuario.
Resultado	Éxito.

Resultado prueba 16: Eliminación correcta de la categoría

Identificador	CP-17
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Se eliminará una categoría creada por la aplicación No es posible la eliminación de la categoría.
Resultado	Éxito.

Resultado prueba 17: Eliminación fallida de categoría por defecto

Identificador	CP-18
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a una categoría. Navegación hacia un elemento. Borrado de un elemento. Se comprobará que se elimina correctamente.
Resultado	Éxito.

Resultado prueba 18: Eliminación de elemento

Identificador	CP-19
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a la sección de configuración. Guardar la configuración sin modificar ninguna opción. Se comprobará que no se producen cambios en la configuración.
Resultado	Éxito.

Resultado prueba 19: Modificación de la configuración

Identificador	CP-20
Descripción	Inicio de la aplicación. Identificación correcta. Navegación a la sección de configuración. Modificar la opción de cifrado. Se comprobará que se guardan los cambios correctamente.
Resultado	Éxito.

Resultado prueba 20: Modificar la opción de cifrado

Identificador	CP-21
Descripción	Inicio de la aplicación. Identificación correcta. Comprobación del listado. Se comprobará que el listado está ordenado alfabéticamente.
Resultado	Éxito.

Resultado prueba 21: Comprobación de ordenación del listado

Identificador	CP-22
Descripción	Inicio de la aplicación. Identificación correcta. Navegación hacia una categoría. Comprobación del listado. Se comprobará que el listado está ordenado alfabéticamente.
Resultado	Éxito.

Resultado prueba 22: Comprobación de ordenación de elementos

4.3.3 Conclusión de las pruebas

Se han realizado todas las pruebas y el resultado ha sido satisfactorio en cada una de ellas. Por lo que se produce la aceptación y el paso a producción del sistema probado.

Capítulo 5: Gestión del proyecto

5.1 Introducción

En este capítulo se trata de los aspectos relativos a la gestión del proyecto. Ésta abarca desde el tiempo empleado en la ejecución del mismo, los recursos utilizados, tanto humanos como materiales, estudio de la desviación del proyecto. Además se detalla el apartado económico, lo invertido como los beneficios esperados de la venta del producto.

5.2 Planificación del proyecto

A continuación se detallan las etapas y las tareas pertenecientes a cada una de ellas. Las principales etapas del proyecto son las siguientes:

- **Planificación:** Etapa más importante en la gestión del proyecto.
- **Análisis:** Está compuesta por todas las tareas de toma de requisitos, tanto funcionales como no funcionales y estudio de tecnologías.
- **Diseño:** En esta etapa se determina la arquitectura del sistema, los componentes y la tecnología a utilizar.
- **Implementación:** Etapa de desarrollo de los requisitos establecidos en los puntos anteriores.
- **Pruebas:** Ejecución de pruebas detalladas en la etapa de diseño.
- **Documentación:** Etapa de creación y control de documentos implicados en el desarrollo del proyecto.

5.2.1 Planificación

Se establecen las tareas iniciales del proyecto.

- **Estudio de viabilidad:** Se comprobará si la consecución del proyecto es posible tanto en conocimientos como económicamente.
- **Estimación del sistema:** Estudio del tamaño de la aplicación a construir.
- **Estimación de costes:** Establecimiento de los costes previstos a la hora de realizar el proyecto.
- **Estimación de tiempo:** Tiempo que se prevé que consuma el proyecto.
- **Planificación inicial:** Programación inicial del desarrollo del proyecto.

5.2.2 Análisis

Se establecen las tareas para la etapa de análisis.

- **Captura de requisitos:** Captura de la funcionalidad de la aplicación. Requisitos funcionales y no funcionales.
- **Estudio de dispositivos:** En esta tarea se realizó un estudio exhaustivo de los distintos dispositivos objetivo de la aplicación del proyecto.
- **Estudio del sistema:** Formación sobre el sistema operativo del dispositivo elegido. En el caso del proyecto iOS.
- **Algoritmos de cifrado:** Estudio de los distintos algoritmos que el sistema operativo ofrece para el proyecto.
- **Sistemas de bases de datos:** Formación sobre el tipo de sistemas gestores de información persistente disponibles en el dispositivo.
- **Casos de uso:** Análisis de la utilidad de la aplicación.
- **Casos de prueba:** Tarea en la que se demuestra el cumplimiento de la funcionalidad por parte de la aplicación.

5.2.3 Diseño

Se establecen las tareas para la etapa de diseño.

- **Arquitectura del software:** En esta tarea se establece la estructura de construcción del software.
- **Controladores de vista:** Especificación de los gestores de las vistas.
- **Base de datos:** En esta tarea se define el sistema gestor de bases de datos que va a utilizar la aplicación.
- **Interfaz de usuario:** Se define la estructura, colores, imágenes que van a utilizar las distintas pantallas de la aplicación.

5.2.4 Implementación

Se establecen las tareas para la etapa de diseño.

- **Adquisición del equipo:** Tarea de compra del equipo necesario para la realización del proyecto.
- **Adquisición del software:** Adquisición e instalación del software utilizado en la realización del proyecto.
- **Sistema gestor base de datos:** Desarrollo del sistema gestor de bases de datos utilizado en la aplicación
- **Sistema localizador:** Desarrollo del sistema de localización utilizado en la aplicación.
- **Sistema de cifrado:** Desarrollo de los sistemas de cifrado utilizados en la aplicación.
- **Interfaz de usuario:** Desarrollo de las pantallas de la aplicación.

5.3 Planificación inicial del proyecto

A continuación se muestra el listado de tareas con las duraciones planificadas inicialmente, Figura 58.

		Nombre	Duración	Inicio	Terminado	Predecesores
1		Inicio de Proyecto	0 days	13/04/11 8:00	13/04/11 8:00	
2		**Planificación**	**6 days**	**13/04/11 8:00**	**21/04/11 8:00**	
3		Estudio de viabilidad	1 day	13/04/11 8:00	13/04/11 17:00	1
4		Estimación del sistema	1 day	14/04/11 8:00	14/04/11 17:00	3
5		Estimación de costes	1 day	15/04/11 8:00	15/04/11 17:00	4
6		Estimación de tiempo	1 day	18/04/11 8:00	18/04/11 17:00	5
7		Planificación inicial	2 days	19/04/11 8:00	20/04/11 17:00	6
8		Fin planificación	0 days	21/04/11 8:00	21/04/11 8:00	7
9		**Análisis**	**43 days**	**21/04/11 8:00**	**21/06/11 8:00**	
10		Captura de requisitos	3 days	21/04/11 8:00	25/04/11 17:00	8
11		Estudio de dispositivos	1 day	26/04/11 8:00	26/04/11 17:00	10
12		Estudio del sistema	7 days	27/04/11 8:00	5/05/11 17:00	11
13		Algoritmos de cifrado	15 days	6/05/11 8:00	26/05/11 17:00	12
14		Sistemas de bases de datos	10 days	27/05/11 8:00	9/06/11 17:00	13
15		Casos de uso	3 days	10/06/11 8:00	14/06/11 17:00	14
16		Casos de prueba	4 days	15/06/11 8:00	20/06/11 17:00	15
17		Fin análisis	0 days	21/06/11 8:00	21/06/11 8:00	16
18		**Diseño**	**33 days**	**21/06/11 8:00**	**5/08/11 8:00**	
19		Arquitectura del software	5 days	21/06/11 8:00	27/06/11 17:00	17
20		Controladores de vista	10 days	28/06/11 8:00	11/07/11 17:00	19
21		Base de datos	5 days	12/07/11 8:00	18/07/11 17:00	20
22		Interfaz de usuario	13 days	19/07/11 8:00	4/08/11 17:00	21
23		Fin diseño	0 days	5/08/11 8:00	5/08/11 8:00	22
24		**Implementación**	**54 days**	**5/08/11 8:00**	**20/10/11 8:00**	
25		Adquisición del equipo	0,5 days	5/08/11 8:00	5/08/11 12:00	23
26		Adquisición del software	0,5 days	5/08/11 12:00	5/08/11 17:00	25
27		Sistema gestor base de d...	5 days	8/08/11 8:00	12/08/11 17:00	26
28		Sistema localizador	3 days	15/08/11 8:00	17/08/11 17:00	27
29		Sistema cifrador	20 days	18/08/11 8:00	14/09/11 17:00	28
30		Interfaz de usuario	25 days	15/09/11 8:00	19/10/11 17:00	29
31		Fin implementación	0 days	20/10/11 8:00	20/10/11 8:00	30
32		**Pruebas**	**6 days**	**20/10/11 8:00**	**28/10/11 8:00**	
33		Pruebas	6 days	20/10/11 8:00	27/10/11 17:00	31
34		Fin pruebas	0 days	28/10/11 8:00	28/10/11 8:00	33
35		**Documentación**	**30 days**	**28/10/11 8:00**	**8/12/11 17:00**	

PFC - página1

Figura 58: Planificación del proyecto

Se estima que el proyecto se inicie el día 13 de abril de 2011. Las fechas de inicio y fin de cada fase se detallan a continuación:

- Planificación:
  - Inicio: 13/04/2011
  - Fin: 21/04/2011
  - Duración: 6 días
- Análisis:
  - Inicio: 21/04/2011
  - Fin: 21/06/2011
  - Duración: 43 días
- Diseño:
  - Inicio: 21/06/2011
  - Fin: 5/08/2011
  - Duración: 33 días
- Implementación:
  - Inicio: 5/08/2011
  - Fin: 20/10/2011
  - Duración: 54 días
- Pruebas:
  - Inicio: 20/10/2011
  - Fin: 28/10/2011
  - Duración: 6 días
- Documentación:
  - Inicio: 28/10/2011
  - Fin: 8/12/2011
  - Duración: 30 días

5.3 Recursos asignados al proyecto

En esta sección no se entra en detalle sobre los recursos humanos dedicados al proyecto puesto que solo hay una persona asignada a él. Los recursos materiales utilizados para el proyecto se pueden agrupar en:

- Hardware.
- Software.

5.3.1 Hardware

A continuación se muestra el equipo utilizado para la realización del proyecto, Tabla 15.

Equipos de trabajo	
Portátil Apple MacBook, procesador Intel i5 2.4 GHz, 4 GB 1067 MHz DDR3	Mac mini, procesador Intel i5 2.7 GHz, 4 GB 1600 MHz DDR3
Banco de pruebas: iPhone 4S	iOS 6

Tabla 15: Equipos de trabajo

5.3.2 Software

A continuación se muestra la parte de software utilizado para la realización del proyecto, Tabla 16.

Software		
MS Office 2007 para Mac.	http://www.microsoft.com	Licencia usuario.
OpenProj.	http://sourceforge.net/projects/openproj/	Shareware.
Firefox. Plugin - SQLITE Manager.	http://www.firefox.com/	Shareware.
xCode 4.1.	http://developer.apple.com/	Licencia usuario.
Adobe Photoshop 5.1 CSS.	http://www.adobe.com/	Licencia usuario.
OmniGraffle.	http://www.omnigroup.com/products/omnigraffle/	Licencia usuario.

Tabla 16: Listado de Software

5.4 Estudio económico

En esta sección se detallará el análisis económico realizado para los tres conjuntos del proyecto. Coste de los recursos humanos, recursos materiales e indirectos.

Además se hará un estudio de la desviación producida en el coste final del proyecto debido al retraso en tiempo producido.

5.4.1 Estimación coste de los recursos humanos

El único recurso humano que ha participado en el proyecto es Jesús Martín Chicharro Gallego. Para este coste se va a utilizar la tarifa actual del mercado para un consultor Junior en la empresa *Realtech Consulting*, 32 €/hora.

En un principio se había establecido como jornada laboral 8 horas diarias. Sin embargo, por motivos laborales no se pudo ejecutar dicha jornada. Se ha establecido como jornada 2 horas diarias.

Por lo tanto, se describe el coste de los recursos humanos de la estimación final en la siguiente tabla, Tabla 17. En esta estimación se incluyen los gastos generados por la contratación como son el seguro de desempleo, seguridad social, contingencias.

Concepto	Tiempo	Honorarios	Coste total
Consultor Junior	100 días, 2 horas 72 días, 8 horas	32 €/ h	24.832 €

Tabla 17: Recursos Humanos

5.4.1 Estimación coste de los recursos materiales

En esta sección se desglosarán los recursos materiales utilizados en el desarrollo del proyecto. Por un lado los recursos hardware y por otro los relacionados con el software.

5.4.1.1 Recursos hardware

A continuación se desglosa el coste por equipo, Tabla 18. Los cálculos del coste se han realizado teniendo en cuenta que se realiza una inversión en la compra del ordenador portátil, *MacBook Pro* y que ya se contaba con el ordenador *Mac Mini*.

Concepto	Vida útil	Coste unitario	Coste total
MacBook Pro	4 años	1.749,00 €	1.749,00 €
Mac Mini	4 años	599 €	0 €
iPhone 4S	6 años	599 €	8,00 €
			1.757,00 €

Tabla 18: Equipos de trabajo

5.4.1.2 Recursos software

En la realización del proyecto se ha intentado utilizar software libre. Sin embargo en algunos casos se ha tenido que utilizar software privado para la realización de alguna tarea.

Además de las herramientas software, se ha de adquirir una licencia de desarrollador para iOS. Esta licencia se paga anualmente y permite distribuir la aplicación mediante el App Store de Apple. Los cálculos de los costes se realizan teniendo en cuenta la vida útil y la utilización de ese recurso en el proyecto, Tabla 19.

Concepto	Vida útil	Coste unitario	Coste total
MS Office 2007 para Mac	4 años	139 €	34 €
OpenProj	4 años	Gratuito	0 €
Firefox. Plugin - SQLITE Manager	4 años	Gratuito	0 €
xCode 4.1	1 año	Gratuito	0 €
Adobe Photoshop 5.1 CSS	4 años	119 €	29,75 €
Licencia de desarrollador	1 año	99 €	99 €
			162,75 €

Tabla 19: Coste del Software

5.4.1.3 Costes indirectos

Además se presenta un tipo de gasto que no se puede enmarcar en las dos categorías anteriores. En la siguiente tabla, Tabla 20, se establecen estos costes.

Tipo	Cuota	Coste para el proyecto	Coste unitario	Coste total
Luz	40 €/mes	20%	8 €/mes	71 €
Conexión a Internet	20 €/mes	20%	4 €/mes	36 €
				107 €

Tabla 20: Costes indirectos

5.4.1.4 Resumen de los costes

A continuación se muestra el resumen del coste del proyecto, Tabla 21.

Tipo	Coste total
Coste de los recursos humanos	24.832 €
Coste de los recursos materiales – Hardware	1.757,00
Coste de los recursos materiales – Software	162,75 €
Costes indirectos	107 €
	26.858 €

Tabla 21: Resumen de los costes

El presupuesto final de la realización del proyecto han sido 26.858 €. Se observa que el desglose que más presupuesto se lleva es el de los recursos humanos. Además hay que tener en cuenta que se ha producido una inversión inicial en cuanto a los equipos como a las herramientas software que se han utilizado y que en los proyectos siguientes no hay que realizar.

5.5 Distribución de la aplicación

En este apartado se analizará la mejor forma de venta de la aplicación en el App Store de Apple. Los métodos de distribución que se utilizan en la actualidad son:

- Aplicación Lite y versión PRO.
- Aplicación venta no gratuita.
- Aplicación venta gratuita con gestor de anuncios.

5.5.1 Aplicación Lite y versión PRO

Este método es el que mejor acogida tiene por parte del usuario debido a que deja probar una versión de prueba y en caso de que la aplicación sea del agrado del cliente este puede adquirir la versión PRO de la misma. Desechamos este tipo de distribución puesto que no se ha planificado realizar dos versiones de la aplicación.

5.5.2 Aplicación venta no gratuita

Método clásico de distribución de aplicaciones. El cliente paga por la versión final de la aplicación. A continuación se muestran el número de descargas necesarias para hacer frente a la inversión del proyecto y obtener beneficio.

El primer punto a tener en cuenta es que el 30% del beneficio de la venta es para Apple, 1% para el gestor de la tarjeta de crédito y el 69% restante para el desarrollador.

Para calcular el número de aplicaciones que se tienen que vender para tener un beneficio del 20% sobre lo invertido se utiliza el indicador "Valor de retorno de la inversión" (ROI).

Para un precio de venta de 1 €, obteniendo un beneficio de 0,69 € brutos para el desarrollador obtenemos, Tabla 22:

Precio de venta	Beneficio bruto	Beneficio	Número de aplicaciones vendidas	Beneficio de retorno
1 €	0,69 €	20 %	48.769	33.651 €
0,79 €	0,54 €	20 %	62.314	33.651 €
0,5 €	0,34 €	20 %	98.973	33.651 €
1,44 €	0,99 €	20 %	33.989	33.651 €

Tabla 22: Estudio del número de aplicaciones vendidas

El precio medio de venta de las aplicaciones en el App Store está en torno a 1,44 €. El precio que se aproxima más a la realidad en este caso es el de 1 €.

Además del beneficio, es importante conocer el tiempo necesario para obtener de vuelta la inversión. De nada sirve que se cumpla el valor especificado de retorno si este se obtiene pasado mucho tiempo.

A continuación se muestran los tiempos de retorno estimando un número de descargas diarias. El precio de venta unitario se establece en 1 € por descarga, Tabla 23.

Precio de venta	Descargas diarias	Años para el ROI
1 €	20	6,6 años
1 €	40	3,3 años
1 €	80	1,67 años
1 €	160	0,83 años

Tabla 23: Descargas Diarias

5.5.2 Aplicación gratuita e ingresos a partir de publicidad.

Una alternativa a la venta directa de la aplicación, es ofrecerla gratuitamente introduciendo en ella un contenedor, ADBannerView, de publicidad.

Apple ofrece a los desarrolladores el sistema *iAds*. Gestiona automáticamente el acceso a la publicidad ofreciendo beneficios a través de visualización de publicidad. A diferencia de otros sistemas de publicidad, la experiencia publicitaria que ofrece el sistema creado por Apple es más rica en cuanto a contenido interactivo.

El sistema se basa en dos medidores:

- CPM: Coste por millar, midiendo el número de impresiones. 15 € por cada 1000 impresiones.
- CPC: Coste por clic, midiendo el número de personas que pulsan sobre el anuncio. 2 € por cada clic.

Actualmente el número de aplicaciones descargadas del App Store superan los 15.000 millones. Además de esto, Apple ha lanzado cifras de más de 250 millones de dispositivos con iOS instalado. Con estos datos podemos obtener el número de aplicaciones media por usuario.

Número Medio = 15000/250 = 60 aplicaciones por usuario.

De las cuales el 10% de las aplicaciones son de pago. El número medio de aplicaciones gratuitas de media por usuario es 54 aplicaciones.

Es un número bastante alto de aplicaciones susceptibles de contener iAds. Sin embargo el uso medio de las aplicaciones gratuitas es de dos veces al mes. Esto implica que en muchos casos las descargas gratuitas se utilizan una vez y la aplicación cae en el olvido.

Este dato es muy importante debido a que los beneficios en este caso se basan en una alta utilización por parte del usuario.

Debido a que la aplicación es gratuita, se supone tener 250 descargas por día. Además hay que tener en cuenta de que la aplicación una vez descargada, se vuelve a reutilizar. En este primer caso se supone que la utilización de la aplicación es de una vez por descarga.

A continuación, Tabla 24, se detalla el estudio de los beneficios obtenidos a través de este sistema, suponiendo que no se recibe aportación por clic en el banner.

ROI	CPC	CPM	Clic por día	Impresiones por día	Años para el ROI
20%	2 €	15 €	0	250	24 años

Tabla 24: Beneficios por publicidad I

Obteniendo 15 € cada 4 días tenemos:

33.651 / 15 € = 2.243 * 4 días = 8.972 días. = 24 años.

Como se puede observar, para obtener un ROI del 20% deberían pasar 24 años. Un plazo que no permitiría ganarse la vida con este tipo de desarrollos.

Sin embargo hemos supuesto que no se va a recibir beneficio al realizar clics en los banners. Suponiendo que el usuario va a realizar un clic al día manteniendo las mismas impresiones al día tenemos, Tabla 25.

ROI	CPC	CPM	Clic por día	Impresiones por día	Años para el ROI
20%	2 €	15 €	1	250	19 años

Tabla 25: Beneficios por publicidad II

Obteniendo 19 € cada 4 días tenemos:

33.651 / 19 € = 1771 * 4 días = 7084 días. = 19 años.

Como se puede observar el tiempo ha disminuido bastante. Esto es debido a que el negocio está en "obligar" al usuario a pulsar en los banners o tener una aplicación capaz de generar unas impresiones muy altas.

Este tipo de negocio es viable para aplicaciones de grandes empresas que ofrecen un servicio al usuario que en muchos casos es necesario.

5.5.3 Conclusiones

Analizando los resultados obtenidos en los dos casos estudiados. Parece clara la elección de la venta directa de la aplicación. Ya que en un primer momento no se espera un nivel de descarga tan alto como para obtener beneficios en un periodo razonable de tiempo a través del sistema de anuncios que ofrece Apple.

Por lo tanto el sistema elegido para la distribución de la aplicación es el de venta directa.

Capítulo 6: Conclusiones y líneas futuras

6.1 Introducción

En este capítulo se describe el futuro que ha abierto para mí como desarrollador del proyecto en el mercado laboral.

6.2 Conclusiones del proyecto

El proyecto nació como un estudio de la seguridad en los dispositivos móviles al que se le unió la idea de una identificación relativa a la posición del GPS. Consiguiendo un sistema gestor de información privada relacionado parcialmente con la posición actual del dispositivo. Digo parcial por que es posible descifrar los datos con un par de valores usuario/contraseña.

Además se trata de utilizar distintos sistemas de cifrado. El usuario es muy exigente a la hora de tratar con información privada y el seleccionar el tipo de cifrado que el más utilice es un valor añadido para la aplicación.

En un primer momento la aplicación iba a ser monousuario, sin embargo y debido a que es posible que para un mismo usuario la información del trabajo debe ser totalmente privada en su casa. Se creó un sistema multiusuario para resolver este tipo de problemática.

Otro punto a favor, es que la aplicación fue diseñada para sustituir a elementos propios del sistema del dispositivo como es el Keychain, o la aplicación de notas.

6.3 Conclusiones personales

La etapa de desarrollo de este proyecto ha sido una de las más provechosas de la carrera, gracias a la realización de esta aplicación se me abrieron muchas puertas en el mercado laboral.

En un momento de crisis mundial en el que en nuestro país el paro aumenta día tras día y que incluso el estar titulado no te garantiza un puesto de trabajo acorde con los estudios realizados. El desarrollo de aplicaciones para dispositivos móviles crece día a día y la demanda de profesionales en el ámbito de la movilidad aumenta también.

Además, me ha servido para conocer nuevas tecnologías y a aumentar mi capacidad de trabajo en nuevos proyectos de los que no conozco nada más que el nombre de la tecnología a emplear.

6.4 Líneas futuras y próximos empleos

Dado que durante la realización de este proyecto el mercado de los dispositivos móviles creció a un nivel comparable al de la burbuja inmobiliaria, bien sufrida por nuestro país, es posible que se abran nuevas vías de negoció para las aplicaciones móviles.

Uno de los posibles caminos a tomar a partir de aquí, es portar la aplicación a sistemas basados en Android o crear una nueva interfaz gráfica para dispositivos *iPad*.

Personalmente, encontré trabajo como desarrollador iOS en una consultora madrileña, pero durante el periodo de realización del proyecto y gracias a la experiencia adquirida en él, las ofertas de trabajo para la realización de distintos proyectos, incluso como autónomo, no han dejado de llegar.

Además la plataforma en la que se ha desarrollado la aplicación, básicamente para dispositivos *iPhone, iPod Touch*, ha evolucionado hasta el *iPhone 5S*. Con su novedoso asistente *Siri,* lector de huellas digitales. Sería posible utilizar dicho asistente para la gestión de la información incluso integrar la aplicación con otras aplicaciones del sistema, para poder utilizar los datos a través del navegador o utilizar el lector de huellas digitales para utilizarlo junto a la geo posición para identificar al usuario.

Otro punto en el que se podría ampliar la aplicación es en el tipo de venta. Se habló en el capítulo anterior de crear varios tipos de aplicaciones. Una aplicación gratuita con publicidad para el usuario y una aplicación de pago sin publicidad que amplíe la funcionalidad del sistema.

Con el avance de las tecnologías se podría implementar el acceso a almacenaje en la nube con *iCloud* o *Dropbox*, permitiendo que al tener la aplicación en varios dispositivos ésta esté sincronizada en todo momento. Además esto serviría también como copia de seguridad de la base de datos.

Capítulo 7: Glosario de términos

Android:

Es un sistema operativo basado en Linux, diseñado principalmente para dispositivos móviles con pantalla táctil como teléfonos inteligentes o tabletas o inicialmente desarrollados por Android, Inc que posteriormente fue comprado por Google, Inc.

App Store:

Es un 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 OSX. La página web de este servicio es http://store.apple.com/es.

Banners:

(En español: banderola) es un formato publicitario en Internet. Esta forma de publicidad online consiste en incluir una pieza publicitaria dentro de una página web. Prácticamente en la totalidad de los casos, su objetivo es atraer tráfico hacia el sitio web del anunciante que paga por su inclusión.

BlackBerry App World:

BlackBerry App World es un servicio de distribución de aplicaciones desarrollado por *BlackBerry Ltd* destinado a los dispositivos de la misma compañía. El servicio permite a los usuarios buscar, descargar y actualizar aplicaciones para sus dispositivos.

BlackBerry:

Es un conjunto de dispositivos inalámbricos y servicios diseñados y vendidos por la empresa *Blackberry Ltd*. Estos dispositivos se caracterizan por tener un teclado físico en sus modelos más antiguos. Sin embargo los nuevos dispositivos se basan en una pantalla táctil con un teclado virtual.

Google Play:

Google Play (antes *Android Market*) es una tienda de software en línea desarrollada por Google para los dispositivos con sistema operativo Android. Es una aplicación que está preinstalada en la mayoría de los dispositivos Android y que permite a los usuarios buscar, obtener información y descargar aplicaciones publicadas por desarrolladores terceros.

IDE:

Un entorno de desarrollo integrado, llamado también IDE (siglas en inglés de *integrated development environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

iOS:

iOS es un sistema operativo móvil de la empresa *Apple Inc,* Originalmente desarrollado para el *iPhone* (*iPhone OS*), siendo después usado en dispositivos como el *iPod Touch*, *iPad* y el *Apple TV*. *Apple, Inc*. no permite la instalación de iOS en hardware de terceros.

Jailbreak:

El *Jailbreak* es el proceso de eliminar las limitaciones impuestas por Apple Inc, en dispositivos que utilicen el sistema operativo iOS mediante el uso de *kernels* modificados.

SDK:

Un kit de desarrollo de software o SDK (siglas en inglés de *software development kit*) es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto.

iPhone Explorer

Software de terceros desarrollado por MacroPlant que permite el acceso al sistema de ficheros del dispositivo.

Bibliografía

[1] Ranking de dispositivos en Europa
<http://www.comscore.com/Press_Events/Press_Releases/2011/9/Android_Captures_number_2_Ranking_Among_Smartphone_Platforms_in_EU5>

[2] Piratería en Android e iOS.
<http://www.cultofandroid.com/27547/android-piracy-outnumbers-ios-piracy-by-141-driving-devs-to-freemium-only-model/>

[3] Estudio de piratería en Android.
<http://www.yankeegroup.com/ResearchDocument.do?id=57254>

[4] Descompilar APK. *<http://www.youtube.com/watch?v=d_imW9jAct0>*

[5] Algoritmo de cifrado CAST, Universidad de Michigan
<http://www.umich.edu/~x509/ssleay/rfc2144.html>

[6] Los peligros de repetir patrones
<http://eprint.iacr.org/2013/241.pdf>

[7] CS 193P iPhone Application Development (Stanford)
<http://www.stanford.edu/class/cs193p/cgi-bin/drupal/>

[8] Apple Developer
< https://developer.apple.com/>

[9] Android developer Guide
< http://developer.android.com/guide/components/index.html/>

[10] HTML5 Reference
< http://dev.w3.org/html5/html-author/>

[11] Android developer Guide
< http://developer.android.com/guide/components/index.html/>

[12] Android developer Guide
< http://developer.android.com/guide/components/index.html/>

[13] Android developer Guide
< http://developer.android.com/guide/components/index.html/>

[14] Android developer Guide
< http://developer.android.com/guide/components/index.html/>